Regression 1: Linear Regression

Marco Baroni

Practical Statistics in R

Outline

Outline

Outline

The general setting

- In many many many research contexts, you have a number of measurements (variables) taken on the same units
- You want to find out whether the distribution of a certain variable (response, dependent variable) can be, to a certain extent, predicted by a combination of the others (explanatory, independent variables), and how the latter are affecting the former
- We look first at case in which response is continuous (or you can reasonably pretend it is)
- Simple but extremely effective model for such data is based on assumption that response is given by *weighted sum* of explanatory variables, plus some random noise (the error term)
- We must look for a good setting of the weights, and at how well the weighted sums predict observed response distribution (the *fit* of the model)

The linear model

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_n x_{1n} + \epsilon_1 \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_n x_{2n} + \epsilon_2 \\ &\cdots \\ y_m &= \beta_0 + \beta_1 x_{m1} + \beta_2 x_{m2} + \cdots + \beta_n x_{mn} + \epsilon_m \end{aligned}$$

The matrix-by-vector multiplication view

$$\vec{y} = \mathbf{X}\vec{\beta} + \vec{\epsilon}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \times \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \cdots \\ \beta_n \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdots \\ \epsilon_m \end{pmatrix}$$

The matrix-by-vector multiplication view

$$\vec{y} = \mathbf{X}\vec{\beta} + \vec{\epsilon}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{pmatrix} = \beta_0 \begin{pmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{pmatrix} + \beta_1 \begin{pmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{m1} \end{pmatrix} + \beta_2 \begin{pmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{m2} \end{pmatrix} + \cdots + \beta_n \begin{pmatrix} x_{1n} \\ x_{2n} \\ \cdots \\ x_{mn} \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdots \\ \epsilon_m \end{pmatrix}$$

The linear model

- Value of continuous response is given by weighted sum of explanatory continuous or discrete variables (plus error term)
- Simplified notation:

$$y = \beta_0 + \beta_1 \times \mathrm{x}_1 + \beta_2 \times \mathrm{x}_2 + ... + \beta_n \times \mathrm{x}_\mathrm{n} + \epsilon$$

- The intercept β_0 is the "default" value of the response when all explanatory variables are set to 0 (often, not a meaningful quantity by itself)
- Steps of linear modeling:
  - Construct model
  - Estimate parameters, i.e., the β weights and the variance of the error term ϵ (assumed to be normally distributed with mean 0)
  - Look at model fit, check for anomalies, consider alternative models, evaluate predictive power of model...
  - Think of what results mean for your research question

Outline

Choosing the independent variables

- Typically, you will have one or more variables that are of interest for your research
- plus a number of "nuisance" variables you should take into account
- E.g., you might be really interested in the effect of colour and shape on speed of image recognition, but you might also want to include age and sight of subject and familiarity of image among the independent variables that might have an effect
- General advice: it is better to include nuisance independent variables than to try to build artificially "balanced" data-sets
- In many domains, it is easier and more sensible to introduce more independent variables in the model than to try to control for them in an artificially dichotomized design
  - Free yourself from stiff ANOVA designs!
  - As usual, with moderation and commonsense

Choosing the independent variables

- Measure the correlation of independent variables, and avoid highly correlated variables
  - Use a chi-square test to compare categorical independent variables
- Intuitively, if two independent variables are perfectly correlated you could have an infinity of weights assigned to the variables that would lead to exactly the same response predictions
- More generally, if two variables are nearly interchangeable you cannot assess their effects separately
- Even if no pair of independent variables is strongly correlated, one variable might be highly correlated to a linear combination of all the others ("collinearity")
  - With high collinearity, the fitting routine will die

Choosing the independent variables

- How many independent variables can you get away with?
- If you have as many independent variables as data points, you are in serious trouble
- The more independent variables, the harder it is to interpret the model
- Various techniques for variable selection: more on this below, but always keep core modeling questions in mind (does a model with variables X, Y and Z make sense?)

Dummy coding of categorical variables

- Categorical variables with 2 values coded by a single 0/1 term
- E.g., male/female distinction might be coded by term that is 0 for male subjects and 1 for females
- Weight of this term will express (additive) difference in response for female subjects
  - E.g., if response is reaction times in milliseconds and weight of term that is set to 1 for female subjects is -10, model predicts that, all else being equal, female subjects take 10 less milliseconds than males to respond
- Multi-level categorical factors are split into $n-1$ binary (0/1) variables
  - E.g., from 3-valued "concept class" variable (animal, plant, tool) to:
    - is animal? (animal=1; plant=0; tool=0)
    - is plant? (animal=0; plant=1; tool=0)
- Often, choosing sensible "default" level (the one mapped to 0 for all binary variables) can greatly improve the qualitative analysis of the results

Interactions

- Suppose we are testing recognition of animals vs. tools in males and females, and we suspect men recognize tools faster than women
- We need a male-tool interaction term (equivalently, female-animal, female-tool, male-animal), created by entering a separate weight for the product of the male and tool dummy variables:

$$y = \beta_0 + \beta_1 \times \text{male} + \beta_2 \times \text{tool} + \beta_3 \times (\text{male} \times \text{tool}) + \epsilon$$

- Here, β_3 will be added only in cases in which a male subject sees a tool (both male and tool variables are set to 1) and will account for any differential effect present when these two properties co-occur
- Categorical variable interactions are the easiest to interpret, but you can also introduce interactions between categorical and continuous or two continuous variables

Pre-processing

- Lots of potentially useful transformations I will skip
- E.g., take logarithm of response and/or of some explanatory variables
- Center variable so that mean value will be 0, scale it (these operations will not affect fit of model, but they might make the results easier to interpret)
- Look at documentation for R's `scale()` function

Outline

Estimation (model fitting)

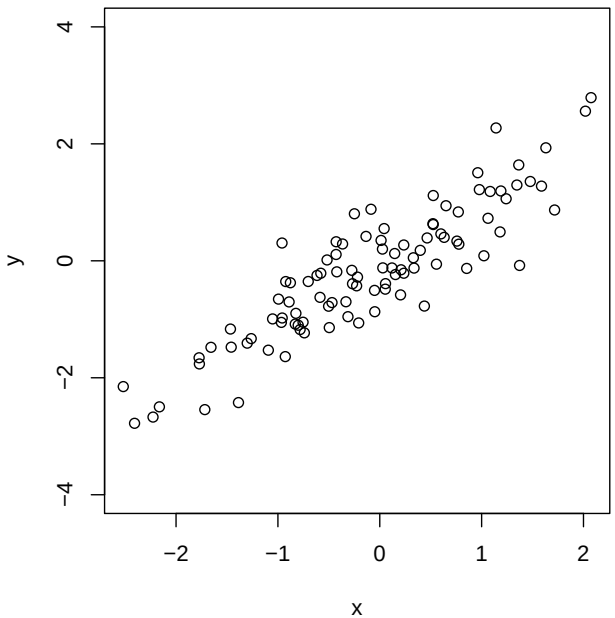
- The linear model:

$$y = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + ... + \beta_n \times x_n + \epsilon$$

- ϵ is normally distributed with mean 0
- We need to estimate (assign values) to the β weights and find the standard deviation σ of the normally distributed ϵ variable
- Our criterion will be to look for β's that minimize the error terms
- Intuitively, the smaller the ϵ's, the better the model

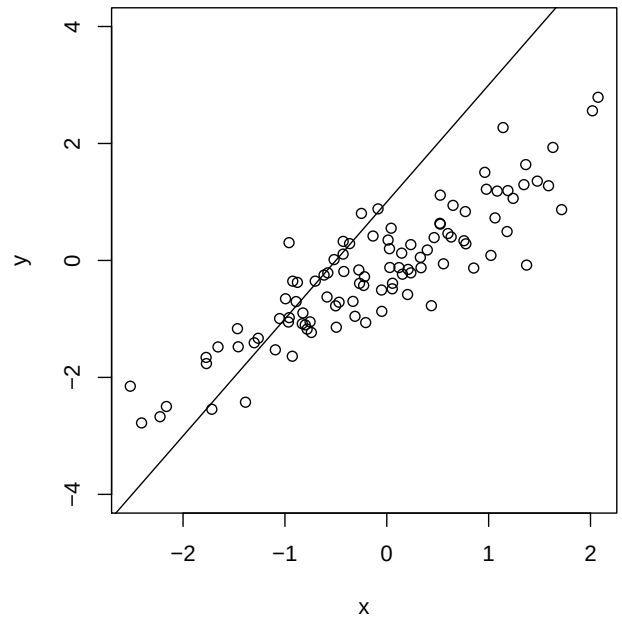
Big and small ϵ's

Some (unrealistically neat) data

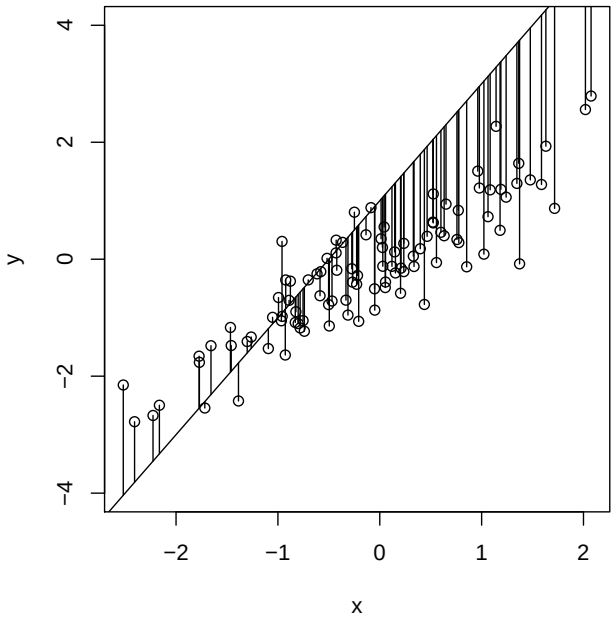


Big and small ϵ's

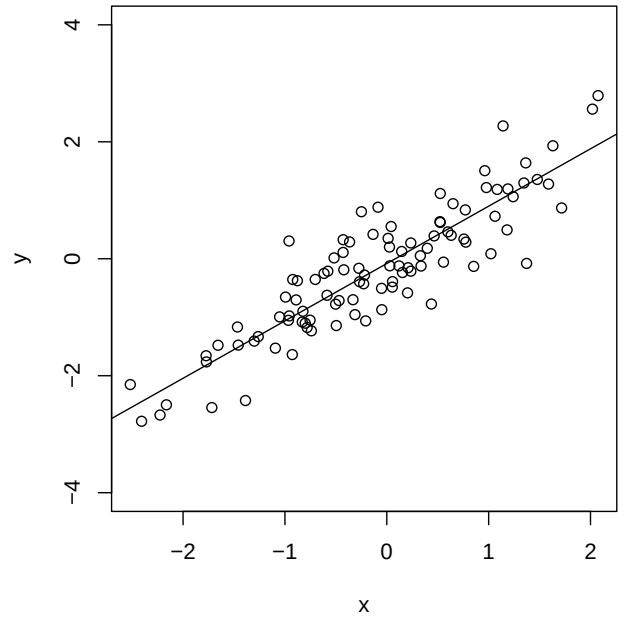
Bad fit



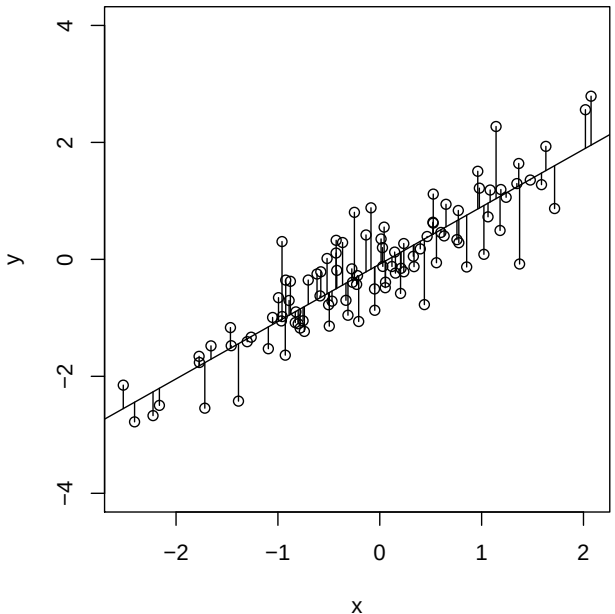
Big and small ϵ's
Large ϵ's



Big and small ϵ's
Good fit



Big and small ϵ's
(Relatively) small ϵ's



Minimizing the sum of square errors
The method of least squares

- Rewrite:

$$y = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + ... + \beta_n \times x_n + \epsilon$$

- as:

$$\epsilon = y - (\beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + ... + \beta_n \times x_n)$$

- Sum of ϵ across rows (data points) should be zero (why?)
- However, we can minimize sum of squared ϵ's
- I.e., pick β's such that the sum of squared ϵ's, computed as above, takes its minimum value

Minimizing the sum of square errors

- There is a closed-form expression to find the β's that minimize the sum of squared errors
- This means that we have an easy-to-compute formula to find the optimal β's, no need to look for them by trial-and-error
- (Incidentally, the β's satisfying the least square criterion are also the maximum likelihood estimates, i.e., the estimates that make the seen data maximally likely (ignore this for now if it doesn't make sense to you))
- Once we estimate the β's, we can also find the parameter σ expressing the standard deviation of the error term

Standard errors of the β estimates

- The standard errors of the β's (indicating the margin of uncertainty in our estimates) depend on:
  - the number of data points (more data points, more precise estimates)
  - the variance of the corresponding independent variable x_i (the larger the range of variation of the independent variable, the more precise the β_i estimate)
  - σ (the more unexplained error we are left with, the less certain we are about our β estimates)

Outline

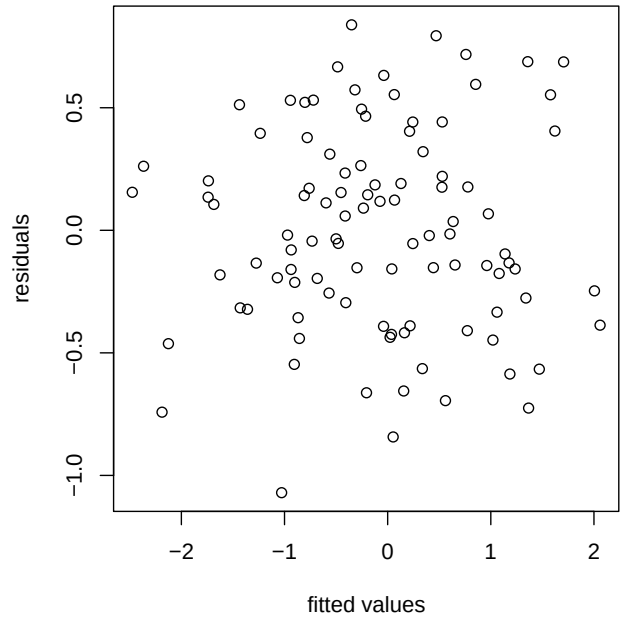
Sanity checks after fitting

- Plenty of sanity checks we might want to perform after estimating a model
- See, e.g., Baayen, Gelman/Hill, Dalgaard, ...

Residual plots

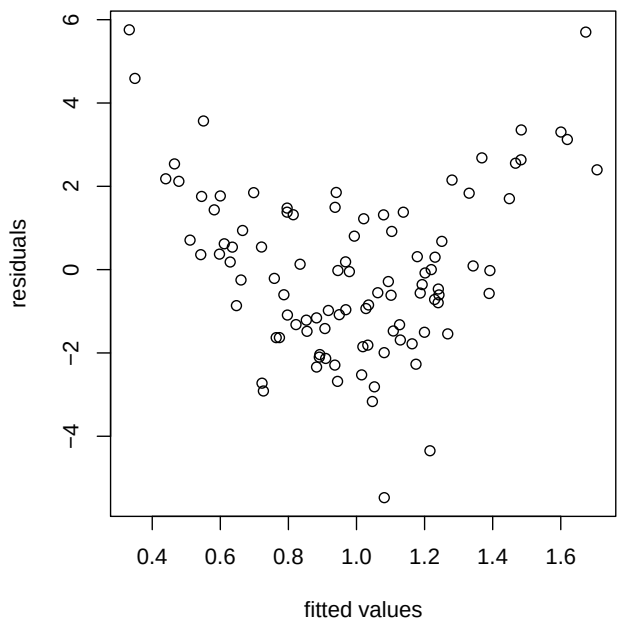
- A pretty easy and potentially revealing diagnostic looks at the distribution of residuals over the range of fitted values
  - Fitted values: the *y*'s predicted by the model for the data points in the set
  - Residuals: the attested *y*'s minus the predicted *y*'s, i.e., the empirical error terms in the estimated model
- If residuals do not look random across fitted values, our modeling assumption that response can be explained by a weighted sum of the explanatory variables plus random noise does not hold (noise is not random)
- What to do in case?
  - Ignore the problem (a common strategy)
  - Transform the response (e.g., take the logarithm, square root. . .) or the independent variables (e.g., square one of the variables)
  - Try other forms of analyses (e.g., could the data be more appropriate for an ordinal logistic regression?)
  - etc.

Good-looking residuals



Very suspicious-looking residuals

How could you try to fix this?



Goodness of fit

- How well does your model fit the data?
- How much of the variance in the *y*'s does the model "explain"?
- Variance in the unmodeled *y*'s:

$$s_y^2 = \frac{\sum_i (\mu_y - y_i)^2}{m - 1}$$

- Variance of residuals ("what is left to get from the model-predicted line to the actual *y*'s"):

$$\sigma^2 = \frac{\sum_i (y_i - (\beta_0 + \beta_1 \mathrm{x_{i1}} + \beta_2 \mathrm{x_{i2}} + \cdots + \beta_n \times \mathrm{x_{in}}))^2}{m - n}$$

- R^2, the standard measure of fitness, is given by 1 - the ratio of residual to response variance: $R^2 = 1 - (\sigma^2 / s_y^2)$
- R^2 ranges from 0 (no variance explained) to 1 (perfect fit)
- NB: this statistic is called *adjusted* R^2 in the R linear model output

Interpreting the β's

- Positive β's correspond to positive effects on the dependent variable, vice versa for negative β's: the higher $|\beta/s.e.(\beta)|$, the stronger the corresponding effect
- If a β coefficient is 0 (i.e., the corresponding explanatory variable has no effect whatsoever on the response), then $\beta/s.e.(\beta)$ has a *t* distribution with $m - n$ degrees of freedom (m: number of data points; n: number of weights to estimate)
- In "quick and dirty" terms, if $\beta \pm 2 \times s.e.(\beta)$ crosses 0, the corresponding explanatory variable (or interaction) is not having a significant effect on the response at $\alpha = 0.05$, given the current model

Comparing nested models

- Does it help to include variable x_i (that might be an interaction), when we already include $x_0, \ldots, x_{i-1}$? (the "subset" model is said to be "nested" in the model with extra-variable(s))
- Adding a variable will typically improve the fit (just because we have an extra weight to play around with) – but is the fit improvement good enough to justify complicating the model? Or is it just in line with what we could expect by chance?

Comparing nested models

- We assess the improvement by comparing the sum of squared residuals of the smaller and larger models ($SS = \sum_i (y_i - \vec{x}_i\beta)^2$):

$$F = \frac{SS_{smaller} - SS_{larger}/(n_{larger} - n_{smaller})}{SS_{larger}/(m - n_{larger})}$$

- Intuitions:
  - The larger the difference in squared residuals between the smaller and the larger model, the stronger the impact of the extra terms
  - The more extra terms we added, the less surprising it is that the difference in squared residuals increased
- Significance of F value can be found in the $F_{n_{larger}-n_{smaller}, m-n_{larger}}$ table you might be familiar with from ANOVA (not by chance, in R we will perform this sort of comparison with the `anova()` command)

Comparing nested models

- Multiple F-tests comparing increasingly larger models (and/or increasingly smaller models) can be used for incremental model building
- However, it is all too easy to get carried away with this
- Don't forget that the model is indeed a "model" of the phenomenon you are studying: it should make sense from a qualitative point of view
- Some practical advice:
  - Start with a model with the constant (intercept) term and the important nuisance terms: keep these in any case
  - When you add an interaction, always keep the corresponding single variable terms
  - If a set of independent variables constitute a meaningful cluster (e.g., the variables that measure different aspects of colour: hue, saturation and brightness), keep them all in the model

Output statistics: summary

- While experimenting with nested models, look at the F value of increasingly complex models to decide whether the increase in complexity is justified
- Once you settle on a model, look at
  - R^2 for the overall fit of the model
  - the β's and their standard errors (and the corresponding t values) to assess the separate impact of the explanatory variables in the final model
- NB: it is easy to get confused between F and t values: the former compare different regression models; the latter look at the impact of a single variable in a specific model

Prediction

- Ultimately, we hope to have a model that will predict responses in the "population" of interest, not only in our sample
  - E.g., we want to draw conclusions about how various factors influence image perception in women and men in general, not only in the data collected for our experiment
- The more parameters we have, the less data we have, the less "truly random" our sample is...
- the more we run the risk of "overfitting": estimating a model that is adapted to idiosyncratic aspects of our data that we would not encounter in other samples from the population we are interested in

Prediction

- While statistical inference will give us an idea of how stable our results should be across similar experiments, it does not shield us from the risk that there is something really idiosyncratic about some of our data, and that's what our model "learned" to simulate
  - Statistical inference simply tells us how well our model would generalize to more data with the same idiosyncrasies!
- The ideal test for the model would be how well it predicts data that have not been used for estimation
  - This is the standard division between "training" and "test" data in machine learning, where emphasis shifts almost entirely on prediction of unseen data
- Modern validation techniques such as bootstrapping or n-fold cross-validation estimate the model on a subset of the data, and use the remaining part to evaluate it, iterating the process with different data used as estimation and evaluation sets

Bootstrap validation

- In bootstrap validation, we randomly resample m data points from our original m data points *with replacement*
  - I.e., at each step all data points have the same probability of being sampled, including those that we have already sampled in a previous step
- The estimation set has same number of data points m as original data-set, but contains repeated data points and misses some original data points
- We fit model to estimation set, but measure its goodness-of-fit on the original data-set (that will contain data points not used for estimation)
- We repeat this process many times, and we use averages of R^2 (or any other statistics) across the estimation and full sets to assess how well the model generalizes

Outline

Outline

The `english` data

- Load Baayen's `languageR` library, that in turn loads various useful packages and data-sets:

```
> library(languageR)
```

- We load and attach Baayen's `english` data-set:

```
> data(english)
> ?english
> summary(english)
> attach(english)
```

The `english` data

- We will only look at a few variables from this data-set, namely Familiarity, WordCategory, and WrittenFrequency
- Suppose that our theory of familiarity (Familiarity) predicts that nouns feel in general more familiar than verbs (WordCategory); we want to test this while taking into account the obvious fact that familiarity is affected by the frequency of usage of the word (WrittenFrequency)
- We will thus use only two independent variables (and their interaction), but in a real life regression you should look at the potential impact of many more variables

Preliminary checks

- The dependent variable looks reasonably "continuous":

```
> summary(Familiarity)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.100   3.000   3.700   3.796   4.570   6.970
```

- A look at the distribution of WrittenFrequency (or at the documentation!) reveals that these are log frequencies:

```
> summary(WrittenFrequency)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   3.761   4.832   5.021   6.247  11.360
```

- And we have so many nouns and so many verbs:

```
> summary(WordCategory)
    N    V
 2904 1664
```

Preliminary checks

- A sanity t-set suggests that WrittenFrequency and WordCategory are not so strongly related as to worry about using them as part of the same regression:

```
> t.test(WrittenFrequency[WordCategory=="V"],
  WrittenFrequency[WordCategory=="N"])
...
t = 1.0129, df = 3175.109, p-value = 0.3112
...
sample estimates:
mean of x mean of y
 5.058693  4.999629
```

Outline

The linear model in R

- We estimate the model using the R "formula" notation:
  `dependent ~ indep1 + indep2 + ...`
- In our case:

```
> modelFrequencyCategory<-
  lm(Familiarity~WrittenFrequency+WordCategory)
```

- `lm()` (for **l**inear **m**odel) returns an R object that contains various information about the fitted model
  - Other functions, such as `t.test()`, also return an object, except that in that case we rarely need to store the information for further post-processing and analysis
- We store the object produced by fitting the regression in the modelFrequencyCategory variable

A look at the estimated model with `summary()`

```
> summary(modelFrequencyCategory)

Call:
lm(formula = Familiarity ~ WrittenFrequency + WordCategory)

Residuals:
     Min       1Q   Median       3Q      Max
-2.43148 -0.45094 -0.04207  0.41231  2.71238

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.225880   0.030547   40.13   <2e-16
WrittenFrequency 0.492205   0.005546   88.75   <2e-16
WordCategoryV    0.269772   0.021243   12.70   <2e-16

Residual standard error: 0.6909 on 4565 degrees of freedom
Multiple R-Squared: 0.6388, Adjusted R-squared: 0.6387
F-statistic:  4037 on 2 and 4565 DF,  p-value: < 2.2e-16
```

The summary line by line

- The model we specified (a useful reminder when we start having many variables containing lm objects):

```
Call:
lm(formula = Familiarity ~ WrittenFrequency + WordCategory)
```

- Distribution of residuals should be normal with mean 0 (and thus medians should not be far from 0):

```
Residuals:
     Min       1Q   Median       3Q      Max
-2.43148 -0.45094 -0.04207  0.41231  2.71238
```

- This looks reasonably symmetrical and centered reasonably near 0

The summary line by line

- The β coefficient estimates and their standard errors:

```
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.225880   0.030547   40.13   <2e-16
WrittenFrequency 0.492205   0.005546   88.75   <2e-16
WordCategoryV    0.269772   0.021243   12.70   <2e-16
```

- The intercept (β_0) is significantly different from 0, which is not a particularly interesting fact
  - A noun with log frequency 0 (frequency 1) should get on average a 1.22588 familiarity rating
- Not surprisingly, frequency has a significantly positive impact on familiarity
  - Notice that $0.492205/0.005546 = 88.75$

The summary line by line

- The β coefficient estimates and their standard errors:

```
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.225880   0.030547   40.13   <2e-16
WrittenFrequency 0.492205   0.005546   88.75   <2e-16
WordCategoryV    0.269772   0.021243   12.70   <2e-16
```

- WordCategory has been recoded as the dummy variable WordCategoryV (1 for verbs, 0 for nouns), thus we get a β for the verbs, nouns are the "default" level
- Interestingly, WordCategoryV has a significantly positive effect, i.e., all else being equal, verbs "feel" more familiar than nouns

Changing the default level

- In this case, with only two levels of WordCategory, there is no particular reason to pick nouns or verbs as the default, but in other cases picking the right default level can simplify the interpretation of the results
- Levels are ordered alphabetically by default, use `relevel()` to pick a different default; Try:

```
> levels(WordCategory)
> WordCategoryBis<-relevel(WordCategory,"V")
> levels(WordCategoryBis)
> summary(lm(Familiarity~WrittenFrequency+
  WordCategoryBis))
```

- What changed?

The summary line by line

- Next line gives standard error of residuals:

```
Residual standard error: 0.6909 on 4565 degrees of freedom
```

- And then we get the R^2 and adjusted R^2 value:

```
Multiple R-Squared: 0.6388, Adjusted R-squared: 0.6387
```

R^2 and adjusted R^2

- The R^2 value is 1 minus the ratio of the variance of the residuals to the variance of the original variable (note the `residuals()` function):

```
> 1-(var(residuals(modelFrequencyCategory))/
  var(Familiarity))
[1] 0.6388438
```

- The adjusted R^2 corrects the variance estimates for the number of parameters (weights) that have been estimated (penalizing for the fact that the more parameters we have to play with, the easier it is to fit the data):

```
> correctedResVar<-
  (sum((residuals(modelFrequencyCategory)
  -mean(residuals(modelFrequencyCategory)))^2)/
  (length(Familiarity)-3))
> 1-(correctedResVar/var(Familiarity))
[1] 0.6386856
```

R^2 and adjusted R^2

- Notice that the square root of the corrected residual variance is the residual standard error reported in the summary:

```
> sqrt(correctedResVar)
[1] 0.6908534
```

- Relation between unadjusted and adjusted R^2:

```
> unadjustedRsquared<-1-
  (var(residuals(modelFrequencyCategory))/
  var(Familiarity))
> m<-length(Familiarity)
> n<-3
> 1-((1-unadjustedRsquared)*((m-1)/(m-n-1)))
[1] 0.6386064
```

R^2 and adjusted R^2

- As $(m-1)/(m-n-1)$ approaches 1 (because m, the number of data points, is very high, and/or n, the number of weights, is very low) the unadjusted and adjusted R^2 become virtually identical
- A rule-of-thumb I've read somewhere on the Web: if unadjusted and adjusted R^2 differ by more than 5%, you should worry that you are using too many dependent variables for the amount of data you've got

The summary line by line

- Finally, we see a F value:

```
F-statistic:  4037 on 2 and 4565 DF,  p-value: < 2.2e-16
```

- This is a test for the hypothesis that our model is significantly better at explaining variance than an "empty" model that predicts the dependent variable mean for all data points: not a terribly interesting result
- Different from the F test we use in step-wise model building, as discussed above and illustrated below

Interactions

- Is there an interaction between the effect of word frequency and the one of word category?

```
> modelInteraction<-
  lm(Familiarity~WrittenFrequency+WordCategory
  +WrittenFrequency:WordCategory)
```

- Equivalent shorthand notation I don't particularly like:

```
> modelInteraction<-
  lm(Familiarity~WrittenFrequency*WordCategory)
```

- What does the `summary()` tell us?

Comparing nested models with `anova()`

- Incremental model construction confirms what we found out by looking at the significant β coefficients in the largest model, i.e., we should keep both the frequency and category models in the model, whereas we don't have strong evidence for an effect of their interaction:

```
> modelFrequency<-lm(Familiarity~WrittenFrequency)
> modelCategory<-lm(Familiarity~WordCategory)

> anova(modelFrequency,modelFrequencyCategory)
> anova(modelCategory,modelFrequencyCategory)
> anova(modelFrequencyCategory,modelInteraction)
```

Comparing nested models with `anova()`

- Recall that we compare models computing the F statistic as follows:

$$F = \frac{SS_{smaller} - SS_{larger}/(n_{larger} - n_{smaller})}{SS_{larger}/(m - n_{larger})}$$

  with $n_{larger} - n_{smaller}$ and $m - n_{larger}$ degrees of freedom

- Let us compute the relevant values for the comparison of modelFrequencyCategory and modelInteraction

Comparing nested models with `anova()`

- Sum of squared residuals for the larger and smaller models:

```
> ss_smaller<-sum(residuals(modelFrequencyCategory)^2)
> ss_larger<-sum(residuals(modelInteraction)^2)
```

- Total data points, parameters estimated in the two models:

```
> m<-length(Familiarity)
> n_smaller<-length(modelFrequencyCategory$coefficients)
> n_larger<-length(modelInteraction$coefficients)
```

- The F-statistic:

```
> myF<-((ss_smaller-ss_larger)/(n_larger-n_smaller))
       /(ss_larger/(m-n_larger))
```

- The corresponding p-value:

```
1-pf(myF,n_larger-n_smaller,m-n_larger)
```

- Compare the results (and also the various quantities we compute along the way) with the output of `anova()`

What to report

- After you picked the model you want to focus on, did all the relevant checks, gave some serious thought to your statistical model in the light of your theoretical model, etc., you might consider reporting the following results in your paper:
  - The number of data points used to estimate the model
  - The model (mention also if you explored other smaller or larger models via F-tests for nested models)
  - The β values with their standard errors (marking the significant ones)
  - The R^2 or adjusted R^2 value (assuming they are reasonably similar)

Outline

A further look at the lm object

- The lm object stores many goodies – you can see the full list in the Value section of the `lm()` documentation
- Among other things, you can extract coefficients (weights), residuals and fitted values:

```
> coefficients(modelFrequencyCategory)
> modelFrequencyCategory$coefficients
> modelFrequencyCategory$coefficients[1]
> head(fitted.values(modelFrequencyCategory))
> head(modelFrequencyCategory$fitted.values)
> head(residuals(modelFrequencyCategory))
> head(modelFrequencyCategory$residuals)
```

- Use these data to make a plot of residuals by fitted values
- How does it look?

Practice with the lm output

- Use the fitted values and Familiarity to create a residuals vector; check that it is (nearly) identical to `residuals(modelFrequencyCategory)` (small differences might be due to rounding errors)
- Create the WordCategoryV vector:

```
> WordCategoryV<-sapply(WordCategory,
  function(x){if (x=="V"){1}else{0}})
```

- Now, use the coefficients from the model and the WrittenFrequency and WordCategoryV vectors to recreate the fitted values; then compare the ones obtained this way with the fitted values stored in the lm object

The fitted values by matrix multiplication

```
> modelFrequencyCategory<-
  lm(Familiarity~WrittenFrequency+WordCategory,
  x=T,y=T)
> myFitted<-
  modelFrequencyCategory$x %*%
  modelFrequencyCategory$coefficients
```

Plotting the model lines

- This is relatively straightforward here, since we have only one continuous and one two-level discrete independent variables

```
# plot familiarity in function of frequency separately
# for nouns and verbs
> plot(WrittenFrequency[WordCategory=="N"],
  Familiarity[WordCategory=="N"],col="red",pch=1,cex=.5)
> points(WrittenFrequency[WordCategory=="V"],
  Familiarity[WordCategory=="V"],col="green",pch=1,cex=.5)

# the noun line: beta0 + beta1*frequency
> abline(a=modelFrequencyCategory$coefficients[1],
  b=modelFrequencyCategory$coefficients[2],col="red")

# the verb line: beta0 + beta2 + beta1*frequency
> vIntercept<-modelFrequencyCategory$coefficients[1]+
  modelFrequencyCategory$coefficients[3]
> abline(a=vIntercept,
  b=modelFrequencyCategory$coefficients[2],col="green")
```

Plotting the lines with the interaction

```
# plot familiarity in function of frequency separately
# for nouns and verbs
> plot(WrittenFrequency[WordCategory=="N"],
  Familiarity[WordCategory=="N"],col="red",pch=1,cex=.5)
> points(WrittenFrequency[WordCategory=="V"],
  Familiarity[WordCategory=="V"],col="green",pch=1,cex=.5)

# the noun line: beta0 + beta1*frequency
> abline(a=modelInteraction$coefficients[1],
  b=modelInteraction$coefficients[2],col="red")

# the verb line: beta0 + beta2 + (beta1+beta3)*frequency
> vIntercept<-modelInteraction$coefficients[1]+
  modelInteraction$coefficients[3]
> vSlope<-modelInteraction$coefficients[2]+
  modelInteraction$coefficients[4]
> abline(a=vIntercept,b=vSlope,col="green")
```

Prediction

- We need to use a different function (part of the `Design` library, that was loaded by `languageR`) to fit a linear model that then we can use with the `validate()` function (also part of the `Design` library):

```
> modelFrequencyCategory<-
  ols(Familiarity~WrittenFrequency+WordCategory,
  x=TRUE,y=TRUE)
```

- The *x* and *y* arguments tell the function to store the original independent variable matrix (*x*) and dependent variable values (*y*) in the resulting object; these are necessary to perform the validation experiments
- We repeat the bootstrapping procedure 1,000 times:

```
> validate(modelFrequencyCategory,B=1000)
```

Prediction

- We only look at the bootstrap estimates of R^2
- Different simulations will produce slightly different results, but the `training` column reports the average R^2 computed on the same bootstrapped sets used for estimation whereas `test` reports the average R^2 of the bootstrapped models as computed on the whole data-set
- The `optimism` column is the difference between these two values, and the `index.corrected` is the original R^2 minus this optimism score
- The larger the optimism (or, equivalently, the larger the difference between original and corrected R^2), the more the model is overfitting the training data, and the worse it will be at predicting unseen data

Outline

Back to the Elena Nava's cochlear implant data

- Fields in the `cochlear-1.txt` file

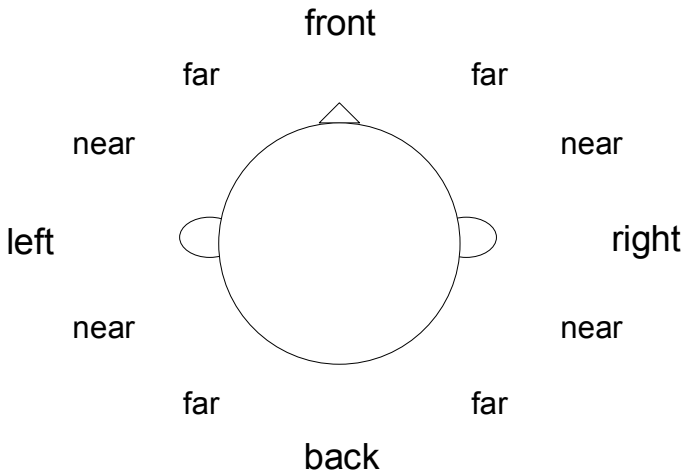
source	mono or bilateral
session	1st, 2nd, 3d or 4th meeting with the subject
stimulus	position of stimulus wrt subject, in degrees
response	response, in degrees
error	absolute difference, in degrees, between stimulus and response

- For stimulus and response, possible values are 30, 60, 120, 150 (source on the right, degrees measured clockwise from 0 in front of subject), -30, -60, -120, -150 (source on the left, negative degrees measures counterclockwise from 0 in front of subject)
- Possible error values: 0, 30, 60, 90, 120, 150, 180
- It is not entirely appropriate to treat this as a continuous dependent variable!

Decomposing the stimulus

- Possible stimulus values are 30, 60, 120, 150 (source on the right, degrees measured clockwise from 0 in front of subject), -30, -60, -120, -150 (source on the left, negative degrees measures counterclockwise from 0 in front of subject)
- Ears are at -90 and 90 degrees
- We can break down the stimulus into more primitive variables:
  - Is stimulus presented from left (< 0) or right (> 0)?
  - Is stimulus in front ($|stim| < 90$) or to the back ($|stim| > 90$) of subject?
  - Is stimulus "nearer" ($60 \leq |stim| \leq 120$) or further away from the ears?

Decomposing the stimulus



Decomposing the stimulus

```
> stimlr<-sapply(stimulus,
  function(x)if (x<0)"left"else"right")
> stimfb<-sapply(stimulus,
  function(x)if (abs(x)<90)"front"else"back")
> stimdist<-sapply(stimulus,
  function(x)if ((abs(x)==60) || (abs(x)==120))
  "near"else"far")

# we also scale the dependent variable to the 0-6 range
> normError<-error/30
```

Analyze!

- Try various linear models of `normErr` using (a selection of) the following independent variables: `source`, `session`, `stimlr`, `stimdist`, `stimfb`
  - You might get a few “variable ’X’ converted to a factor” warnings: to avoid them, wrap the variable creation commands above with the `as.factor()` function
- Look at the models’ fit, look at the β’s and think of what they mean, explore some interactions, compare nested models, relevel some variable (you will have to use `as.factor()` to be able to relevel the variables created above)
- Inspect the distribution of residuals (not very reassuring)
- Run a bootstrap validation