Chapter 1: Linear models

Big picture

This course builds on an understanding of the mechanics of linear models. Here we introduce some key topics that will facilitate future understanding hierarchical models.

Learning goals

- linear regression with `lm`
- intercepts, "categorical" effects
- varying model structure to estimate effects and standard errors
- interactions as variation in slope estimates for different groups
- centering input variables and interpreting resulting parameters
- assumptions and unarticulated priors
- understanding residual variance (Gaussian)
- understanding all of the above graphically
- understanding and plotting output of lm
- notation and linear algebra review: $X\beta$

Linear regression, ANOVA, ANCOVA, and multiple regression models are all species cases of general linear models (hereafter "linear models"). In all of these cases, we have observed some response variable y, which is potentially modeled as a function of some covariate(s) $x_1, x_2, ..., x_p$. As we shall see, it ultimately makes no difference whether the covariates are continuous, categorical, or ordinal. In fact, it even makes sense to think of a linear model with no covariates at all.

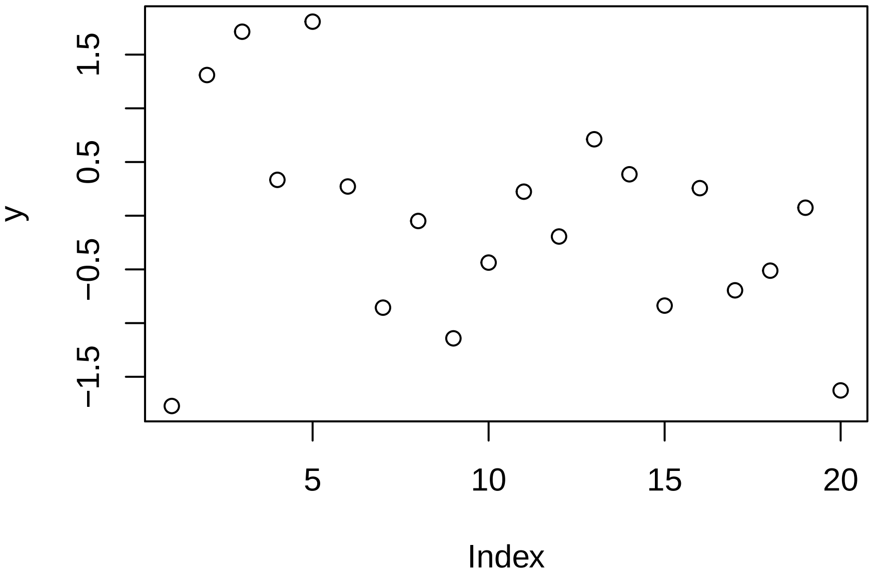
Model of the mean

Imagine a situation where you've been asked predict some measurement for a single individual from a population of individuals. You don't know anything else about this individual or the population. In fact, you don't even know what kind of thing the individual is, nor do you know what kind of measurement you are trying to estimate. What's your best guess? To give you a little help, imagine that instead of having just the measurement for your focal individual, you have measurements for a larger sample of n individuals. Now, it might make sense that your best guess for any given individual will be the average of measurement for all the individuals in the population. While this underlying average is completely unknown, you can get a decent idea of what the underlying population average is from your sample of individuals.

Statistically speaking, in this case we have no covariates (extra information about individuals) of interest. Instead, we are interested in estimating the population mean and variance of the random variable Y (our measurement of interest) based on n observations, corresponding to the values $y_1, ..., y_n$. Here, capital letters indicate the random variable, and lowercase corresponds to realizations of that variable. This model is sometimes referred to as the "model of the mean".

First, let's simulate our situation using the *rnorm* function in R by drawing 20 Y values from an underlying normal distribution with mean equal to zero and standard deviation equal to 1:

```r
# simulating a sample of 20 y values from a normal distribution
# with mean = 0 and standard deviation = 1
y <- rnorm(n = 20, mean = 0, sd = 1)
plot(y)
```



We have two parameters to estimate: the mean of Y, which we'll refer to as μ, and the variance of Y, which we'll refer to as σ^2. Here, and in general, we will use greek letters to refer to parameters. If its reasonable to think that Y is normally distributed, then we can assume that the realizations or samples y that we observe are also normally distributed: $y \sim N(\mu, \sigma^2)$. Here and elsewhere, the $\sim$ symbol represents that some quantity "is distributed as" something else (usually a probability distribution). You can also think of $\sim$ as meaning "is sampled from". A key concept here is that we are performing statistical inference, meaning we are trying to learn about (estimate) population-level parameters with sample data. In other words, we are not trying to learn about the sample mean $\bar{y}$ or sample variance of y. These can be calculated and treated as known once we have observed a particular collection of y values. The unknown quantities μ (the underlying mean of the population) and σ^2 (the underlying variance) are the targets of inference.

Fitting this model (and linear models in general) is possible in R with the `lm` function. For this rather simple model, we can estimate the parameters as follows:

```r
# fitting a model of the mean with lm
m <- lm(y ~ 1)
summary(m)
```

```
##
## Call:
## lm(formula = y ~ 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.72048 -0.67883  0.06428  0.39840  1.85860
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.05161    0.21960  -0.235    0.817
##
## Residual standard error: 0.9821 on 19 degrees of freedom
```

The summary of our model object `m` provides a lot of information. For reasons that will become clear shortly, the estimated population mean is referred to as the "Intercept". Here, we get a point estimate for the population mean μ: -0.052 and an estimate of the residual standard deviation σ: 0.982, which we can square to get an estimate of the residual variance σ^2: 0.965. So, with 20 individuals sampled we didn't to bad.

Linear regression

Now imagine that we have an additional measurement X made on our 20 individuals, and we believe that knowing something about an individual's value of X will give us a better estimate for that individual's Y than simply using the estimated population mean. Keep in mind that this X could be a simple categorical variable (think gender) or a continuous measurement. Either way, knowing X may help us better predict Y.

To put things more statistically, we are interested in estimating the mean of Y as a function of some other variable X. That is, for any given value of X, what is the underlying mean of the Y values for all individuals that share the same value for X. If X is a categorical variable, we are interested in estimating the mean Y value for each group denoted by all the possible values of X (e.g. males and females). If X is continuous, we are interested in estimating what the mean value of Y would be for each possible value of X. Standard linear regression typically falls in the latter category but in fact there is no conceptual difference between the two.

Simple linear regression assumes that y is again sampled from a normal distribution, but this time the mean or expected value of y is a function of x:

$$y_i \sim N(\mu_i, \sigma^2)$$

$$\mu_i = \alpha + \beta x_i$$

Here, subscripts indicate which particular value of y and x we're talking about. Specifically, we observe n pairs of values: $(x_i, y_i), ..., (x_n, y_n)$, with all x values known exactly. Note that in this case the expected mean value of y is shifted up or down relative to the global mean depending on the value of x, and that when x_i equals zero, the model collapses to the global mean (meaning that our estimate for individuals with a value of zero for their covariate should equal the overall population mean – more on this below).

Linear regression models can equivalently be written as follows:

$$y_i = \alpha + \beta x_i + \epsilon_i$$

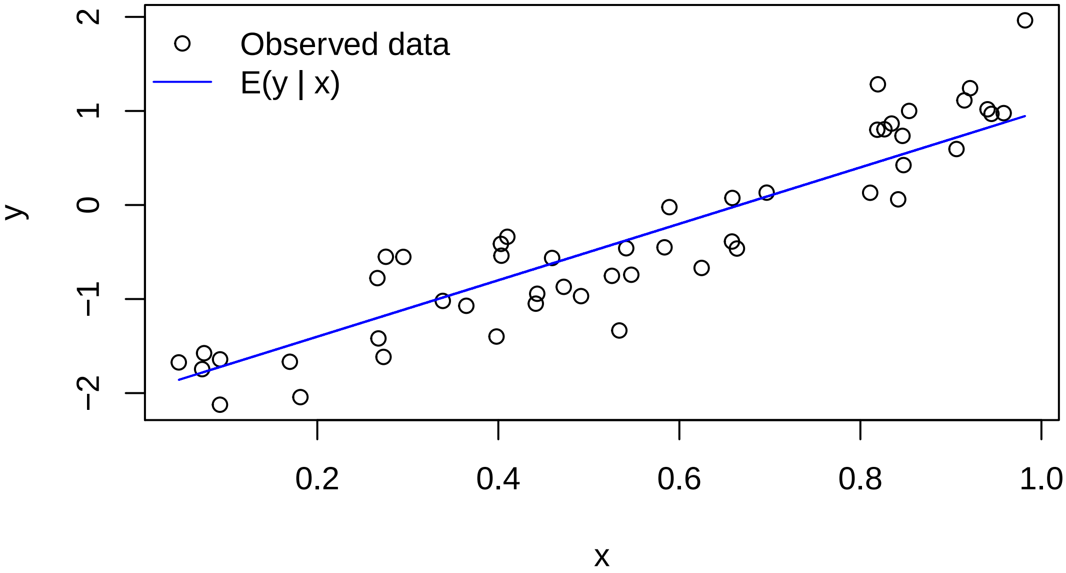
$$\epsilon_i \sim N(0, \sigma^2)$$

Key assumptions here are that each of the error terms $\epsilon_1, ..., \epsilon_n$ are normally distributed around zero with some variance (i.e., the error terms are identically distributed), and that the value of ϵ_1 does not affect the value of any other ϵ (i.e., the errors are independent). This combination of assumptions is often referred to as "independent and identically distributed" or i.i.d. Equivalently, given some particular x_i and a set of linear regression parameters, the distribution of y_i is normal. A common misconception is that linear regression assumes the distribution of y is normal. This is technically wrong - linear regression assumes that the error terms are normally distributed. The assumption that the variance σ^2 is constant for all values of x is referred to as homoskedasticity. Rural readers may find it useful to think of skedasticity as the amount of "skedaddle" away from the regression line in the y values. If the variance is changing across values of x, then the assumption of homoskedasticity is violated and you've got a heteroskedasticity problem.

Let's go ahead and simulate a larger data set of 50 individuals as well as some random X values drawn from a uniform distribution between 0 and 1 (i.e. a continuous covariate). We'll then make up some values for *alpha*, *beta*, and *sigma* so that we can plot a hypothetical relationship between x and y:

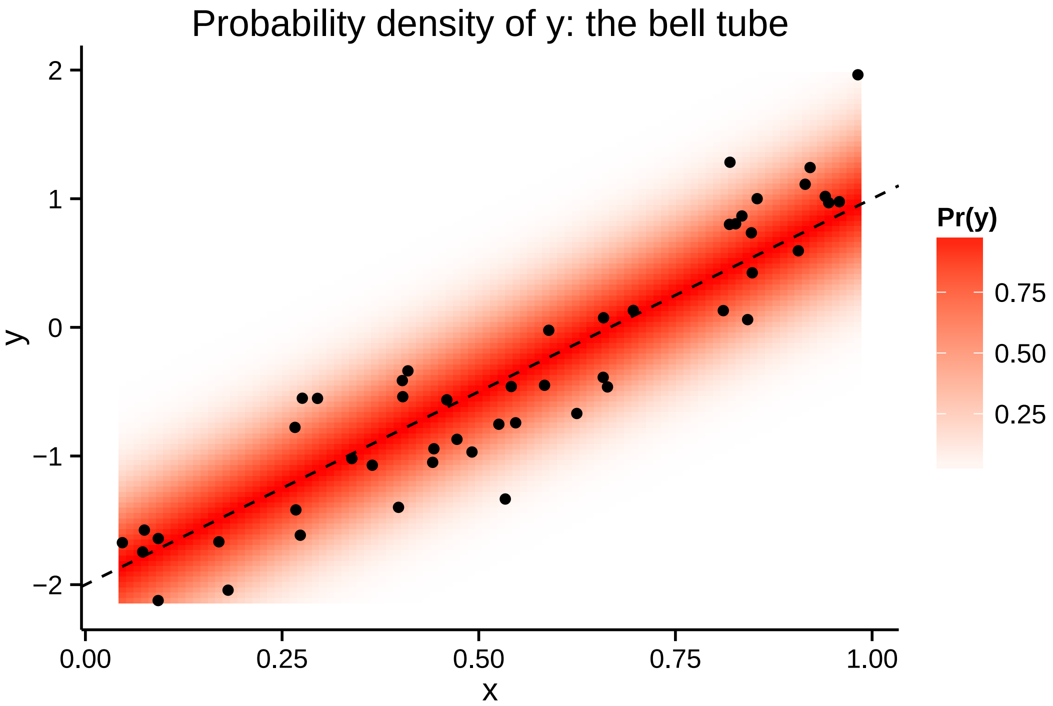
```r
# simulate x values
n <- 50
x <- runif(n, min = 0, max = 1)

# designate the underlying parameters for our hypothetical population
```

```r
alpha <- -2
beta <- 3
sigma <- .4

# simulate values of y based on our randomly generated values of x
# and our underlying parameters
y <- rnorm(n, mean = alpha + beta * x, sd = sigma)

# plot the values of x and y
plot(x, y)

# add known mean function
lines(x = x, y = alpha + beta * x, col='blue')
legend('topleft',
       pch = c(1, NA), lty = c(NA, 1),
       col = c('black', 'blue'),
       legend = c('Observed data', 'E(y | x)'),
       bty = 'n')
```



The normality assumption means that the probability density of y is highest at the value $\alpha + \beta x$, where the regression line is, and falls off away from the line according to the normal probability density. In other words, for any given value of x, our best guess for y is indicated by the fitted regression line, with the probably of other values for y falling off gradually directly above and below the regression line for that value of x

Graphically, this looks like a bell 'tube' along the regression line, adding a dimension along x to the classic bell 'curve'.



Model fitting

Linear regression parameters α, β, and σ^2 can be estimated with `lm`. The syntax is very similar to the previous model, except now we need to include our covariate `x` in the formula (the first argument to the `lm` function).

```
m <- lm(y ~ x)
summary(m)
```

```
## 
## Call:
## lm(formula = y ~ x)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.93197 -0.30831 -0.00562  0.26076  0.86905 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  -2.1845     0.1240  -17.62   <2e-16 ***
## x             3.3393     0.2018   16.55   <2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
## 
## Residual standard error: 0.3954 on 48 degrees of freedom
## Multiple R-squared:  0.8509, Adjusted R-squared:  0.8478 
## F-statistic: 273.8 on 1 and 48 DF,  p-value: < 2.2e-16
```

The point estimate for the parameter α is called “(Intercept)”. This is because our estimate for α is the y-intercept of the estimated regression line when $x = 0$. If you need convincing, recall that $y_i = \alpha + \beta x_i + \epsilon_i$.

If you substitue 0 in for x, the equation for why collapses down to $y_i = \alpha + \epsilon_i$, which is the original "model of the mean".

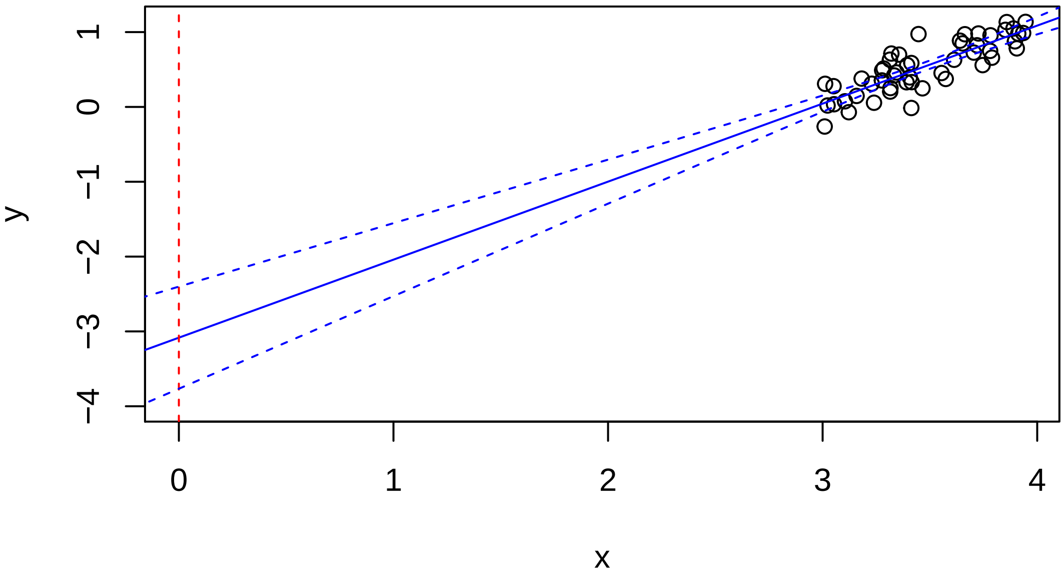
The estimate for β is called "x", because it is a coefficient associated with the variable "x" in this model. This parameter is often referred to as the "slope", because it represents the increase in the expected value of y for a one unit increase in x (the rise in y over run in x). Keep in mind, though, that *lm* does not care about the actual distribution of x you give it. The values of x could be all zeros and ones (or all -3.4 or 2.5) and lm would treat x the same way.

Point estimates for the standard deviation and variance of ϵ can be extracted as before (`summary(m)$sigma` and `summary(m)$sigma^2`).

Centering covariates

Often, it's a good idea to "center" covariates so that they have a mean of zero ($\bar{x} = 0$). This is achieved by subtracting the sample mean of a covariate from the vector of covariate values ($x - \bar{x}$).

If covariates are not centered, then it is common to observe correlations between estimated slopes and intercepts. Consider the following graph, which shows the the middle, lower, and upper range of the estimated slopes and intercepts using an uncentered covariate (the center is at x = 3.5):



Note that for the larger slope values (steeper dashed blue line), the estimated value for the intercept (where the dashed blue line crosses the dashed red line) is smaller, and vice versa for smaller slope values. So, we expect that in this case, the estimates for the intercept and slope must be negatively correlated.

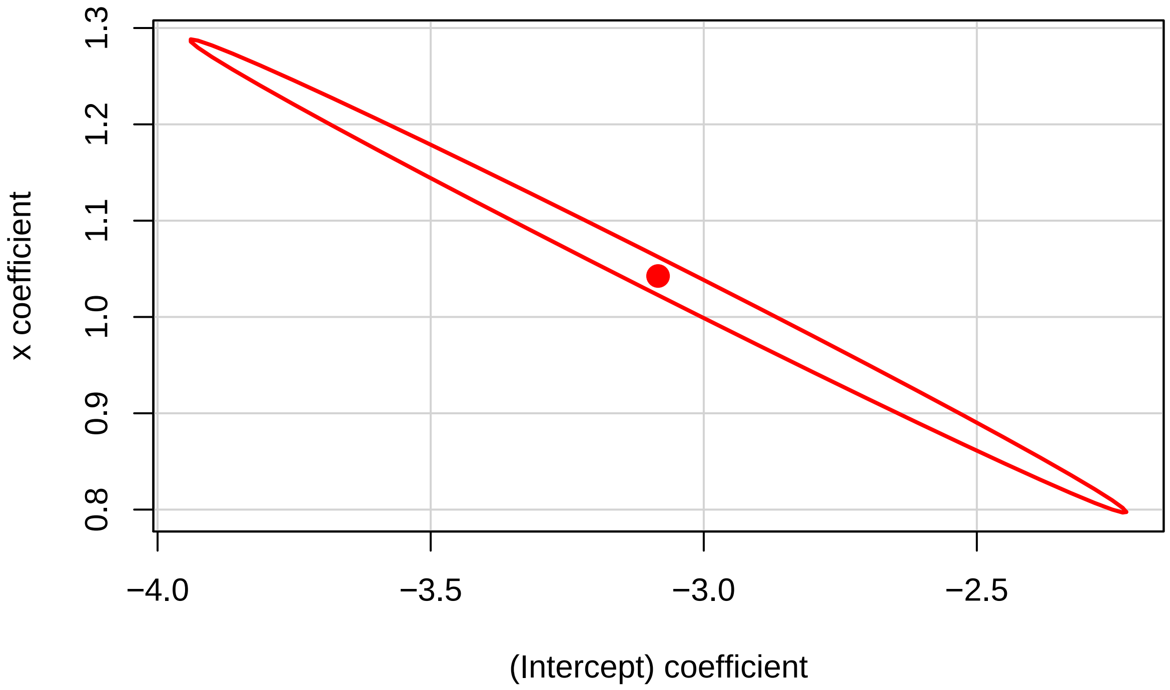
Usually, people inspect univariate confidence intervals for our estimates of α and β. , e.g.,

```
# calculate 95% confidence intervals for the parameters estimated in model m
confint(m, level = 0.95)
```

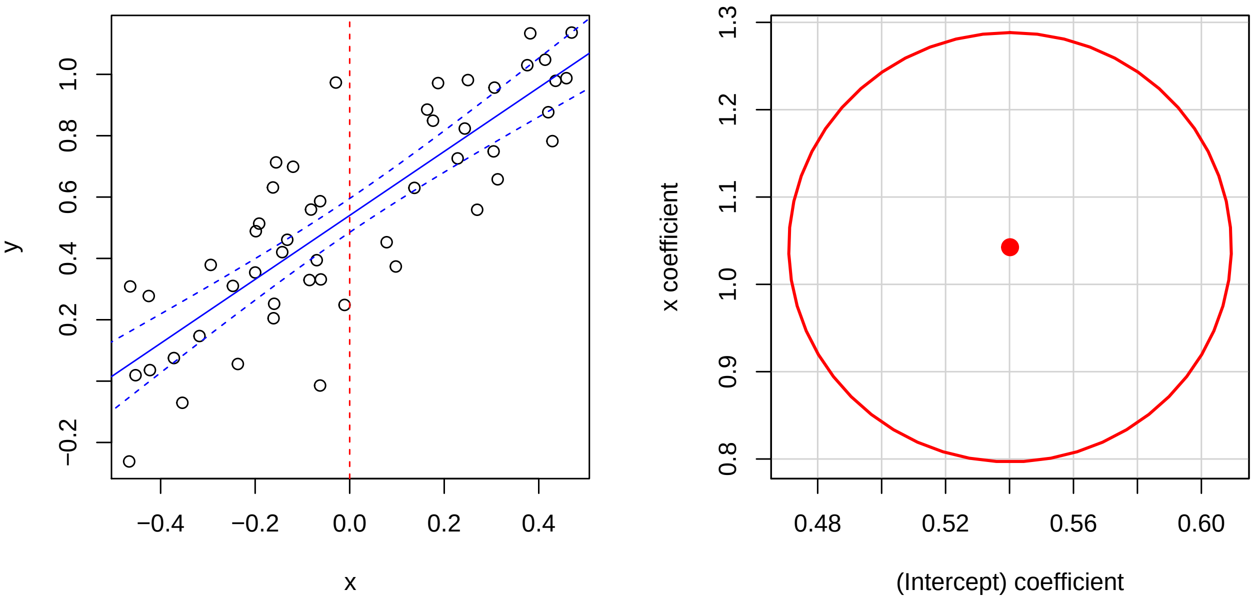
```
##                  2.5 %    97.5 %
## (Intercept) -3.7659585 -2.401361
## x            0.8468737  1.238166
```

However, these *univariate* confidence intervals are somewhat misleading because, as we shall see, our estimates for these parameters are correlated. For any given value of the intercept, there are only certain values of the slope within the univariate confidence interval that are actually supported. To assess this possibility, consider the *bivariate* confidence ellipse for these two parameters. We can evaluate this quantity graphically as follows with some help from the `car` package:

```r
library(car)
confidenceEllipse(m)
```



This is not great, because the possible values for each parameter are heavily cconstrained by the estimate for the other, and we cannot take our univarite confidence intervals at face value. Our problem can be solved by centering x:



Now there is no serious correlation in the estimates and we are free to use the univariate confidence intervals without needing to consider the joint distribution of the slope and intercept. This trick helps with interpretation, but it will also prove useful later in the course in the context of Markov chain Monte Carlo (MCMC) sampling.

Scaling covariates

It is usually also useful to scale covariates. There are two reasons for scaling covariates. First, if values of X are extremely small (e.g. 10^-6) or extremely large (e.g. 10^6 or greater) then it doesn't make sense to think about changes of 1 x unit. For example, if x is a distance measurement where all the values are on the scale of 1000km, the β value for a change in 1 kilometer may not be helpful. If we scale the covariate by dividing by 1000, then the β estimate now gives the change in y for every 1000km – much more useful!

Second, when more than one covariate is present, it is useful to be able to compare the magnitudes of the β estimates for each covariate in a meaningful way. There are a lot of possibilities, depending on the types of covariates. One good option is to divide each covariate by its standard deviation, such that a change in 1 x unit would correspond to a change in 1 standard deviation of x. This is especially useful when all covariates are continuous.

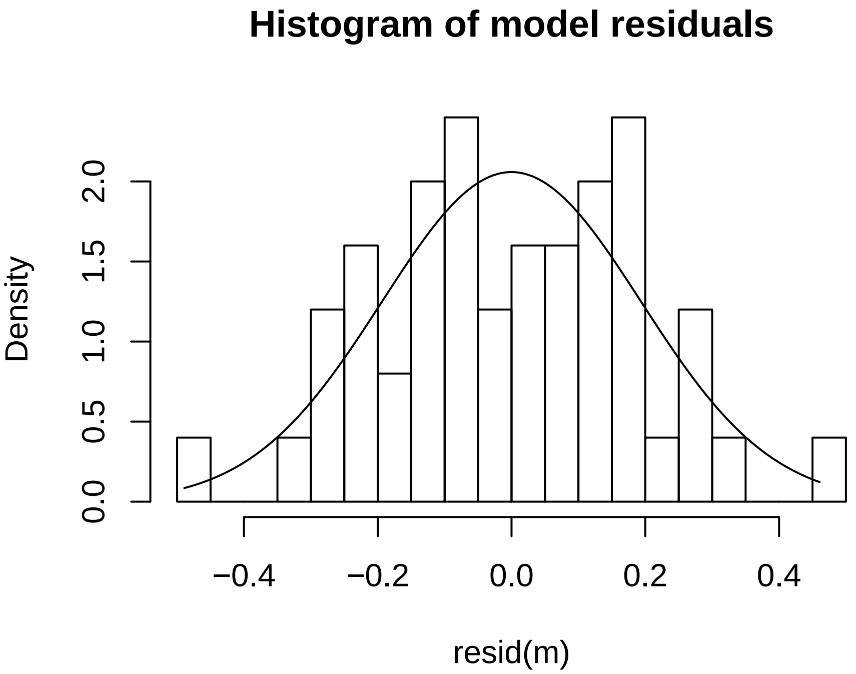
If one covariate is binary, it may make sense to divide by twice the standard deviation (s_x) (as recommended by German and Hill p. 57). This has the two effects. First, a 1 unit change in X now corresponds to a the change from 1 standard deviation below the mean to one standard deviation above the mean. Second, when binary variables are present, dividing by twice the standard deviation transforms binary covariates from $x \in \{0, 1\}$ to $x_t \in \{-0.5, 0.5\}$, where x_t is the transformed covariate: $x_t = \frac{x-\bar{x}}{2s_x}$. Now 1 unit change in a binary covariate corresponds exactly to the change from on binary category to the other.

Checking assumptions

One of the most important aspects of fitting statistical models is checking assumptions. This not only ensures that we have used the correct type of model for our data, but more importantly can give us insight into how to improve the model and, hence, any inferences derived from our modeling.

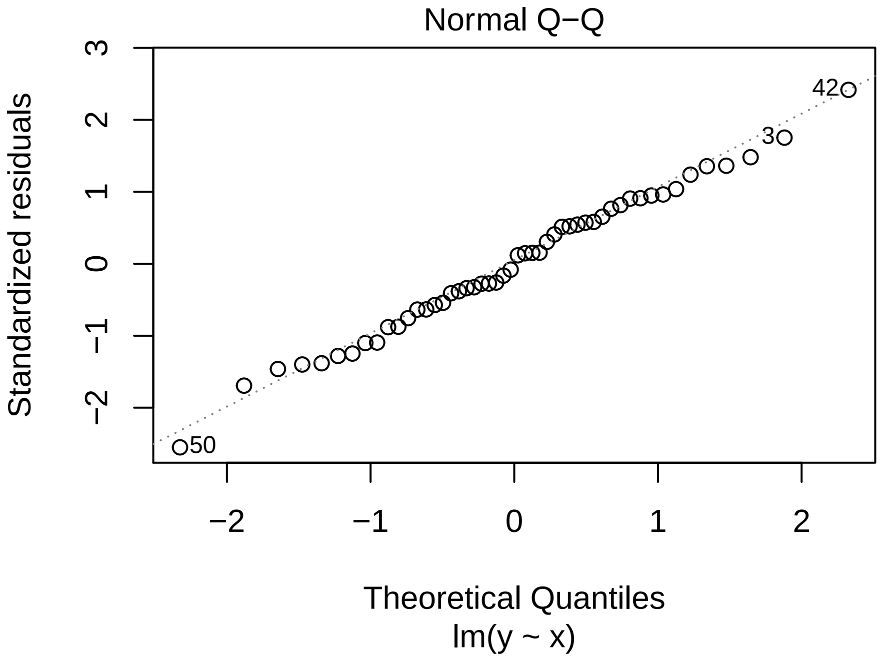
In the above cases we have assumed that the distribution of error terms is normally distributed, and this assumption is worth checking. Below, we plot a histogram of the residuals (another name for the ϵ parameters) along with a superimposed normal probability density so that we can check normality.

```r
# generate a histogram of the residuals from the m model
hist(resid(m), breaks = 20, freq = F,
     main = 'Histogram of model residuals')

# generate x-values for a normal curve
curve_x <- seq(min(resid(m)), max(resid(m)), .01)

# add the normal curve line
lines(curve_x, dnorm(curve_x, 0, summary(m)$sigma))
```



Even when the assumption of normality is correct, it is not always obvious that the residuals are normally distributed. Another useful plot for assessing normality of errors is a quantile-quantile or Q-Q plot. If the residuals do not deviate much from normality, then the points in a Q-Q plot won't deviate much from the dashed one-to-one line. If points lie above or below the line, then the residual is larger or smaller, respectively, than expected based on a normal distribution.

```r
plot(m, 2)
```



To assess heteroskedasticity, it is useful to inspect a plot of the residuals vs. fitted values, e.g. `plot(m, 1)`. If it seems as though the spread or variance of residuals varies across the range of fitted values, then it may be worth worrying about homoskedasticity and trying some transformations to fix the problem.

Categorical Covariates

Sometimes, the covariate of interest is not continuous but instead categorical (e.g., “chocolate”, “strawberry”, or “vanilla”). We might again wonder whether the mean of a random variable Y depends on the value of this covariate. However, we cannot really estimate a meaningful “slope” parameter, because in this case x is not continuous. Instead, we might formulate the model as follows:

$$y_i \sim N(\alpha_{j[i]}, \sigma^2)$$

Where α_j is the mean of group j, and we have J groups total. The notation $\alpha_{j[i]}$ represents the notion that the i^{th} observation corresponds to group j, and we are going to assume that all observations in the j^{th} group have the same mean, α_j. The above model is perfectly legitimate, and our parameters to estimate are the group means $\alpha_1, ..., \alpha_J$ and the residual variance σ^2. This parameterization is called the “means” parameterization, and though it is perhaps easier to understand than the following alternative, it is less often used.

This model is usually parametrized not in terms of the group means, but rather in terms of an intercept (corresponding to the mean of one “reference” group), and deviations from the intercept (differences between a group of interest and the intercept). For instance, in R, the group whose mean is the intercept (the “reference” group) will be the group whose name comes first alphabetically. Either way, we will estimate the same number of parameters. So if our groups are “chocolate”, “strawberry”, and “vanilla”, R will assign the group “chocolate” to be the intercept, and provide 2 more coefficient estimates for the difference between the estimated group mean of strawberry vs. chocolate, and vanilla vs. chocolate.

This parameterization can be written as

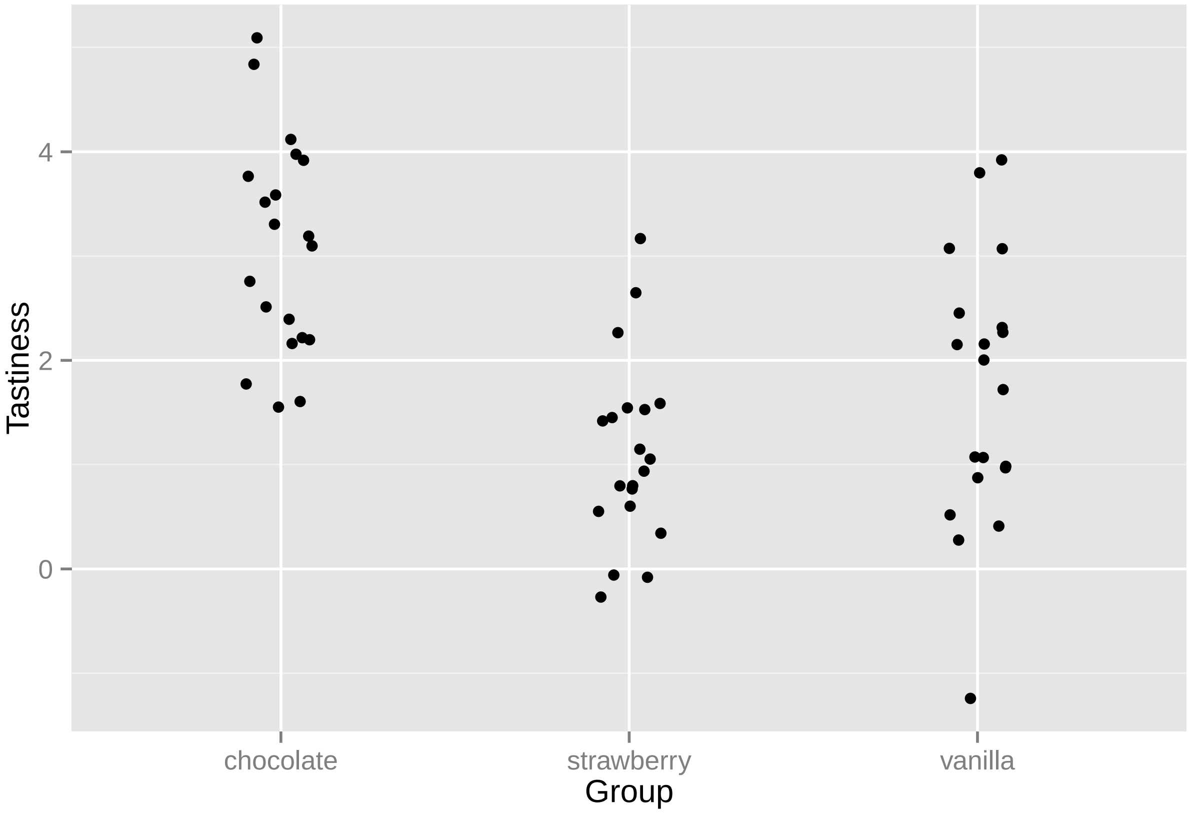
$$y_i \overset{iid}{\sim} N(\mu_0 + \beta_{j[i]}, \sigma^2)$$

where μ_0 is the “intercept” or mean of the reference group, and β_j represents the difference in the population mean of group j compared to the reference group (if j is the reference group, the $\beta_j = 0$).

Traditionally, this model (liner model with a single categorical predictor) is analyzed using a one-way analysis of variance (ANOVA), where essentially the goal is to determine whether incorporating the categorical predictor in the model provides a better estimate for y than simply using the global mean (by applying an F-test). When viewed from the point of view of simply getting the best estimates we can for the mean of each category, we see that the model used to estimate an ANOVA is simply another special case of a linear model. Indeed, note the resemblance between this formulation and the equation for a linear model with a continuous covariate from earlier.

The following example illustrates some data simulation, visualization, and parameter estimation in this context. Specifically, we assess 60 humans for their taste response to three flavors of ice cream. We want to extrapolate from our sample to the broader population of all ice cream eating humans to learn whether in general people think ice cream tastiness varies as a function of flavor.

```r
# simulate and visualize data
n <- 60
x <- rep(c("chocolate", "strawberry", "vanilla"), length.out = n)
x <- factor(x)
sigma <- 1
mu_y <- c(chocolate = 3.352, strawberry = .93, vanilla = 1.5)
y <- rnorm(n, mu_y[x], sigma)

library(ggplot2)
ggplot(data.frame(x, y), aes(x, y)) +
  geom_jitter(position = position_jitter(width=.1)) +
  xlab('Group') +
  ylab('Tastiness')
```



Model fitting

We can estimate our parameters with the `lm` function (this should be a strong hint that there are not huge differences between linear regression and ANOVA). The syntax is exactly the same as with linear regression. The only difference is that our input `x` is not numeric, but a character vector.

```
m1 <- lm(y ~ x)
summary(m1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.93472 -0.73470  0.02245  0.58698  2.22928
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0791     0.2411  12.771  < 2e-16 ***
## xstrawberry  -1.9689     0.3410  -5.775 3.35e-07 ***
## xvanilla     -1.3856     0.3410  -4.064 0.000149 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 1.078 on 57 degrees of freedom
## Multiple R-squared:  0.3817, Adjusted R-squared:   0.36
## F-statistic:  17.6 on 2 and 57 DF,  p-value: 1.118e-06
```

Because chocolate comes first alphabetically, it is the reference group and the “(Intercept)” estimate corresponds to the estimate of the group-level mean for chocolate. The other two estimates are contrasts between the other groups and this reference group, i.e. “xstrawberry” is the estimated difference between the group mean for strawberry and the reference group.

However it may make more sense to use the means parameterization to directly estimate the mean for each group. We need to tell R to suppress the intercept term in our model:

```r
m <- lm(y ~ 0 + x)
summary(m)
```

```
##
## Call:
## lm(formula = y ~ 0 + x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.93472 -0.73470  0.02245  0.58698  2.22928
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## xchocolate    3.0791     0.2411  12.771  < 2e-16 ***
## xstrawberry   1.1101     0.2411   4.605 2.36e-05 ***
## xvanilla      1.6934     0.2411   7.024 2.90e-09 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 1.078 on 57 degrees of freedom
## Multiple R-squared:  0.8039, Adjusted R-squared:  0.7936
## F-statistic: 77.88 on 3 and 57 DF,  p-value: < 2.2e-16
```

Arguably, this approach is more useful because it simplifies the construction of confidence intervals for the group means:

```r
confint(m)
```

```
##                 2.5 %   97.5 %
## xchocolate  2.5963035 3.561853
## xstrawberry 0.6273717 1.592922
## xvanilla    1.2106613 2.176211
```

One important philosophical point. Note that a frequentist ANOVA test:

```r
anova(m1)
```

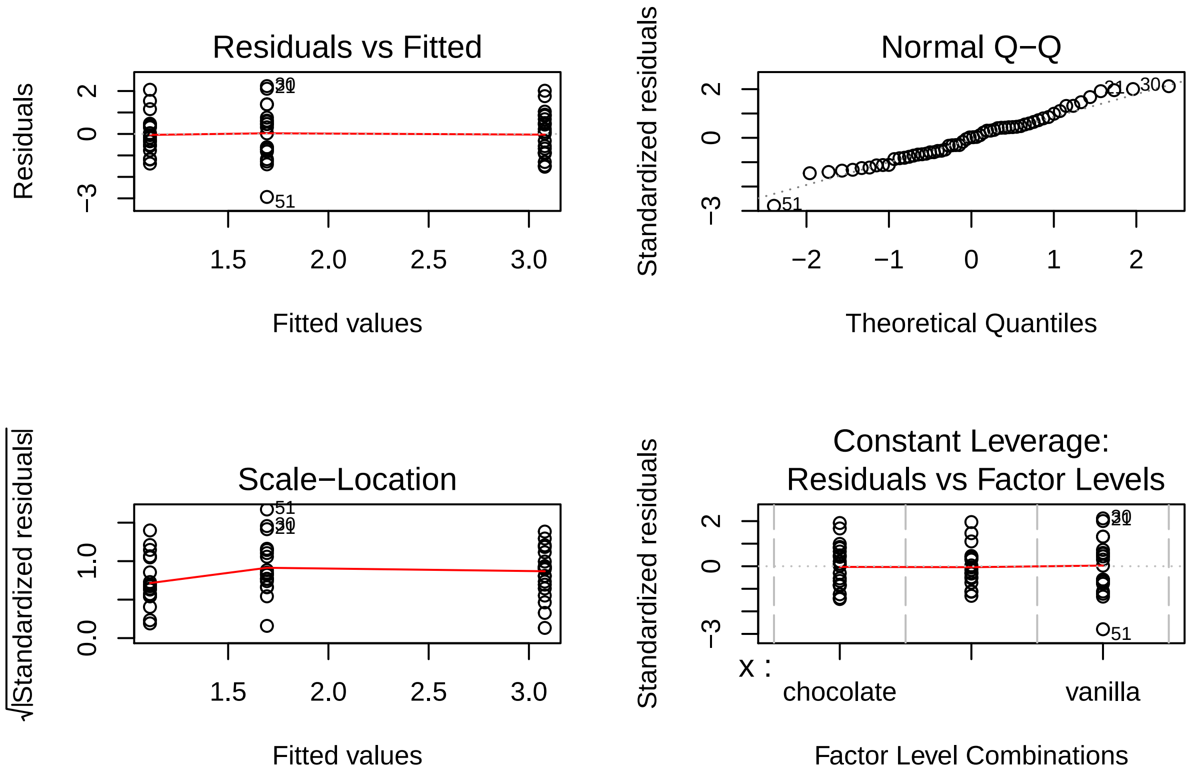
```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## x          2 40.913 20.4564  17.597 1.118e-06 ***
## Residuals 57 66.262  1.1625
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

indicates that the observed data are not terribly consistent with the null hypothesis that humans think there is no difference in ice cream flavors. However, it doesn't tell us a whole lot about just how different humans think these flavors are, nor which is the most flavorful, or whether certain flavors don't taste any better than others. Of course, you could wrap yourself in knots trying a battery of post-hoc tests after running the ANOVA, but perhaps its better just to use the means parameterization, the estimated confidence intervals, and little common sense to say that humans probably think chocolate tastes better than strawberry or vanilla, but that the latter two or more or less equivalent.

Checking assumptions

Our assumptions in this simple one way ANOVA context are identical to our assumptions with linear regression. Specifically, we assumed that our errors are independently and identically distributed, and that the variance is constant for each group (homoskedasticity). The built in `plot` method for `lm` objects is designed to return diagnostic plots that help to check these assumptions.

```
par(mfrow=c(2, 2))
plot(m)
```



Note that there are now just three "fitted values" (one for each group) with residuals around those fitted values, but that the residuals are still consistent with a normal distribution (see Q-Q plot).

General linear models

We have covered a few special cases of general linear models, which are usually written as follows:

$$y \overset{iid}{\sim} N(X\beta, \sigma^2)$$

Where y is a vector consisting of n observations, X is a "design" matrix with n rows and p columns, and β is a vector of p parameters. There are multivariate general linear models (e.g., MANOVA) where the response variable is a matrix and a covariance matrix is used in place of a scalar variance parameter, but we'll stick to univariate models for simplicity. The key point here is that *the product of X and β provides the mean of the normal distribution from which y is drawn.* From this perspective, the difference between the model of the mean, linear regression, ANOVA, etc., lies in the structure of X and subsequent interpretation of the parameters β. This is a very powerful idea that unites many superficially disparate approaches. It also is the reason that these models are considered "linear", even though a regression line might by quite non-linear (e.g., polynomial regression). These models are linear in their parameters, meaning that our expected value for the response y is a **linear combination** (formal notion) of the parameters. If a vector of expected values for y in some model cannot be represented as $X\beta$, then it is not a linear model.

In the model of the mean, X is an n by 1 matrix, with each element equal to 1 (i.e. a vector of ones). With linear regression, X's first column is all ones (corresponding to the intercept parameter), and the second column contains the values of the covariate x. In ANOVA, the design matrix X will differ between the means and effects parameterizations. With a means parameterization, the entries in column j will equal one if observation (row) i is in group j, and entries are zero otherwise. If you are not comfortable with matrix multiplication, it's worth investing some effort so that you can understand why $X\beta$ is such a powerful construct.

> Can you figure out the structure of X with R's default effects parameterization? You can check your work with `model.matrix(m)`, where `m` is a model that you've fitted with `lm`.

Interactions between covariates

Often, the effect of one covariate depends on the value of another covariate. This is referred to as "interaction" between the covariates. Interactions can exist between two or more continuous and/or nominal covariates. These situations have special names in the classical statistics literature. For example, models with interactions between nominal covariates fall under "factorial ANOVA", while those with interactions between a continuous and a nominal covariate are referred to as "analysis of covariance (ANCOVA)". Here we prefer to consider these all as special cases of general linear models.

Interactions between two continuous covariates

Here we demonstrate simulation and estimation for a model with an interaction between two continuous covariates. In the simulation, we exploit the $X\beta$ construct to generate a vector of expected values for y.

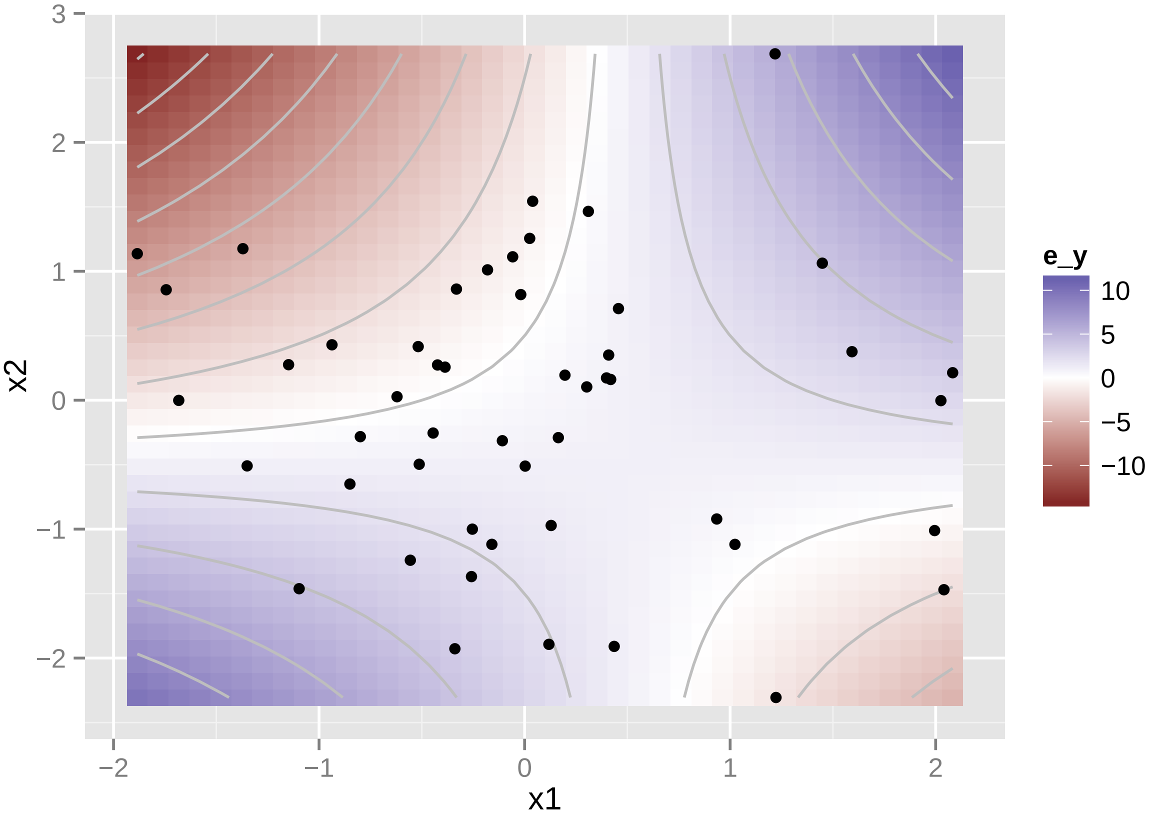
```r
# set up
n <- 50
x1 <- rnorm(n)
x2 <- rnorm(n)
beta <- c(.5, 1, -1, 2)
sigma <- 1

# generate the design matrix X with n rows and four columns
# corresponding to the intercept, two main effects, and the interaction effect
```

```
X <- matrix(c(rep(1, n), x1, x2, x1 * x2), nrow=n)

# rather than use a linear equation (a + bx1 + bx2 + ...)
# we use matrix multiplication to generate mean y values
mu_y <- X %*% beta

# now we simulate values for y using the simulated means and sigma = 1
y <- rnorm(n, mu_y, sigma)

# estimate the parameters
m <- lm(y ~ x1 + x2 + x1:x2)
summary(m)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x1:x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4175 -0.6959 -0.1166  0.6015  1.8164
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6454     0.1230   5.248 3.81e-06 ***
## x1            1.0717     0.1262   8.492 5.62e-11 ***
## x2           -0.7983     0.1231  -6.486 5.40e-08 ***
## x1:x2         2.0854     0.1257  16.584  < 2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.8637 on 46 degrees of freedom
## Multiple R-squared:  0.8845, Adjusted R-squared:  0.877
## F-statistic: 117.5 on 3 and 46 DF,  p-value: < 2.2e-16
```

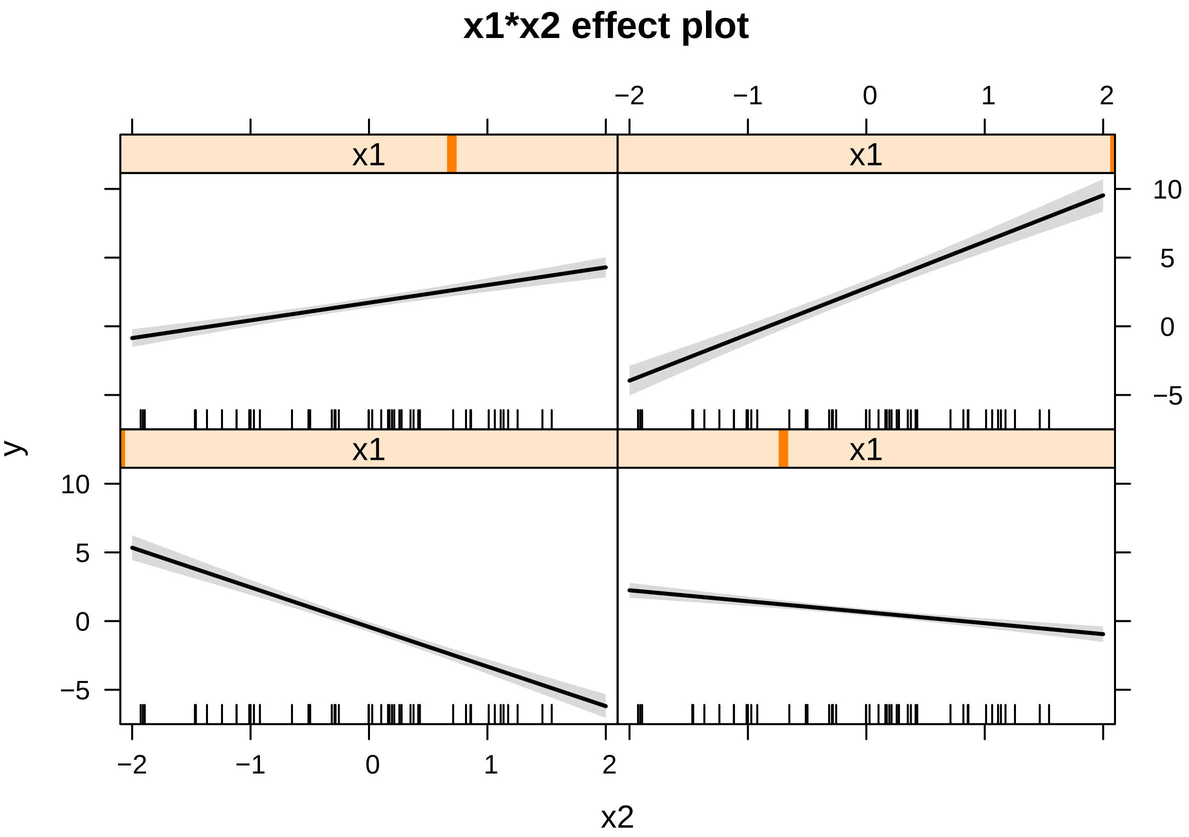
Visualizing these models is tricky, because we are in 3d space (with dimensions x_1, x_2, and y), but contour plots can be effective and leverage peoples’ understanding of topographic maps.

```
# visualizing the results in terms of the linear predictor
lo <- 40
x1seq <- seq(min(x1), max(x1), length.out = lo)
x2seq <- seq(min(x2), max(x2), length.out = lo)
g <- expand.grid(x1=x1seq, x2=x2seq)
g$e_y <- beta[1] + beta[2] * g$x1 + beta[3] * g$x2 + beta[4] * g$x1 * g$x2
ggplot(g, aes(x=x1, y=x2)) +
  geom_tile(aes(fill=e_y)) +
  stat_contour(aes(z=e_y), col=’grey’) +
  scale_fill_gradient2() +
  geom_point(data=data.frame(x1, x2))
```



Alternatively, you might check out the `effects` package:

```
library(effects)
plot(allEffects(m))
```



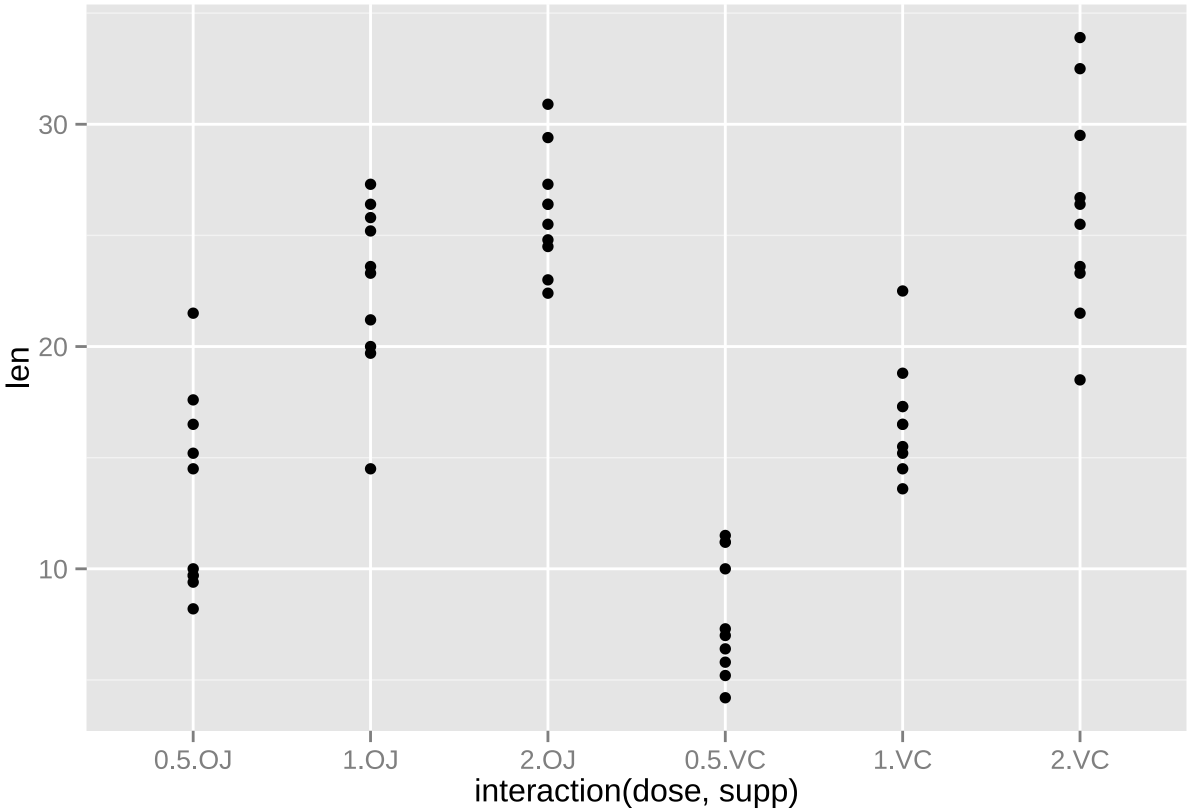
Interactions between two categorical covariates

Here we demonstrate interaction between two categorical covariates, using the `ToothGrowth` dataset. Specifically we are interested in how the the length of odontoblasts (teeth) in each of 10 guinea pigs varies among three dose levels of Vitamin C (0.5, 1, and 2 mg) and two supplement delivery methods (orange juice or ascorbic acid). It's possible that the effect of the dose might vary as a function of the delivery method, indicating an interaction between dose and supplement.

```
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  3 variables:
##  $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
##  $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dose: Factor w/ 3 levels "0.5","1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

```
ToothGrowth$dose <- factor(ToothGrowth$dose)
ggplot(ToothGrowth, aes(x=interaction(dose, supp), y=len)) +
  geom_point()
```



In general, visualizing the raw data is a good idea. However, we might also be interested in a table with group-wise summaries, such as the sample means, standard deviations, and sample sizes.

```
library(dplyr)
ToothGrowth %>%
  group_by(dose, supp) %>%
  summarize(mean = mean(len),
            sd = sd(len),
            n = n())
```

```
## Source: local data frame [6 x 5]
## Groups: dose [?]
##
##     dose   supp  mean       sd     n
##   (fctr) (fctr) (dbl)    (dbl) (int)
## 1    0.5     OJ 13.23 4.459709    10
## 2    0.5     VC  7.98 2.746634    10
## 3      1     OJ 22.70 3.910953    10
## 4      1     VC 16.77 2.515309    10
## 5      2     OJ 26.06 2.655058    10
## 6      2     VC 26.14 4.797731    10
```

We can construct a model to estimate the effect of dose, supplement, and their interaction.

```
m <- lm(len ~ dose * supp, data = ToothGrowth)
summary(m)
```

```
##
## Call:
## lm(formula = len ~ dose * supp, data = ToothGrowth)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -8.20  -2.72  -0.27   2.65   8.27
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     13.230      1.148  11.521 3.60e-16 ***
## dose1            9.470      1.624   5.831 3.18e-07 ***
## dose2           12.830      1.624   7.900 1.43e-10 ***
## suppVC          -5.250      1.624  -3.233  0.00209 **
## dose1:suppVC    -0.680      2.297  -0.296  0.76831
## dose2:suppVC     5.330      2.297   2.321  0.02411 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.631 on 54 degrees of freedom
## Multiple R-squared:  0.7937, Adjusted R-squared:  0.7746
## F-statistic: 41.56 on 5 and 54 DF,  p-value: < 2.2e-16
```

This summary gives the effects-parameterization version of the summary. The “(Intercept)” refers to individuals with the specific combination of factor levels that occur first alphabetically: in this case, those that received a dose of 0.5 with the “OJ” supplement. The coefficients for `dose1` and `dose2` represent estimated contrasts for these pigs that received doses 1 and 2 using the “OJ” supplement relative to the intercept. The coefficient for `suppVC` represents the contrast between the “VC” and “OJ” levels of supplement when the dose is 0.5. The interaction terms represent the difference in the effect of VC for `dose1` and `dose2` relative to a dose of 0.5 using “VC”. None of this is particularly intuitive, but this information can be gleaned by inspecting the design matrix X produced by `lm` in the process of fitting the model. Inspecting the design matrix along with the dataset to gives a better sense for how X relates to the factor levels:

```
cbind(model.matrix(m), ToothGrowth)
```

```
##    (Intercept) dose1 dose2 suppVC dose1:suppVC dose2:suppVC  len supp dose
## 1            1     0     0      1            0            0  4.2   VC  0.5
## 2            1     0     0      1            0            0 11.5   VC  0.5
## 3            1     0     0      1            0            0  7.3   VC  0.5
## 4            1     0     0      1            0            0  5.8   VC  0.5
## 5            1     0     0      1            0            0  6.4   VC  0.5
## 6            1     0     0      1            0            0 10.0   VC  0.5
## 7            1     0     0      1            0            0 11.2   VC  0.5
## 8            1     0     0      1            0            0 11.2   VC  0.5
## 9            1     0     0      1            0            0  5.2   VC  0.5
## 10           1     0     0      1            0            0  7.0   VC  0.5
## 11           1     1     0      1            1            0 16.5   VC    1
## 12           1     1     0      1            1            0 16.5   VC    1
## 13           1     1     0      1            1            0 15.2   VC    1
## 14           1     1     0      1            1            0 17.3   VC    1
## 15           1     1     0      1            1            0 22.5   VC    1
## 16           1     1     0      1            1            0 17.3   VC    1
## 17           1     1     0      1            1            0 13.6   VC    1
## 18           1     1     0      1            1            0 14.5   VC    1
## 19           1     1     0      1            1            0 18.8   VC    1
## 20           1     1     0      1            1            0 15.5   VC    1
## 21           1     0     1      1            0            1 23.6   VC    2
## 22           1     0     1      1            0            1 18.5   VC    2
## 23           1     0     1      1            0            1 33.9   VC    2
## 24           1     0     1      1            0            1 25.5   VC    2
## 25           1     0     1      1            0            1 26.4   VC    2
## 26           1     0     1      1            0            1 32.5   VC    2
## 27           1     0     1      1            0            1 26.7   VC    2
## 28           1     0     1      1            0            1 21.5   VC    2
## 29           1     0     1      1            0            1 23.3   VC    2
## 30           1     0     1      1            0            1 29.5   VC    2
## 31           1     0     0      0            0            0 15.2   OJ  0.5
## 32           1     0     0      0            0            0 21.5   OJ  0.5
## 33           1     0     0      0            0            0 17.6   OJ  0.5
## 34           1     0     0      0            0            0  9.7   OJ  0.5
## 35           1     0     0      0            0            0 14.5   OJ  0.5
## 36           1     0     0      0            0            0 10.0   OJ  0.5
## 37           1     0     0      0            0            0  8.2   OJ  0.5
## 38           1     0     0      0            0            0  9.4   OJ  0.5
## 39           1     0     0      0            0            0 16.5   OJ  0.5
## 40           1     0     0      0            0            0  9.7   OJ  0.5
## 41           1     1     0      0            0            0 19.7   OJ    1
## 42           1     1     0      0            0            0 23.3   OJ    1
## 43           1     1     0      0            0            0 23.6   OJ    1
## 44           1     1     0      0            0            0 26.4   OJ    1
## 45           1     1     0      0            0            0 20.0   OJ    1
## 46           1     1     0      0            0            0 25.2   OJ    1
## 47           1     1     0      0            0            0 25.8   OJ    1
## 48           1     1     0      0            0            0 21.2   OJ    1
## 49           1     1     0      0            0            0 14.5   OJ    1
## 50           1     1     0      0            0            0 27.3   OJ    1
## 51           1     0     1      0            0            0 25.5   OJ    2
## 52           1     0     1      0            0            0 26.4   OJ    2
## 53           1     0     1      0            0            0 22.4   OJ    2
## 54           1     0     1      0            0            0 24.5   OJ    2
## 55           1     0     1      0            0            0 24.8   OJ    2
## 56           1     0     1      0            0            0 30.9   OJ    2
## 57           1     0     1      0            0            0 26.4   OJ    2
## 58           1     0     1      0            0            0 27.3   OJ    2
## 59           1     0     1      0            0            0 29.4   OJ    2
## 60           1     0     1      0            0            0 23.0   OJ    2
```

Interpreting Models

Now for future work in make guinea pig teeth grow, we may want to know whether the dose and method of supplementation interact in their influence on length. From a null hypothesis significance testing perspective, we can evaluate the 'significance' of the interaction term as follows:

```
anova(m)
```

```
## Analysis of Variance Table
```

```
##
## Response: len
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## dose       2 2426.43 1213.22  92.000 < 2.2e-16 ***
## supp       1  205.35  205.35  15.572 0.0002312 ***
## dose:supp  2  108.32   54.16   4.107 0.0218603 *
## Residuals 54  712.11   13.19
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case we found that the interaction was significant, but what does that mean, really? Well it means that if we ran this experiment repeatedly (really, an infinite number of times) and there was no actual interaction, than we would be unlikely to have observed the data *as extreme or more extreme* as the data in our particular experiment (if we used a p-value cutoff of 0.05 than we can be more specific and say that the observed (or more extreme) data would occur in less than 5% of the hypothetical infinite amount of experiments.) In our case the frequentist statistician would say, yes, use the interaction to predict tooth growth in guinea pigs (and use the point estimate generated from your particular observed data set as your best guess for the strength of the interaction).

Of course, if our data had been ever so slightly less extreme, such the calculated p-value for the interaction was now 0.06 (or 0.11), our frequentist statistician would say there is no strong evidence for an interaction and that we should not use it to predict tooth length in the future.

Although this is far from intuitive, it has been widely used (and abused)

An alternative approach would be to use information theoretical to decide whether the interaction is warranted. In the past decade following Burnham and Anderson's book on the topic, ecologists have leaned heavily on Akaike's information criterion (AIC), which is a relative measure of model quality (balancing goodness of fit with model complexity). The essential idea behind the use of AIC is that while the interaction may help us predict our observed data better than a model without an interaction, is it really going to help us fit future data well? At one extreme you could imagine a model that has a separate parameter for each observations. Voila, you've fit your model perfectly, but its utter crap when it comes to predicting future values. AIC and similar metric penalize us for adding more parameters than absolutely necessary

We can compare how the two models we are considering (with and without interaction) compare using AIC:

```
m2 <- lm(len ~ dose + supp, data = ToothGrowth)
AIC(m, m2)
```

```
##    df      AIC
## m   7 332.7056
## m2  5 337.2013
```

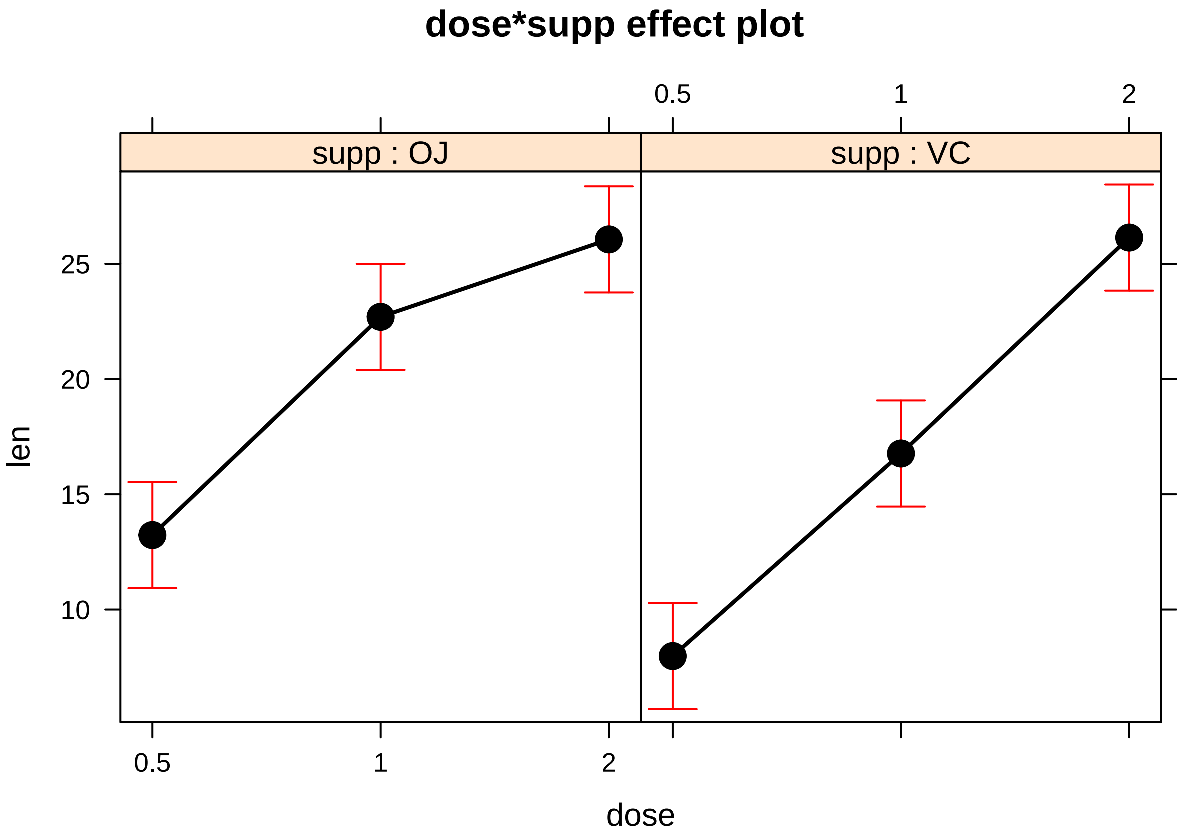
Here we see that the original model `m` with interaction has a lower AIC value, and is therefore better supported. Note that AIC is not without its own drawbacks (just how much lower AIC is really *lower*?). AIC can be considered to be similar to cross validation, approximating the ability of a model to predict future data.

In chapter 3 we will introduce a more streamlined procedure that 1) does not assume that the effect is zero to begin with 2) does not invoke a hypothetical infinite number of replicated realizations of the data, conditional on one particular parameter value.

Plotting the data

Being somewhat lazy, we might again choose to plot the results of our interaction model using the `effects` package.

```
plot(allEffects(m))
```



This is less than satisfying, as it does not show any data. All we see is model output (which looks pretty snazzy!), but if the model doesn't actually fit the data well at all (i.e. the model is crap), then the output is meaningless. Ideally we want to be able to juxtapose our observed data with what what we would expect from the model. Fortunately, its possible to plot both the observed data points along with the estimated group means and some indication of uncertainty in our estimates of those means.

For starters, if we weren't quite so lazy, we could use the `predict` function to obtain confidence intervals for the means of each group.

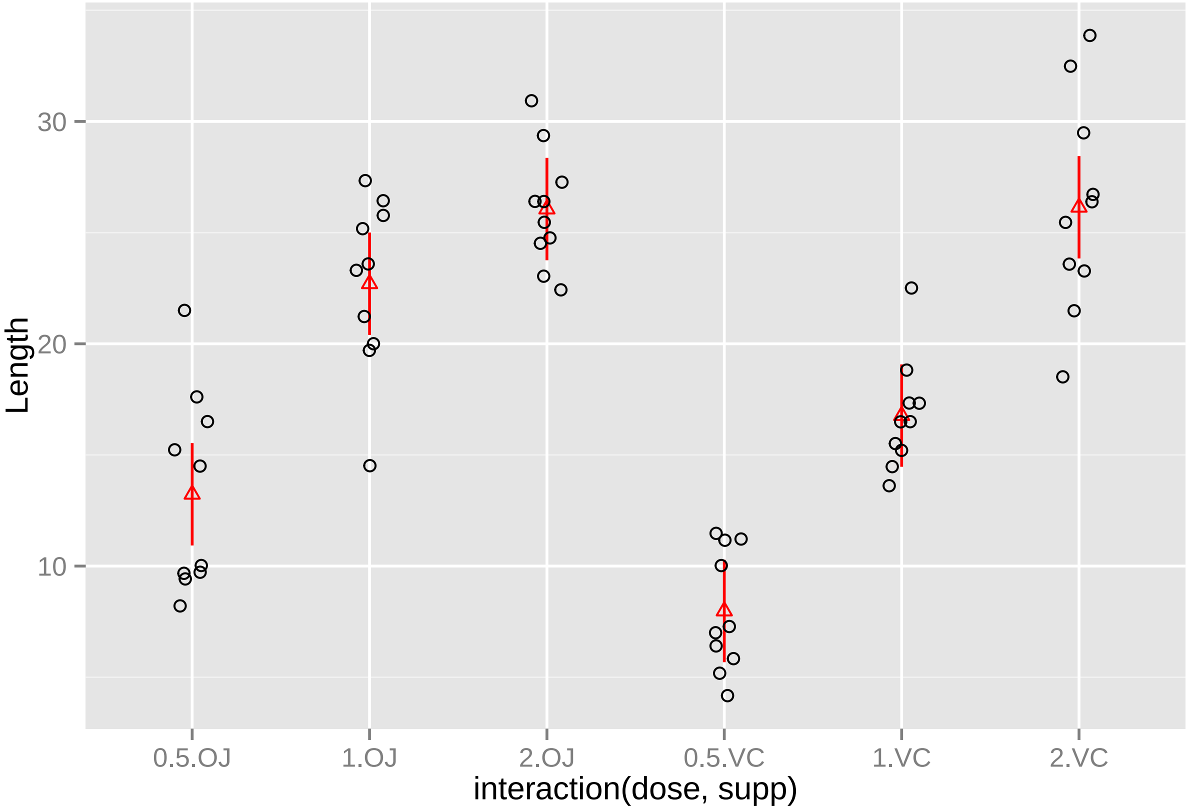
```
# construct a new data frame for predictions
g <- expand.grid(supp = levels(ToothGrowth$supp),
                 dose = levels(ToothGrowth$dose))
p <- predict(m, g, interval = 'confidence', type='response')
predictions <- cbind(g, data.frame(p))
predictions
```

```
##   supp dose   fit       lwr      upr
## 1   OJ  0.5 13.23 10.927691 15.53231
## 2   VC  0.5  7.98  5.677691 10.28231
## 3   OJ    1 22.70 20.397691 25.00231
## 4   VC    1 16.77 14.467691 19.07231
## 5   OJ    2 26.06 23.757691 28.36231
## 6   VC    2 26.14 23.837691 28.44231
```

Now we have the model fits plus 95% confidence intervals for predictions (for now let's assume the confidence intervals represent our uncertainty in the estimate of the mean and not what they actually indicate, which is an interval that, if we calculated it for our each of our infinite repetitions of the same experiment, the “true” population mean would fall within the calculated confidence interval 95 times out of 100–what a mouthful!)

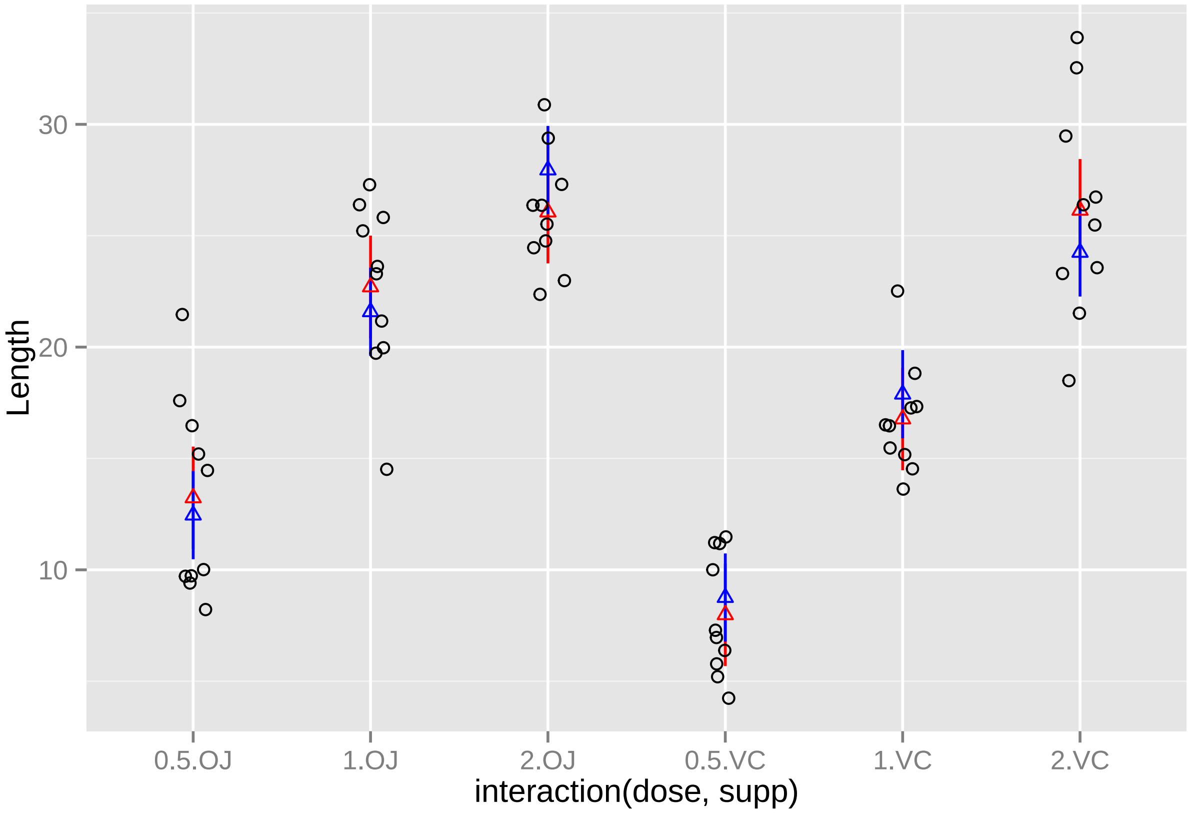
We could try to compare these using the table, but lets illustrate them graphically along with the raw data:

```
ggplot(ToothGrowth, aes(x=interaction(dose, supp), y=len)) +
  geom_segment(data=predictions,
               aes(y=lwr, yend=upr,
                   xend=interaction(dose, supp)), col='red') +
  geom_point(data=predictions, aes(y=fit), color='red', size=2, shape=2) +
  geom_jitter(position = position_jitter(width=.1), shape=1) +
  ylab("Length")
```



This plot is nice because we can observe the data along with the model output. This makes it easier for readers to understand how the model relates to, fits, and does not fit the data. If you wish to obscure the data, you could make a bar plot with error pars to represent the standard errors. Although “dynamite” plots are common, we shall not include one here and we strongly recommend that you never produce such a plot (more here).

For kicks, lets just look at the predictions (in blue) if we assume the interaction is zero:



So, not that different, but perhaps we are overshooting the 2:OJ and 1:VC categories and undershooting the 1:OJ and 2:VC categories a bit.

Interactions between continuous and categorical covariates

Sometimes, we're interested in interactions between continuous or numeric covariates and another covariate with discrete categorical levels. Again, this falls under the broad class of models used in analysis of covariance (ANCOVA).

Once again, let's simulate some hypothetical data where we have two groups and a continuous covariate, and the continuous covariate differs for the two groups (note again the use of the design matrix):

```
x1 <- rnorm(n)
x2 <- factor(sample(c('A', 'B'), n, replace=TRUE))

# generate random intercepts first and second groups
a <- rnorm(2)

# generate random slopes
b <- rnorm(2)

sigma <- .4

# the design matrix (note that )
X <- matrix(c(ifelse(x2 == 'A', 1, 0),
              ifelse(x2 == 'B', 1, 0),
              ifelse(x2 == 'A', x1, 0),
              ifelse(x2 == 'B', x1, 0)
```

```
            ), nrow=n)

mu_y <- X %*% c(a, b)
y <- rnorm(n, mu_y, sigma)
```

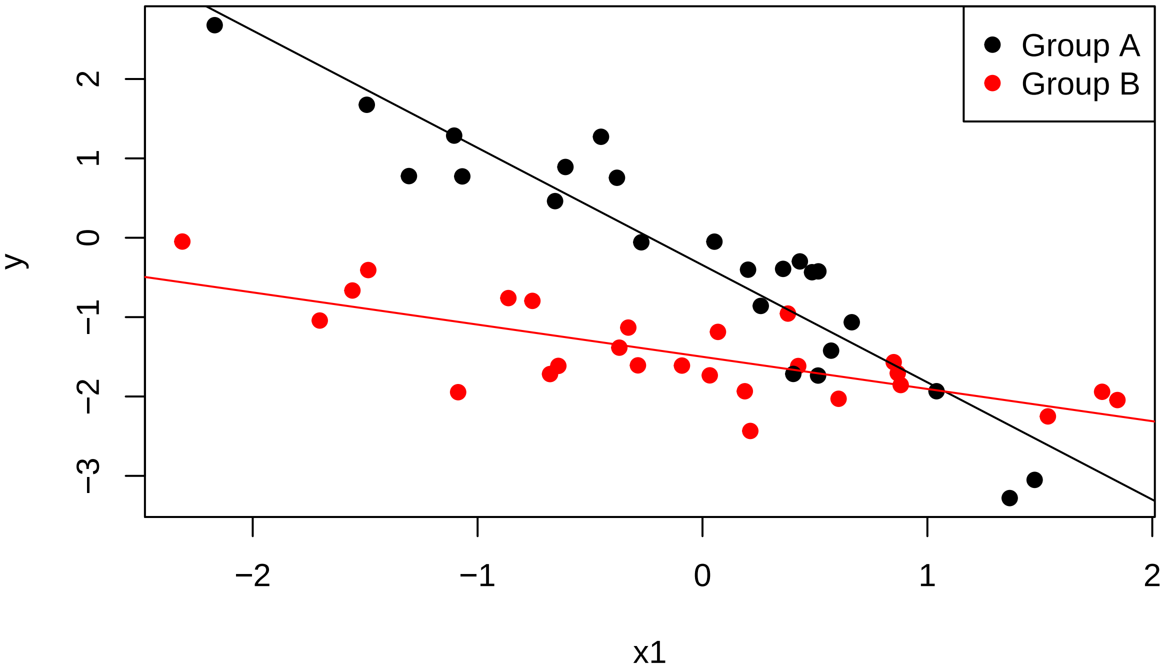
When we fit an ANCOVA we can assume that the slope of our continuous covariate does not differ between the two groups (and fit only a single predictor that applies to both groups), or we can fit an “interaction” between the continuous and categorical factors, which implicitly tells R to assume that slopes differ and to fit them separately. Let’s go ahead and try the latter

```
m <- lm(y ~ x1 + x2 + x1:x2)
summary(m)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x1:x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91716 -0.23330 -0.01746  0.34014  0.94970
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.34436    0.09683  -3.556 0.000885 ***
## x1          -1.47738    0.10749 -13.744  < 2e-16 ***
## x2B         -1.15543    0.13446  -8.593 4.01e-11 ***
## x1:x2B       1.07188    0.13958   7.679 8.82e-10 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.4737 on 46 degrees of freedom
## Multiple R-squared:  0.8623, Adjusted R-squared:  0.8534
## F-statistic: 96.05 on 3 and 46 DF,  p-value: < 2.2e-16
```

Let’s plot the lines of best fit along with the data.

```
plot(x1, y, col=x2, pch=19)
legend(’topright’, col=1:2, legend=c(’Group A’, ’Group B’), pch=19)
abline(coef(m)[1], coef(m)[2])
abline(coef(m)[1] + coef(m)[3], coef(m)[2] + coef(m)[4], col=’red’)
```



The `abline` function, used above, adds lines to plots based on a y-intercept (first argument) and a slope (second argument). Do you understand why the particular coefficients that we used as inputs provide the desired intercepts and slopes for each group? If not, evaluate the design matrix X via `model.matrix(m)` and interpret the coefficients β corresponding to each column via `coef(m)`.

The general strategy of building a design matrix X is useful in many other contexts, such as when the errors are not normally distributed. Specifically, we will soon explore situations where our data may consist of binary observations or integer counts, and the $X\beta$ formulation still plays a central role. Indeed, later on in the course when we begin discussing random effects, it will be possible to represent specific types of model structures with two design matrices: one for the fixed effects X, and one for the random effects Z.

Further reading

Schielzeth, H. 2010. Simple means to improve the interpretability of regression coefficients. Methods in Ecology and Evolution 1:103–113.

Enqvist, L. 2005. The mistreatment of covariate interaction terms in linear model analyses of behavioural and evolutionary ecology studies. Animal Behaviour 70:967–971.

Gelman and Hill. 2009. *Data analysis using regression and multilevel/hierarchical models.* Chapter 3-4.