Andrew's Two-Day R Course
Day Two

Andrew Stewart

Andrew.Stewart@manchester.ac.uk

@ajstewart_lang

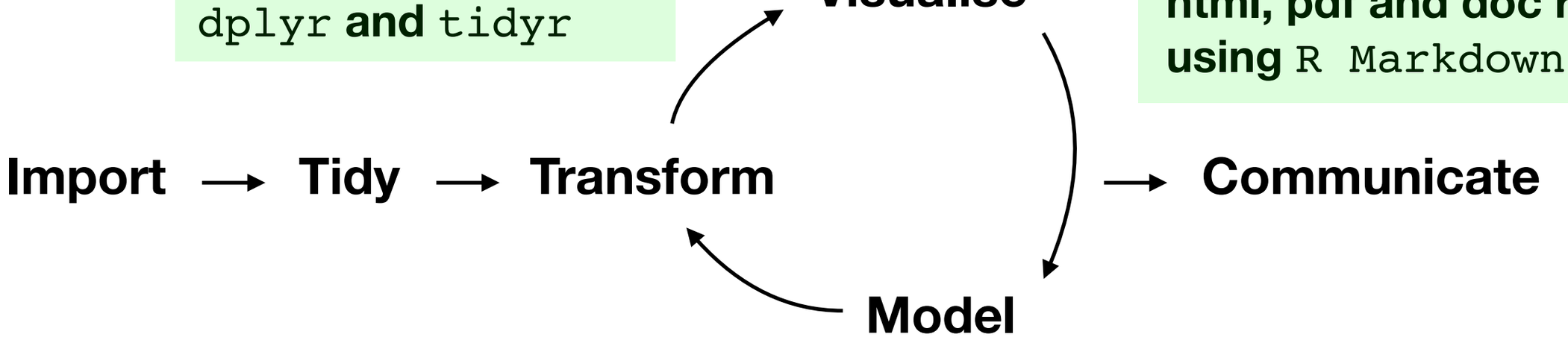
Plan for Today

- Tutorial this morning looking at linear models (regression and AN(C)OVA) and (generalised) linear mixed models.
- LMMs allow for models with a combination of fixed and random effects (intercepts and slopes).
- We will look at designs with one factor with several levels, and 2 x 2 designs for continuous and dichotomous data.
- Examination of measures of model fit, and using *emmeans* to interpret interactions.
- You doing all of the above in this afternoon's lab.

Workflow

Data plots using `ggplot()` **and** `pirateplot()`

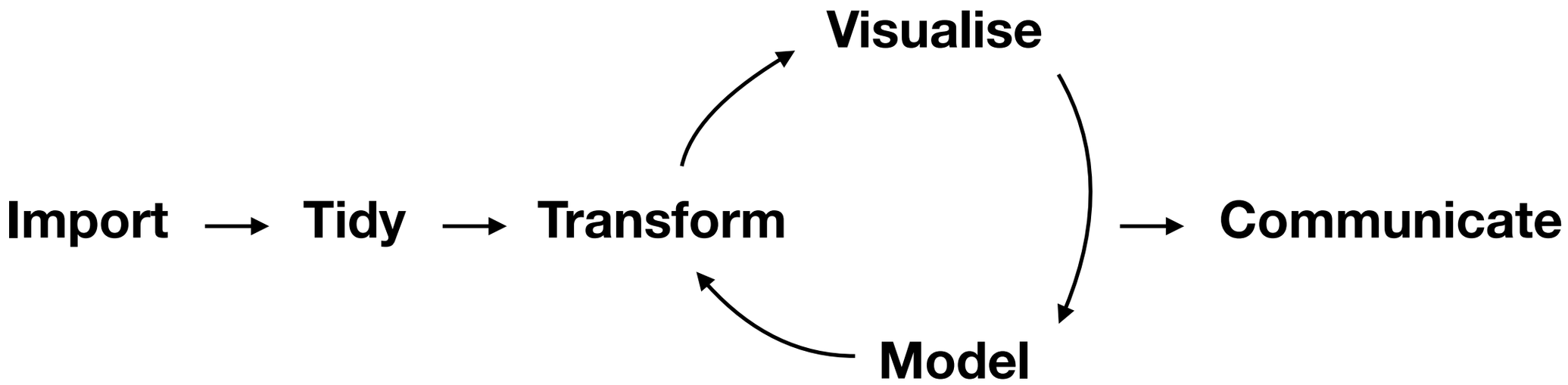
Tidying and transforming using `dplyr` **and** `tidyr`



html, pdf and doc reports using `R Markdown`

ANO(C)VA using `afex()` **and** `aov()`
Linear regression using `lm()` **and** `step()`
(Generalised) linear mixed models using `lmer()` **and** `glmer()`

Workflow



ANO(C)VA using `afex()` and `aov()`
Linear regression using `lm()` and `step()`
(Generalised) linear mixed models using `lmer()` and `glmer()`

Statistical Models

- Most of what we do in applying statistics in Psychology is model building. We build a statistical model and test whether it is a good fit for our data - in other words, whether it describes our data well.

- All models are an approximation of reality, and some are better than others…

- Or to paraphrase the statistician George Box, **all models are wrong but some are useful**…

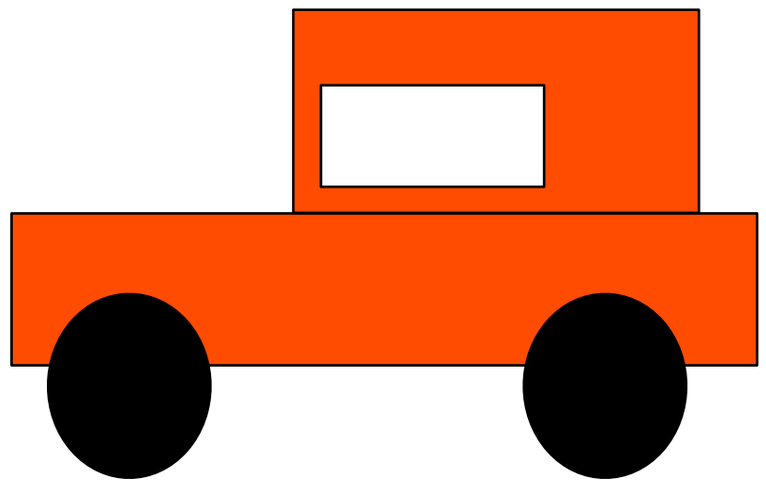
Regression

- Regression is where we want to predict the value of one variable (called our Outcome variable) on the basis of the value of one or more predictor variables.

- Simple regression is when we have one predictor, multiple regression is when we have more than one…

- Most commonly used regression type is OLS (ordinary least squares) which works by minimising the distance (deviation) between the observed data and the linear model.

Real data



Model 1

Model 2

- So how do we tell if a particular statistical model is a good fit to our data?
- We can look at the extent to which our data deviate from a particular model (where deviation = error)…

Real data

=

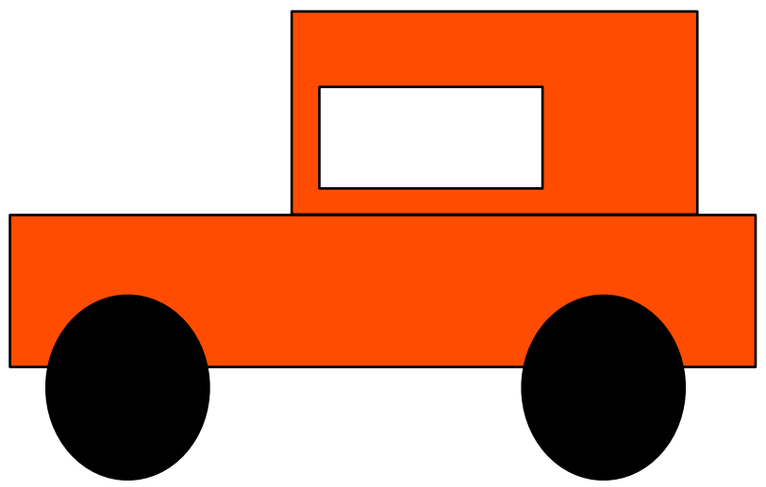
Model 1

+ Error

Real data

=

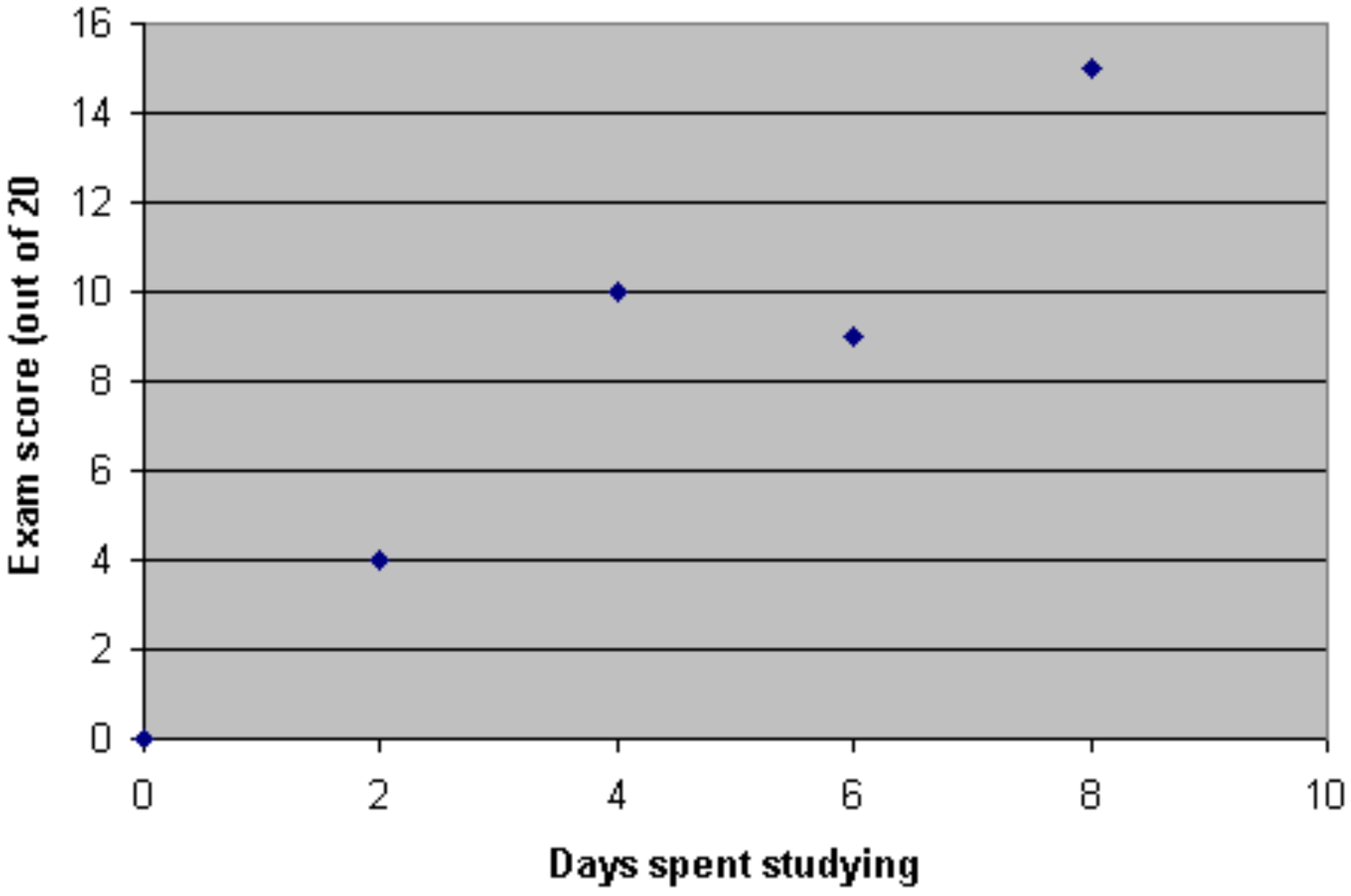
Model 2



+ Error

- We want to select the model which has the smallest error (aka model residuals)…

Regression



We can plot data on exam performance and days spent studying.

Wouldn't it be helpful if we could draw a straight line such that if we know the value on one axis (x say), we could predict the value on the other (y say) ?

Plotting a straight line

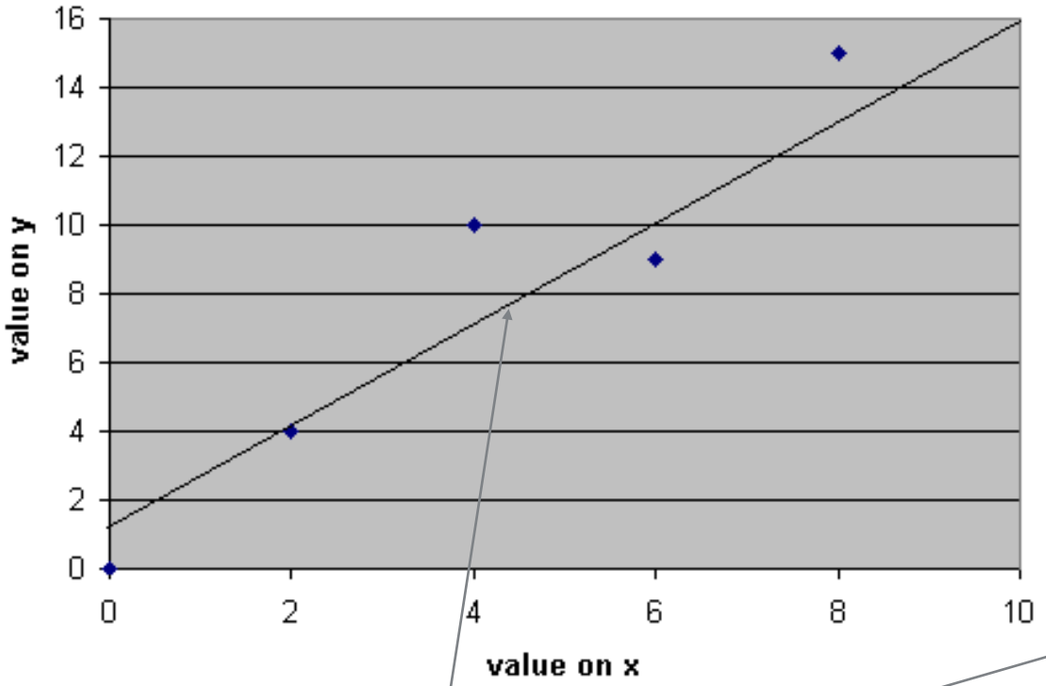
- For any data plots such as on the previous slide, when we have one predictor (x) we could plot many straight lines.

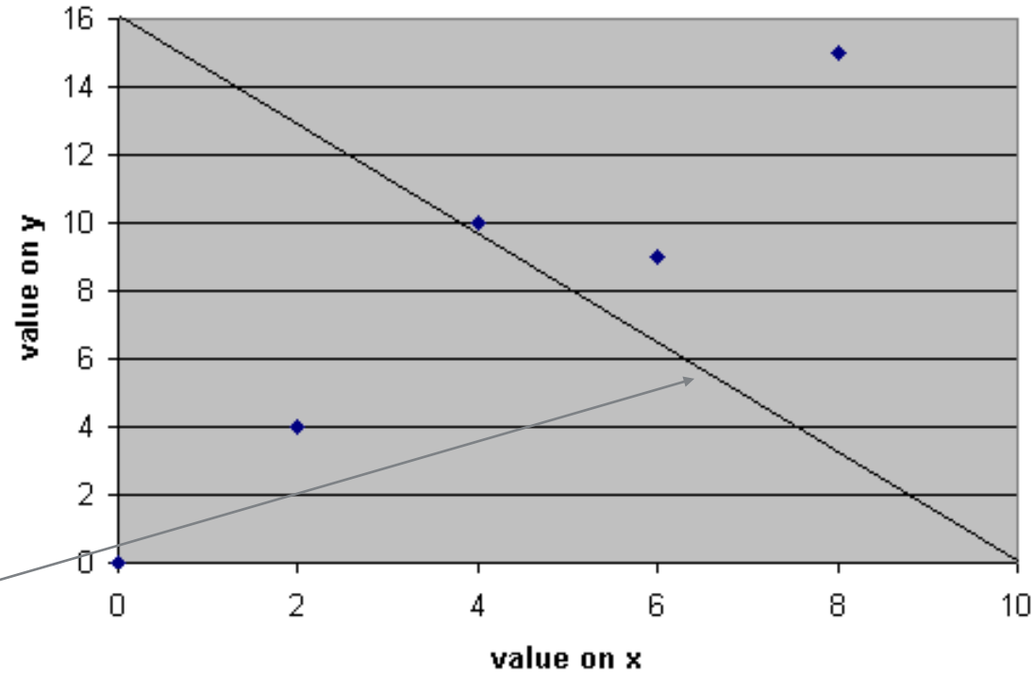
$$y = \beta x_i + \beta_o + residual_i$$

β = gradient of the line

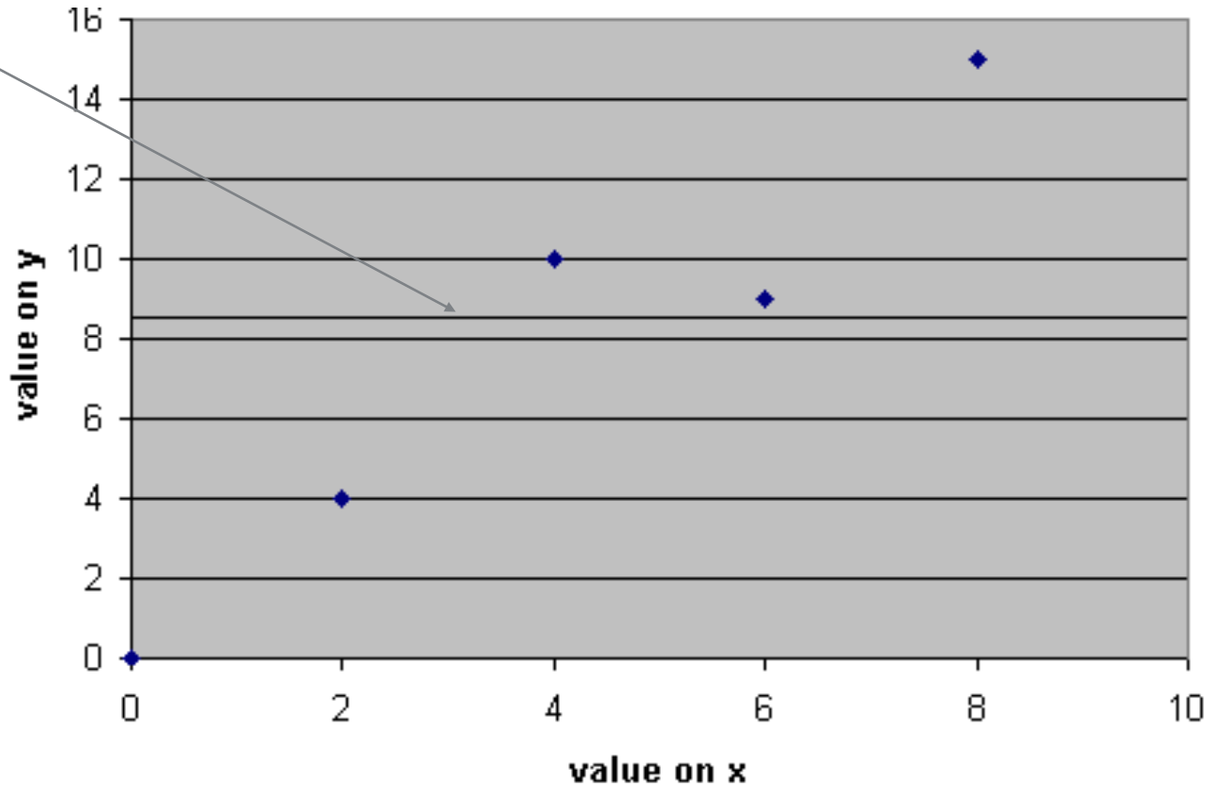
β_o = intercept (when x=0)

$residual_i$ = difference between predicted score and actual score for participant i





Which line seems the best fit ?

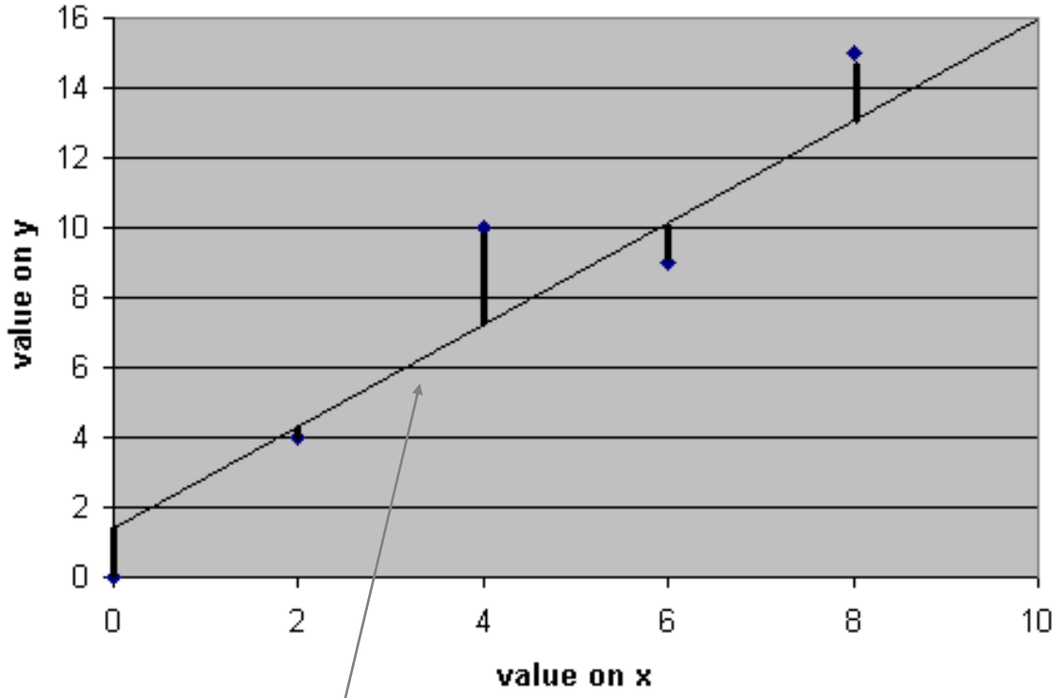


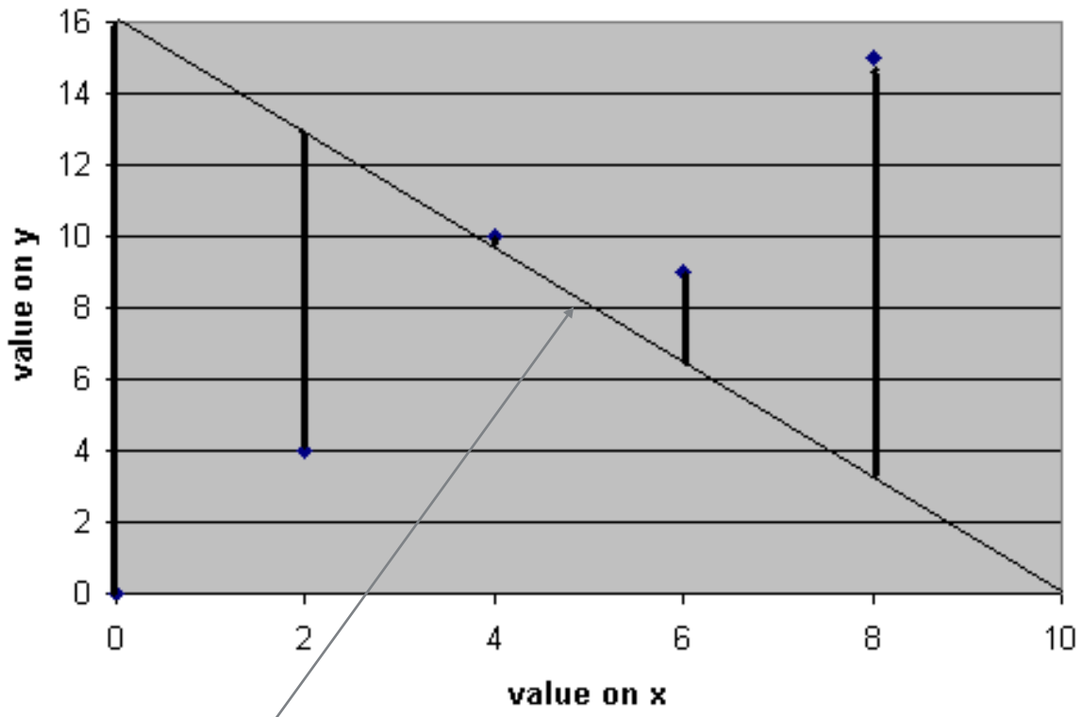
Determining the best line

- For any line, we can calculate what's known as the Least Squares.

- The Least Squares method in regression provides us with a line that results in the least differences between the values predicted by the line and the data themselves….

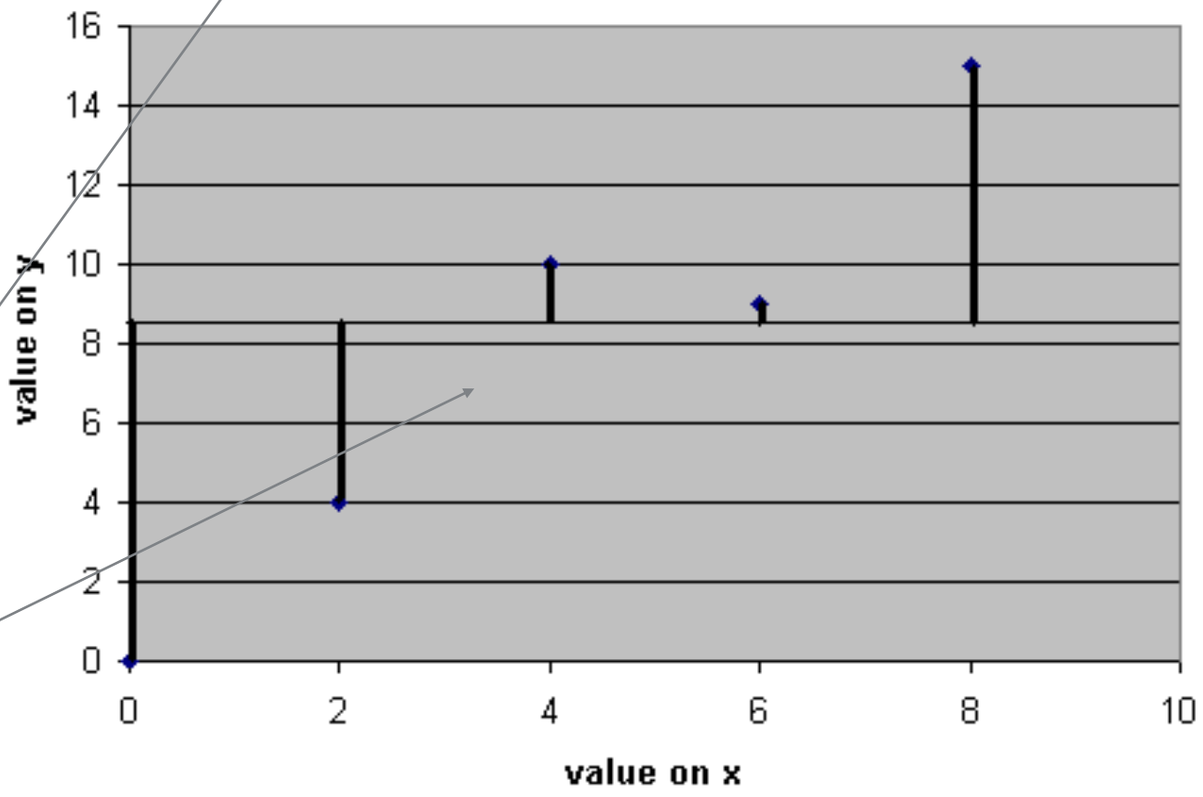
- So, for the three possible lines we just looked at….





We can see that this line seems to be the best fit as it leads to the least error between the predicted data (the line) and our observed data (the points).

These two lines aren't much good as they lead to a lot of error between predicted and observed data.



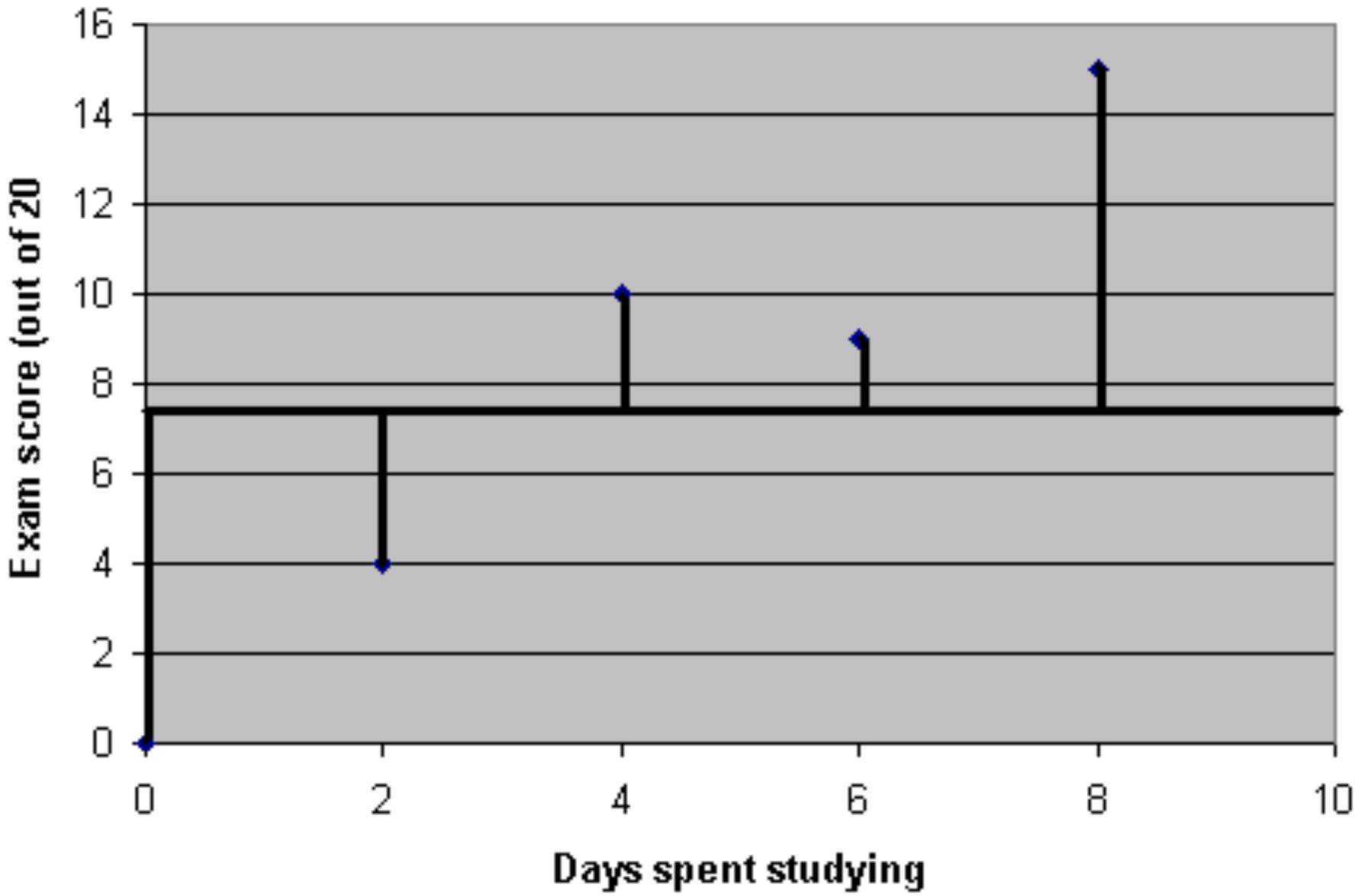
How do we determine how good a fit our line is ?

- We could work out by how much each observed value differs from the mean of y.

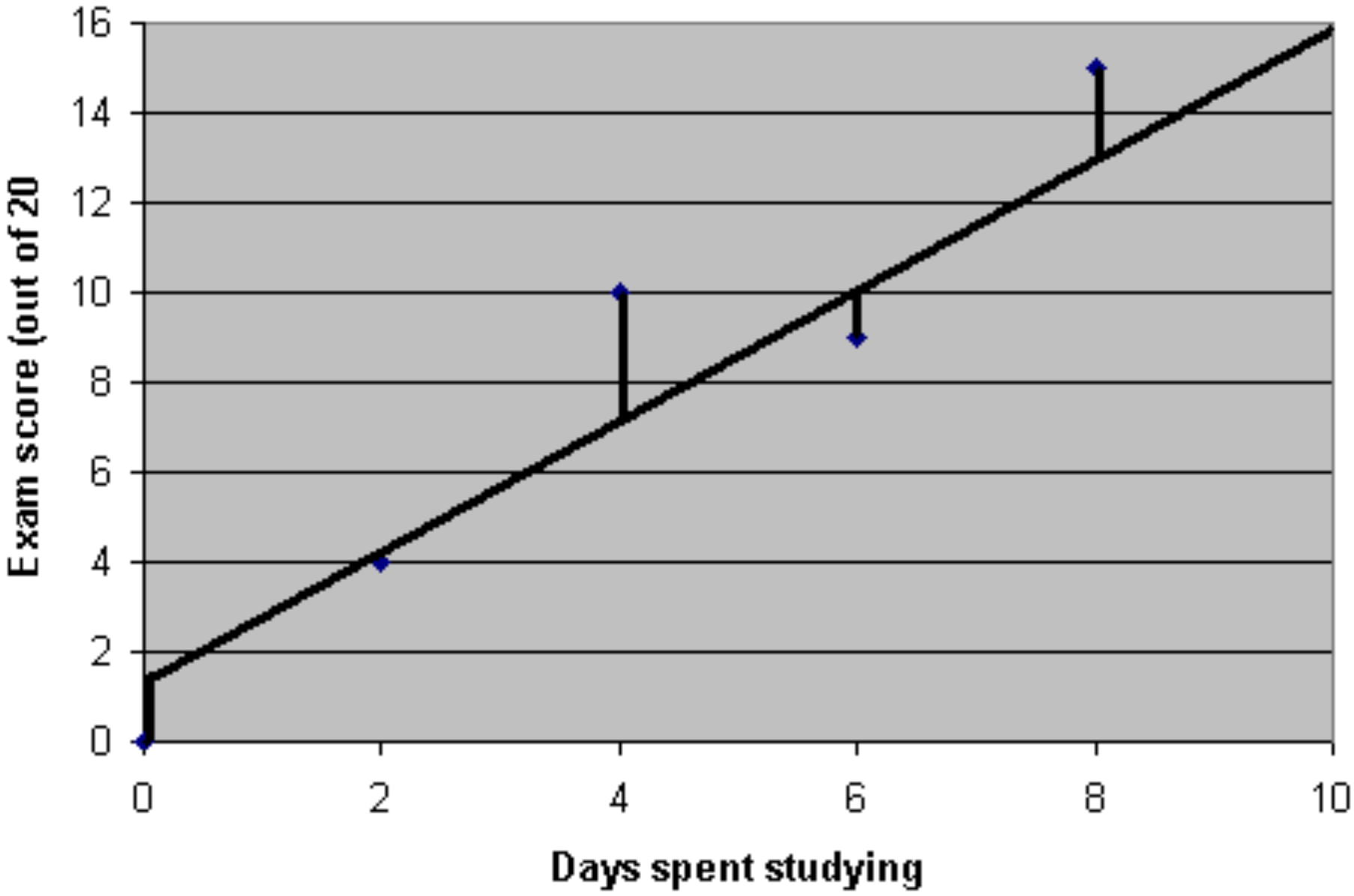
- We could work out by how much each observed value differs from the regression line.

- We could work out by how much the mean value of y differs from the regression line (for different values of x).

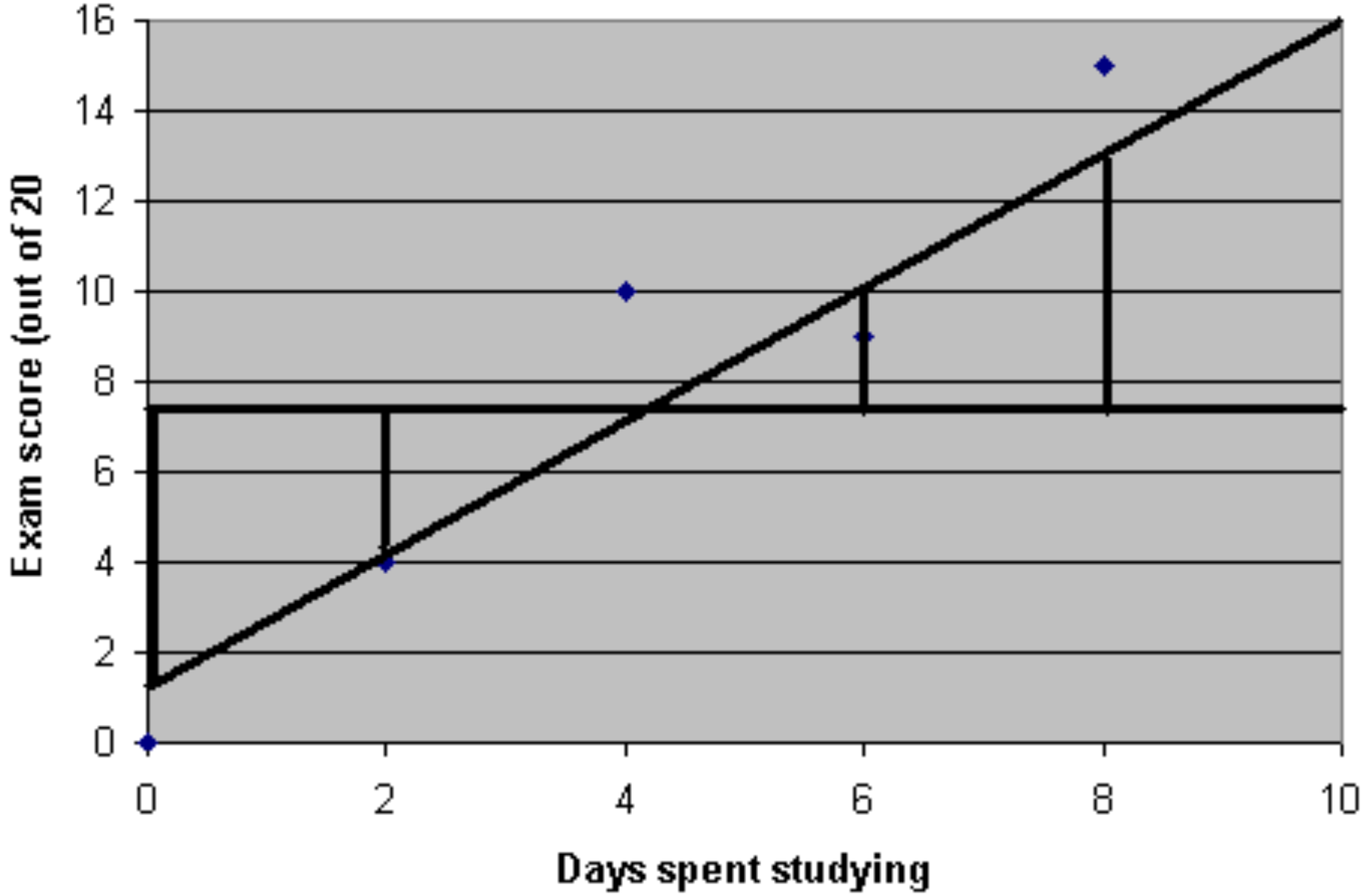
How much each observed value differs from the mean of y (called SS_T):



How much each observed value differs from the regression line (called SS_R):



How much the mean value of Y differs from the regression line for different values of x (called SS_M):



- If SS_M is large, then the regression model is better than the mean in terms of predicting values of the outcome variable.

- If SS_M is small, then the regression model is not much better than the mean in terms of predicting values of the outcome variable.

- We can calculate the proportion of improvement in prediction by looking at the ratio of SS_M to SS_T.

- Actually, this is called R^2 so:

$$R^2 = \frac{SS_M}{SS_T}$$

And this is the <u>same</u> R^2 that you get when squaring the Pearson correlation coefficient…..

- We can also assess how good our model is by using the F-test.

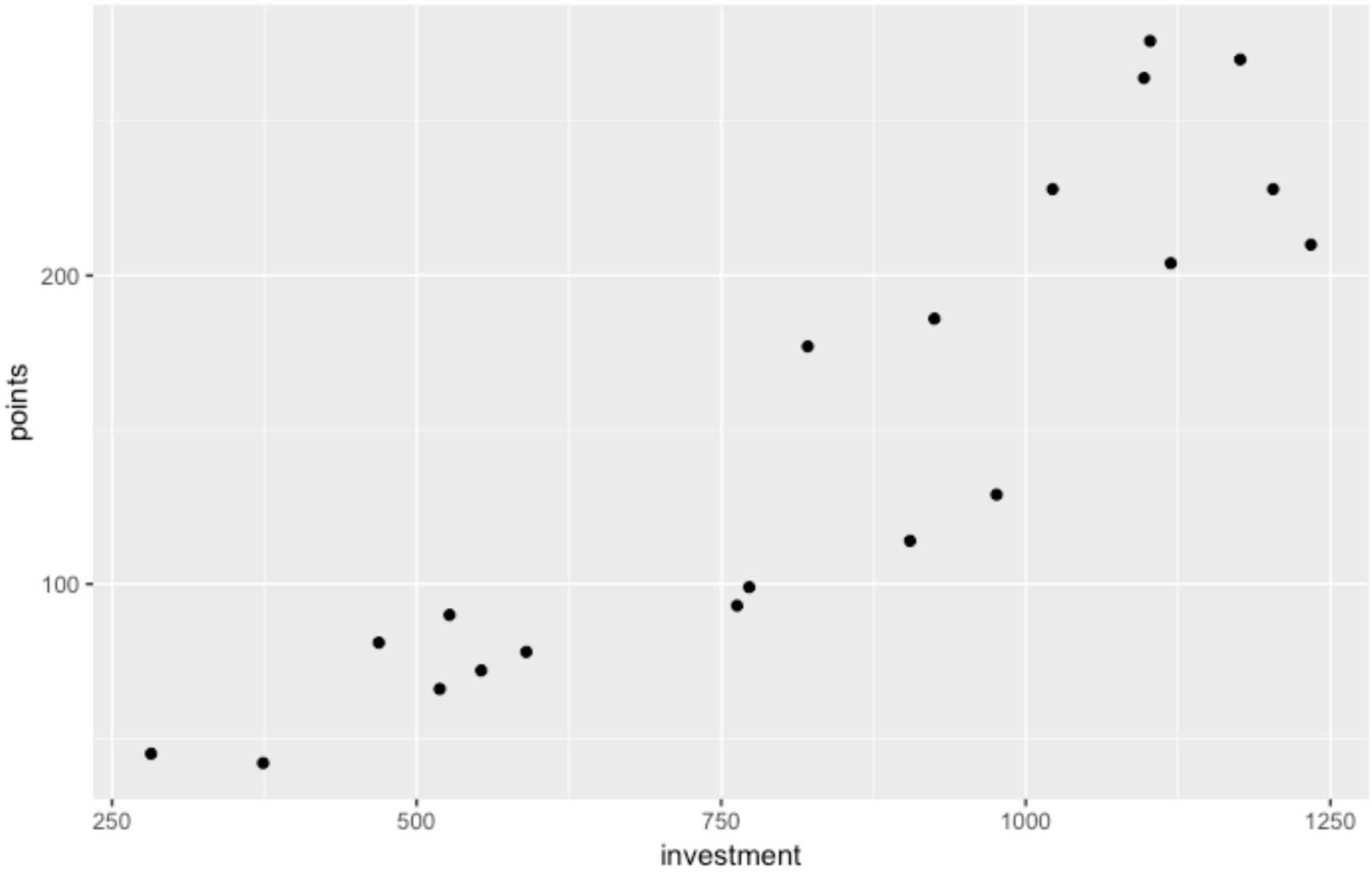
$$F = \frac{MS_M}{MS_R}$$

- The F-test is based on the ratio of the improvement due to the model (MS_M) and the difference between the model and the observed data (MS_R).

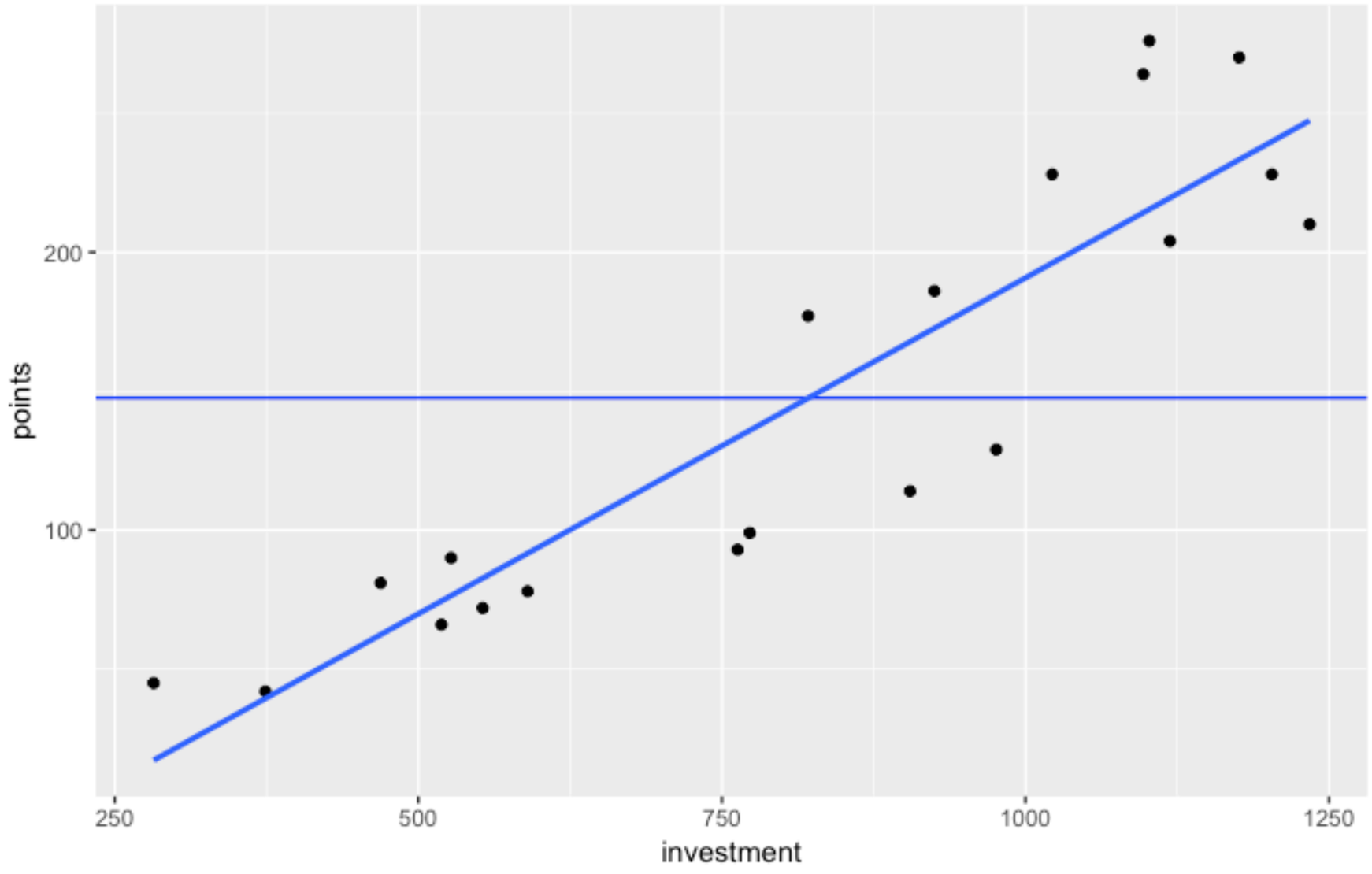
An example

Imagine that you are Formula 1 team director. You're interested in understanding how the number of points that a team scores is predicted by the amount of money invested in the team. As well as being in charge of F1, you also have a secret interest in statistical analysis. In "dataset1" you will find (for each of the 20 drivers) the amount of money invested in their particular car (in £100,000s) plus the total number of points they were awarded over the season. Work out the simple linear regression equation that captures the relationship between investment (as our predictor) and points awarded (as our outcome).

```
> library(tidyverse) # Contains the ggplot2 package
> library(Hmisc) # Needed for correlation

> #let's do a plot first
> ggplot(dataset1, aes (x=investment, y=points)) + geom_point()
```



```
> # Let's add a regression line and a line of our outcome mean
> ggplot(dataset1, aes(x = investment, y = points)) + geom_point() +
geom_hline(yintercept = mean(dataset1$points), colour = "blue") +
geom_smooth(method = "lm", se = FALSE)

> # Let's calculate Pearson's r
> rcorr(dataset1$investment, dataset1$points)
```



Pearson's r = 0.9, p < .001

Building a simple linear model

```
> # Let's do regression with just the one predictor

> model0 <- lm(points ~ 1, data = dataset1)
> model1 <- lm(points ~ investment, data = dataset1)
```

We have built two models - *model0* is a model with just the intercept (so the mean of our outcome) predicting the outcome (*points*) while *model1* is a model with *investment* predicting the outcome (*points*).

```
> # You can compare the two models to each other

> anova(model0, model1)
```

```
> anova(model0, model1)
Analysis of Variance Table

Model 1: points ~ 1
Model 2: points ~ investment
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     19 120827
2     18  22046  1     98781 80.654 4.547e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The F-ratio comparing our two models is 80.654 indicating our model with our predictor (*investment*) is a better fit than our model with just the intercept (the mean).

```
> summary(model1)

Call:
lm(formula = points ~ investment, data = dataset1)

Residuals:
    Min      1Q  Median      3Q     Max
-55.936 -20.840  -2.978  28.212  60.615

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -50.92329   23.44967  -2.172   0.0435 *
investment    0.24166    0.02691   8.981 4.55e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 35 on 18 degrees of freedom
Multiple R-squared:  0.8175,   Adjusted R-squared:  0.8074
F-statistic: 80.65 on 1 and 18 DF,  p-value: 4.547e-08
```

Here we have our parameter estimates.
Here we have the t-test associated with our predictor (investment).
Here are the R-squared and Adjusted R-squared values (which reflects the number of predictors in our model).

We would conclude from this that the amount of money spent on a driver does indeed predict the number of points they score in a season of F1. Specifically, for every £24,1666 spent on them they will score one additional point.

Remember, regression is nothing more than prediction - a simple regression model allows us to predict the value of a variable future on the basis of knowing about that variable (and it’s relationship to another variable) now…

Multiple Regression in R

We are interested in whether house prices in 250 regions of the UK can be predicted by:

(a) Population size

(b) Crime rate (per 10,000 people)

(c) Average age of people in the region

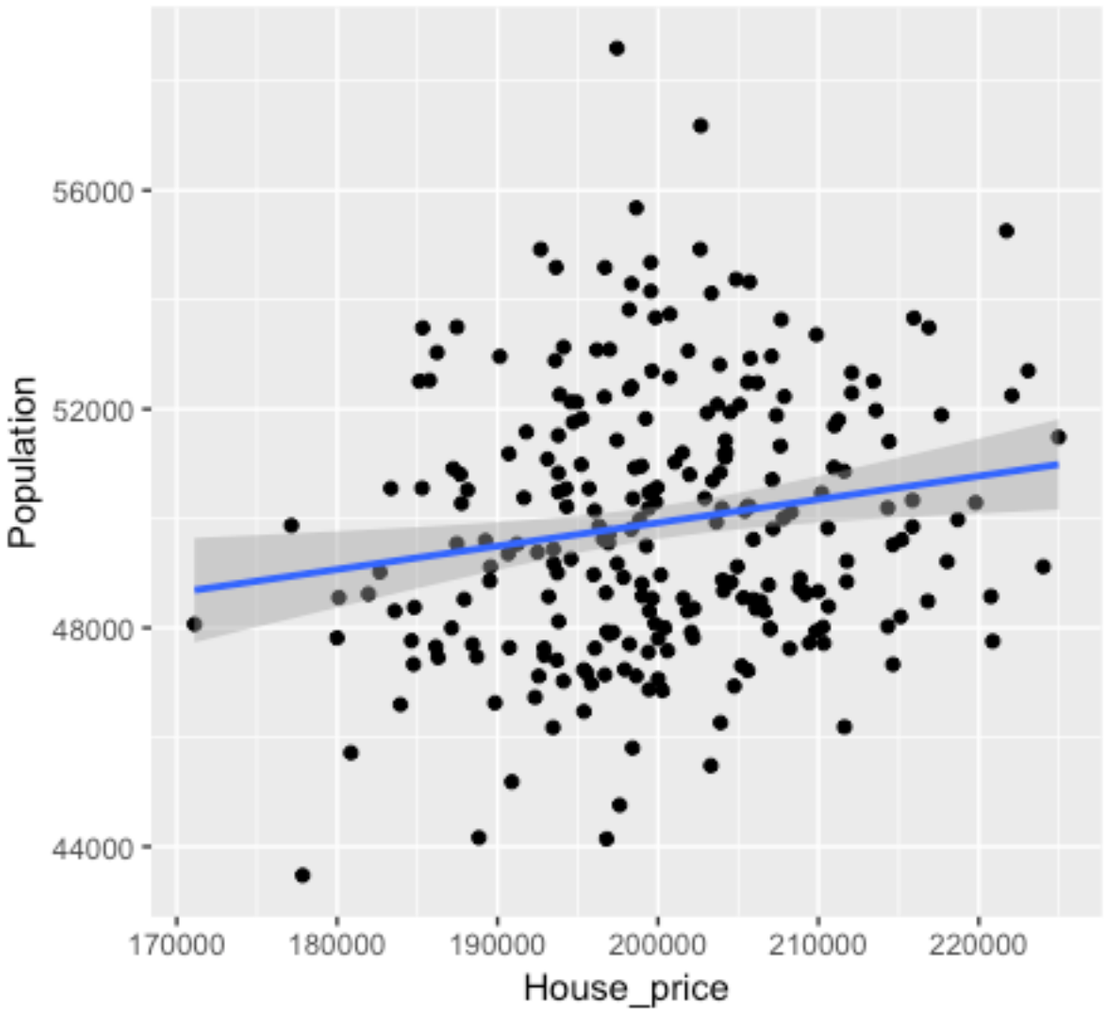
(d) Average household income in the region.

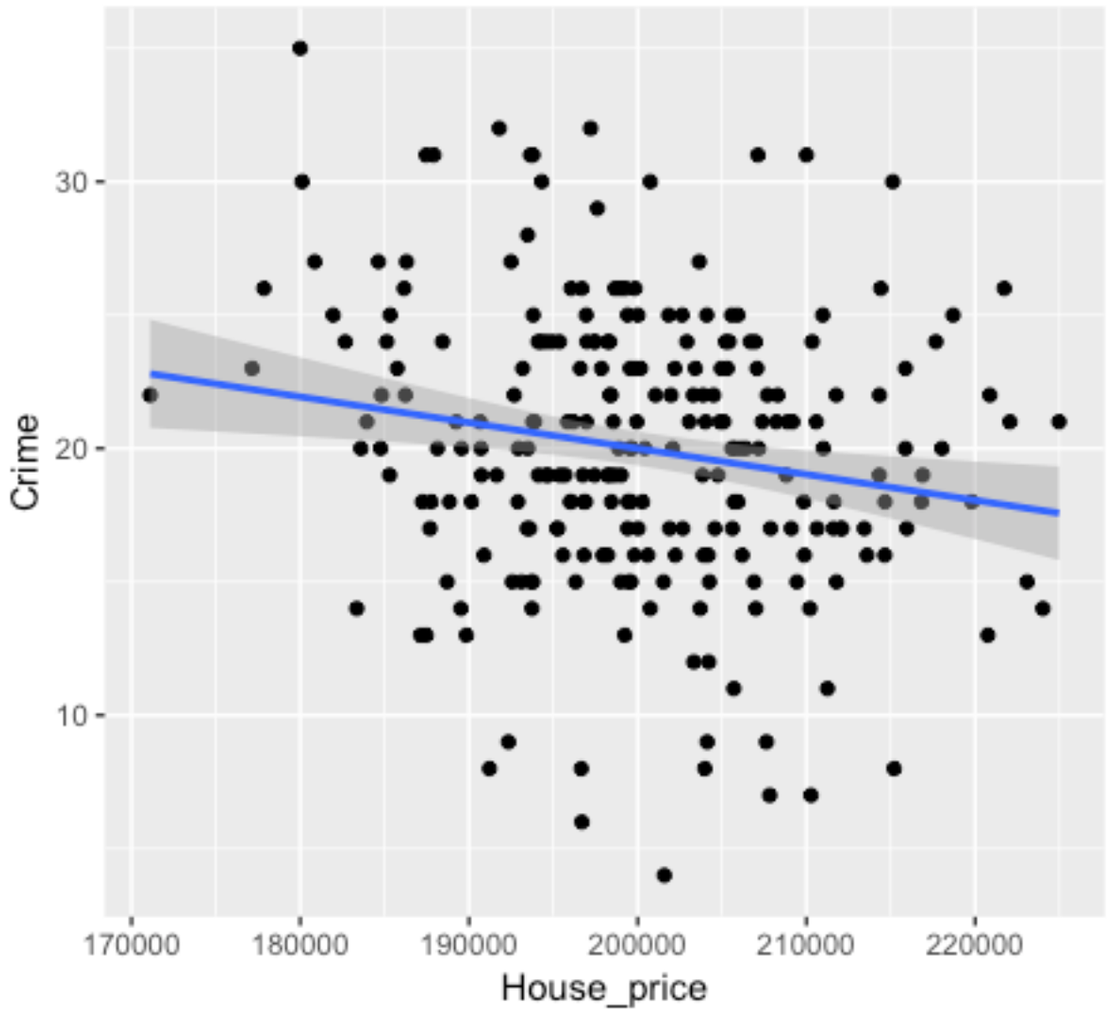
Including our predictor and a column identifying our Regions, our datasets consists of 6 variables.

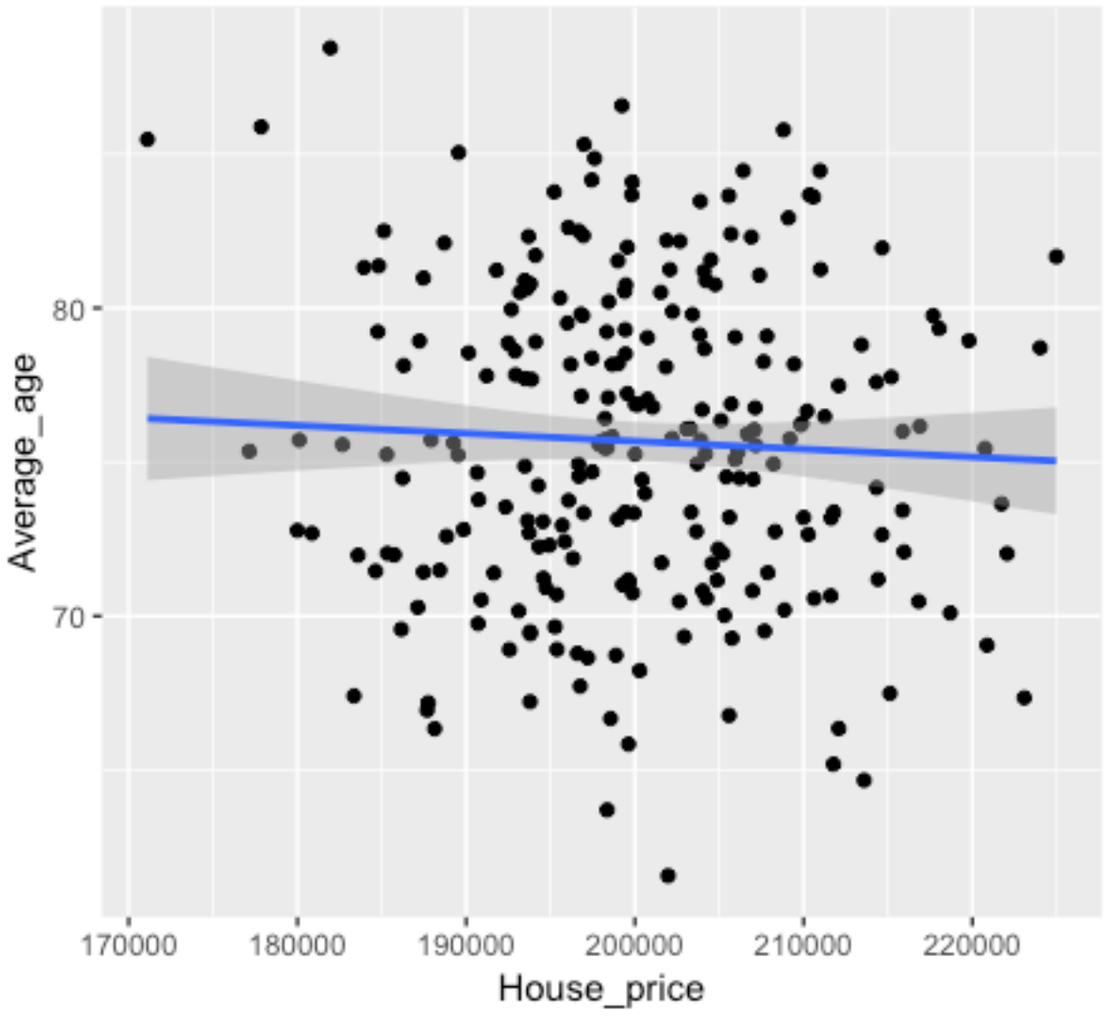
```
> str(data)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 250 obs. of  6 variables:
 $ Region          : num  1 2 3 4 5 6 7 8 9 10 ...
 $ House_price     : num  193735 201836 191643 215952 203295 ...
 $ Population      : num  49004 48307 50379 53664 45481 ...
 $ Crime           : num  14 25 19 17 22 32 21 21 20 25 ...
 $ Average_age     : num  72.7 78.1 71.4 72.1 76.1 ...
 $ Household_income: num  20843 19130 20411 16863 19964 …
```

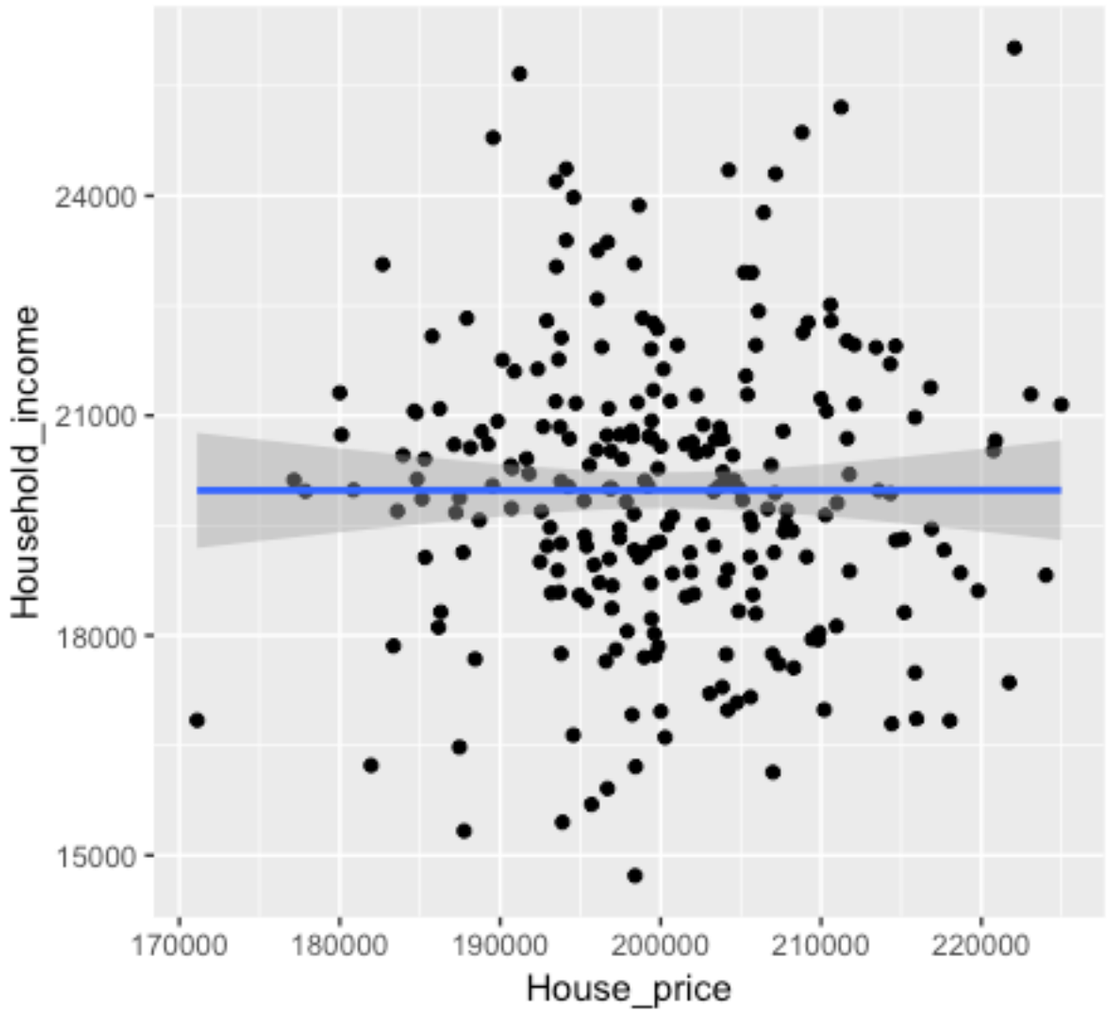
```
> head(data)
# A tibble: 6 x 6
  Region House_price Population Crime Average_age Household_income
   <dbl>       <dbl>      <dbl> <dbl>       <dbl>            <dbl>
1      1      193735      49004    14        72.7            20843.
2      2      201836      48307    25        78.1            19130.
3      3      191643      50379    19        71.4            20411.
4      4      215952      53664    17        72.1            16863.
5      5      203295      45481    22        76.1            19964.
6      6      191795      51582    32        81.2            20207.
```

Let’s first build some plots looking at the possible relationship between each predictor and our outcome variable.






Average_age
80
70
170000
180000
190000
200000
210000
220000
House_price


Household_income
24000
21000
18000
15000
170000
180000
190000
200000
210000
220000
House_price

First we will build a linear model with all 4 predictors, then a second model with just the intercept (i.e., the mean) - we then compare them - is the model with the 4 predictors a better fit to our data than the model with just the mean?

```
> model0 <- lm (House_price ~ 1, data = data)
> model1 <- lm (House_price ~ Population + Crime + Average_age + Household_income,
data = data)
> anova(model0, model1)
Analysis of Variance Table

Model 1: House_price ~ 1
Model 2: House_price ~ Population + Crime + Average_age + Household_income
  Res.Df        RSS Df  Sum of Sq      F  Pr(>F)
1    249 2.3069e+10
2    245 2.1622e+10  4 1446836735 4.0985 0.00311 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The F-ratio comparing our two models is 4.0985 indicating our model with all the predictors is a better fit than our model with just the intercept (the mean). We then need to get our parameter estimates using the function `summary()`

```
> summary(model1)

Call:
lm(formula = House_price ~ Population + Crime + Average_age
+
    Household_income, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-26460.7  -6011.9   -386.4   6331.8  24591.6

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.807e+05  1.680e+04  10.754  < 2e-16 ***
Population       6.577e-01  2.453e-01   2.682  0.00782 **
Crime           -3.358e+02  1.153e+02  -2.913  0.00391 **
Average_age     -8.218e+01  1.186e+02  -0.693  0.48915
Household_income -1.957e-02  3.033e-01  -0.065  0.94861
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

Residual standard error: 9394 on 245 degrees of freedom
Multiple R-squared:  0.06272, Adjusted R-squared:  0.04741
F-statistic: 4.098 on 4 and 245 DF,  p-value: 0.00311
```

Here we have our parameter estimates and the t-tests associated with our predictors.

Here are the R-squared and Adjusted R-squared values (which reflects number of predictors in our model).

```
# Notice that Average_age and Household_income do not seem to predict house prices
# Let's drop them in model2
model2 <- lm (House_price ~ Population + Crime, data = data)
anova(model2, model1)
```

```
> anova(model2, model1)
Analysis of Variance Table

Model 1: House_price ~ Population + Crime
Model 2: House_price ~ Population + Crime + Average_age + Household_income
  Res.Df        RSS Df Sum of Sq      F Pr(>F)
1    247 2.1666e+10
2    245 2.1622e+10  2  43401593 0.2459 0.7822
```

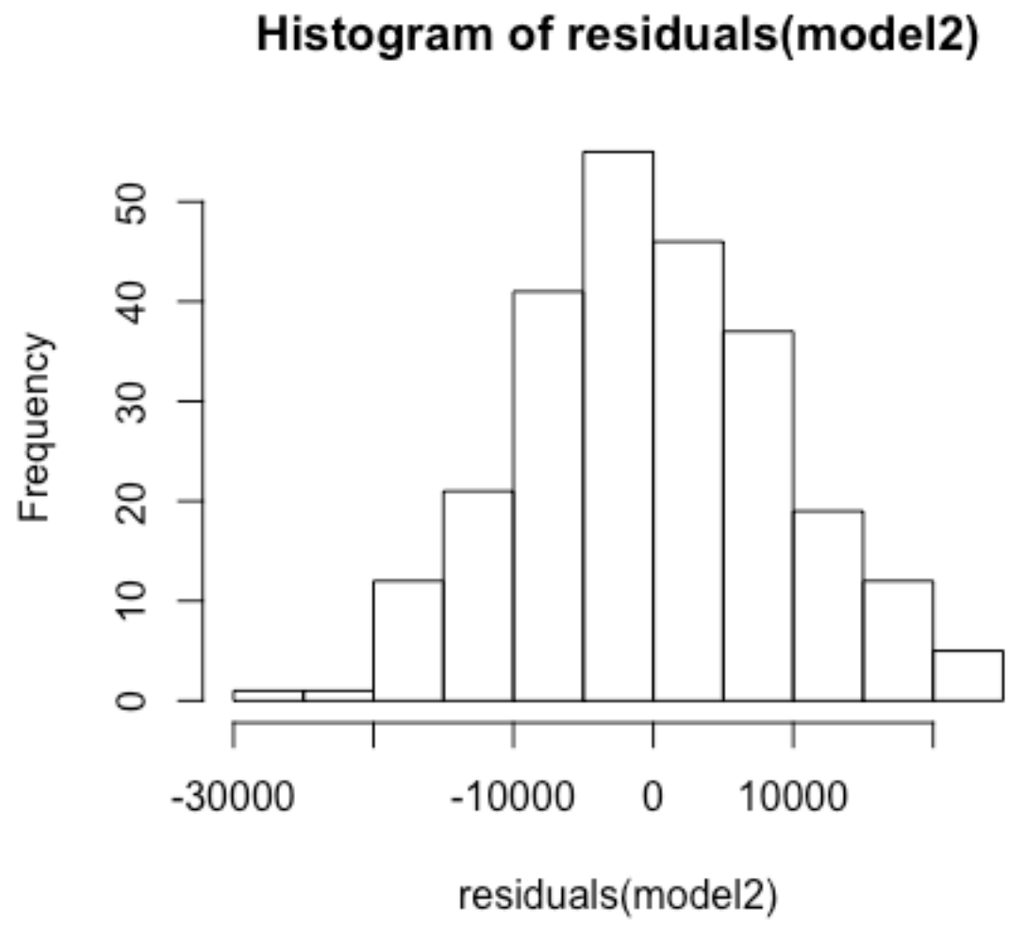
OK, so the models do not differ significantly by this test - we can use another measure of goodness-of-fit - AIC (Aikaike Information Criterion). AIC tells us how much information in our data is not captured by each model - lower values are better - can only be interpreted in a relative sense (i.e., comparing one model to another)…

```
> AIC(model1)
[1] 5290.354
> AIC(model2)
[1] 5286.855
```

We defined `model2` as having just two predictors - as `model2` has the lower AIC value (so more information in our data is explained by `model2` than by `model1`), we would be justified in selecting that as our ‘best’ model. AIC penalises models with increasing number of parameters (but not as much as BIC) so gives us a good trade-off of fitting our data and model complexity.

In regression our residuals need to be normally distributed - the easiest way to check this is to plot them:

```
> hist(residuals(model2))
```

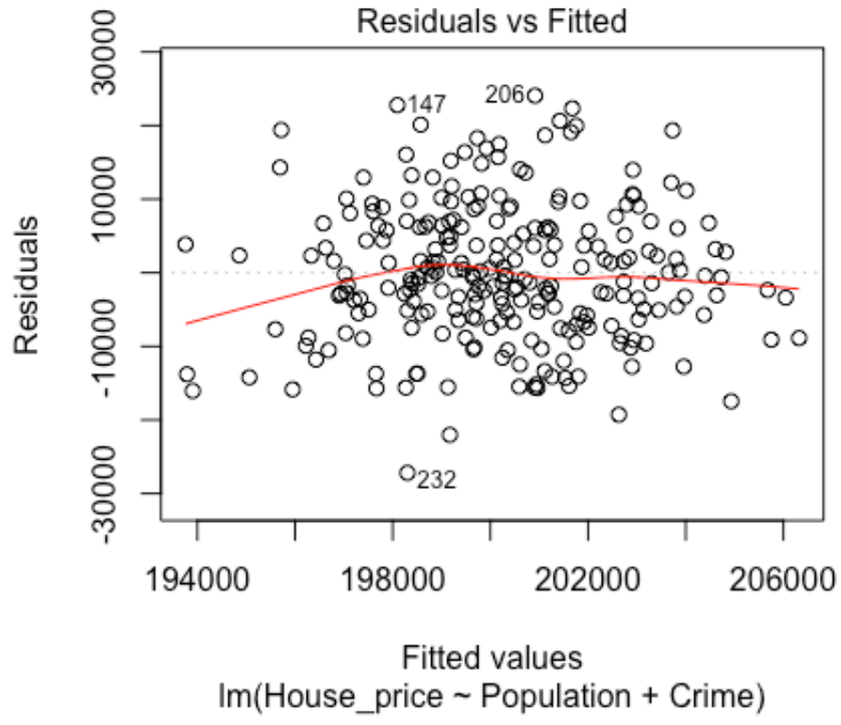


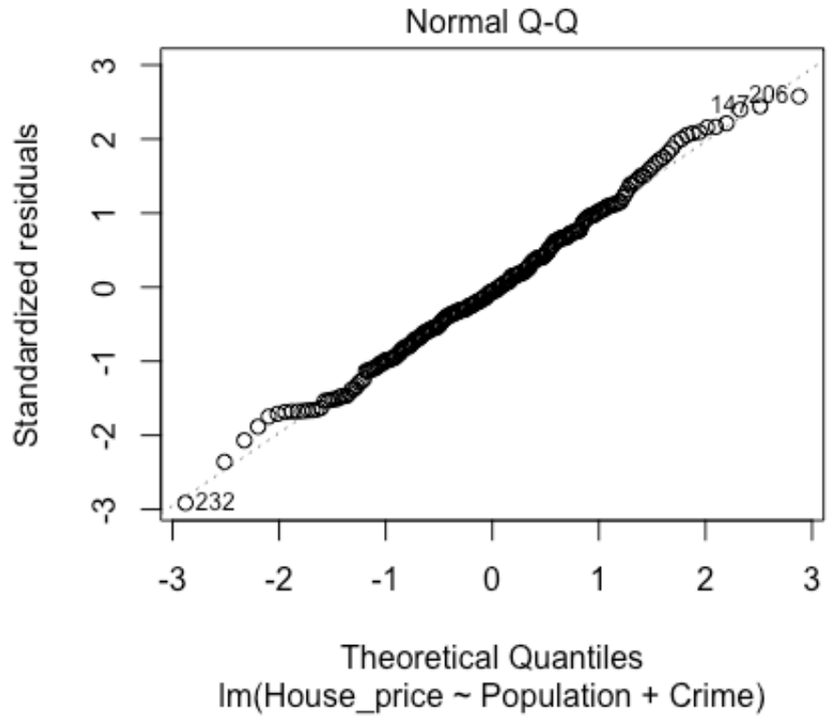
Now let's look at a number of diagnostic plots…

We can use the following command to get some visual representations of model fit:

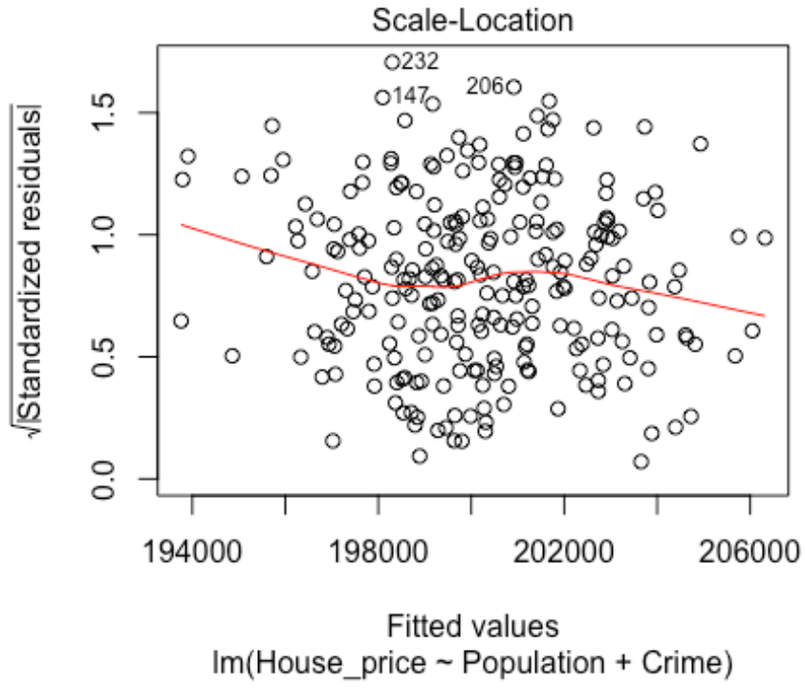
```
> plot(model2)
```

We should have a similar distribution of points (via LOESS Curve Fitting) either side of zero - if we don't it would suggest non-random errors (see Durbin Watson test later). In the Q-Q plot we should see a diagonal line if our residuals are normally distributed.

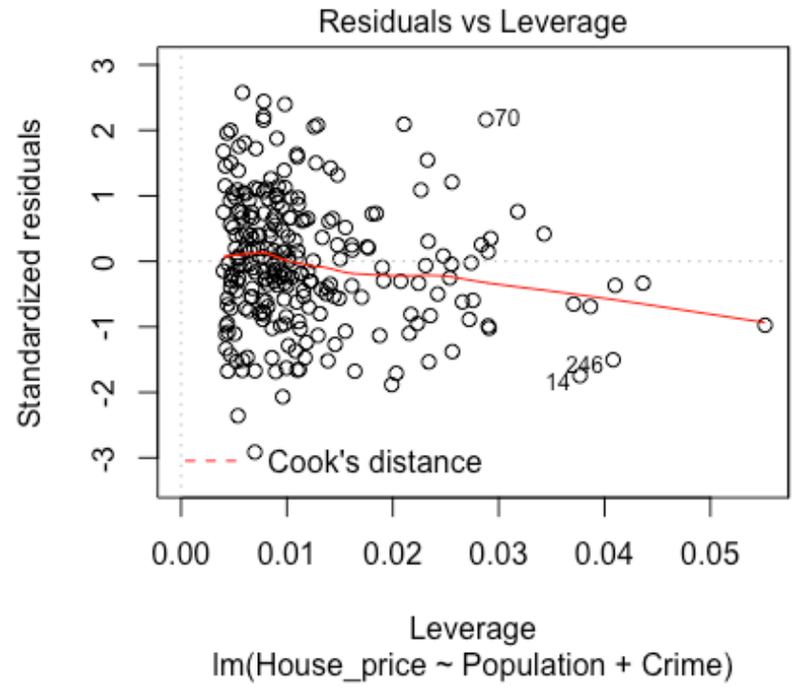




The Scale-Location plot shows if residuals are spread equally along the ranges of predictors. We used this to check the assumption of equal variance (homoscedasticity). We really want to see a horizontal line with equally, randomly spread points.



The Residuals vs. Leverage plot tells us about influential outliers (i.e., outliers that are affecting our model) - when cases are outside of Cook's distance (beyond the dashed line) it means they are having an influential affect on the regression model - we'd might want to exclude these points and rebuild our model.



Durbin Watson Test

- This tests for the non-independence of errors - our errors need to be independent (one of the assumptions of regression). This test needs the `car` package to be loaded.

```
> durbinWatsonTest(model2)
 lag Autocorrelation D-W Statistic p-value
   1     -0.03048832      2.055433    0.66
 Alternative hypothesis: rho != 0
```

A D-W value of 2 means that there is no autocorrelation in the sample - our calculated value is pretty close to that - p = .66 so we conclude our errors are independent of each other.

Stepwise Regression Based on AIC Improvement

Rather than building our regression model step by step manually, we can use the `step` function in R - it takes a starting model, and then uses forwards or backwards procedures (or a combination of both) to produce the best model.

We need to install the `MASS` library.

Let's apply the procedure to `model0` and `model1` as our limits - we can specify the stepwise procedure with the parameter "direction":

```
> library (MASS)

> steplimitsboth <- step(model0, scope = list (upper = model1), direction =
"both")
```

- Let’s focus on the combined method that adds predictors which improve model fit, and removes ones that don’t - based on minimising AIC:

```
> summary(steplimitsboth)

Call:
lm(formula = House_price ~ Crime + Population, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-27192.2  -6161.4   -555.2   6203.4  24061.0

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.736e+05  1.243e+04  13.973  < 2e-16 ***
Crime       -3.343e+02  1.147e+02  -2.915  0.00388 **
Population   6.662e-01  2.442e-01   2.729  0.00682 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9366 on 247 degrees of freedom
Multiple R-squared:  0.06084,  Adjusted R-squared:  0.05323
F-statistic:     8 on 2 and 247 DF,  p-value: 0.0004301

> AIC(steplimitsboth)
[1] 5286.855
```

We can see the procedure has settled on the model with Crime and Population. AIC value is 5286.855. In this case the stepwise model is the same as what we arrived at manually.

We can also estimate the confidence intervals for each of our parameters using the `confint()` function - this tells us that 95% of the time the true parameter value will lie somewhere between these points

```
> confint(steplimitsboth, level = 0.95)
                    2.5 %        97.5 %
(Intercept)  1.491596e+05 198110.856517
Crime       -5.602084e+02   -108.461481
Population   1.853052e-01      1.147126
```

Collinearity?

- We can apply the `vif()` function to our model - it will work out the VIF values for each of our variables - `vif()` is in the `car` package so don't forget to load that…

```
> vif(steplimitsboth)
    Crime Population
 1.000012   1.000012
```

- As a rule of thumb VIF greater than 10 suggests a multicollinearity issue (although greater than 5 is sometimes used as it is more conservative).
- For our case, we don't have a collinearity problem as the VIF values are low.

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.736e+05  1.243e+04  13.973  < 2e-16 ***
Crime       -3.343e+02  1.147e+02  -2.915  0.00388 **
Population   6.662e-01  2.442e-01   2.729  0.00682 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Using these regression coefficents, we could write our regression equation as something like:

```
House price = 173,600 - 334.2(Crime) + 0.6662(Population) + residual
```

- So, crime has a *negative* influence on house prices (more crime = lower prices) while population size has a *positive* influence on house prices (more people = higher house prices).

ANOVA for factorial designs

- A particularly good package for factorial ANOVA is by Henrik Singmann and called `afex`.
- Built to work like ANOVA in SPSS - uses Type III Sums of Squares with e*ffect* coding of contrasts. This overrides the default contrast coding in *R* which is for *dummy* coding.

AN(C)OVA

- Let's look at how double espresso vs. single espresso vs. water drinking (our IV) might influence motor performance (our DV).

- Imagine we sampled from a group of participants - and we think other factors that we are not manipulating might also influence the DV – e.g., practice with computer games.

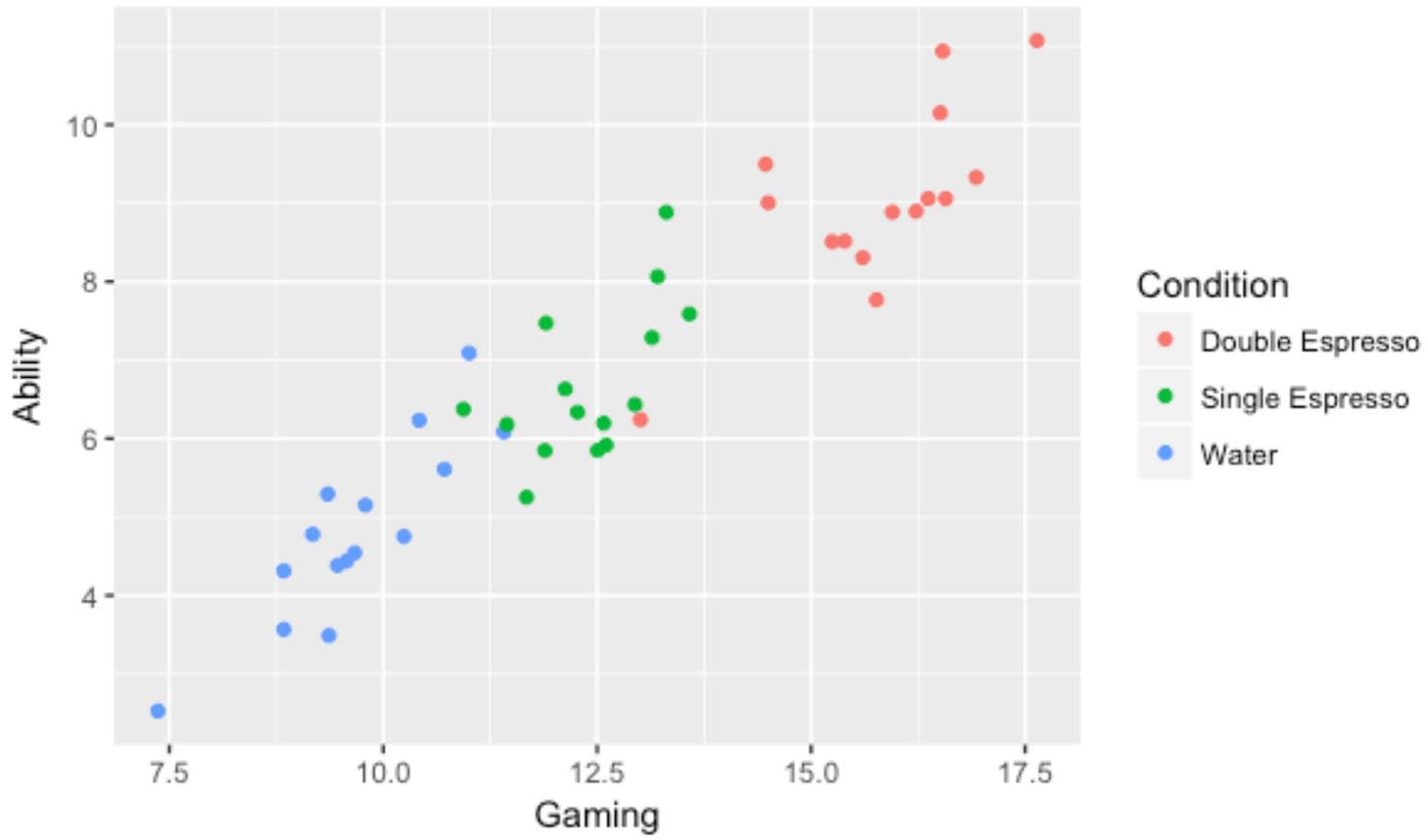
- What we want is to be able to see the effect on our DV of our IV after we have removed the effects of other things (computer gaming frequency in this case).

- Now, imagine we have a measure of computer games frequency - perhaps hours per week people play computer games…

- So, in addition to manipulating the type of beverage we're giving people (i.e., double espresso vs. single espresso vs. water) we also measure how often they play computer games…

- Let's do a plot first with our DV (Ability) on the y-axis, and our covariate (Gaming Frequency) on the x-axis…

```
> ggplot(cond, aes(x = Gaming, y = Ability, colour = Condition)) + geom_point()
```



- So we can see there's a relationship between our DV (Ability) and our covariate (Gaming Frequency)…

- We can also see our Gaming Ability groups appear to be clustering in our data by Condition…

Running a 1-way between participants ANOVA (and ignoring the covariate)…

```
> model1 <- aov_4(Ability ~ Condition + (1 | Participant), data = cond)
Contrasts set to contr.sum for the following variables: Condition
> anova(model1)
Anova Table (Type 3 tests)

Response: Ability
          num Df den Df    MSE      F     ges    Pr(>F)
Condition      2     42 1.2422 53.432 0.71786 2.882e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor Condition is significant with an F = 53.432. We would erroneously conclude that our manipulation has had an effect…

The effect size is measured by *ges* which stands for generalised effect size (η_G^2).

But now let's control for the effect of our co-variate (which we first need to scale and centre)…

```
> cond$Gaming <- scale(cond$Gaming)
> model_ancova <- aov_4(Ability ~ Gaming + Condition + (1 | Participant),
data = cond, factorize = FALSE)
Contrasts set to contr.sum for the following variables: Condition
> anova(model_ancova)
Anova Table (Type 3 tests)

Response: Ability
          num Df den Df     MSE       F     ges    Pr(>F)
Gaming         1     41 0.55171 53.5636 0.56643 5.87e-09 ***
Condition      2     41 0.55171  0.8771 0.04103    0.4236
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor Condition is now <u>not</u> significant with an F < 1. However, our covariate *Gaming Frequency* is significant. Adding it means a lot of the variance we previously attributed to our experimental factor is actually explained by our covariate.

Rather than calculating over the raw means which are:

Water Group = 4.82

Double Espresso Group = 9.02

Single Espresso Group = 6.69

```
> describeBy(cond$Ability, group = cond$Condition)

 Descriptive statistics by group
group: Double Espresso
   vars  n mean   sd median trimmed  mad  min   max range  skew kurtosis   se
X1    1 15 9.02 1.19   9.01    9.07 0.73 6.24 11.07  4.83 -0.26     0.16 0.31
------------------------------------------------------------------------------
group: Single Espresso
   vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 15 6.69 0.98   6.37    6.63 0.78 5.25 8.88  3.63 0.69    -0.53 0.25
------------------------------------------------------------------------------
group: Water
   vars  n mean   sd median trimmed mad  min  max range skew kurtosis  se
X1    1 15 4.82 1.16   4.75    4.82 0.8 2.53 7.09  4.56 0.03    -0.57 0.3
```

The calculation is performed over the *adjusted* means (which take into consideration the influence of the covariate). We use the `emmeans()` function in the emmeans package to work out the adjusted means and run pairwise comparisons:

Water Group = 7.33

Double Espresso Group = 6.32

Single Espresso Group = 6.87

```
> emmeans(model_ancova, pairwise ~ Condition, adjust = “none")
$emmeans
 Condition         emmean        SE df lower.CL upper.CL
 Double Espresso 6.319464 0.4152816 41 5.480786 7.158142
 Single Espresso 6.871614 0.1934303 41 6.480974 7.262255
 Water           7.327960 0.3931110 41 6.534056 8.121864

Confidence level used: 0.95
```

If our experimental factor in the ANCOVA *had* been significant, we could have looked at the pairwise comparisons reported by *emmeans* to determine what condition was different from what other condition…

```
$contrasts
 contrast                                estimate        SE df t.ratio p.value
 Double Espresso - Single Espresso -0.5521505 0.4779448 41  -1.155  0.2547
 Double Espresso - Water           -1.0084959 0.7614421 41  -1.324  0.1927
 Single Espresso - Water           -0.4563454 0.4179276 41  -1.092  0.2812
```

But once we take account of the influence of our covariate we found no effect of Condition…

Note that there is an `aov()` function in base R that uses Type I Sums of Squares, but we use `aov_4()` in the afex package and it uses Type III by default (although this can be changed).

Type I vs. II vs. III Sums of Squares

- Type I Sum of Squares is calculated sequentially - e.g., first for Factor A main effect, then for Factor B main effect, then for the interaction. The order in which they are calculated matters and can be misleading for unbalanced design or cases where predictors are correlated. Total SS is the sum of the individual effect SS.

- Type II Sum of Squares assumes no interaction(s) when testing main effects or higher order interaction(s) when testing lower order interaction(s).

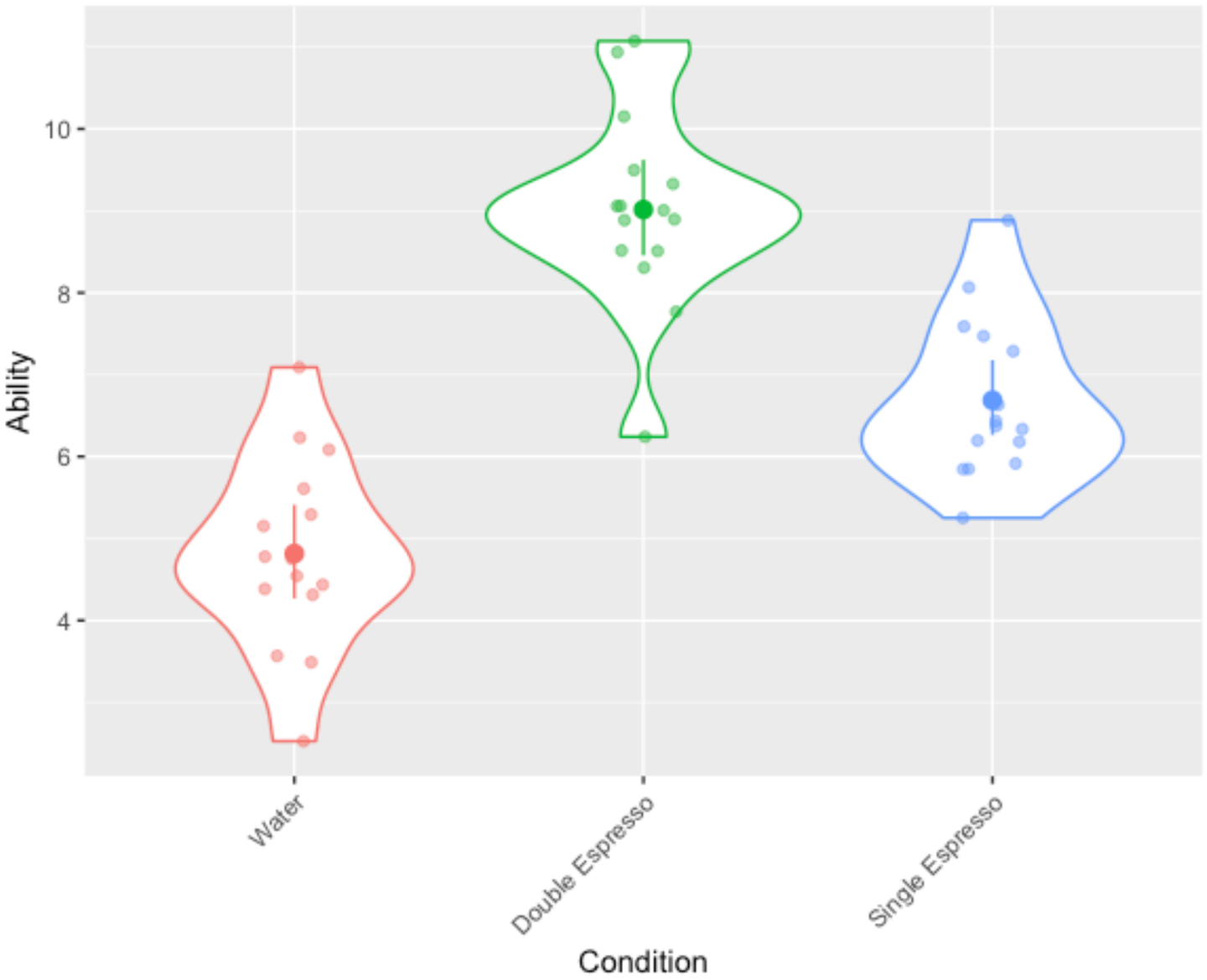
- Type III Sum of Squares tests for effects adjusted for the presence of the other effects (so does not depend on the order of terms).

- Much debate about which one is 'correct' - each has their own purpose - for factorial designs where you're interested in testing an interaction (or when your predictors correlate), Type III is most commonly used.

AN(C)OVA as a special case of regression…

- Let’s return to the example we looked at for ANCOVA - and let’s forget the co-variate for a moment…

- We looked at how double espresso vs. single espresso vs. water drinking (our IV) might influence people’s gaming ability (our DV).



Water mean = 4.82
Double Espresso mean = 9.02
Single Espresso mean = 6.69

- First we need to use dummy coding of the levels of our experimental factor - which is the default coding in R for factors…

```
> # Set up the Water level as the reference level and check the contrasts
> cond$Condition <- relevel(cond$Condition, ref = 3)
> contrasts(cond$Condition)
                 Double Espresso Single Espresso
Water                          0                0
Double Espresso                1                0
Single Espresso                0                1
```

Ability = Intercept + $\beta 1$(Double Espresso) + $\beta 2$(Single Espresso) + ε

The Intercept is our reference category (Water) with coding (0, 0), while the dummy coding for Double Espresso is (1, 0) and for Single Espresso (0, 1)

Ability = Intercept + *β1*(Double Espresso) + *β2*(Single Espresso) + *ε*

We want to calculate *β1* and *β2*

```
> lm1 <- lm(Ability ~ Condition, data = cond)
> lm1

Call:
lm(formula = Ability ~ Condition, data = cond)

Coefficients:
             (Intercept)  ConditionDouble Espresso  ConditionSingle Espresso
                   4.817                     4.199                     1.871
```

The intercept is 4.817 (which is the mean of our Water group), *β1 i*s 4.2, and *β2* is 1.87

To work out the mean Ability of our Double Espresso Group:

Ability = Intercept + $\beta 1$(Double Espresso) + $\beta 2$(Single Espresso) + ε

Ability = 4.82 + 4.2(1) + 1.87(0) + ε

Ability = 4.82 + 4.2 + ε

Ability = 9.02 + ε

To work out the mean Ability of our Single Espresso Group:

Ability = Intercept + $\beta 1$(Double Espresso) + $\beta 2$(Single Espresso) + ε

Ability = 4.82 + 4.2(0) + 1.87(1) + ε

Ability = 4.82 + 1.87 + ε

Ability = 6.69 + ε

Which are the exact same means generated by the ANOVA when we ignored the covariate…

Water mean = 4.82
Double Espresso mean = 9.02
Single Espresso mean = 6.69

Why Linear Mixed Models?

(Generalized) linear mixed models have taken the biological and behavioural sciences by storm.

(G)LMMs can be more flexible and nuanced than linear models, allowing for multiple simultaneous random effects (e.g., subjects and items), subject and item covariates, nesting, unbalanced designs, normal and non-normal data distributions, cope with missing data, allow you to model both continuous and categorical IVs and DVs, operate over trial-level data, and allow you to determine the best statistical models to fit to your data that make the most theoretical sense…

Linear Mixed Models

What happens when we have many observations per person that we want to model?

Imagine we are interested in how a person's reaction time varies whether they're responding to Large or Small target items.

We observe the same 10 people each responding to 5 Large and 5 Small target items.

We have 10 observations per person. These observations are not independent of each other as (which is an assumption of a linear model).

- We can get around the lack of independence by treating participants as a random effect such that each participant has their own *individual* reaction time baseline.
- This gives us a separate random intercept value for each participant - in other words, our model can account for individual variation.
- This is a *mixed effects model*:

rt ~ condition + (1 | subject) + error

This is our random effect and assumes a different intercept for each participant.

- Imagine also that we have different Target Items e.g., 10 different items that were presented in either in Large or Small format)
- Each Target Item might have been a little different. One particular Target might just be easier to respond to quickly - in other words, the Target Items will also have different baselines.

- We can capture the random effect of Item in the same way we did for participants:

```
rt ~ condition + (1 | subject) + (1 | item) + error
```

Filter

	subject	condition	item	rt
1	1	small	1	1127.4384
2	1	large	1	968.2830
3	1	small	2	1133.4436
4	1	large	2	1051.7208
5	1	small	3	952.1512
6	1	large	3	1131.0116
7	1	small	4	1242.9841
8	1	large	4	999.4708
9	1	small	5	1085.0351
10	1	large	5	865.3554

Showing 1 to 10 of 100 entries

10 participants, and 10 items. Each item appeared in two versions - Small vs. Large.

Fixed vs. Random Effects

Fixed effect Data has been gathered from all the levels of the factor that are of interest. (Typically your experimental factors and maybe factors like gender).

Random effect The factor has many possible levels, interest is in all possible levels, but only a random sample of levels is included in the data. (Typically participants and items). Typically need > 5 levels in order to estimate effects.

For mixed effects linear modelling in R, we need to install the package *lme4*. This is the mixed effects model equivalent of *lm* which we used previously. We also want the *lmerTest* package and the *emmeans* package.

```
> install.packages("lme4")

> install.packages("lmerTest")

> install.packages("emmeans")
```

Gives us p-values for our model estimates.

Allows us to do pairwise comparisons.

Remember then to load them:

```
> library(lme4)

> library(lmerTest)

> library(emmeans)
```

```
> mixed_model <- lmer(rt ~ condition + (1 | subject) + (1 | item), data = fulldata)
> summary(mixed_model)
Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest']
Formula: rt ~ condition + (1 | subject) + (1 | item)
   Data: fulldata

REML criterion at convergence: 1276.5

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.59882 -0.62360  0.07231  0.57203  2.91523

Random effects:
 Groups   Name        Variance Std.Dev.
 subject  (Intercept)  7952.1   89.17
 item     (Intercept)   436.3   20.89
 Residual             20938.7  144.70
Number of obs: 100, groups:  subject, 10; item, 5

Fixed effects:
               Estimate Std. Error      df t value Pr(>|t|)
(Intercept)     1067.99      36.07   12.62   29.61 4.82e-13 ***
conditionsmall   187.83      28.94   85.00    6.49 5.46e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
            (Intr)
conditnsmll -0.401
```

More variability in subjects than in items.

The intercept corresponds to the RT to the Large Condition - going from Large to Small contexts increases RT by around 188 ms.

- To determine whether our mixed effects model is significant, we need to know whether it differs from what we'd expect if Condition didn't influence Reaction Times.

```
mixed_model_null <- lmer(rt ~ (1 | subject) + (1 | item), data = fulldata)
```

- This model which we call `mixed_model_null` removes Condition as a predictor - in other words, it simply contains our random effects.

We can now compare the two models with each other using the anova function:

```
> anova(mixed_model, mixed_model_null)
```

This performs a likelihood ratio test on our 2 models and tells us whether they are significantly different from each other - this test only works with **nested** models - i.e., when one model is a subset of the other.

```
> anova(mixed_model, mixed_model_null)
refitting model(s) with ML (instead of REML)
Data: fulldata
Models:
mixed_model_null: rt ~ (1 | subject) + (1 | item)
mixed_model: rt ~ condition + (1 | subject) + (1 | item)
                  Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
mixed_model_null   4 1336.4 1346.8 -664.18   1328.4
mixed_model        5 1303.8 1316.8 -646.91   1293.8 34.534      1   4.19e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

This is the important bit as the chi-squared test tells us whether our models differ from each other. It does. Note the AIC, BIC, and deviance values are all lower for the model with our fixed effect.

Note, deviance equals the residual sum of squares in linear models.

- So far we have accounted for the possibility that our participants and items might have different reaction time baselines - that some people are faster at responding that others (which is why we introduced the separate random intercepts).

- But what if the *magnitude* of the effect of Condition is different for different participants, and also what if the effect of Condition is different for different items?

- All this means is that the slopes of our lines might vary as a function of participant (so the difference between the two levels of our Condition factor might be bigger for one person than for another) and as a function of item (so the difference between the two levels of our Condition factor might also be bigger for one item than for another).

```
> coef(mixed_model)
$subject
   (Intercept) conditionsmall
1     983.9157         187.825
10   1149.1395         187.825
2    1069.1966         187.825
3    1155.6409         187.825
4     975.1408         187.825
5     938.9609         187.825
6    1073.0511         187.825
7    1069.8418         187.825
8    1161.5529         187.825
9    1103.5083         187.825

$item
  (Intercept) conditionsmall
1    1078.522         187.825
2    1081.053         187.825
3    1059.536         187.825
4    1055.631         187.825
5    1065.233         187.825
```

The different intercepts for each item and for each participant take into account individual baseline differences. However, it doesn't take into account the fact our effect might be bigger for some participants than for others (and for some items than for others). In other words, the slopes are all currently the same (187.825).

```
mixed_model <- lmer(rt ~ condition + (1 + condition | subject)
+ (1 + condition | item), data = fulldata)
```

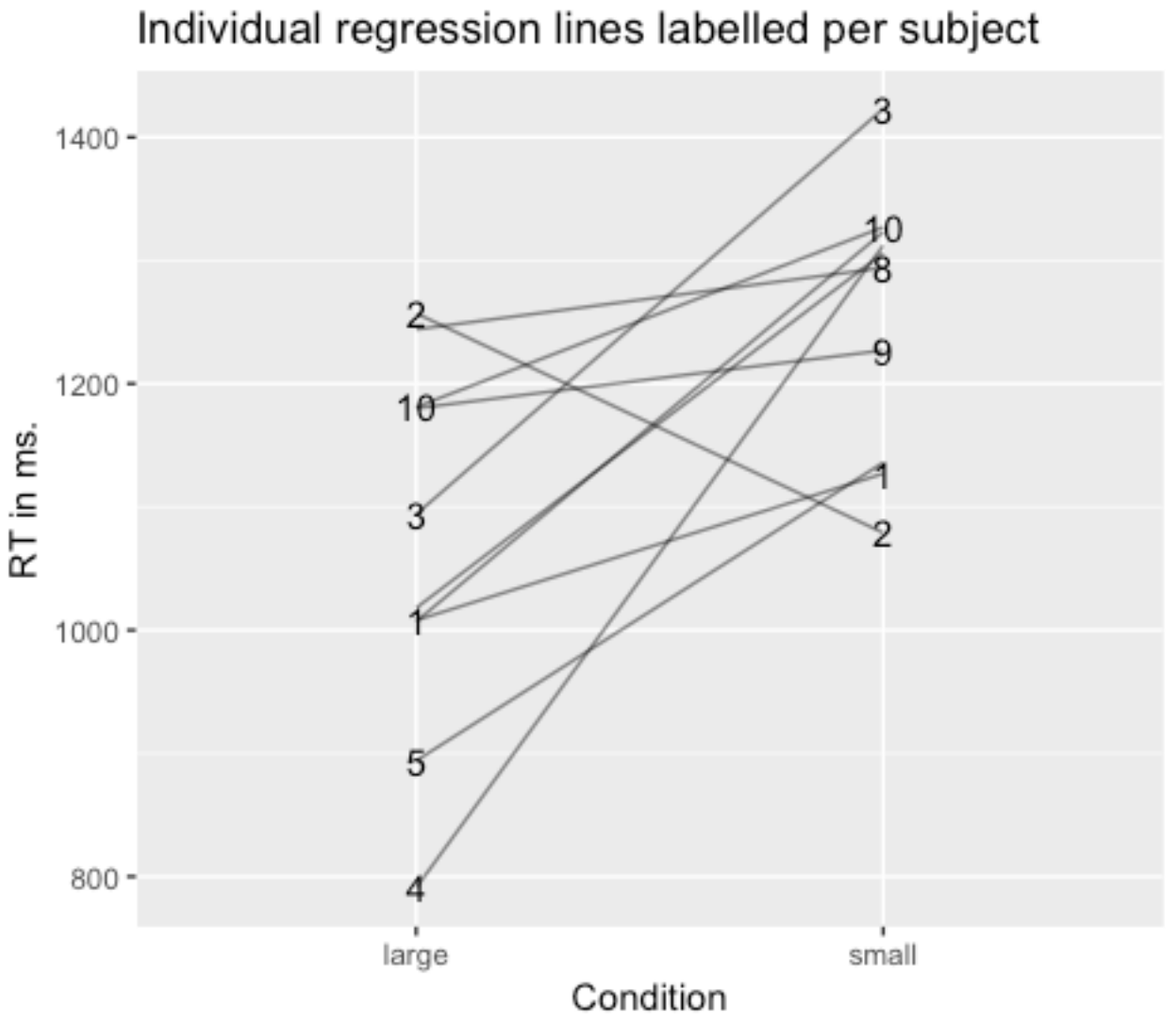
These modified terms tell the model to expect different intercepts for Condition (which we had before) as well as differing slopes as a function of the factor Condition. These are our random effects.

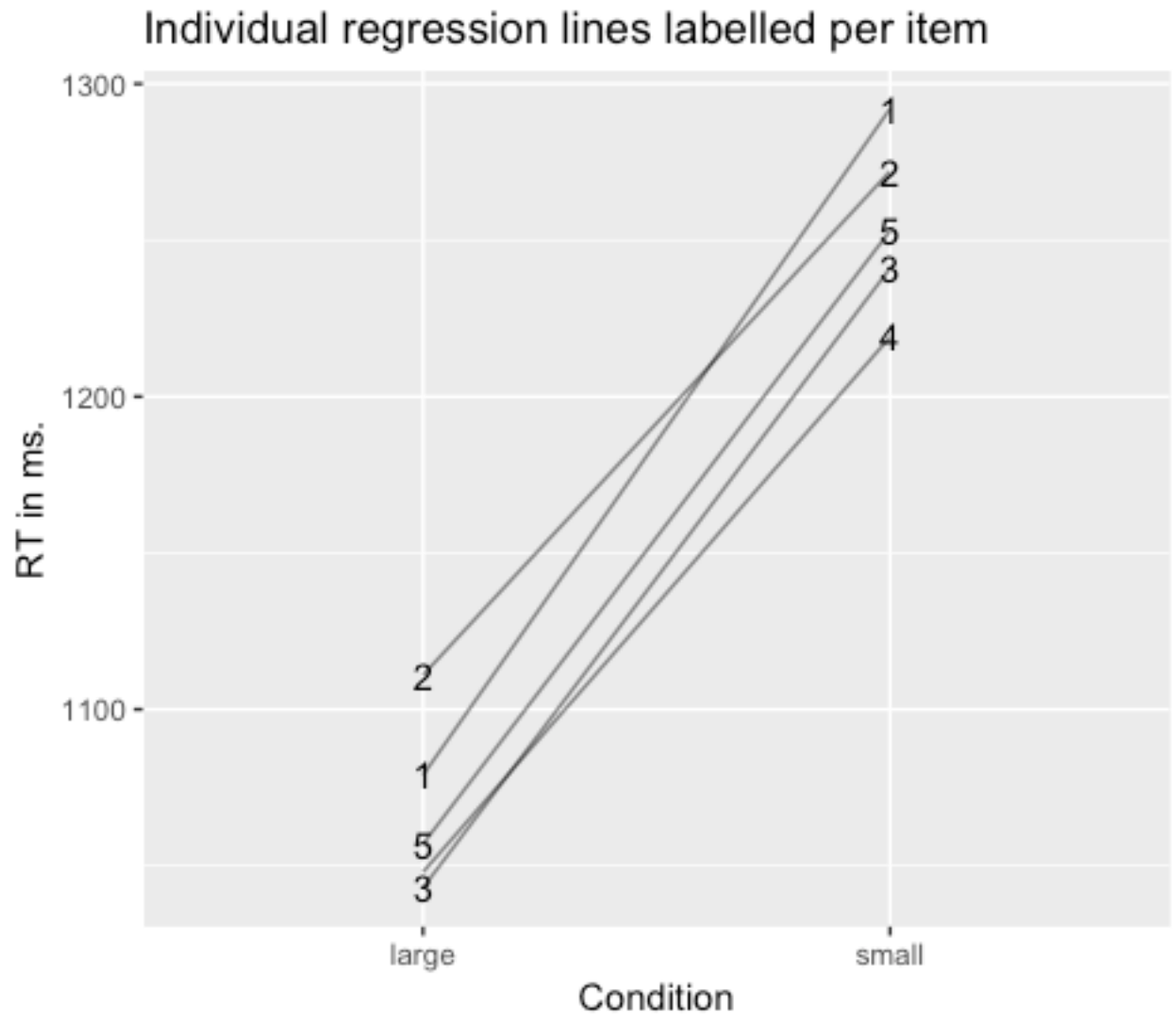
```
> coef(mixed_model)
$subject
   (Intercept) conditionsmall
1    1008.7029      118.34480
10   1181.1532      146.17094
2    1257.5895     -178.00567
3    1094.7593      328.35749
4     791.8356      520.54188
5     894.3442      241.83682
6    1007.5660      316.37449
7    1018.4388      288.19304
8    1244.7464       49.85042
9    1180.8127       46.58613

$item
  (Intercept) conditionsmall
1    1079.387       213.2465
2    1111.258       160.7852
3    1043.521       198.1986
4    1048.069       171.1310
5    1057.739       195.7639
```

The slopes between the two levels of our Condition differ for each participant…

…and for each item.

Plotting the slopes of our Condition factor





We see quite a lot of variability in our participants - incl. participant 2 who is going the other way (RT for Small targets is shorter than RT for Large targets)

Partial Pooling in LMMs

- LMMs use *partial pooling* to estimate the parameters of the model coefficients.

- Partial pooling takes account of the individual slopes and intercepts for each level of the random effect structure, but also the slope and intercept of the overall model (which ignores how things vary from one participant to the next).

The Dataset

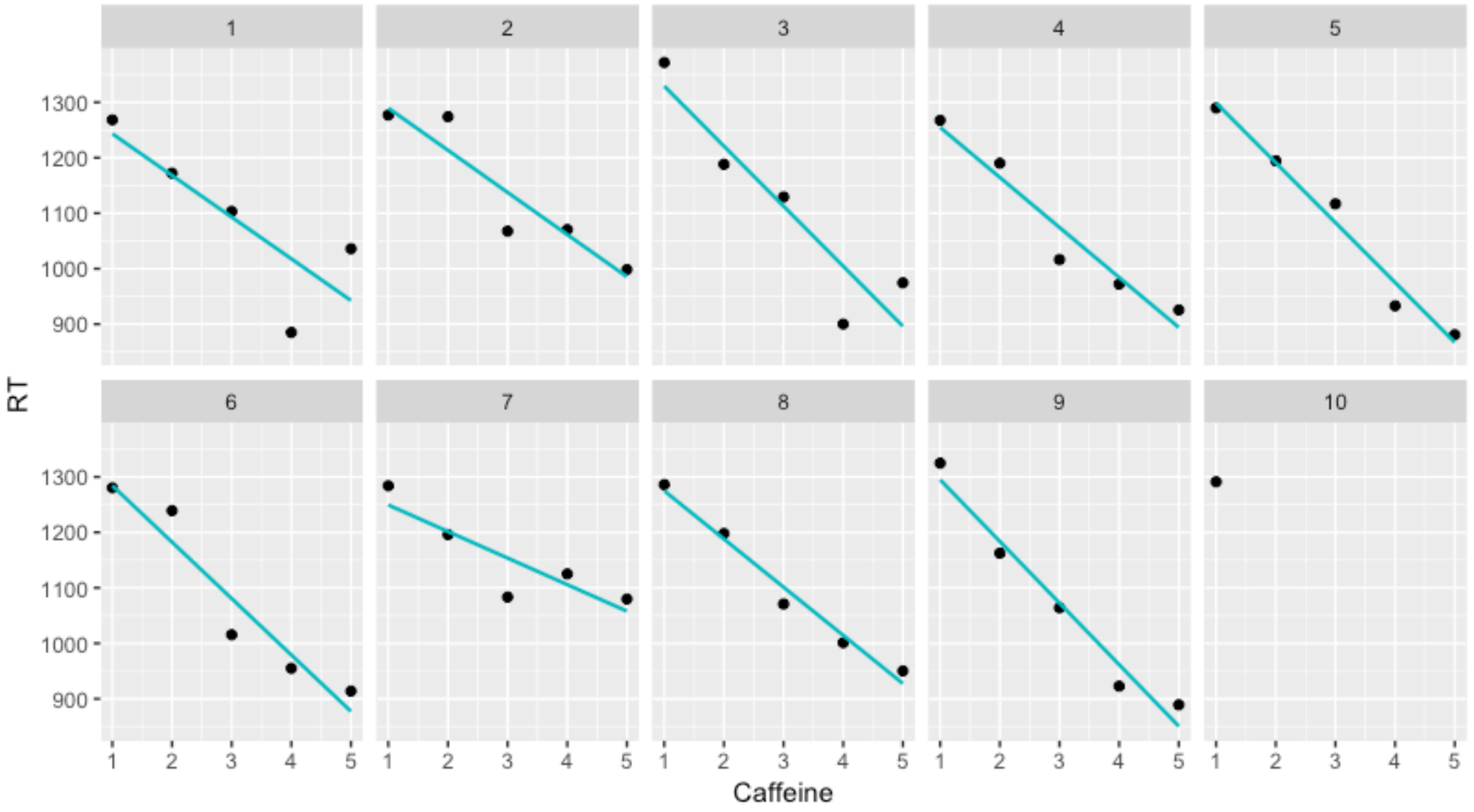
- We are going to use data from 10 participants with measures of reaction time and caffeine consumption.

```
> str(data_all)
'data.frame':  50 obs. of  3 variables:
 $ subject : int  1 1 1 1 1 2 2 2 2 2 ...
 $ caffeine: int  1 2 3 4 5 1 2 3 4 5 ...
 $ rt      : num  1268 1172 1103 885 1036 ...
```

- Contains 50 observations from 10 participants with measures of reaction time and caffeine consumed (measured in cups of coffee).

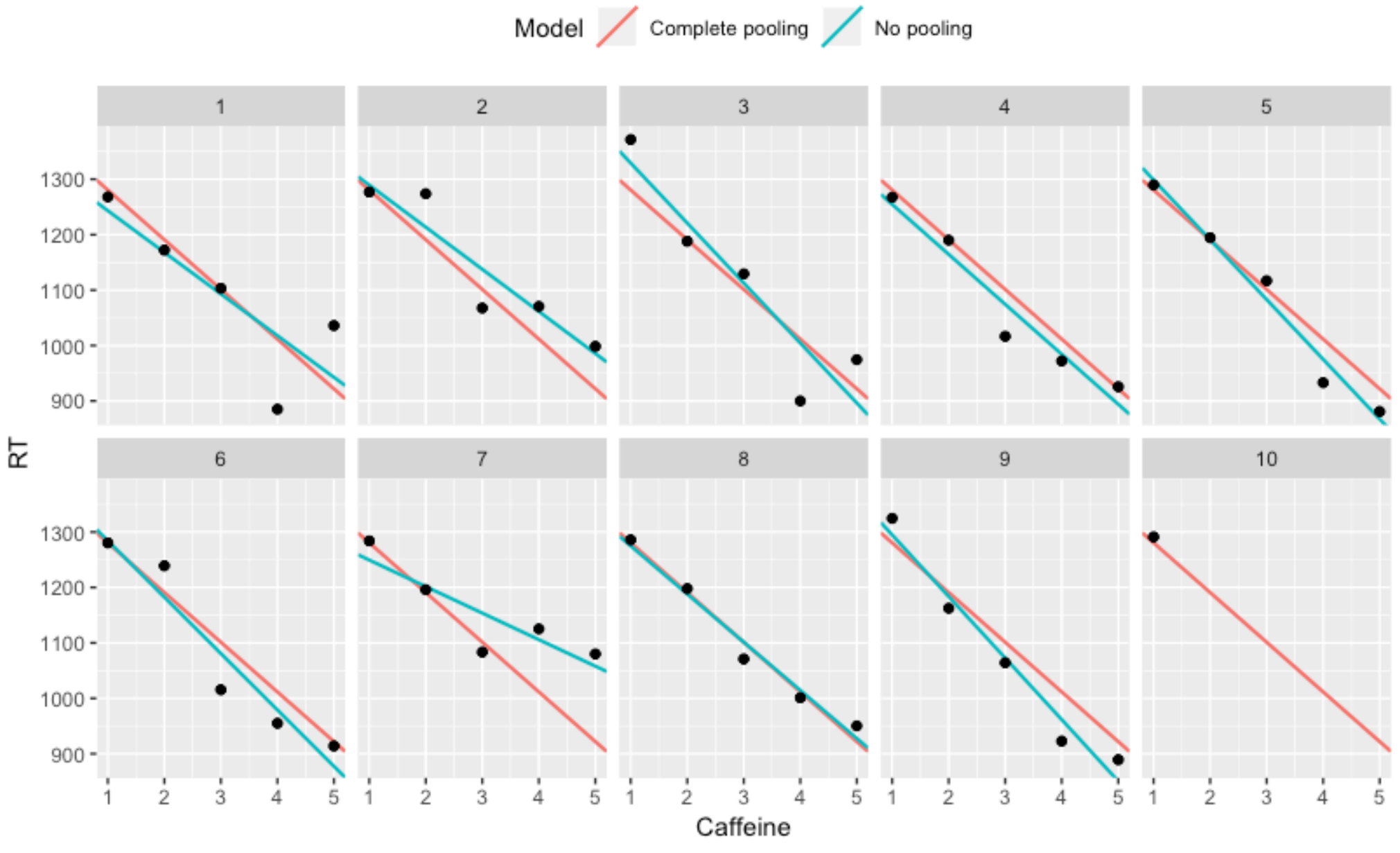
No Pooling

- We plot Reaction Time against Coffee Consumption separately for each Subject we are also adding a regression line by Subject. This is known as No Pooling.



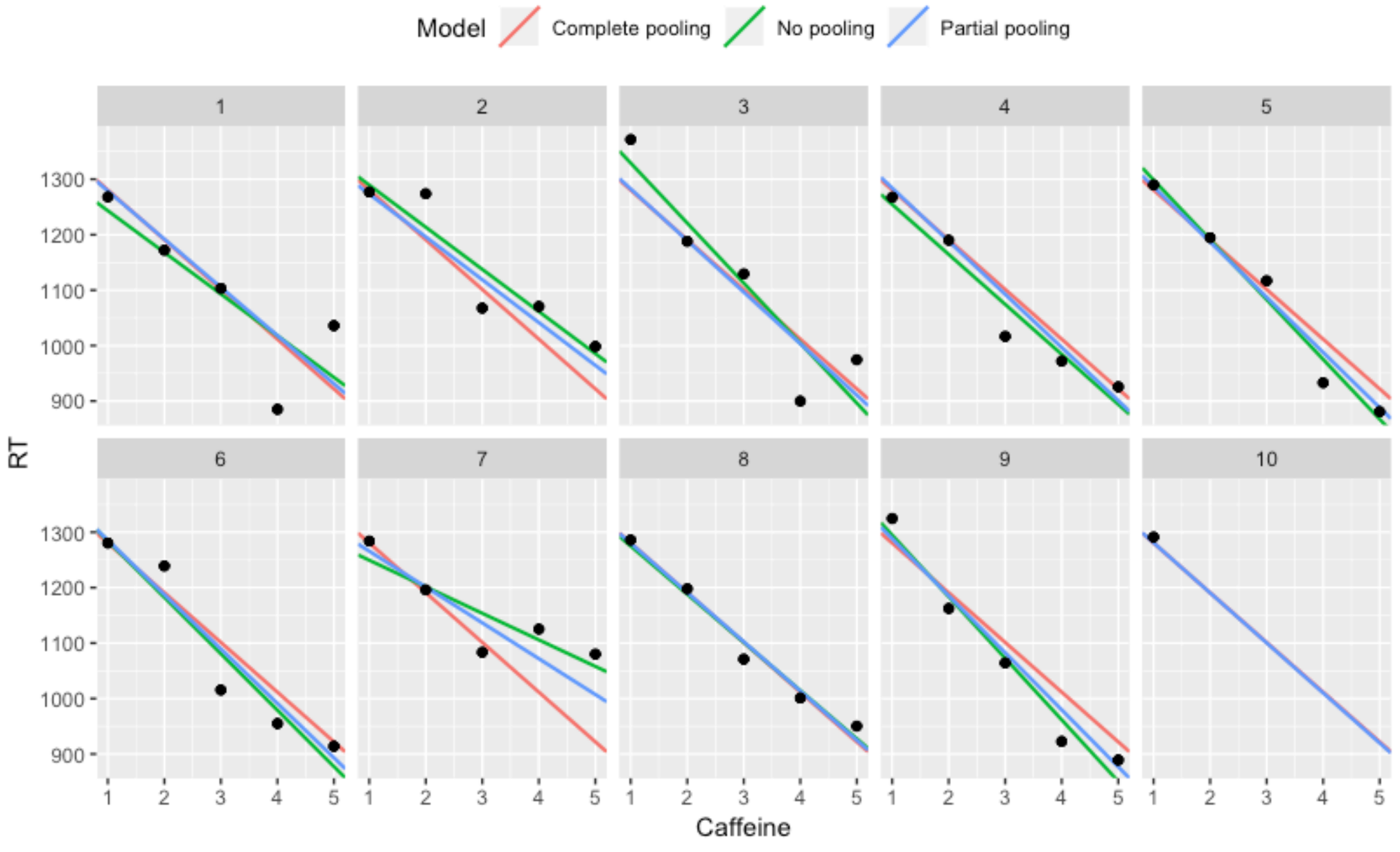
Complete Pooling

- In Complete Pooling, we fit an overall regression line to our entire dataset ignoring differences from one participant to the next.



Partial Pooling

- In Partial Pooling, we pool information from both sets of lines to improve our parameter estimates.



Partial Pooling

- Most of the time the partial pooling and no pooling lines are similar to each other - when they differ, it's because the partial pooling line is being drawn towards the complete pooling line. In other words, it's being affected by the dataset in its entirety.
- For participants with incomplete data, the partial pooling model is like the complete pooling model. The complete pooling and the partial pooling lines are basically parallel - i.e, they have the same slope. That's a reasonable guess given so little information.
- The process by which partial pooling pulls more extreme estimates towards an overall average (i.e., the complete pooling line) is known as *shrinkage*. Subject 7 is a good example of this happening.

Partial Pooling

- The use of partial pooling is one reason why LMMs are so powerful - they can cope with missing data (by being sensitive to properties of the overall dataset) and are not too affected by extreme data points (because they know these are quite unlikely in the context of the larger dataset - *shrinkage* reduces the influence of these extreme values on your parameter estimates).

Examples of LMMs for Factorial Designs

- In the first case, we will look at a model where we have one factor with three levels. We have two sets of data we want to analyse - one is eye gaze duration data, the other is the number of times people re-read a section of text.
- In the second case, we will look at a model for a 2 x 2 repeated measures design - this time just with eye gaze duration data as people read a section of text.

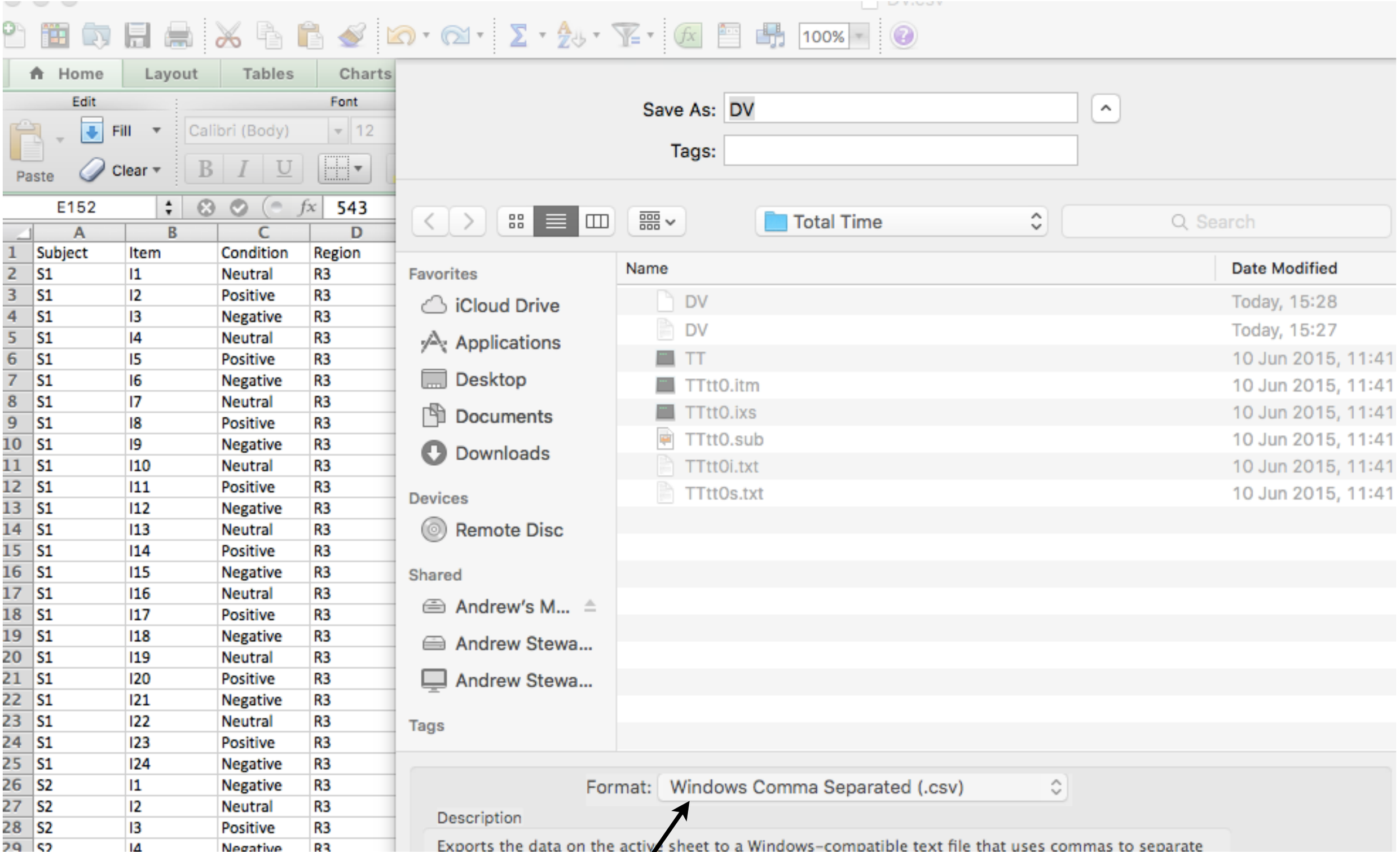
One factor with Three levels

- We are going to analyse eye movement data associated with reading a segment of text in one of three conditions - Positive, Negative, or Neutral.

```
#install the lme4, lsmeans and lmertest packages first
install.packages ("lme4")
install.packages ("lmerTest")
install.packages ("emmeans")
library (lme4)
library (lmerTest)
library(emmeans)
```

```
#C1 = Neutral condition
#C2 = Negative condition
#C3 = Positive condition
```

The *lmerTest* package gives us *p*-values for our fixed effects, while the *emmeans* allows us to conduct pairwise comparisons.



	A	B	C	D
1	Subject	Item	Condition	Region
2	S1	I1	Neutral	R3
3	S1	I2	Positive	R3
4	S1	I3	Negative	R3
5	S1	I4	Neutral	R3
6	S1	I5	Positive	R3
7	S1	I6	Negative	R3
8	S1	I7	Neutral	R3
9	S1	I8	Positive	R3
10	S1	I9	Negative	R3
11	S1	I10	Neutral	R3
12	S1	I11	Positive	R3
13	S1	I12	Negative	R3
14	S1	I13	Neutral	R3
15	S1	I14	Positive	R3
16	S1	I15	Negative	R3
17	S1	I16	Neutral	R3
18	S1	I17	Positive	R3
19	S1	I18	Negative	R3
20	S1	I19	Neutral	R3
21	S1	I20	Positive	R3
22	S1	I21	Negative	R3
23	S1	I22	Neutral	R3
24	S1	I23	Positive	R3
25	S1	I24	Negative	R3
26	S2	I1	Negative	R3
27	S2	I2	Neutral	R3
28	S2	I3	Positive	R3
29	S2	I4	Negative	R3

- First I need to re-save my data in Excel as a .csv file

- Our data file is called DV and looks like this:

analysis_submitted.R* × | DV ×

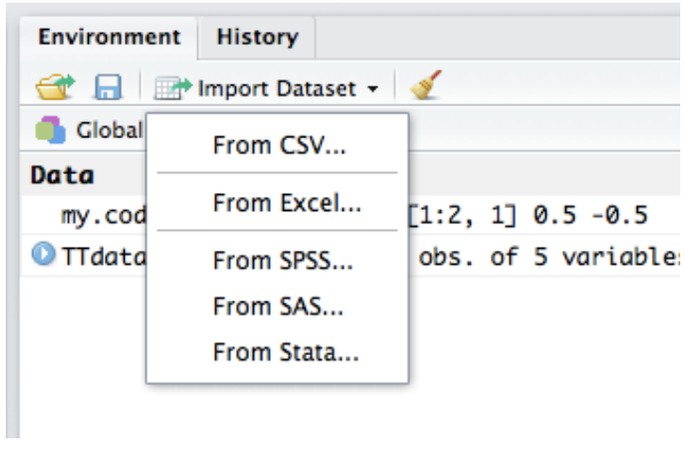
Filter

	Subject	Item	Condition	Region	Gaze
1	S1	I1	Neutral	R3	867
2	S1	I2	Positive	R3	1061
3	S1	I3	Negative	R3	771
4	S1	I4	Neutral	R3	626
5	S1	I5	Positive	R3	1283
6	S1	I6	Negative	R3	846
7	S1	I7	Neutral	R3	547
8	S1	I8	Positive	R3	1135
9	S1	I9	Negative	R3	1000
10	S1	I10	Neutral	R3	926
11	S1	I11	Positive	R3	1176
12	S1	I12	Negative	R3	847
13	S1	I13	Neutral	R3	761

Showing 1 to 13 of 574 entries

- The columns correspond to our Subject Number, our Item Number, our Condition, the Region of Text and the Gaze time (ms.)

- You then need to import the data file:



- Make sure you check that R correctly recognises your factors. In this case, it initially doesn't:

Data Preview:

Subject (character)	Item (character)	Condition (character)	Region (character)	Gaze (integer)
S1	I1	Neutral	R3	867
S1	I2	Positive	R3	1061
S1	I3	Negative	R3	771
S1	I4	Neutral	R3	626
S1	I5	Positive	R3	1283
S1	I6	Negative	R3	846
S1	I7	Neutral	R3	547
S1	I8	Positive	R3	1135
S1	I9	Negative	R3	1000
S1	I10	Neutral	R3	926
S1	I11	Positive	R3	1176
S1	I12	Negative	R3	847
S1	I13	Neutral	R3	761

Previewing first 50 entries.

The names of the columns you will use in your model (incl. the names of the random effects).

- For Condition, you need to select it as a Factor (not as a character string). Click on the down arrow, and then select Factor. Enter the levels separated by commas.

You can also use the function *as.factor* to turn your variable into a factor:

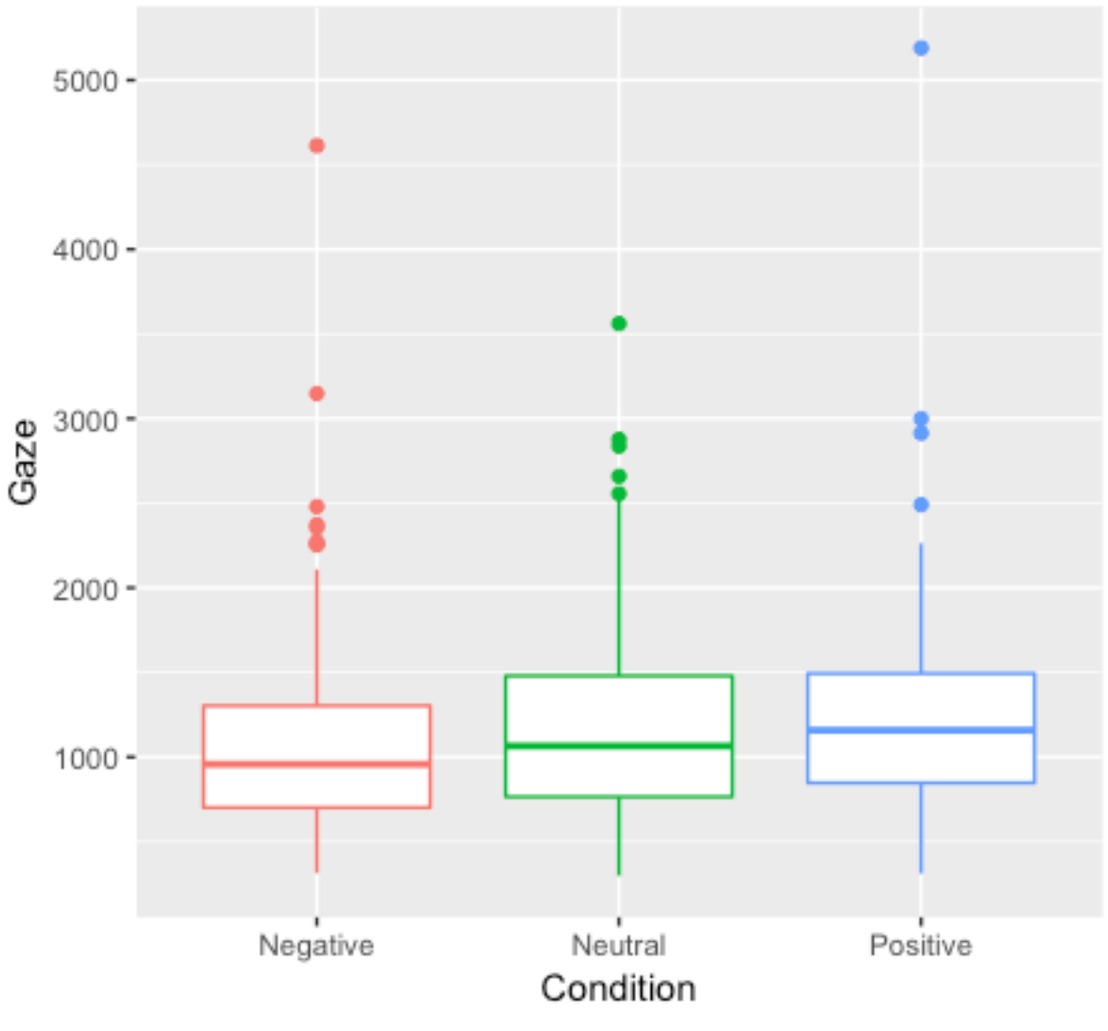
```
> DV$Condition <- as.factor(DV$Condition)
```

You can type the following to check the number of levels of the factor:

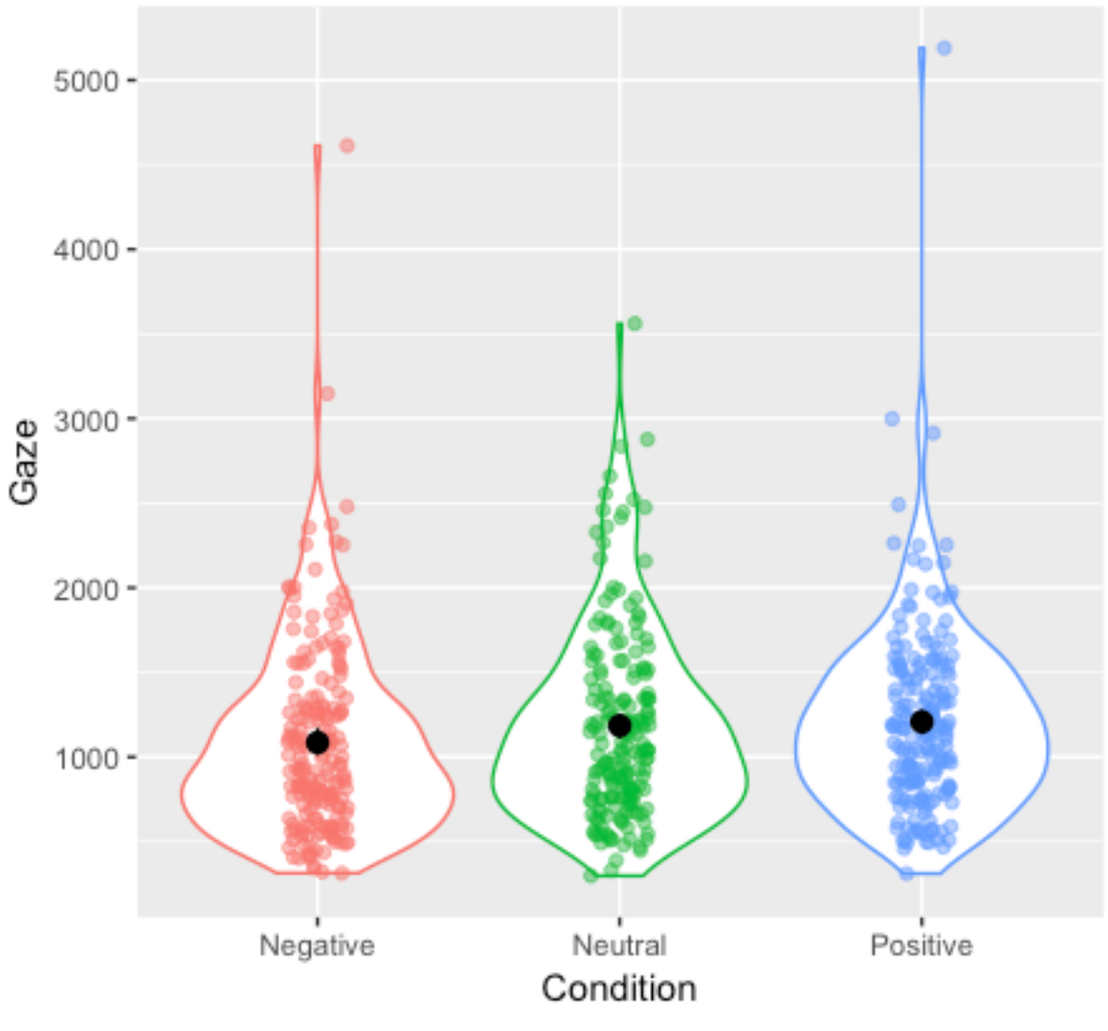
```
> levels(DV$Condition)
```

Visualising the Data

```
ggplot(DV, aes(x = Condition, y =
Gaze, colour = Condition)) +
geom_boxplot() + guides(colour =
FALSE)
```




```
ggplot(DV, aes(x = Condition, y = Gaze,
colour = Condition)) + geom_violin() +
geom_jitter(width = .1, alpha = .5) +
  stat_summary(fun.data =
"mean_cl_boot", colour = "black") +
guides(colour = FALSE)
```



```
25  model.null <- lmer (Gaze ~ (1 + Condition| Subject) + (1 + Condition| Item), data=DV, REML=TRUE)
26  model.full <- lmer (Gaze ~ Condition + (1 + Condition| Subject) + (1 + Condition| Item), data=DV, REML=TRUE)
27  anova (model.null, model.full)
28  summary (model.full)
```

- Line 25 creates a variable called model.null associated with just random effects of Subjects and Items. Note there is no fixed effect.
- Line 26 create a variable called model.full which includes both the random and fixed effects.
- Line 27 tests where the model.full is a better fit to our data and model.null. If it is, it means adding the fixed effect means we are able to explain our data better than if we don't add it.
- Line 28 then asks for the model.full parameters to be displayed.

The Output

```
> anova(model.null, model.full)
refitting model(s) with ML (instead of REML)
Data: DV
Models:
object: Gaze ~ (1 + Condition | Subject) + (1 + Condition | Item)
..1: Gaze ~ Condition + (1 + Condition | Subject) + (1 + Condition |
..1:     Item)
       Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
object 14 8773.4 8834.4 -4372.7   8745.4
..1    16 8771.2 8840.8 -4369.6   8739.2 6.236      2    0.04425 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

For model comparisons, a different parameter estimator must be used (R will do this for you). REML should be used to estimate parameters when you report them.

- Our two models differ significantly from each other. The one that fits our data the best has the lower AIC value. AIC is the Akaike Information Criterion and measures how much ‘information’ is not captured by our model (values that are relatively lower are better). NOTE - absolute AIC values cannot be interpreted - they have to be compared with the AIC value of another model.

```
Random effects:
 Groups   Name                Variance Std.Dev. Corr
 Subject  (Intercept)         108205   328.95
          ConditionNeutral      2589    50.88   -1.00
          ConditionPositive     6425    80.16   -1.00  1.00
 Item     (Intercept)          32985   181.62
          ConditionNeutral      1296    36.00    0.00
          ConditionPositive     3897    62.42   -0.54  0.84
 Residual                      204916   452.68
Number of obs: 574, groups:  Subject, 24; Item, 24

Fixed effects:
                  Estimate Std. Error      df t value Pr(>|t|)
(Intercept)        1083.76      83.40   30.15  12.994 6.88e-14 ***
ConditionNeutral    101.04      48.05   52.01   2.103   0.0403 *
ConditionPositive   123.54      50.70   22.73   2.437   0.0231 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- This is what we're mainly interested in. We know the model itself is significantly better than the null model. These comparisons tells us what differences are driving the effect.

- Think of these like the contrasts that are used to interpret significant ANOVAs. In this case, the Neutral and Positive conditions are each being compared to the Negative condition (or the intercept of the regression line). The estimates tell us that the intercept is 1084 (which is the Negative condition mean). The Neutral mean is 1084+101, while the Positive mean is 1084+124.

A few points to note so far…

- Models can only be compared to each other using the ANOVA function if they are nested - in other words, if one model is a subset of the other. Models with different fixed and random effects structures cannot be compared in this way - use AIC or BIC comparisons.

- If using treatment coding for Contrasts, sometimes the Intercept (or reference level condition) chosen by R isn't the one you might want. You can change it using: `DV$Condition <- relevel (DV$Condition, ref = 3)` where ref corresponds to the level of the factor Condition you want as the intercept, DV corresponds to the datafile, and Condition corresponds to the factor you want to relevel.

What if our DV isn't a continuous variable?

- In eye movement work, we measure both gaze time (ms.) and also the number of times people re-read a region of text. For any one person reading a region of text, they either re-read it, or they don't. Thus, the data are binary (not continuous). In our data set, 1 corresponds to a region being re-read, 0 to not being re-read.

for workshop.R* | RO | DV

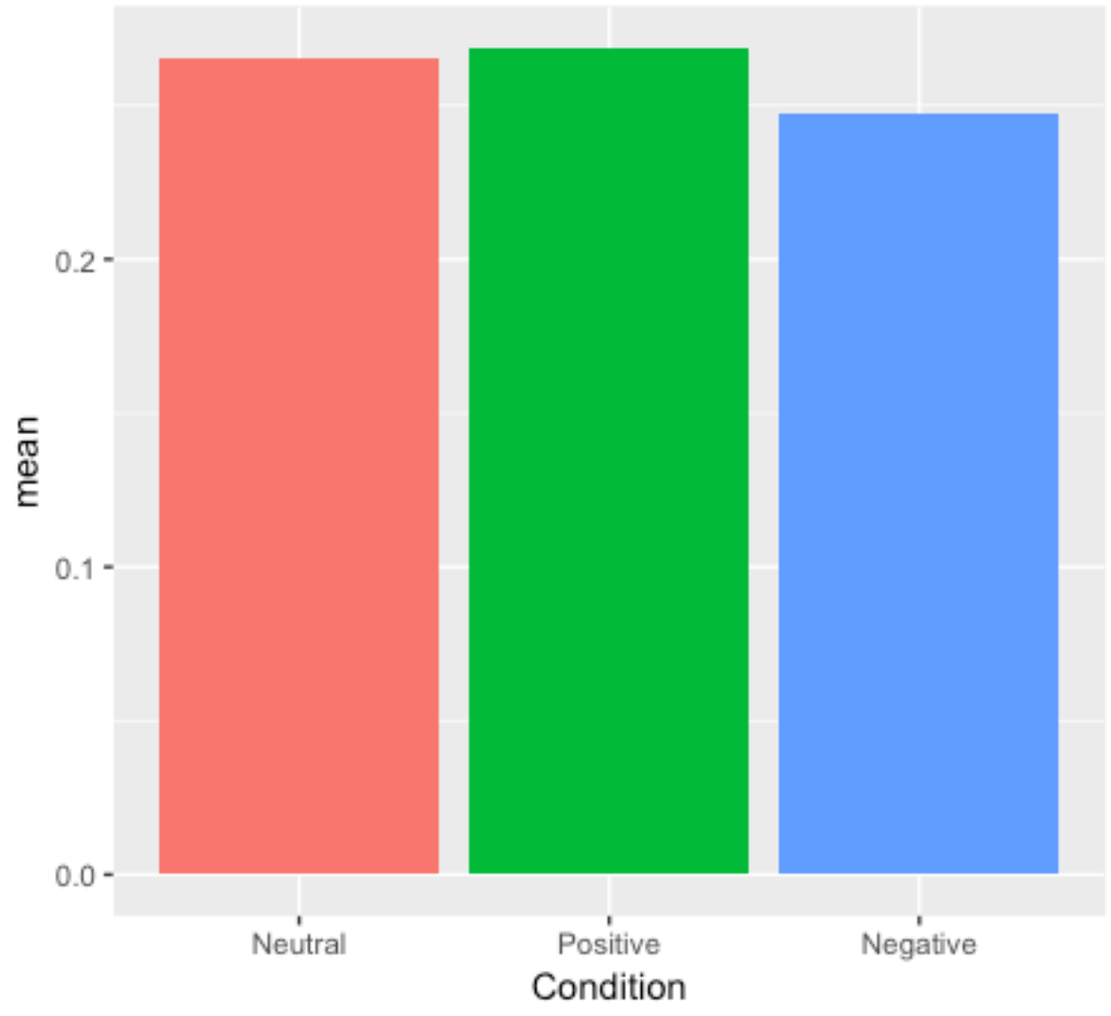
Filter

	Subject	Item	Condition	Region	DV
1	S1	I2	Positive	R3	0
2	S1	I3	Negative	R3	0
3	S1	I4	Neutral	R3	0
4	S1	I5	Positive	R3	0
5	S1	I6	Negative	R3	0
6	S1	I7	Neutral	R3	1
7	S1	I8	Positive	R3	0
8	S1	I9	Negative	R3	0
9	S1	I10	Neutral	R3	0
10	S1	I11	Positive	R3	0
11	S1	I12	Negative	R3	0
12	S1	I13	Neutral	R3	0

Showing 1 to 12 of 553 entries

- Here is our data file - our DV is categorical - either 1 or 0.

- Looks like we might have *slightly* fewer regressions in the Negative condition.



- For binomial data, we have to use the generalised linear model (`glmer`) and the binomial distribution. This is the syntax for such a model with both fixed and random effects:

```
model.full <- glmer (DV ~ Condition + (1 + Condition|Subject) + (1 + Condition|Item), data=RO, family=binomial)
```

- When we run it, we get an error (that you will get used to seeing again and again!)

```
> model.full <- glmer (DV ~ Condition + (1 + Condition|Subject) + (1 + Condition|Item), data=RO, family=binomial)
Warning message:
In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model failed to converge with max|grad| = 0.0171702 (tol = 0.001, component 1)
```

- So what can we do? We need to simplify the random effects structures. We can do this by dropping terms one by one until we find a model that can be fitted to our data. For example, we could drop the random slope from our items random effect first.

- For this particular example, the most complex model that converges involves only random intercepts.

```
> model.interceptonly <- glmer(DV ~ Condition + (1|Subject) + (1|Item) , data=RO,
family=binomial)
> model.null <- glmer (DV ~  (1|Subject) + (1|Item), data=RO, family=binomial)
> anova (model.interceptonly, model.null)
```

```
> anova (model.interceptonly, model.null)
Data: RO
Models:
model.null: DV ~ (1 | Subject) + (1 | Item)
model.interceptonly: DV ~ Condition + (1 | Subject) + (1 | Item)
                    Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
model.null           3 601.97 614.91 -297.98   595.97
model.interceptonly  5 605.70 627.28 -297.85   595.70 0.2617      2     0.8773
```

- Our model with a fixed effect of Condition, and with random intercepts is no better than our model with just the random intercepts. In fact, it's worse - look at the AIC values. So we have no effect of Condition in our re-reading data.

What can we conclude from non-significant results?

- With NHST, a non-significant result means we cannot reject the null hypothesis. But it does not mean our data support the null hypothesis. There may be other hypotheses that our data fit - we just didn't test them.

- What we can do is estimate the Bayes factors associated with our data in support of the null and experimental models.

- Wagenmakers (2007) details a means to estimating the Bayes factor (BF) of our data in support of one model or another using BIC. Essentially, it gives us a measure of the extent to which our data support a particular model.

- BFs are estimated using the BIC value for each model - BIC penalises additional parameters so the BF captures possible overfitting (i.e., too many parameters on our model).

- BF = exp ((BIC2 - BIC1)/2)

```
> anova(model.interceptonly, model.null)
Data: RO
Models:
model.null: DV ~ (1 | Subject) + (1 | Item)
model.interceptonly: DV ~ Condition + (1 | Subject) + (1 | Item)
                    Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
model.null           3 601.97 614.91 -297.98   595.97
model.interceptonly  5 605.70 627.28 -297.85   595.70 0.2617      2     0.8773
```

Interpretation of the Bayes Factor in Terms of Evidence (cf. Raftery, 1995, Table 6)

Bayes Factor BF_{01}	$\Pr(H_0 \mid D)$	Evidence
1–3	.50–.75	weak
3–20	.75–.95	positive
20–150	.95–.99	strong
>150	>.99	very strong

```
> anova (model.interceptonly, model.null)
Data: RO
Models:
model.null: DV ~ (1 | Subject) + (1 | Item)
model.interceptonly: DV ~ Condition + (1 | Subject) + (1 | Item)
                    Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
model.null           3 601.97 614.91 -297.98   595.97
model.interceptonly  5 605.70 627.28 -297.85   595.70 0.2617      2     0.8773
```

BF = exp ((BIC2 - BIC1)/2)

BF = exp ((627.28-614.91)/2)

BF = 485

A BF of 485 is “Very Strong” evidence in support of the null hypothesis.

LMMs for a 2 x 2 Repeated Measures Design

- Now let's take a 2 x 2 repeated measures design. We measured people's eye movements as they read either positive or negative information. Prior context set up expectations that the story was likely to continue with positive vs. negative information.

- Factor 1 is Context (Positive vs. Negative)

- Factor 2 is Sentence Type (Positive vs. Negative)

DV | Workshop script2.R*

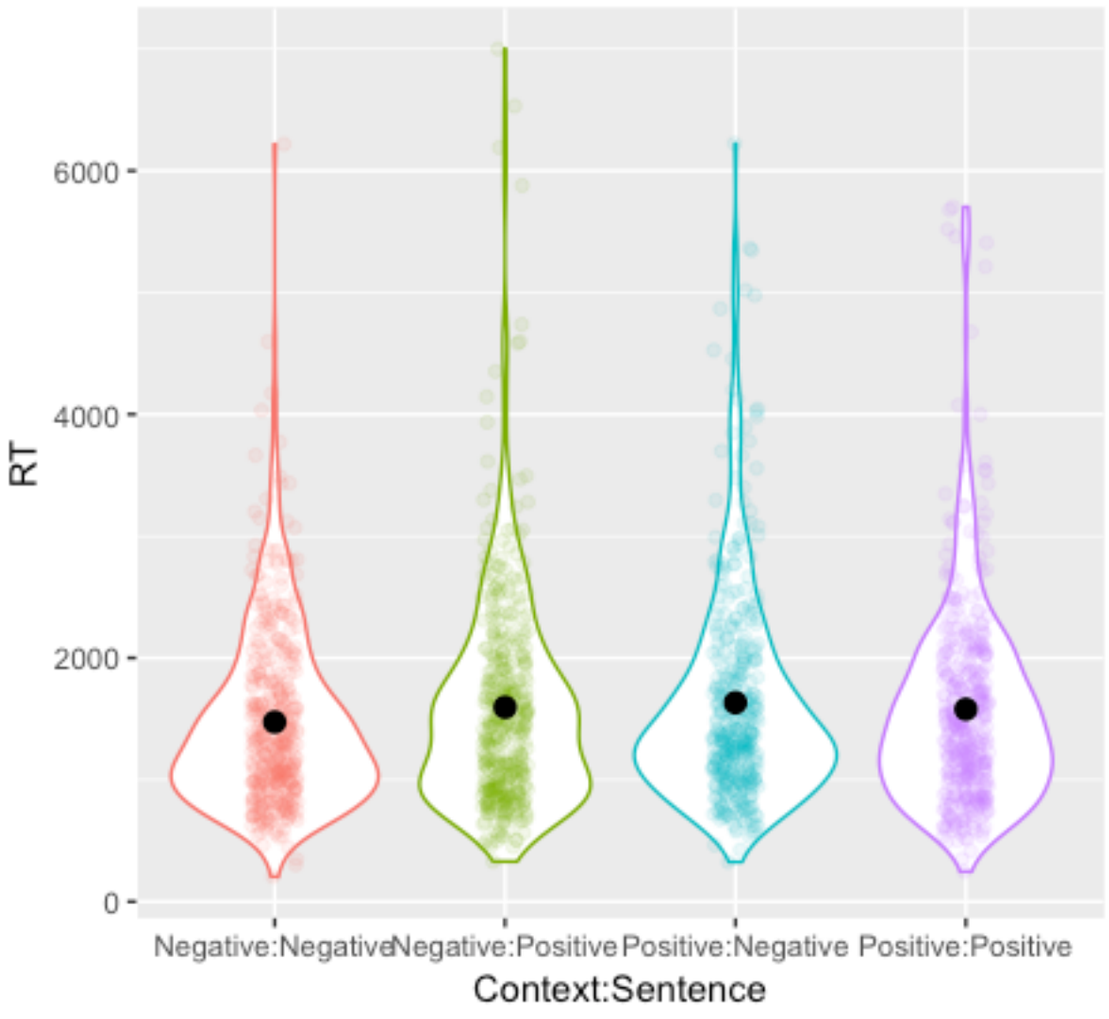
Filter

	Subject	Item	RT	Context	Sentence
1	1	3	1270	Positive	Negative
2	1	7	739	Positive	Negative
3	1	11	982	Positive	Negative
4	1	15	1291	Positive	Negative
5	1	19	1734	Positive	Negative
6	1	23	1757	Positive	Negative
7	1	27	1052	Positive	Negative
8	2	4	1706	Positive	Negative
9	2	8	533	Positive	Negative
10	2	12	1009	Positive	Negative
11	2	16	939	Positive	Negative
12	2	20	1848	Positive	Negative
13	2	24	1435	Positive	Negative
14	2	28	922	Positive	Negative

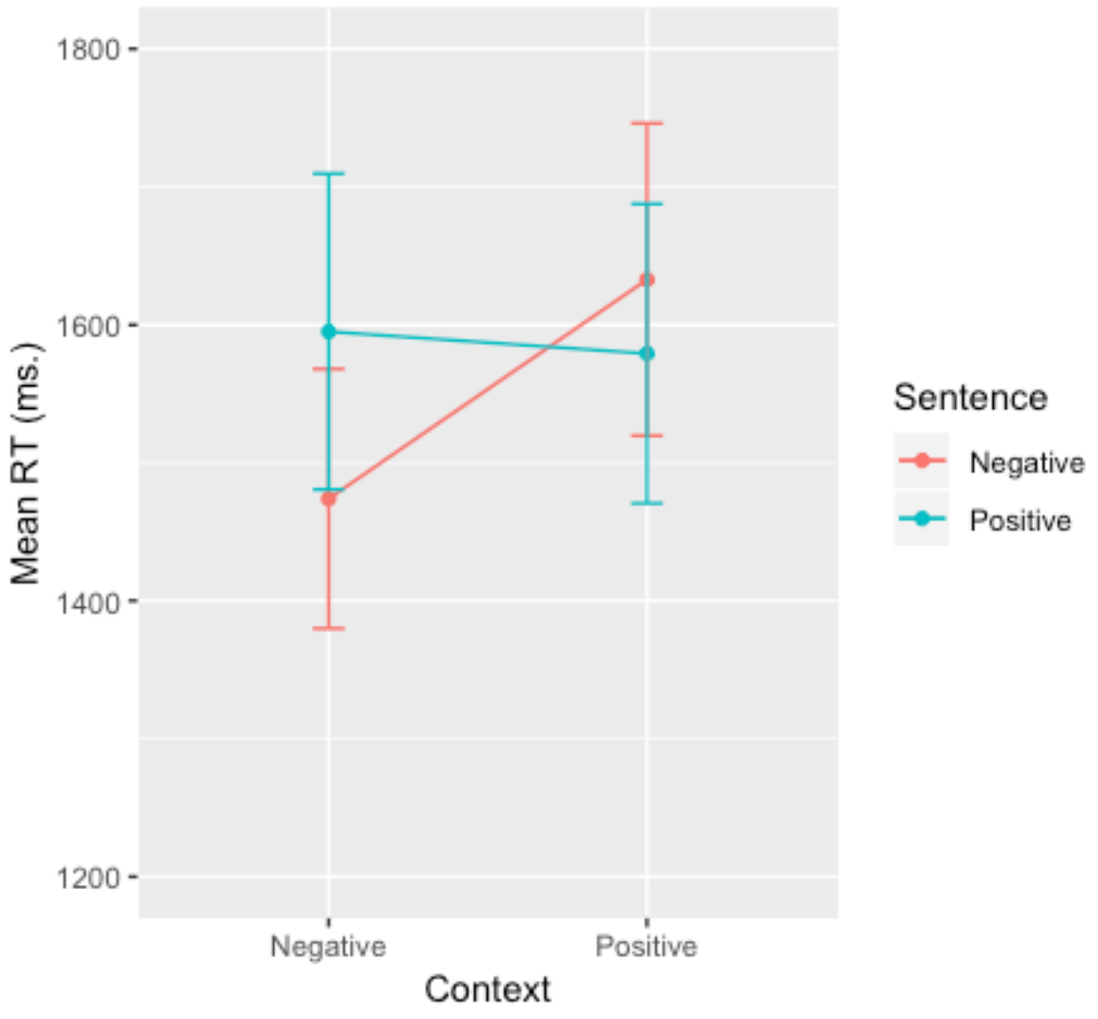
Showing 1 to 15 of 1,680 entries

- We have Subject number, Item number, RT (reading time), Context and Sentence.

Visualise



Raw data



Aggregated data

- The first thing we need to do is to apply contrast weightings to our two factors. By default, the contrasts are dummy or treatment coded. We need to change them to deviation coded. This helps make the coefficients in the LMM make more sense as the intercept of the LMM will correspond to the Grand Mean (i.e., the mean of all four conditions).

```
contrasts(DV$Sentence) <- matrix(c(.5, -.5))
contrasts(DV$Context) <- matrix(c(.5, -.5))
```

- We are going to define our full model with our fixed effects and fully crossed Subject and Item random effects.
- Then we are going to define the null model with only the random effects.

```
model.full <- lmer(RT~Context*Sentence + (1+Context*Sentence|Subject) + (1+Context*Sentence|
Item), data=DV, REML=TRUE)

model.null <- lmer(RT~(1+Context*Sentence|Subject) + (1+Context*Sentence|Item), data=DV,
REML=TRUE)
```

- Note that we define our fixed effect using the notation `Context*Sentence`
- This is equivalent to `(Context + Sentence + Context:Sentence)` which corresponds to a main effect of Context, a main effect of Sentence and the interaction between the two (as represented by the colon symbol).

```
> anova(model.full, model.null)
refitting model(s) with ML (instead of REML)
Data: DV
Models:
model.null: RT ~ (1 + Context * Sentence | Subject) + (1 + Context * Sentence |
model.null:     Item)
model.full: RT ~ Context * Sentence + (1 + Context * Sentence | Subject) +
model.full:     (1 + Context * Sentence | Item)
           Df   AIC   BIC logLik deviance  Chisq Chi Df Pr(>Chisq)
model.null 22 26720 26840 -13338    26676
model.full 25 26718 26853 -13334    26668 8.6625      3    0.03413 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Our model with the fixed effects (as well as the random effects) is a better fit for our data than is the model just with the random effects. Now we need to look at the model parameters using the `summary()` function…

```
> summary(model_full)
```

```
Fixed effects:
                   Estimate Std. Error      df t value Pr(>|t|)
(Intercept)         1568.75      76.24   50.07  20.577   <2e-16 ***
Context1             -69.01      39.87   25.94  -1.731   0.0954 .
Sentence1            -36.20      86.01   29.77  -0.421   0.6768
Context1:Sentence1  -168.73      80.36   25.51  -2.100   0.0458 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- We can see that the interaction is significant. But how do we know what difference(s) is/are driving this effect?
- Think back to ANOVA days - we need to now do something else…

- We can run pairwise comparisons. We can ask for a correction to be applied if we want to, but in this case we're doing to work out that correction by hand. There are only 2 theoretically meaningful pairwise comparisons, so we multiply the reported p value by 2 to manually apply Bonferroni correction.

- We use the `emmeans()` function in the `emmeans` package.

```
> emmeans(model.full, pairwise~Context*Sentence, adjust="none")
$emmeans
 Context  Sentence   emmean       SE    df lower.CL upper.CL
 Negative Negative 1473.962 81.92645 39.71 1308.344 1639.580
 Positive Negative 1627.340 97.87423 41.97 1429.818 1824.862
 Negative Positive 1594.530 96.56707 36.99 1398.865 1790.194
 Positive Positive 1579.181 90.79179 48.12 1396.644 1761.718

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95

$contrasts
 contrast                                estimate       SE    df t.ratio p.value
 Negative,Negative - Positive,Negative -153.37807 50.68254 20.94  -3.026  0.0064
 Negative,Negative - Negative,Positive -120.56791 92.61292 30.57  -1.302  0.2027
 Negative,Negative - Positive,Positive -105.21905 92.22803 29.04  -1.141  0.2633
 Positive,Negative - Negative,Positive   32.81016 97.35194 31.48   0.337  0.7383
 Positive,Negative - Positive,Positive   48.15902 97.23988 26.58   0.495  0.6245
 Negative,Positive - Positive,Positive   15.34886 62.02003 27.31   0.247  0.8064
```

Here we have the descriptive statistics associated with each of our 4 conditions.

Above are all the possible pairwise comparisons - only 2 are of theoretical interest to us:

1. A Negative meaning sentence following a Negative Context vs. the same Negative meaning following a Positive Context.

2. A Positive meaning sentence following a Negative Context vs. the same Positive meaning following a Positive Context.

```
$contrasts
 contrast                                    estimate       SE    df t.ratio p.value
 Negative,Negative - Positive,Negative -153.37807 50.68254 20.94  -3.026  0.0064
 Negative,Negative - Negative,Positive -120.56791 92.61292 30.57  -1.302  0.2027
 Negative,Negative - Positive,Positive -105.21905 92.22803 29.04  -1.141  0.2633
 Positive,Negative - Negative,Positive   32.81016 97.35194 31.48   0.337  0.7383
 Positive,Negative - Positive,Positive   48.15902 97.23988 26.58   0.495  0.6245
 Negative,Positive - Positive,Positive   15.34886 62.02003 27.31   0.247  0.8064
```

- The two key comparisons reveal that Positive sentences are read no more quickly after Positive than after Negative context (1579 vs. 1595 ms.) while Negative Sentences are read more quickly after Negative than after Positive contexts (1474 vs. 1627 ms.)

- Note, the estimates in each contrast pairing corresponds to the difference between the comparison conditions for that pair.

- If we had re-reading (i.e., regression) data, we would also have to run an analysis using the *glmer* function on those data. The code would look like:

```
model.full <- glmer(Regressions ~ Context*Sentence + (1 + Context*Sentence|Subject)
+ (1 + Context*Sentence|Item), data=RO, family=binomial)
```

- To generate the pairwise comparisons (and to report the descriptives using the original measurement scale), we would use:

```
emmeans(model.full, pairwise~ Context*Sentence, adjust="none", type = "response")
```

- If we did not set the `type` parameter, then the descriptives would be on a log odds ratio scale (and harder to interpret).

Citing Packages

Remember to cite the packages you use, plus the version of R itself (with year info.) If you don't do this, you're not doing reproducible research. To find out how to cite a particular package, type:

```
> citation ("lme4")

To cite lme4 in publications use:

  Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models
  Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.
```

And to find out which version of R you are using:

```
> version

                _
platform       x86_64-apple-darwin15.6.0
arch           x86_64
os             darwin15.6.0
system         x86_64, darwin15.6.0
status
major          3
minor          4.3
year           2017
month          11
day            30
svn rev        73796
language       R
version.string R version 3.4.3 (2017-11-30)
nickname       Kite-Eating Tree
```

Addressing lack of convergence

```
Warning message:
In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model failed to converge with max|grad| = 0.0171702 (tol = 0.001, component 1)
```

If you see this message (which you will - again and again and again and again), it means you have to simplify your random effects structure so that a model can be identified.

Addressing lack of convergence

Simplify your random effects structure step by step. For an experiment with two factors (Factor 1 and Factor 2) we could simplify the participant and item random effects like this:

```
(1 + Factor 1*Factor 2| Participant) + (1 + Factor 1*Factor 2| Item)

(1 + Factor 1*Factor 2| Participant) + (1 + Factor 1+Factor 2| Item)

(1 + Factor 1+Factor 2| Participant) + (1 + Factor 1+Factor 2| Item)

(1 + Factor 1+Factor 2| Participant) + (1 + Factor 1| Item)
```

…

If you think your random effects looks too sparse when settling on a model that converges, you could try dropping one effect term entirely and then simplifying the other:

```
(1 + Factor 1*Factor 2| Participant) + (1 + Factor 1*Factor 2| Item)

(1 + Factor 1+Factor 2| Participant)

(1 + Factor 1| Participant)

(1 + Factor 2| Participant)

…
```

You want to avoid random effects with just random intercepts (i.e., no slopes) as that can inflate the Type I error rate (Barr et al., 2013). But you also want a random effects structure supported by your data and theory (Bates et al., 2015) and doesn't involve *overfitting*.

A few other LMM things…

- You can add participant and item covariates as fixed effects, and you can have a variety of continuous and categorical variables in your LMM. LMMs are *very* flexible.

- You'll find that sometimes several models fit your data - always run likelihood comparison tests to determine which is the best fit. If you have a selection where not one is statistically better than the others, choose the model that makes most *theoretical* sense.

Important Point

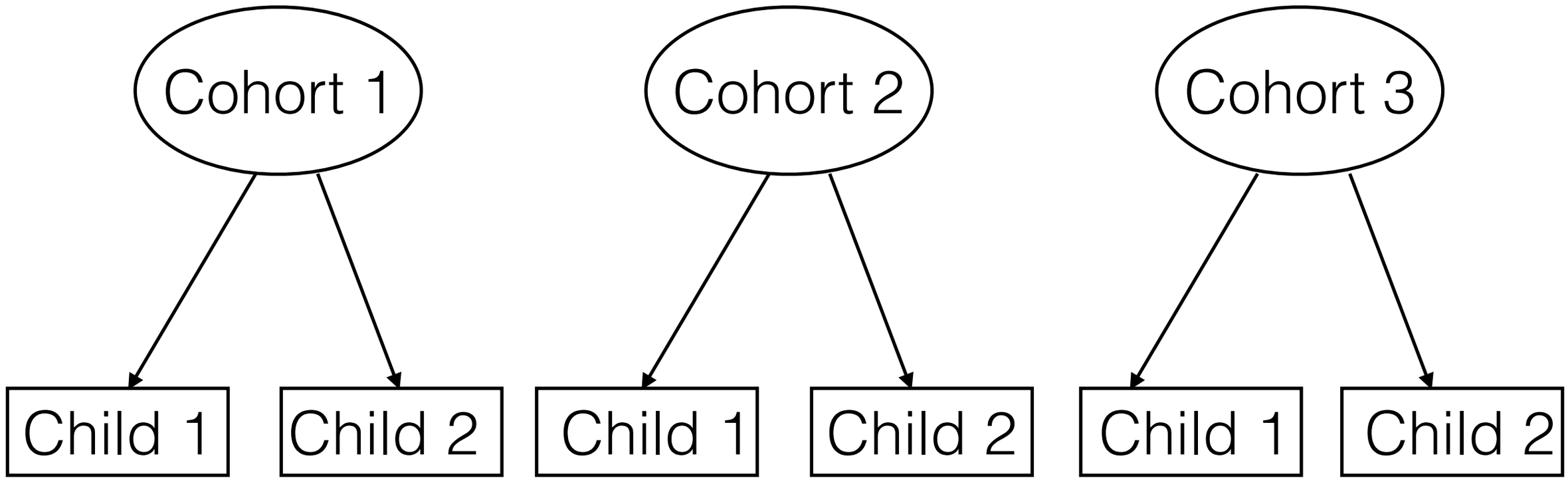
- Add as many random slopes (not just random intercepts) as your experimental design allows - for most cases, we expect variation between participants in terms of how they'll respond to different levels of an experimental condition (which is why we add participants as a random slope) and also variation between our experimental items to different levels of an experimental condition (which is why we also add items as a random slope).

- If the full model with random slopes and intercepts does not converge, then gradually simplify your random effects structures (e.g., drop an interaction term first, then drop a main effect etc.) until you find a model that does converge.

Crossed vs. Nested Random Effects

- In most experimental designs, your participant and item random factors are likely to be crossed - so random effects notation for a one factor experiment is `(1 + Factor | Subjects) + (1 + Factor | Items)`
- In some cases though, your factors might be *nested*. Nesting is a property of your data.
- To illustrate:

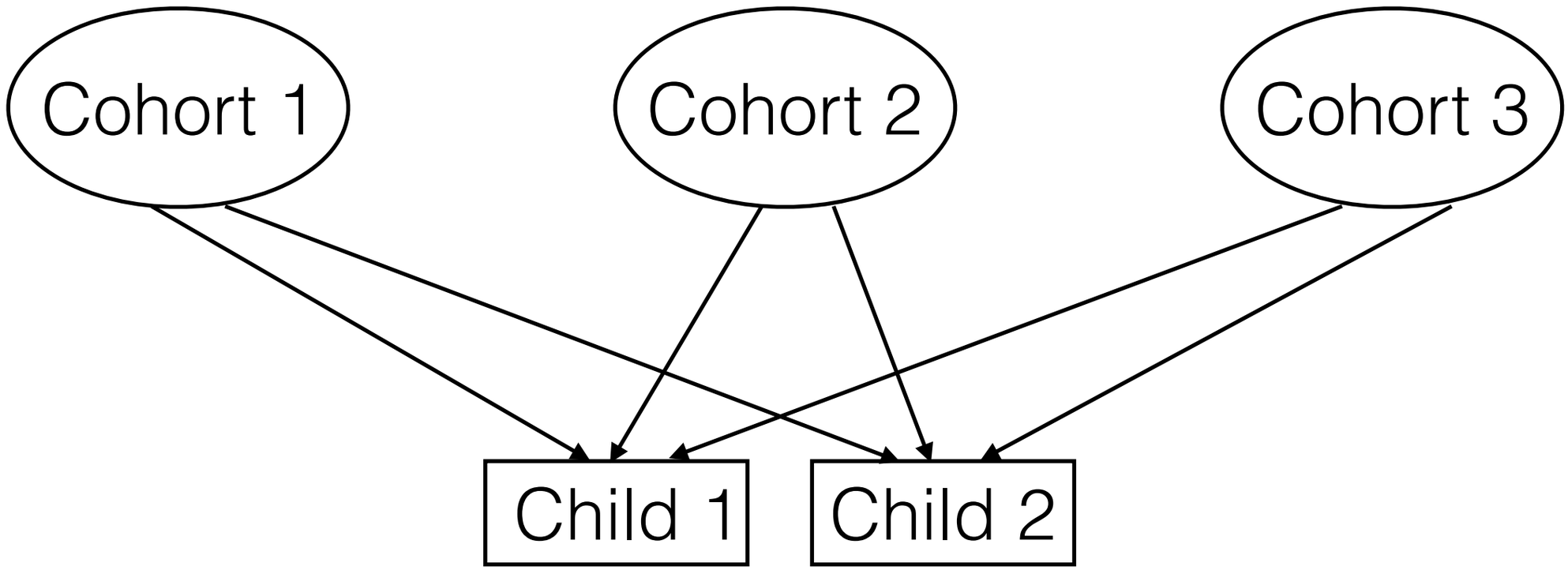
Nested



Child has an identifier that refers to a different child in each Cohort. Each child appears only in one Cohort. Child is nested within Cohort so random effects structure would be:

```
(1 + Factor | Cohort/Child)
```

Crossed



Each child appears in each Cohort. The levels are crossed so random effects structure would be:

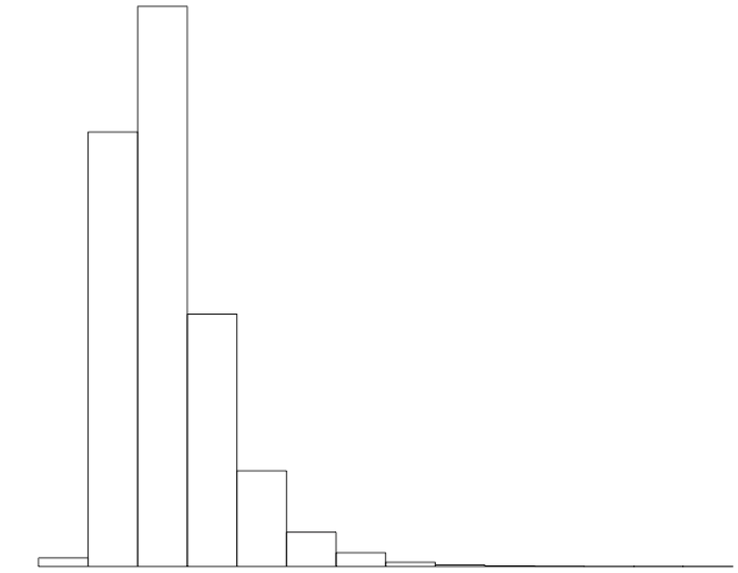
```
(1 + Factor | Cohort) + (1 + Factor | Child)
```

What about normality?

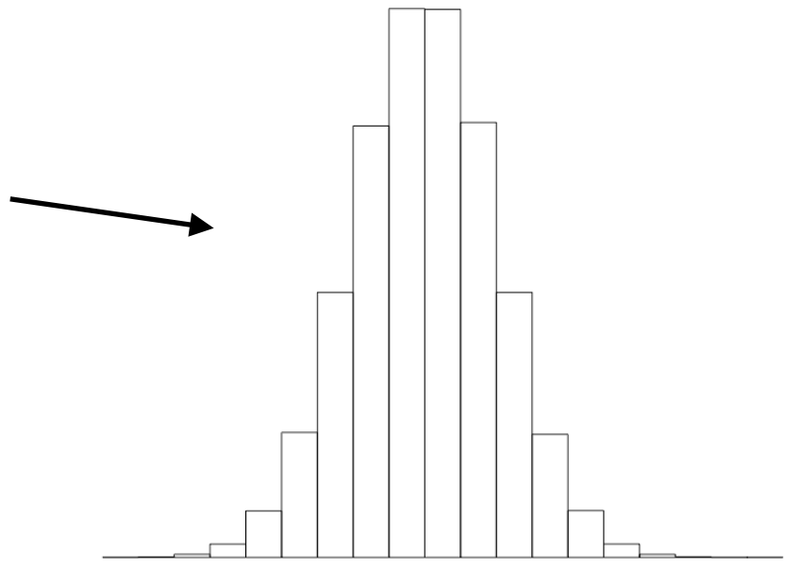
- In LMMs (as with the GLM) we need to worry about the normality of the residuals…
- You can check normality in a number of ways.
- Graphically, you can use the *qqnorm* function (which produces a Q-Q plot), and *hist* (which produces a histogram) applied to the model residuals.
- Statistically, you could use the *shapiro.test* function applied to a distribution of data. Be aware that for large datasets, even small deviations from normality will result in a significant Shapiro test. So best not to use this…

Log transform

Typically, RT data are non-normal and more often the DV looks like this.

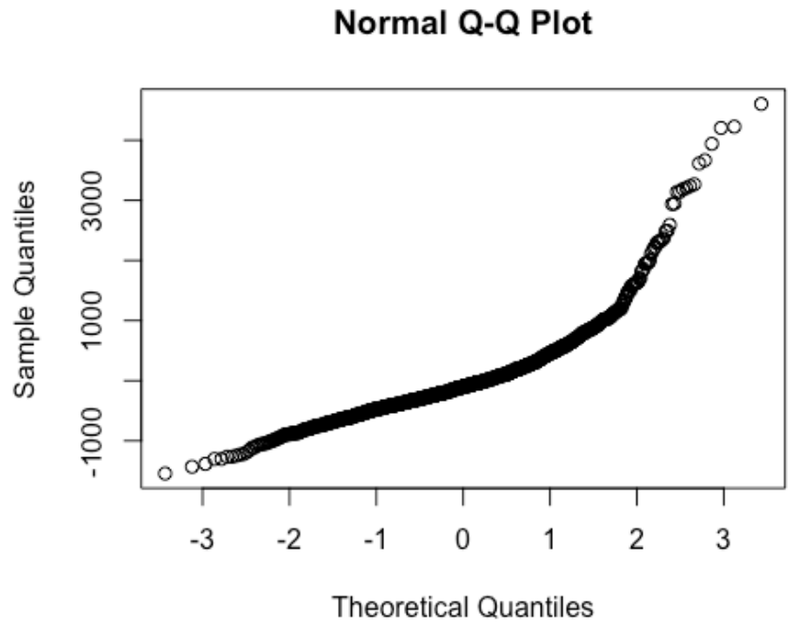


We can log transform our DV to approximate something that looks a bit more like the normal distribution (could also look at inverse RT). But there are risks around transforming data (Lo & Andrews, 2015)



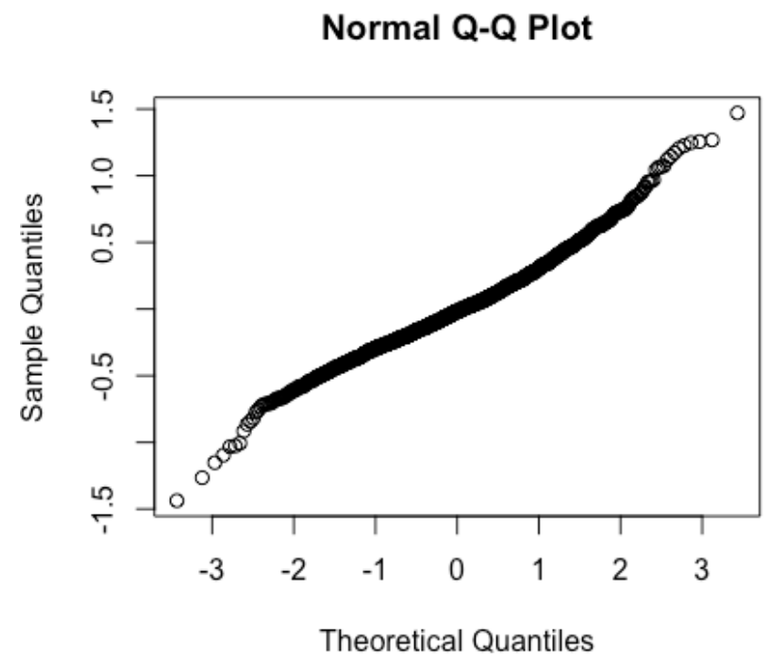
- Normality test on the model residuals from the untransformed data:

```
> qqnorm(residuals(model.full))
> shapiro.test(residuals(model.full))

        Shapiro-Wilk normality test

data:  residuals(model.full)
W = 0.84583, p-value < 2.2e-16
```



- Normality test on the model residuals from the log transformed data:

```
> model.full <- lmer (log(RT) ~ Sentence*Context + (1+Sentenc
e*Context|Subject) + (1+Sentence*Context |Item), data=DV, REM
L=TRUE)
> qqnorm(residuals(model.full))
> shapiro.test(residuals(model.full))

        Shapiro-Wilk normality test

data:  residuals(model.full)
W = 0.98321, p-value = 4.626e-13
```



- The original analysis on the untransformed data:

```
Fixed effects:
                  Estimate Std. Error      df t value Pr(>|t|)
(Intercept)        1568.75      76.24   50.07  20.577   <2e-16 ***
Context1            -36.20      86.01   29.77  -0.421   0.6768
Sentence1           -69.01      39.87   25.93  -1.731   0.0954 .
Context1:Sentence1 -168.73      80.36   25.51  -2.100   0.0458 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- The new analysis on the log transformed data:

```
Fixed effects:
                  Estimate Std. Error       df t value
(Intercept)        7.23975    0.04967 49.13000 145.761
Sentence1          0.01392    0.05278 29.03000   0.264
Context1           0.04316    0.02258 28.62000   1.911
Sentence1:Context1 -0.09333    0.04618 25.55000  -2.021
```

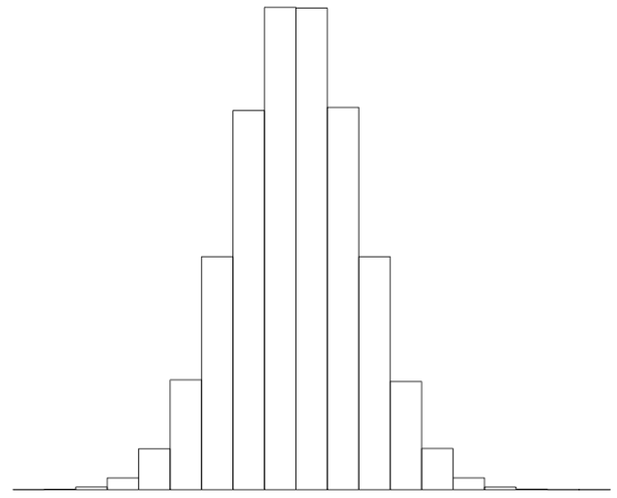
t-value of the interaction smaller than in analysis over untransformed data. With similar dfs, *p* will be bigger.

Other distributions under the GLMM via the function `glmer` are available…

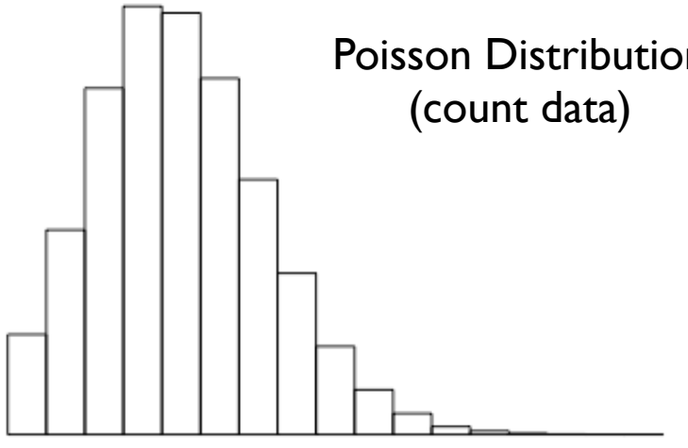
Usage

```
family(object, ...)

binomial(link = "logit")
gaussian(link = "identity")
Gamma(link = "inverse")
inverse.gaussian(link = "1/mu^2")
poisson(link = "log")
quasi(link = "identity", variance = "constant")
quasibinomial(link = "logit")
quasipoisson(link = "log")
```

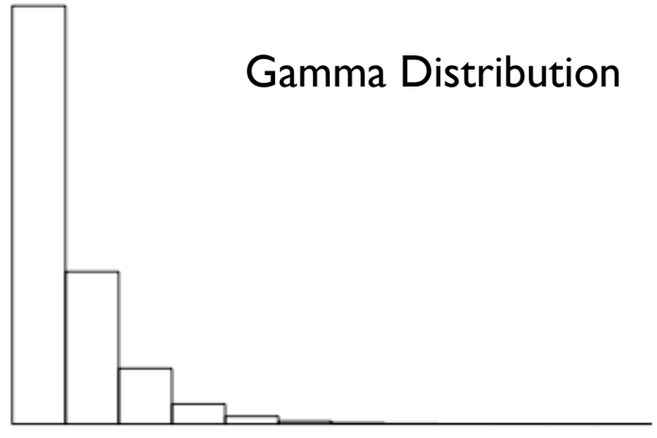
Normal (Gaussian) Distribution



Poisson Distribution
(count data)



Gamma Distribution



- Standard linear model assumes a normal distribution of residuals. In the *generalised* linear mixed model, we can assume a distribution in our model that doesn't involve a normal distribution. We have already looked at the binomial.
- Gamma distribution is another possibility (see Kliegl et al. 2010, Lo & Andrews, 2015, for discussion).

```
model1 <- glmer (RT ~ Sentence*Context + (1+Sentence*Context|Subject) + (1+Sentence*Context|Item), data=DV, family=Gamma)
summary (model1)
```

```
Fixed effects:
                  Estimate Std. Error t value
(Intercept)        7.28232    0.06731  108.20
Sentence1          0.02284    0.07679    0.30
Context1           0.04276    0.01701    2.51
Sentence1:Context1 -0.10806    0.03403   -3.18
```

t-value of the interaction larger than in previous analysis.

So what to do?

- In this example, all three analyses told basically the same story - there is an effect in our interaction term. They differ in terms of the value of the t-statistic associated with testing this.
- It's an issue - but if each possible way of analysing the data produces the same story, probably don't need to worry too much.
- Key is to be transparent in the write-up (did you transform the data? If so, how? What distribution do you assume your data come from?). **Most importantly, publicly archive your data and analysis code so it can be examined by others.**

General R Tips

- Restart R whenever you start a new analysis - and create a new Project for each analysis - you don't want old variable names clogging up your workspace.
- Make sure you remember to install the library packages you need. Remember to check for updates!
- Chances are you are making a mistake related to syntax, capitalisation, or trying to run a package you haven't installed/updated.
- If you get really stuck, look at some of the R advice forums such as on stackoverflow.com

Further Reading

Baayen, R.H., Davidson, D.J., Bates, D.M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language, 59,* 390-412.

Barr, D.J., Levy, R., Scheepers, C., & Tilly, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language, 68,* 255–278.

Kliegl, R., Masson, M. E. J., and Richter, E. M. (2010). A linear mixed model analysis of masked repetition priming. *Visual Cognition, 18,* 655–681.

Lo, S., and Andrews, S. (2015). To transform or not to transform: using generalized linear mixed models to analyse reaction time data. *Frontiers in Psychology,* 6:1171.

Mirman, D. (2014). *Growth curve analysis and visualization using R*. New York, NY: CRC Press.

Winter, B. (2013). Linear models and linear mixed effects models in R with linguistic applications. arXiv:1308.5499. [http://arxiv.org/pdf/1308.5499.pdf]