Stats 20x Handout 1: The Linear Model

1.1 Finding relationships between variables

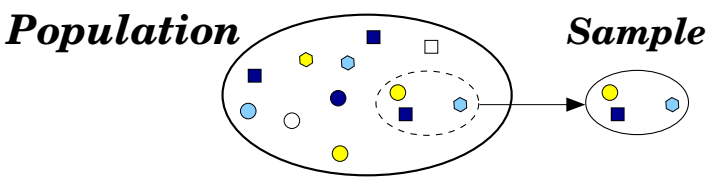
This course is about statistical modelling. What does that mean? In a nutshell, statistical modelling is about *finding relationships between variables*.

Variables are things that we measure or observe. A vast portion of human enterprise is focused on discovering which variables are related to which. This is true in science, business, and any other area in which we aim to understand the world by making observations. The following statements are all examples of relationships between two variables.

- Smoking causes lung cancer.
  Subjects: people.
  Variables: smoke yes/no: the *predictor* variable;
  lung cancer yes/no: the *response* variable.
- Heart disease is associated with high cholesterol levels.
  Subjects: people.
  Variables: cholesterol level (numeric): ***predictor variable;***
  heart disease yes/no: ***response variable.***
- Having a university degree is associated with higher lifetime income.
- Users of Apple smartphones do more online shopping than Android users.
- The chance of a baby being a boy decreases with the father's age.
- Cows that are given names (e.g. Daisy or Ermintrude) produce more milk.

Relationships between variables are everywhere around us. Some relationships are so well-known that it is surprising to think they once had to be 'discovered'. For example, as recently as 1960, only 30% of US doctors believed in the link between smoking and lung cancer.

This brings us to our key role as statistical modellers. How do we establish that a relationship exists? We can't observe all possible smokers that might ever exist, and compare their lung cancer outcomes to those of all possible non-smokers. Instead, we have to learn about the world through ***samples.***



We can only work with samples, but we want to know about the ***population:*** e.g. all smokers and non-smokers that can ever exist. The process of drawing conclusions about a population, based on data from a sample, is called ***inference.*** We can't *observe* the whole population, so instead we must focus on what we can legitimately *infer* about the population, based on our sample data.

Key questions

Here are the key questions we will be addressing as we explore ideas in statistical modelling.

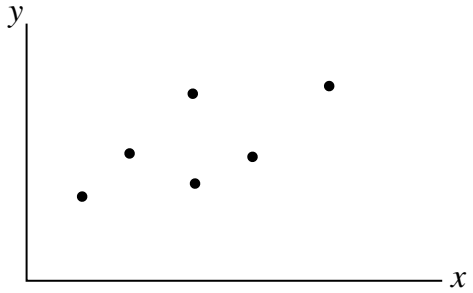
- How can we be sure a relationship exists?
  — *We might need to convince a sceptical or resistant audience: e.g. the smoking / lung cancer link was suppressed by the powerful tobacco industry.*
- If it does exist, how can we describe it? What shape is the relationship, and how can we quantify it numerically?
- How can we assess the quality of a relationship: e.g. how tightly connected two variables are?
- When there are multiple variables, how can we decide which *suite* of predictor variables best explains the response variable?
  — *E.g. if Apple users do more online shopping than Android users, could it just be that Apple is more popular in richer countries and Android in poorer countries?*

1.2 The Simple Linear Model

We'll start with a classic relationship scenario: the simple linear model.
We have n *subjects*, e.g. people, whom we'll label $i = 1, 2, \ldots, n$.

We have two numeric variables of interest: ***X and Y.*** Because they are numeric variables, and not labels like 'yes/no', they can be plotted on a scatterplot. Each of our subjects has a pair of observations, which we can write as x_i and y_i. An easy example to think of is where x_i is the height of person i, and y_i is their weight.

Person i has a pair of observations, (x_i, y_i), which we plot on a scatterplot.



We are interested in the relationship between the variables X and Y, and particularly in whether ***X can be said to 'predict' or 'explain' Y.***

For this reason we call X the *predictor* or *explanatory* variable, and Y the *response* variable.

(You might have heard X called the 'independent' variable, and Y called the 'dependent' variable, but this terminology is confusing and best avoided.)

We use CAPITAL LETTERS when we refer to X and Y as variables. Think of capital letters as denoting an unspecified or random quantity.

We use lower-case letters when referring to observations of these variables. For example, if X is height and Y is weight, then we use (x_i, y_i) to describe the height and weight of person i, after they have been observed.

When thinking about how X predicts Y, we often treat X as if it is already decided, and treat Y as random. For example, if I have selected a person of height $X = 175$cm, I might want to know their range of likely weights, Y. In this case, we often use the notation of ***conditioning:*** we are interested in the distribution of weight, Y, given a height of $X = 175$cm. We then write Y_i with a capital letter, but x_i with a lower-case letter, because we are thinking of Y_i as a random response to the fixed choice of x_i.

We are now ready to formulate what we mean by a linear model. A simple linear model is a ***straight-line relationship between X and Y, with scatter.***

We write:

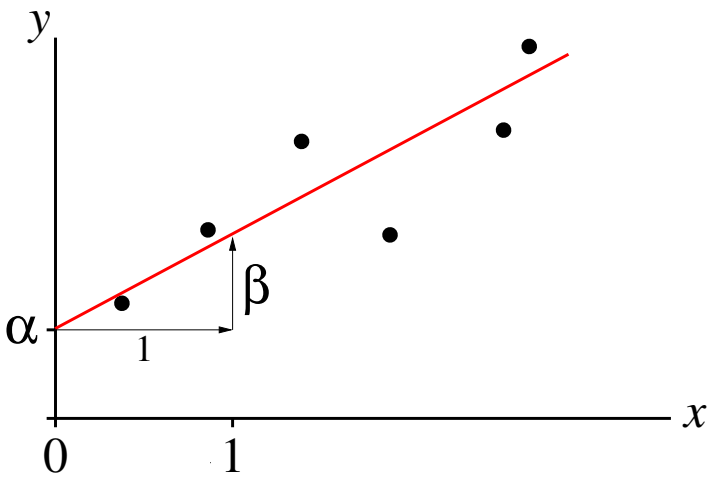
$$Y_i = \alpha + \beta x_i + \epsilon_i \,.$$

Here, $\alpha + \beta x_i$ is the ***explained part of the relationship.***

The values α and β are *parameters* that we want to estimate. Estimating these parameters is equivalent to ***finding the best-fit line through the scatterplot.***

The value α is the ***intercept:*** the intersection of the line with the y-axis when $x = 0$.

The value β is the ***slope:*** β specifies the increase in y for each unit increase in x. E.g. if $\beta = 1.2$, then we expect a person's weight, y, to increase by 1.2kg for every 1cm increase in their height, x, on average.



The explained, or systematic, part of the model is $\alpha + \beta x_i$. This describes the average relationship between Y and X. To be precise, it specifies the ***average value of Y, when X is fixed at x_i.***

There are various notations for the $\alpha + \beta x_i$ portion. We often write

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

where it is implicitly understood that μ_i ***is the average of*** Y_i***, when we know that our*** X***-observation is*** x_i***.***

Another notation is:

$$\mathbb{E}(Y_i \,|\, X = x_i) = \alpha + \beta x_i.$$

This is the precise statistical formulation: we are looking at the *mean*, or *expected value*, of the distribution of Y, given our knowledge that $X = x_i$.

In the height-weight example, $\mathbb{E}(Y_i \,|\, X = x_i)$ describes the *average weight for height* x_i. For example $\mathbb{E}(Y_i \,|\, X = 175\text{cm})$ is the average weight for people of height 175cm.

Sometimes this is abstracted further and written as $\mathbb{E}(Y \,|\, X) = \alpha + \beta X$.

Scatter model, or 'Error model'

The second part of our model formula is ϵ_i, the ***'scatter term' or 'error term'.***
Remember

$$Y_i = \alpha + \beta x_i + \epsilon_i.$$

We know that variable Y cannot be *completely* explained by variable X: there will always be some scatter, or what we call 'residual variation'. The word 'residual' is important: it means ***what's left over*** after we have described the systematic relationship between Y and X as best we can — i.e. by selecting the best possible α and β to give the best-fit line.

Selecting an appropriate scatter model is an important part of statistical modelling. To begin with, we will focus on one type of scatter model:

$$\epsilon_i \sim \text{Normal}(0, \sigma^2).$$

This means that all observations Y_i are assumed to have the ***same scatter*** about the line, regardless of where on the line they are. You can see this because σ^2 ***does not depend upon*** i ***or on*** x_i***: it is constant.***

The parameter σ^2 describes the ***amount of scatter.*** The precise definition of σ^2 is the mean squared deviation of a point from the best-fit line.

Although σ^2 is another parameter we will have to estimate, it is typically not of very much interest in its own right. We call it a *nuisance parameter.*

We also assume that all errors ϵ_i are *independent* of each other. This means that ***none of the observations in the study are influenced by other observations.***

This typically makes sense: my weight is not influenced by yours.

The two assumptions of independence and constant scatter are encapsulated in the statement:

$$\epsilon_i \sim \text{iid Normal}(0, \sigma^2),$$

where iid stands for ***independent and identically distributed.***

We will use this scatter model a lot during the course, but we will also see other scatter models where the amount of scatter does not stay constant along the line.

Best-fit line

Our model is $Y_i = \alpha + \beta x_i + \epsilon_i$. When we write down a model, we are imagining that this is what really happened to generate the response Y_i given the predictor x_i. Our task now is to select the 'best' values of α and β: the intercept and slope that give the best-fit line through our scatterplot.

Our selected values of α and β are called our ***estimates.*** We write estimates with a hat, i.e. $\widehat{\alpha}$ ***and*** $\widehat{\beta}$***.***

Thus:

- α and β are ***true, unknown parameter values.***
  They are the slope and intercept that we believe generated the data in the first place. We believe that they have these true but unknown values in the wider population of interest. In real life we will never know what they are exactly, but we will use our sample data to draw inference on them.
- $\widehat{\alpha}$ and $\widehat{\beta}$ are ***our estimates of*** α ***and*** β ***based on our sample data.***
  These are known numbers computed from our sample to give the 'best' line possible. We need to specify some criterion for what we mean by 'best' line.

  Once we have our estimates $\widehat{\alpha}$ and $\widehat{\beta}$, we can specify the mean relationship we have estimated between Y and X:

$$\begin{aligned} & & \widehat{\mu}_i &= \widehat{\alpha} + \widehat{\beta} x_i \\ \text{Or:} & & \widehat{\mathbb{E}}(Y_i \,|\, X = x_i) &= \widehat{\alpha} + \widehat{\beta} x_i \end{aligned}$$

  The values $\widehat{\mu}_i = \widehat{\alpha} + \widehat{\beta} x_i$ are called the ***fitted values or predicted values.*** They trace the path of the best-fit line.

The fitted values $\widehat{\mu}_i = \widehat{\alpha} + \widehat{\beta} x_i$ are often used as ***predictions:*** they specify the 'expected' value of Y for a given value of X.

For example, if we fitted the linear model Weight$= \alpha + \beta \times$ Height and obtained $\widehat{\alpha} = -115$ and $\widehat{\beta} = 1.1$, then we would make the following predictions:

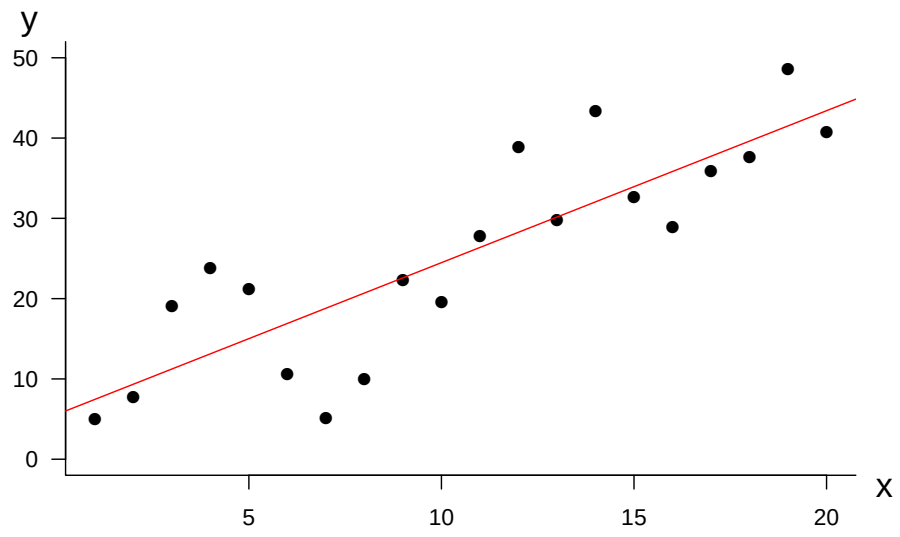
- Predicted weight at height 170cm: $-115 + 1.1 \times 170 = 72$***kg.***
- Predicted weight at height 180cm: $-115 + 1.1 \times 180 = 83$***kg.***

The 'prediction' is our estimated mean: we estimate that a person of height 170cm will weigh 72kg on average.
The value 72kg is our ***fitted value when*** $X = 170$***cm.***

Question: Suppose our model $Y_i = \alpha + \beta x_i + \epsilon_i$ is really true in the population at large. We draw two different samples from this population. Which of the following statements is true?

A. α and β are the same for both samples, but $\widehat{\alpha}$ and $\widehat{\beta}$ will be different.
B. $\widehat{\alpha}$ and $\widehat{\beta}$ are the same for both samples, but α and β will be different.
C. All of α, β, $\widehat{\alpha}$, and $\widehat{\beta}$ will be the same for the two samples.

Best-fit line in the simple linear model



For the simple linear model, we select $\widehat{\alpha}$ and $\widehat{\beta}$ using ***least-squares:***

$$\widehat{\alpha} \text{ and } \widehat{\beta} \text{ minimize } \sum_{i=1}^{n} \left\{ y_i - (\widehat{\alpha} + \widehat{\beta} x_i) \right\}^2 .$$

Equivalently: $\widehat{\alpha}$ and $\widehat{\beta}$ minimize $\sum_{i=1}^{n} (y_i - \widehat{\mu}_i)^2$: the sum of the squared residuals.

Why do we use the least-squares criterion?

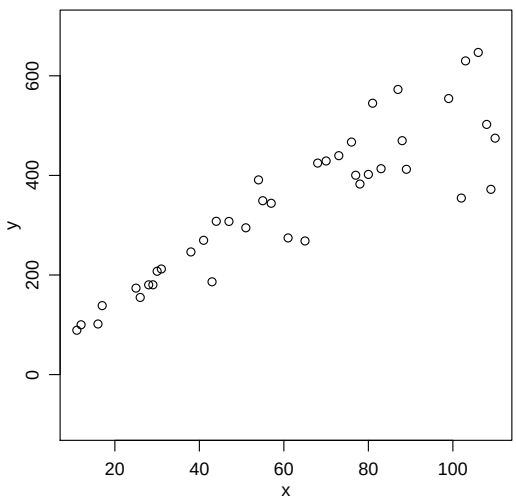
The least-squares criterion seems like a sensible way of selecting $\widehat{\alpha}$ and $\widehat{\beta}$, but you might think there are other sensible ways too. For example, why do we minimize the sum of *squared* distances from the points to the line, $\sum_i (y_i - \widehat{\mu}_i)^2$, instead of the sum of *absolute* distances, $\sum_i |\, y_i - \widehat{\mu}_i \,|$?

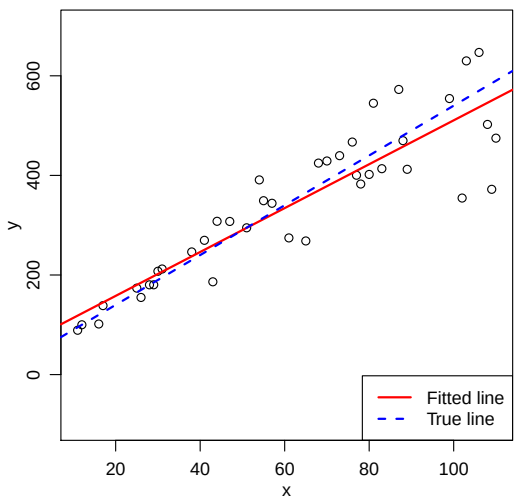
It is important to know the reason behind this, although we will not go into it in detail in Stats 20x.

> The least-squares criterion is appropriate for the simple linear model BECAUSE of the constant-scatter model, $\epsilon_i \sim \text{Normal}(0, \sigma^2)$.

In fact, least-squares is a special case of a general method for estimation called ***maximum likelihood estimation.*** Under our model $Y_i = \alpha + \beta x_i + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$, we can compute *how likely* our observed data y_i are for any value of α and β. We pick $\widehat{\alpha}$ and $\widehat{\beta}$ to maximize this likelihood: so $\widehat{\alpha}$ and $\widehat{\beta}$ are the values under which our data would have the 'best chance' of arising. It so happens that the maximum likelihood estimates (MLEs) for this linear model are the same as the least-squares estimates. We also obtain a MLE for σ, but it is not part of the least-squares sum. These ideas are covered in Stats 210.

We use maximum likelihood estimates because they have superb properties: for example, they are correct-on-average (unbiased), and have the narrowest possible confidence intervals (most precise). However, the method relies upon specifying our model correctly, and things will go wrong if our model is misspecified. The most likely thing to go wrong is confidence interval coverage: you might think you have a 95% confidence interval, but if your model is misspecified then your CIs might contain the true value less than 95% of the time.





This example shows non-constant scatter. The fitted line is pulled down on the right to fit the high-scatter points. With another dataset it might be pulled up too high. The variability of the fitted line is too high.

Fitting the simple linear model in R

Here we will see how to *simulate* data that exactly satisfies the linear model, and then fit the model in R.

First create your simulated data frame. The command below creates a data frame with column $x = 1, 2, \ldots, 20$, and column y such that $y_i = 5 + 2x_i + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, 3^2)$. Thus, we are simulating data y from the model with $\alpha = 5$, $\beta = 2$, $\sigma^2 = 9$.

```
mydat <- data.frame(x=1:20, y=5+2*(1:20)+rnorm(20, 0, 3))
```

Note how we have used the R command for generating random numbers from the Normal distribution: ***rnorm(n, mean, sd).***
Be careful to specify `sd`, the standard deviation: sd $= \sigma = \sqrt{\sigma^2}$. If you enter `sd=3`, then you are specifying a standard deviation of $\sigma = 3$, and a variance of $\sigma^2 = 9$.

Have a look at the whole data-frame you have just generated. Type:

```
mydat
```

You should see output like this. Your values for y will differ, but they will show an increasing trend with x.

```
     x         y
1    1  4.574243
2    2  3.290030
3    3  5.926182
4    4 16.052606
:    :       :
20  20 40.847702
```

To look at just the first few rows of a large dataframe, use the command `head`:

```
head(mydat)
```

Now:

```
## Plot the data:
plot(y~x, data=mydat)
```

```
## Fit the linear model, and assign the output to R object 'myfit'.
## Command 'lm' stands for 'linear model'.
myfit <- lm(y~x, data=mydat)

## Look at the estimated values, alpha-hat and beta-hat.
## These are called 'coefficients' in R, and are extracted by
## command 'coef':
coef(myfit)
```

I get the following output:

```
(Intercept)           x
   7.000682    1.929860
```

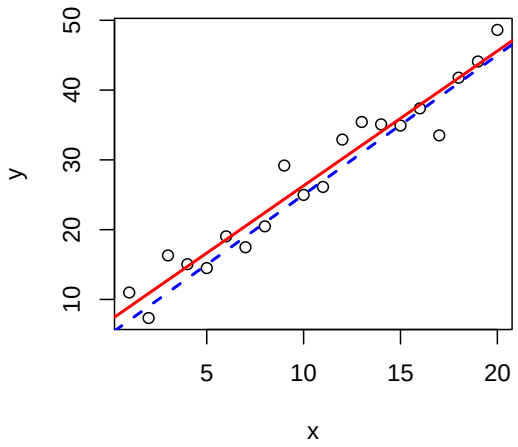
Are these good estimates? What values would you expect or hope to see?
Not bad, but not great: true values are 5 and 2; we've got estimates of 7.0 and 1.9.

```
## Draw the fitted line in red.
## Command 'abline' draws a straight line y = a + bx,
## where a is the first element of the vector provided,
## and b is the second.
## Here, the vector provided is coef(myfit).
abline(coef(myfit), col="red")
```

Because we are working with simulated data, we know the true relationship between x and y in this case. Draw the true line onto the graph with a blue dashed line.

```
abline(5, 2, col="blue", lty=2)
## Using 'lty=2' makes it dashed.  It stands for 'line-type 2.'
## You can alternatively specify the arguments as a vector
## c(5, 2), like this:
abline(c(5, 2), col="blue", lty=2)
```

I get the output at the right. Not too bad!



Here is another go for you to copy-and-paste into R. This time we'll use a larger sample size of 40 instead of 20:

```
## Generate new data and call it 'mydat':
mydat <- data.frame(x=1:40, y=5+2*(1:40)+rnorm(40, 0, 3))
## Plot:
plot(y~x, data=mydat)
## Fit the linear model:
myfit <- lm(y~x, data=mydat)
## Plot the fitted line in red:
abline(coef(myfit), col="red")
## Plot the true line in blue dashed:
abline(5, 2, col="blue", lty=2)
## Look at the coefficients:
coef(myfit)
```

I get the following:

```
(Intercept)           x
   5.417994    1.953421
```

Better this time — as we'd expect with a larger sample size.
Your estimates:

Checking assumptions using residual plots

Under our model,

$$Y_i = \mu_i + \epsilon_i\,, \quad \text{where} \quad \epsilon_i \sim \text{iid Normal}(0, \sigma^2)\,.$$

So

$$Y_i - \mu_i = \epsilon_i \sim \text{iid Normal}(0, \sigma^2)\,.$$

We can use this to check the assumptions of our model. We don't know the true values of $\mu_i = \alpha + \beta x_i$, but we can replace them with our estimated values $\widehat{\mu}_i = \widehat{\alpha} + \widehat{\beta} x_i$. Under the fitted model, if our assumptions are valid, we should have

$$Y_i - \widehat{\mu}_i \sim \textbf{\textit{approx iid Normal}}(0, \sigma^2).$$

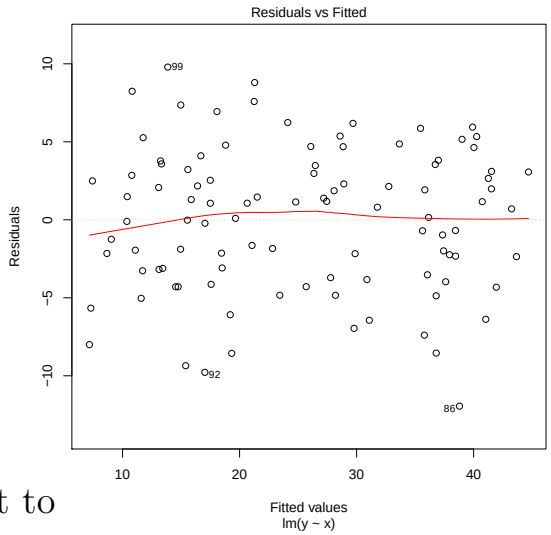
In other words, ***the RESIDUALS, $Y_i - \widehat{\mu}_i$, should be approximately Normally distributed, with constant scatter and no trend.***

We can check whether this is the case by looking at plots of residuals. We should always do this before interpreting the results from our model fit, because if our assumptions aren't valid then our results will not be valid either!

First look to see whether the residuals seem to be approximately trendless with constant scatter. Because the fitted values represent the mean of the distribution of Y, it makes sense to plot the fitted values on the x-axis, so we can check that the residuals are Normally distributed at each point on the fitted line. This corresponds to checking that Y_i is Normally distributed about its mean, regardless of what the mean is.

Here is the command to check the residual plot in R for model `myfit`:

```
## Generate 100 random values of x:
mydat <- data.frame(x=runif(100, 1, 20))
## Generate y = 5 + 2x + epsilon:
mydat$y <- 5 + 2*mydat$x + rnorm(100, 0, 5)
## Plot the data:
plot(y~x, data=mydat)
## Fit the model:
myfit <- lm(y~x, data=mydat)

## Plot residuals versus fitted values:
plot(myfit, which=1)
```



Using `plot(myfit, which=1)` is equivalent to `plot(fitted(myfit), residuals(myfit))`.

Check this by overplotting the latter on top of the current plot using the `points` command with blue crosses:

```
points(fitted(myfit), residuals(myfit), col="blue", pch=4)
```

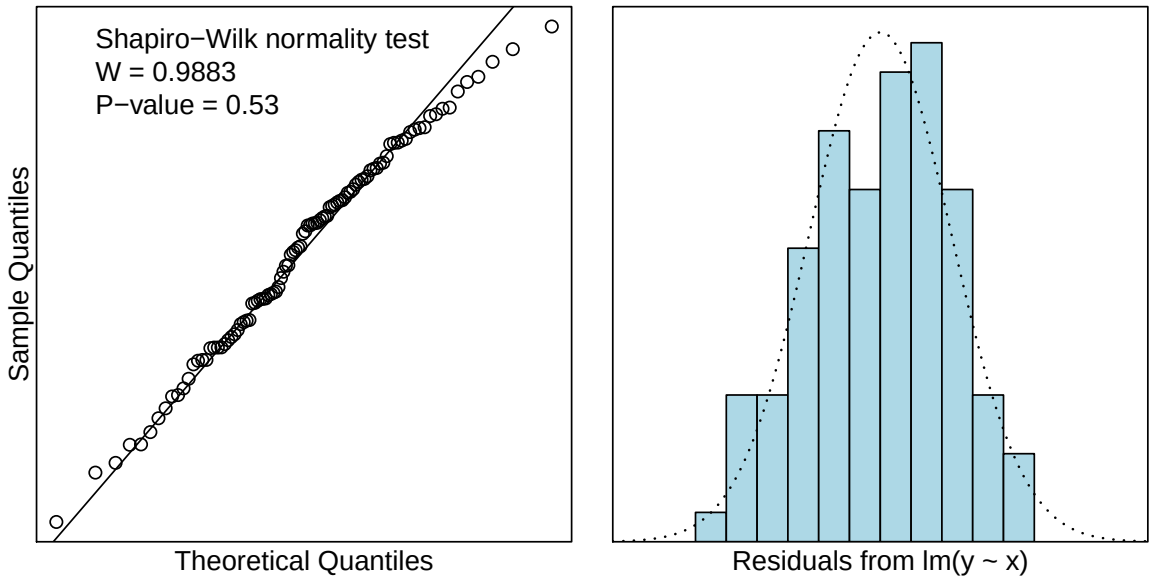
These residuals look fine. There is no systematic trend or obvious change in scatter. By default, R labels the three most extreme points. They do not appear to be outliers in this case.

Having checked for ***patternless*** residuals, we should now do a couple of checks for ***normality*** of residuals. The Stats 20x team have written a library of useful functions which includes `normcheck` for this purpose.

```
library(s20x)
normcheck(myfit, shapiro.wilk=T)
```

You only need to type `library(s20x)` once in every R session: once the library is loaded it is available for the rest of the session.

Using `shapiro.wilk=TRUE` includes a p-value from the Shapiro-Wilk test for normality on the plot.



The first plot from `normcheck` is called a QQ-plot (Quantile-Quantile). It plots the quantiles of our model residuals against the quantiles of a Normal distribution. Roughly speaking, this means that the smallest of our 100 observations should be plotted at the 1% quantile of the Normal distribution; the second smallest at the 2% quantile, and so on. (There is a slight adjustment so the 1st is actually plotted at the 0.5% quantile and the 100th at the 99.5% quantile.)

If the Normal distribution is a good fit, the QQ-plot should be approximately ***a straight line.***

You can check where this plot comes from as follows:

```
## Use the R function qqnorm to find the QQ-plot:
qq.res <- qqnorm(residuals(myfit))
## Plot just the first of the two plots in normcheck:
normcheck(myfit, shapiro.wilk=T, whichPlot=1)
## Overplot the QQ-plot results with blue crosses: they should be identical.
points(y~x, data=qq.res, col="blue", pch=4)
## Check the Shapiro-Wilk p-value supplied against qq.res$y:
shapiro.test(qq.res$y)
```

The p-value of 0.53 is ***large.***

This means ***there is no evidence against the Normal assumption for this data set.***

The second plot in `normcheck` is just a histogram of the residuals, with a Normal density curve plotted over the top. It can be reproduced as follows:

```
hist(residuals(myfit), col="lightblue")
```

Ideally, the histogram should follow the curve quite well. However, the most important thing is that the histogram is reasonably ***symmetric.*** It is not of great concern if it is *short-tailed* relative to the curve, as long as it is symmetric.

Here is the code to scale the histogram so the Normal density can be overplotted:

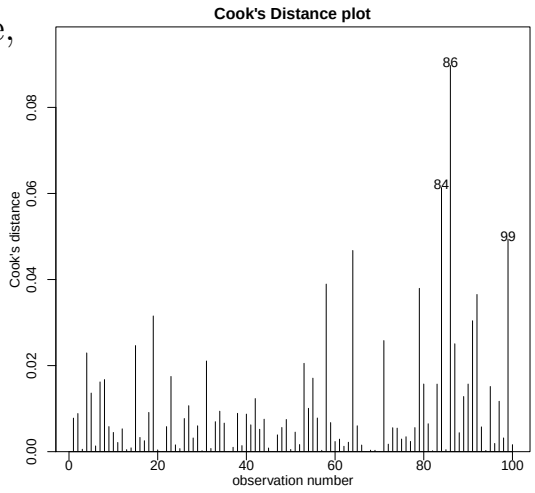
```
hist(residuals(myfit), col="lightblue", probability=T)
## Extract the estimated sigma-hat using summary(myfit)$sigma:
sigmahat <- summary(myfit)$sigma
## Overplot the Normal density curve using dnorm:
lines(seq(-15, 15, by=0.5), dnorm(seq(-15, 15, by=0.5), 0, sigmahat))
```

Check for outliers and points of influence

Another useful `s20x` function is `cooks20x`. This checks for data points of high influence, such that removing the point would change the fitted values substantially. Use:

```
cooks20x(myfit)
```

R labels the three most influential points. A rule of thumb is to inspect any points with a Cook's distance greater than 0.4. It's possible that these points are errors, e.g. due to false data entry, or that they should be removed for some other reason.



1.3 Is there a real relationship between X and Y?

One of our primary goals is to decide whether the evidence in our sample points to a real relationship between X and Y. Is lung cancer *really* associated with smoking, or is it possible that we just had some lucky nonsmokers and unlucky smokers in our sample? At what point can we say that our sample result is so conclusive that it points to a real effect in the population at large?

We can cast this question in terms of ***probabilities.*** If we draw our sample sensibly from the population — e.g. by drawing all subjects randomly and independently — then there are limits to how convincing a fluke trend can look, if there really is no relationship between X and Y. We can compare the trend in our real data to these 'believability limits', and if our trend looks more convincing than any fluke trend born of lucky sampling, then we can conclude ***we have evidence of a real relationship between X and Y.***

The way we do this is to first set up a model in which there is ***no relationship between X and Y:***

$$Y_i = \alpha + \epsilon_i \quad \text{where } \epsilon_i \sim \text{iid Normal}(0, \sigma^2).$$

This indicates that Y_i ***does not depend upon*** x_i**:** it says that $\mathbb{E}(Y_i \,|\, x_i) = \alpha$, which is the same for all x_i. We call this model the ***null model.*** The word 'null' means 'neutral' or 'noncomittal': the model in which nothing is happening.

Compare the null model with our previous model:

$$\begin{aligned} Y_i &= \alpha + \epsilon_i \ : && \text{null model : } X \text{ and } Y \text{ have no relationship.} \\ Y_i &= \alpha + \beta x_i + \epsilon_i \ : && \text{proposed model: } X \text{ and } Y \text{ do have a relationship.} \end{aligned}$$

Clearly, the difference between these two models is that ***the null model specifies that*** $\beta = 0$**.**

So our job is to conduct the following hypothesis test:

$$\begin{aligned} \text{Let} \quad Y_i &= \alpha + \beta x_i + \epsilon_i \quad \text{where } \epsilon_i \sim \text{iid Normal}(0, \sigma^2) \\ H_0 &: \beta = 0 \\ H_1 &: \beta \neq 0 \end{aligned}$$

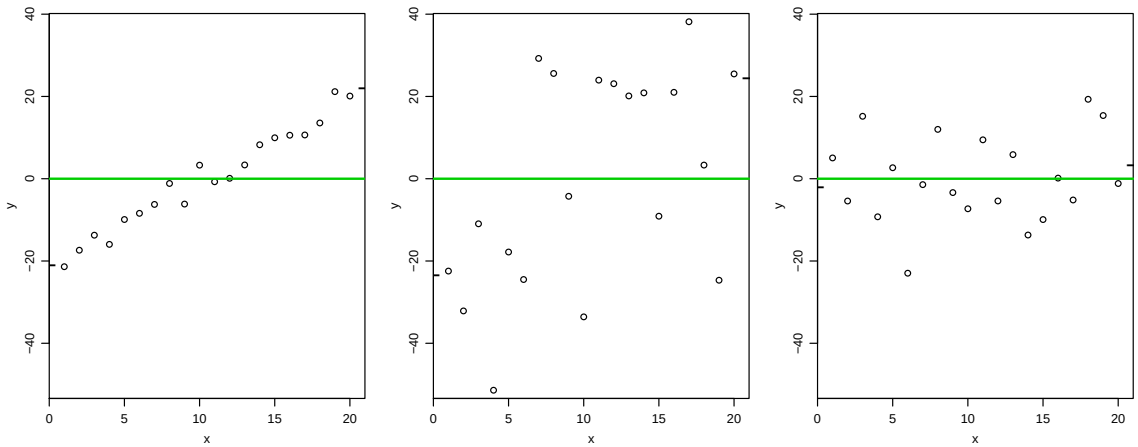
We can compare the data we observe in our sample against ***the sort of data we would obtain if*** $\beta = 0$**.**

If our data are too ***improbable*** for a sample drawn from a population with $\beta = 0$, then we have evidence that $\beta \neq 0$.

We measure evidence using the p-value. ***The*** p***-value measures how likely it is for a trend as clear as the one seen in our sample to arise by chance, if there is really no relationship between*** X ***and*** Y ***in the population at large.***

Do the examples below look as if they could have arisen just by chance from the null model? What sort of p-values would you expect?
The null model is plotted by the green horizontal line.

$p = 0.00 (1.7 \times 10^{-14})$ $\qquad$ $p = 0.02$ $\qquad$ $p = 0.56$



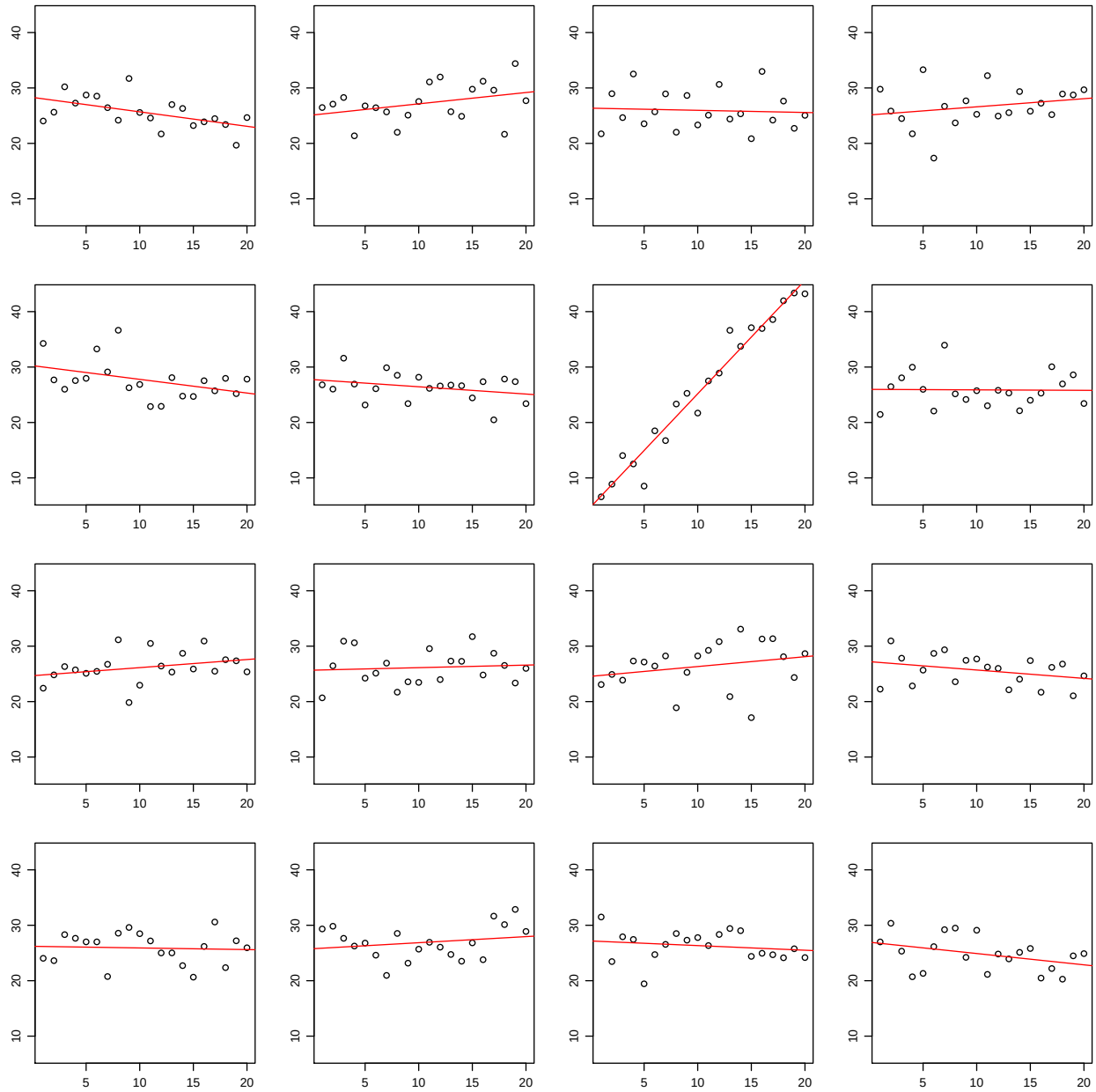
The following graphics show sixteen (x, y) scatterplots.

- Fourteen of them were drawn from the null model, $Y_i = \alpha + \epsilon_i$.
- One was drawn from the trend model $Y_i = \alpha + \beta x_i + \epsilon_i$ with $\beta = 2$.
- One was drawn from the trend model $Y_i = \alpha + \beta x_i + \epsilon_i$ with $\beta = 0.2$.

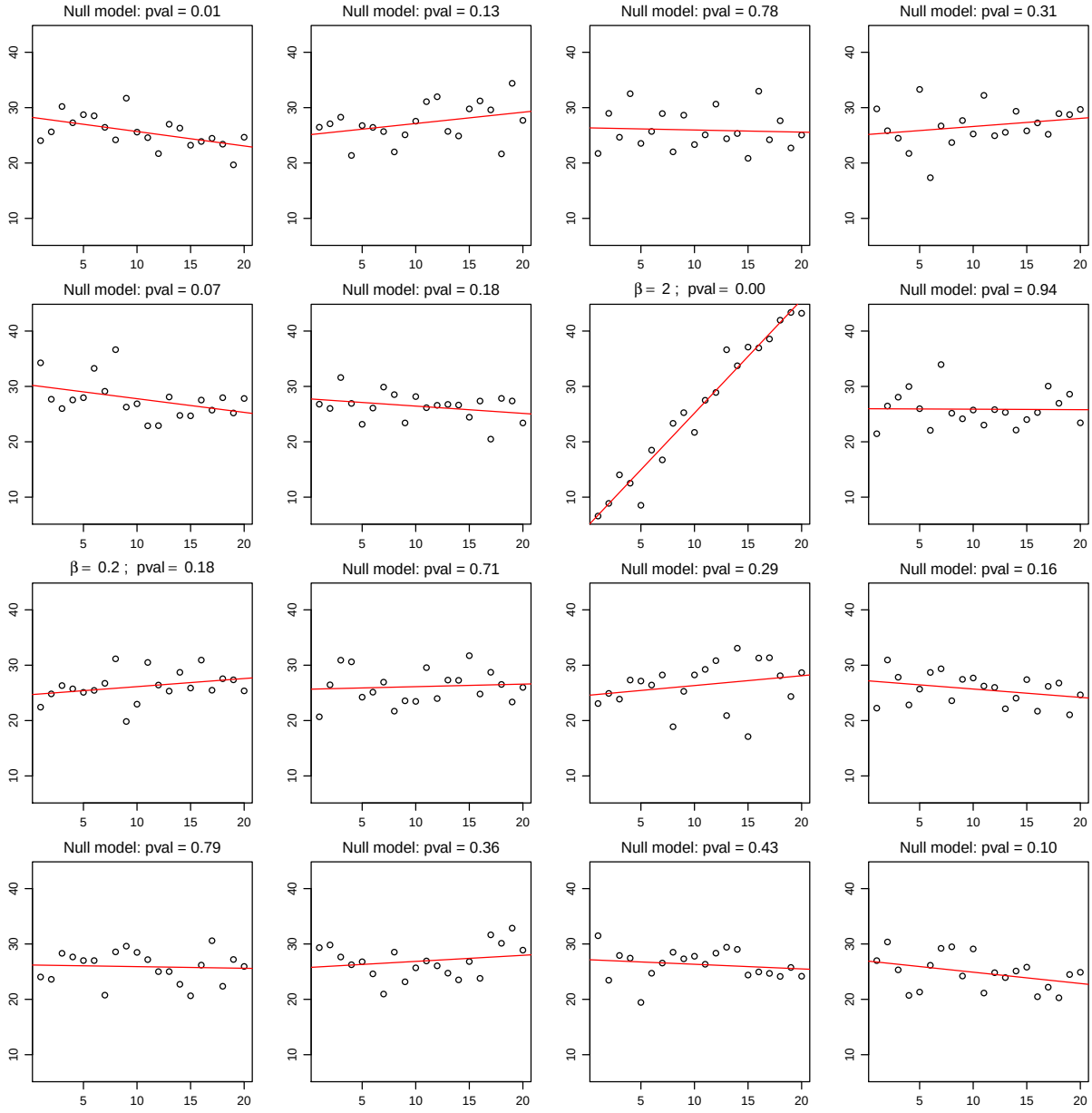
The values of α vary a little to make all plots fit neatly within the same range.

Which one is the odd one out? Can you see why we say that the trend is ***significant,*** to indicate that it is not the sort of data we would expect to see by chance from the null model?

Can you pick out the plot for $\beta = 2$? What about for $\beta = 0.2$?



Here are the same plots again, marking the two β-models and the p-values.



How do we get the p-value output for the $\beta = 2$ data in R?

```
mydat <- data.frame(x=1:20, y=5+2*(1:20)+rnorm(20, 0, 3))
myfit <- lm(y~x, data=mydat)
## Generate the table of estimates and p-values:
summary(myfit)
```

This gives:

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.71190    1.19621   3.939 0.000962 ***
x            2.04667    0.09986  20.496 6.31e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Run the same thing again using data simulated from the null model:

```
nulldat <- data.frame(x=1:20, y=                    )
nulldat.fit <- lm(y~x, data=nulldat)
summary(nulldat.fit)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.15181    1.67183   3.082  0.00643 **
x           -0.03885    0.13956  -0.278  0.78388
```

Which panel on the previous page does this simulation correspond to? Notice that the estimated trend is negative ($\widehat{\beta} = -0.03885$), but of small magnitude.

The p-value for the hypotheses $H_0 : \beta = 0$ against $H_1 : \beta \neq 0$ is $p = 0.78$. This p-value is large and is ***non-significant.***
The non-significance is indicated by the fact there are no **s beside it on the R output.

The interpretation of this p-value is: ***we have no evidence against*** H_0 ***for our dataset 'nulldat'.*** If the assumptions of the linear model are satisfied, there is a high probability (0.78) of seeing a fitted trend as strong as ours, just by chance, if H_0 is true and $\beta = 0$. ***Our sample data are consistent with the null model. There is no evidence in this sample of a relationship between*** X ***and*** Y***.***

What about the example in the first panel overleaf? This was genuinely drawn from the null model, and yet it gave a p-value of 0.01.

```
nulldat2 <- data.frame(x=1:20, y=5+rnorm(20, 0, 3))
nulldat2.fit <- lm(y~x, data=nulldat2)
summary(nulldat2.fit)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.09763    1.14338   6.208 7.39e-06 ***
x           -0.26082    0.09545  -2.733   0.0137 *
```

The conclusion from this p-value of 0.01 is that ***we DO have evidence against*** H_0 ***— even though we know in this case that*** H_0 ***is true!***

Unfortunately, this will happen from time to time: it is a ***false positive or 'Type I Error'.***

The interpretation of the p-value of 0.01 is that an apparent trend as strong as the one seen in `nulldat2` will arise from the null model, just by chance, with a probability of 0.01. This is about ***one time in a hundred.***

It is quite unusual — but not staggeringly unusual. Things that happen 0.01 of the time do happen in everyday life. Lots of events with that level of rarity will happen around you today. (For example — the probability that today is the 9th January is a lot less than 0.01, but you are not so very surprised when it happens!)

In fact, we can be even more precise. You will get a 'false-positive' p-value of 0.01 or less exactly 1% of the time. That is, ***1% of datasets drawn from the null model will appear to provide evidence against*** H_0 ***with a*** p***-value*** ≤ 0.01***.***

Similarly, 5% of datasets drawn from the null model will appear to provide evidence against H_0 with a p-value ≤ 0.05. This is the definition of a p-value.

We say we have evidence against $H_0 : \beta = 0$ at the 5% level if $p \leq 0.05$.
This has a false-positive rate of 5%, by definition.

5% of samples GENUINELY drawn from H_0 will give $p \leq 0.05$.
1% of samples GENUINELY drawn from H_0 will give $p \leq 0.01$.

Saying 'we have evidence against H_0' is NOT the same as saying H_0 must be false.

In fact, the phrase 'we have evidence against H_0', is statistical code for:
EITHER H_0 ***is true and we have observed something quite unusual, OR*** H_0 ***is false.***

As the p-value gets smaller (0.05 down to 0.01, down to 0.001, 0.0001, etc), the interpretation shifts: we are progressively less likely to believe in the first option (H_0 is true and we saw something unusual), and more likely to believe in the second (H_0 is false). However, it is only for VERY small p-values that we can exclude H_0 entirely.

Finally, what went wrong for the $\beta = 0.2$ example? In this case, there was a genuine trend, but the p-value was non-significant:

```
missed.dat <- data.frame(x=1:20, y=5+0.2*(1:20)+rnorm(20, 0, 3))
missed.fit <- lm(y~x, data=missed.dat)
summary(missed.fit)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.1181     1.2587   4.861 0.000126 ***
x             0.1462     0.1051   1.392 0.180986
```

This time, the trend is ***real, but undetected.***
This is a 'false negative', or 'Type II Error'.

The slope coefficient, $\beta = 0.2$, is too weak compared with the amount of scatter, $\sigma^2 = 3^2$, and the sample data don't look sufficiently different from the sort of data that would be generated under the null model.

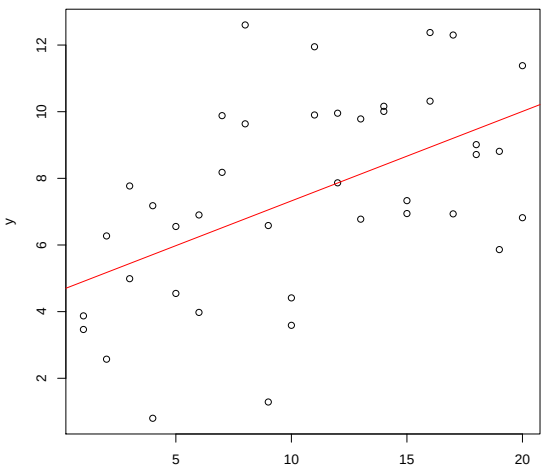
We can only remedy this case by ***increasing our sample size.***

The following commands generate a second data frame from the same model as the original `missed.dat`, using $\beta = 0.2$, and join it to the first using the `rbind` command, where `rbind` stands for 'row-bind'.

```
better.dat <- rbind(missed.dat,
                data.frame(x=1:20, y=5+0.2*(1:20)+rnorm(20, 0, 3)))
better.fit <- lm(y~x, data=better.dat)
plot(y~x, data=better.dat)
abline(coef(better.fit), col="red")
summary(better.fit)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.03996    0.84228   5.984 5.99e-07 ***
x            0.19867    0.07031   2.826  0.00748 **
```

There is now strong evidence against H_0: the p-value is ***0.0075.***

Even though the trend is still weak ($\beta = 0.2$), the combined force of 40 data-points creates a sample that would hardly ever be seen under the null model — only about $0.0075 = 3/400$ null data sets would exhibit a trend this convincing, just by chance.



With our larger sample size, we have now generated strong evidence that there is a relationship between X and Y.

Where does the *p*-value come from?

The theory underlying the p-value for $H_0 : \beta = 0$ versus $H_1 : \beta \neq 0$ is covered in Stats 210 and Stats 310. However, you should recognise the general formulation:

$$\frac{\text{Estimate} - \text{Hypothesized value}}{\text{Standard error of estimate}} \sim t_{\text{df}}.$$

The ***standard error of an estimate*** is an estimate of *how uncertain we are* when we estimate the parameter from our sample data. One of the marvellous things about statistics is that we can use the same data both to *estimate* a parameter, and to say *how reliable our estimate is* for that parameter. The theory is grounded in the Central Limit Theorem and also some of the excellent properties of maximum likelihood estimation.

Let's have another look at the two datasets just above. They were both drawn from a model with a weak trend ($\beta = 0.2$), but `missed.fit` was based on a sample with only 20 data points, whereas `better.fit` was based on 40.

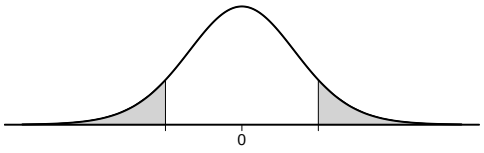
```
summary(missed.fit)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.1181     1.2587   4.861 0.000126 ***
x             0.1462     0.1051   1.392 0.180986

summary(better.fit)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.03996    0.84228   5.984 5.99e-07 ***
x            0.19867    0.07031   2.826  0.00748 **
```

In the first case, `missed.fit` had an estimate of $\widehat{\beta} = 0.1462$ with standard error 0.1051. This is ***very poor. The standard error is almost as big as the estimate!***

If we use the rule of thumb that a 95% confidence interval is approximately ***estimate*** $\pm$ ***two standard errors***, then our confidence interval would extend from $0.1462 - 2 \times 0.1051 = -0.064$, ***to*** $0.1462 + 2 \times 0.1051 = 0.3564$.

So we don't even know if β is positive or negative!

(This is actually equivalent to a non-significant p-value against $H_0 : \beta = 0$ at the 5% level. The p-value is > 0.05 if and only if the 95% confidence interval contains 0.)



The t-statistic for testing $H_0 : \beta = 0$ is:

$$\frac{\text{Estimate} - \text{Hypothesized value}}{\text{Standard error of estimate}} = \frac{0.1462 - 0}{0.1051} = 1.392.$$

If H_0 is true, then statistical theory tells us that this statistic has a t-distribution, with parameter $\text{df} = n - 2 = 20 - 2 = 18$. (We subtract 2 from the number of data points, because we are estimating 2 parameters, α and β. This leaves us with $n - 2$ 'degrees of freedom' in the residual variability.)

The p-value is then calculated from this t_{18} distribution:

`2 * pt(1.392, df=18, lower.tail=FALSE) = 0.1809856`

These values match the R output, with slight differences from rounding errors.

By contrast, for `better.fit`:

$$\frac{\text{Estimate} - \text{Hypothesized value}}{\text{Standard error of estimate}} = \frac{0.19867 - 0}{0.07031} = 2.826,$$

and now we have $n = 40$ data points, so we use df= 38:

`2 * pt(2.826, df=38, lower.tail=FALSE) = 0.00748`

In general, the t-statistic is telling us ***how many standard errors away from the hypothesized value our estimate is.***

If our estimate is too many standard errors away from the hypothesized value $\beta = 0$, it is no longer believable that $\beta = 0$ is the true value. The p-value from the t-test tells us ***how believable it is.***

If the p-value is very small, our data are not believable for the null model, and we have ***evidence against*** H_0**.**

p-value summary

- We began this section by asking, *Is there a real relationship between X and Y?* This translates to the hypothesis test $H_0 : \beta = 0$ versus $H_1 : \beta \neq 0$.
- If $\beta = 0$ and our data really came from the null model, there are limits to how convincing a 'fake trend' can look. We want to compare the trend in our data to these 'believability limits'.
- This means we need to *measure something* about how convincing our trend looks. The thing we measure is called the *test statistic*, in this case a t-statistic. It measures *how many standard errors is $\widehat{\beta}$ away from the hypothesized $\beta = 0$.*
- Under H_0, we know the *probability* of a result as extreme as ours. This is the p-value. If p is very small, our result is very unlikely to arise from the null model, so we have *evidence against H_0*. Our trend is too convincing to be 'fake'.

1.4 Confidence intervals and prediction intervals

One of the key features that distinguishes statistics as a science is that we don't just provide estimates like $\widehat{\alpha}$ and $\widehat{\beta}$: we also provide information about ***how reliable these estimates are.***

Without including reliability measures, estimates by themselves are almost worthless. The primary way of describing the reliability of our estimates is to include ***confidence intervals.***

R provides us with confidence intervals using the command ***confint.***

Let's find CIs for simulated data using $\alpha = 5, \beta = 2, \sigma = 3$:

```
## Generate data and fit the linear model:
mydat <- data.frame(x=1:20, y=5+2*(1:20)+rnorm(20, 0, 3))
myfit <- lm(y~x, mydat)
## Extract the 95% confidence intervals:
confint(myfit)
```

```
                2.5 %   97.5 %
(Intercept) -0.7903782 5.487531
x            1.9038630 2.427933
```

This means the confidence interval for α is ***(-0.79, 5.49)***
and the confidence interval for β is ***(1.90, 2.42).***

In this case, both of the confidence intervals contain the true values: $\alpha = 5$ and $\beta = 2$. The CI for β is much narrower than that for α, indicating that the slope, β, has been estimated with much better precision than the intercept, α.

The CIs have been returned as a matrix of results. To extract the CI just for β, use the R subsetting command to ask for the second row only:

```
confint(myfit)[2,]
```

```
   2.5 %   97.5 %
1.903863 2.427933
```

Will it always happen that the 95% confidence interval encloses the true value $\beta = 2$?

No: the idea of a confidence interval is that it should enclose the true value exactly 95% of the time.

What do we mean by 'exactly 95% of the time?'

Every sample dataset will give us a different interval. We mean that 95% of times that we take a sample, fit the model, and construct the confidence interval, this interval will enclose the true value $\beta = 2$.

The ***interval is random, while $\beta = 2$ stays the same.*** So we should try running these commands many times over and see how the intervals change:

```
mydat <- data.frame(x=1:20, y=5+2*(1:20)+rnorm(20, 0, 3))
myfit <- lm(y~x, mydat)
confint(myfit)[2,]
```

How can we quickly see if the interval does contain the true value $\beta = 2$?

```
beta.CI <- confint(myfit)[2,]
## Ask whether the following logical statement is true:
(beta.CI[1] <= 2) & (2 <= beta.CI[2])
    2.5 %
    TRUE

## Convert to a number, 0 or 1, and remove the unwanted name:
as.numeric((beta.CI[1] <= 2) & (2 <= beta.CI[2]))
    [1] 1
```

If you run the code above a few times, you'll see the occasional time when the CI doesn't enclose the true value. However, we need to repeat it thousands of times to see if the 95% coverage property holds exactly. This is where it becomes very helpful to write your own ***functions.*** Here is a simple function that takes inputs α, β, σ, simulates one data set, and reports whether the CI for β contains the true value.

```
one.confint.func <- function(alpha=5, beta=2, sigma=3){
      ## Generate the data and fit the linear model:
      mydat <- data.frame(x=1:20, y=alpha+beta*(1:20)+rnorm(20, 0, sigma))
      myfit <- lm(y~x, mydat)
      ## Extract the 95% confidence interval for beta:
      beta.CI <- confint(myfit)[2,]
      ## Find whether the confidence interval encloses the true beta:
      CI.res <- as.numeric(beta.CI[1]<=beta & beta<= beta.CI[2])
      ## Print output:
      cat("95% Confidence interval:", format(round(beta.CI, 2), nsmall=2), "\n")
      cat("Contains beta=", beta, "? ", CI.res, "\n")
      ## Finish the function by returning 0 or 1 for whether the CI encloses beta:
      return(CI.res)
}
```

To run this function:

1. *Either:* copy-and-paste the text directly into R;
   Or: save the code in a text file called `mycode.R` and put it in your working directory, then type `source("mycode.R")` to read it into R.
2. Run the code by typing: `one.confint.func()`

By running this many times, you'll see that the CI mostly does, but sometimes doesn't, contain the true β. We need a better function that can run this code thousands of times and report back the exact frequency with which the CI encloses the true β. The function below takes inputs α, β, σ, as before, and also the desired number of simulations, `Nsim`. It then returns the ***CI coverage:*** how many simulations yielded CIs that covered the true β.

```
confint.func <- function(alpha=5, beta=2, sigma=3, Nsim=1000, printit=T){
   ## Create a vector of results:
   CI.res <- numeric(Nsim)
   ## Let simulation i run from 1 to Nsim:
   for(i in 1:Nsim){
      ## Generate the data for this simulation, and fit the linear model:
      mydat <- data.frame(x=1:20, y=alpha+beta*(1:20)+rnorm(20, 0, sigma))
      myfit <- lm(y~x, mydat)
      ## Extract the 95% confidence interval for beta:
      beta.CI <- confint(myfit)[2,]
      ## Find whether the confidence interval encloses the true beta,
      ## and put the result in entry i of CI.res:
      CI.res[i] <- as.numeric(beta.CI[1]<=beta & beta<= beta.CI[2])
      ## Print output for simulation i, if requested via argument "printit":
      if(printit)
         cat("Confidence interval", format(round(beta.CI, 2), nsmall=2),
               "contains beta=", beta, "? ", CI.res[i], "\n")
   }
   ## We now have Nsim results stored in vector CI.res.
   ## Find out how many of the confidence intervals contained beta:
   cat("\n", sum(CI.res), "out of", Nsim,
         "data sets contained the true beta: coverage rate=",
         round(sum(CI.res)/Nsim, 2), "\n")
   ## Finish the function by returning the vector of results:
   return(CI.res)
}
```

Interpreting the confidence interval

The term *95% confidence interval* has a very specific meaning in statistics. It displays the ***precision with which we have estimated the parameters, α and β.***

Our ultimate aim is a narrow confidence interval: this means we can narrow down the parameter values to a small range.

Suppose we fit a linear model and obtain $\widehat{\beta} = 2.05$ with 95% confidence interval $(1.85, 2.25)$. Does this mean:

— The true value β is somewhere between 1.85 and 2.25?
No. We are quite confident it is, but there is a 5% chance our interval does not enclose β.

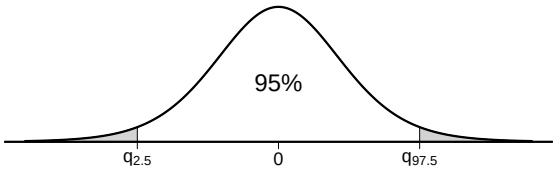
— There is a 95% probability that the true value β is between 1.85 and 2.25?
No: this is poor wording. The true β either is or isn't between 1.85 and 2.25: it can't be true 95% of the time.

— There is a 95% probability that our confidence interval encloses the true β?
Yes. It is correct to say that 95% of confidence intervals will enclose the true β. The probability is taken over all possible samples: each sample leads to its own CI, which will or will not contain β.

Where does the confidence interval come from?

The confidence interval comes from the same t-distribution as the p-value. The degrees of freedom is df $= n -$ (#parameters estimated). If β is the true value, then:

$$\frac{\text{Estimate} - \beta}{\text{Standard error of estimate}} \sim t_{\text{df}}\,.$$



In other words, $$\frac{\widehat{\beta} - \beta}{\text{se}(\widehat{\beta})} \sim t_{\text{df}}\,.$$

So we can find probability intervals for $\dfrac{\widehat{\beta} - \beta}{\text{se}(\widehat{\beta})}$.

Let $q_{2.5}$ be the 2.5% quantile of the t_{df} distribution, and $q_{97.5}$ the 97.5% quantile:

$$q_{2.5} = \texttt{qt(0.025, df)} = -q_{97.5}$$
$$q_{97.5} = \texttt{qt(0.975, df)}$$

Then the statement underlying the 95% confidence interval is:

$$\mathbb{P}\left(q_{2.5} \le \frac{\widehat{\beta} - \beta}{\text{se}(\widehat{\beta})} \le q_{97.5}\right) = 0.95 \qquad \textit{endpoints fixed; middle random}$$

$$\Rightarrow \quad \mathbb{P}\left(\widehat{\beta} - q_{97.5}\,\text{se}(\widehat{\beta}) \le \beta \le \widehat{\beta} + q_{97.5}\,\text{se}(\widehat{\beta})\right) = 0.95 \qquad \textit{middle fixed; endpoints random}$$

The second line is a simple rearrangement of the first, but we must remember that it is the *endpoints* that are random, whereas the β in the middle stays fixed. Thus the 95% CI is:

$$\left(\widehat{\beta} - q_{97.5}\,\text{se}(\widehat{\beta})\,,\ \widehat{\beta} + q_{97.5}\,\text{se}(\widehat{\beta})\right), \quad \text{or equivalently,} \quad \widehat{\beta} \pm q_{97.5,\text{df}}\,\text{se}(\widehat{\beta})\,.$$

Confidence interval for the whole line

Recall that the true mean of Y when $X = x$ is:

$$\mu \;=\; \alpha + \beta x \;=\; \mathbb{E}\,(Y \,|\, X = x)\,.$$

This is a function of the parameters α and β. It is estimated by the *fitted value*:

$$\widehat{\mu} \;=\; \widehat{\alpha} + \widehat{\beta} x \;=\; \widehat{\mathbb{E}}\,(Y \,|\, X = x)\,.$$

Just as we can create confidence intervals for α and β, we can also create them for any function of α and β, including for $\mu = \alpha + \beta x$.

As above, the 95% confidence interval is:

$$\text{Estimate} \;\pm\; q_{97.5,\text{df}} \;\times\; \text{Standard error}\,,$$

so the 95% CI for μ is: $\qquad \widehat{\mu} \;\pm\; q_{97.5,\text{df}}\, \text{se}(\widehat{\mu})\,.$

So all we have to do is to figure out how to compute the standard error of $\widehat{\mu}$, given we already know $\text{se}(\widehat{\alpha})$ and $\text{se}(\widehat{\beta})$. Although this calculation is not covered in this course, you might already know from other courses that:

$$\text{Var}(\widehat{\mu}) = \text{Var}(\widehat{\alpha} + \widehat{\beta} x) = \text{Var}(\widehat{\alpha}) + x^2\,\text{Var}(\widehat{\beta}) + 2\,x\,\text{cov}(\widehat{\alpha}, \widehat{\beta})\,,$$

where cov stands for 'covariance'. The standard error is the square root of the estimated variance:

$$\text{se}(\widehat{\mu}) = \sqrt{\text{se}(\widehat{\alpha})^2 + x^2\,\text{se}(\widehat{\beta})^2 + 2\,x\,\widehat{\text{cov}}(\widehat{\alpha}, \widehat{\beta})}\,, \qquad (\star)$$

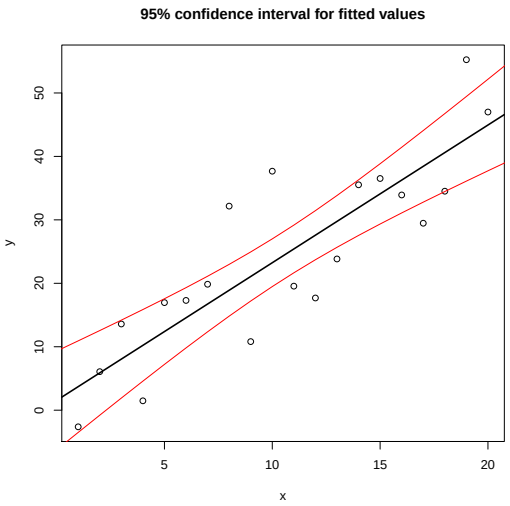
and all terms are given in the model summary from R, using `vcov(myfit)`.

So we have everything we need to calculate the confidence interval. Fortunately, R has a function that does it for us:

```
predict(myfit, interval="confidence")
```

which can be abbreviated to:

```
predict(myfit, interval="conf")
```



The interpretation of the confidence interval remains the same: on 95% of occasions that we draw a sample and build a CI for $\mu = \mathbb{E}(Y \,|\, X = x)$, the interval will contain the true μ.

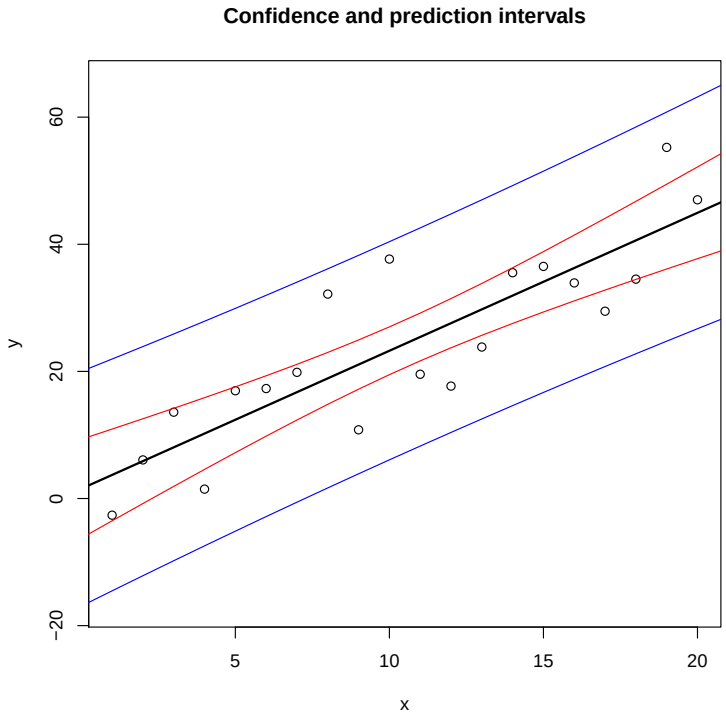
So the CI marked on the figure above reflects ***our uncertainty about where the FITTED LINE goes.***

Prediction intervals

As well as the confidence interval, which describes our uncertainty about where the fitted line goes, we might be interested in another type of interval, called a ***prediction interval.***

A ***prediction interval*** is an interval in which we expect a ***data point*** to lie: in other words it specifies an interval that is likely to enclose the *value of* Y for a particular x. Compare this with the ***confidence interval*** which specifies an interval that is likely to enclose the value of $\mathbb{E}(Y \mid X = x)$, not the raw value of $(Y \mid X = x)$.

The prediction interval incorporates both the uncertainty in estimating the fitted line via $\widehat{\alpha}$ and $\widehat{\beta}$, and the variability in Y around the fitted line which is estimated by $\widehat{\sigma}$. It adds the extra term $\widehat{\mathrm{Var}}(Y) = \widehat{\sigma}^2$ under the square root in eqn $(\star)$ overleaf, where $\widehat{\sigma}$ is extracted by the command `summary(myfit)$sigma`.



Again, there is an R function to extract the prediction interval. Note the use of the `newdata` argument. We can ask R for fitted values and confidence or prediction intervals for any new set of x values, as specified in the `newdata` data-frame. For example:

```
predict(myfit, newdata=data.frame(x=0:21), interval="prediction")
predict(myfit, newdata=data.frame(x=c(5, 10, 15)), interval="prediction")
```

Compare with: `predict(myfit, newdata=data.frame(x=c(5, 10, 15)), interval="conf")`

R will deliver predictions outside of the original range of the data x-values if asked, but it is not a good idea to extrapolate beyond the range of the data.

The function below computes the prediction interval for a selected value of `xpred`. Run the function using `predint.func()`. You should find that the simulated data-value `ypred` lies in the prediction interval for about 95% of simulations.

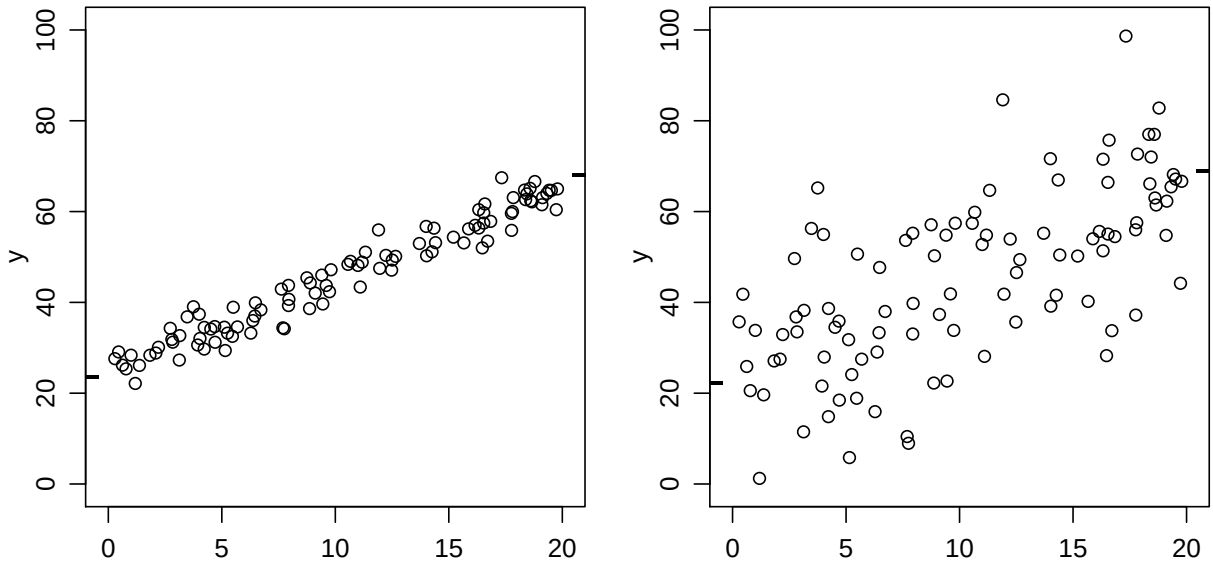
```
predint.func <- function(alpha=5, beta=2, sigma=3, xpred=5, Nsim=1000, printit=T){
  ## Create a vector of results:
  pred.res <- numeric(Nsim)
  ## Let simulation i run from 1 to Nsim:
  for(i in 1:Nsim){
    ## Generate the data for this simulation, and fit the linear model:
    mydat <- data.frame(x=1:20, y=alpha+beta*(1:20)+rnorm(20, 0, sigma))
    myfit <- lm(y~x, mydat)
    ## Generate a new y-value from the true model when x=xpred:
    ypred <- alpha + beta*xpred + rnorm(1, 0, sigma)
    ## Extract the 95% prediction interval for ypred:
    ypred.itvl <- predict(myfit, newdata=data.frame(x=xpred), interval="prediction")
    ## Find whether the prediction interval encloses the simulated value ypred,
    ## and put the result in entry i of pred.res:
    pred.res[i] <- as.numeric(ypred.itvl[,"lwr"]<=ypred & ypred<=ypred.itvl[,"upr"])
    ## Print output for simulation i, if requested via argument "printit":
    if(printit)
      cat("Prediction interval", format(round(ypred.itvl, 2), nsmall=2),
          "contains ypred=",format(round(ypred, 2),nsmall=2),"? ", pred.res[i], "\n")
  }
  ## We now have Nsim results stored in vector pred.res.
  ## Find out how many of the prediction intervals contained ypred:
  cat("\n", sum(pred.res), "out of", Nsim,
      "data sets contained the simulated value of ypred: inclusion rate=",
      round(sum(pred.res)/Nsim, 2), "\n")
  ## Finish the function by returning the vector of results:
  return(pred.res)
}
```

1.5 What is the quality of the relationship between X and Y?

So far we have looked at: (a) evidence of the existence of a relationship between X and Y, using the p-value; (b) what the estimated relationship is between X and Y, using the fitted line; (c) quantifying our uncertainty in the fitted line, using a confidence interval; and (d) specifying an estimated range of values of Y for a given X, using a prediction interval. These ideas address the first two of our key questions in Section 1.1.

Here we look at the next question: how can we assess the ***quality*** of the relationship between X and Y? Another way of phrasing this question is, ***how MUCH does X tell us about Y?***

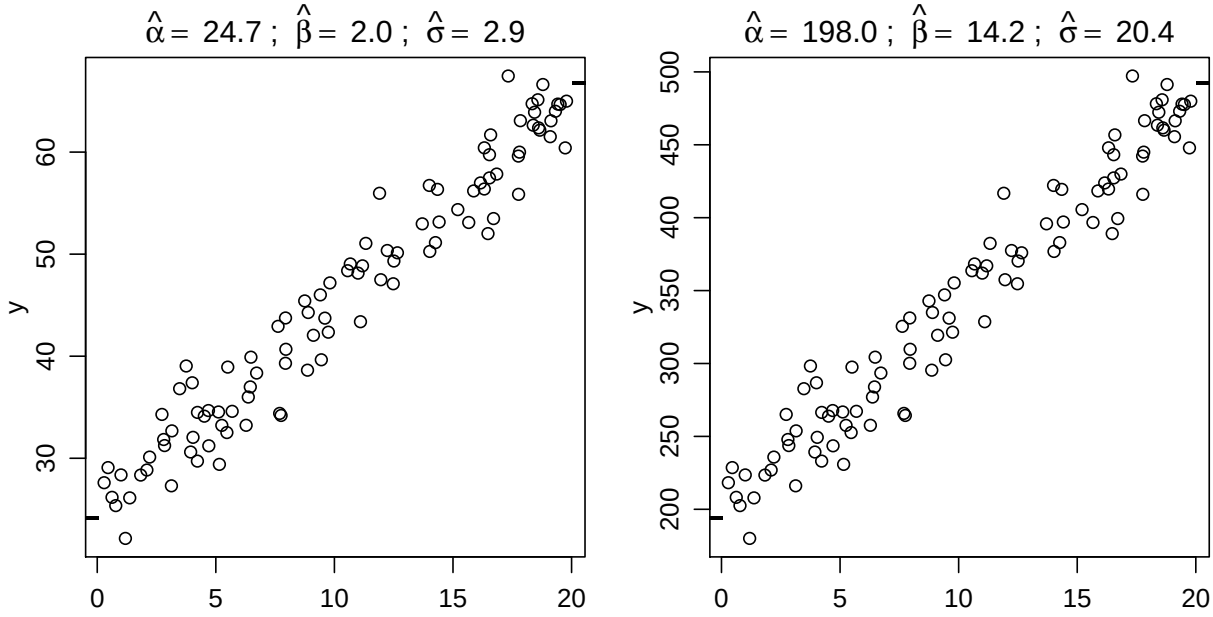
Look at the two examples below. Which of these plots describes a stronger relationship between X and Y?



The left panel shows a stronger relationship.

So perhaps the 'quality' or 'strength' of relationship has something to do with the scatter or variance about the fitted line, in other words σ ?

What about the two plots below? They have very different values of σ, as well as α and β. Should they be assigned the same 'quality' measures, or different?



The strength of relationship looks identical in these cases, even though their $\hat{\sigma}$ values are very different.

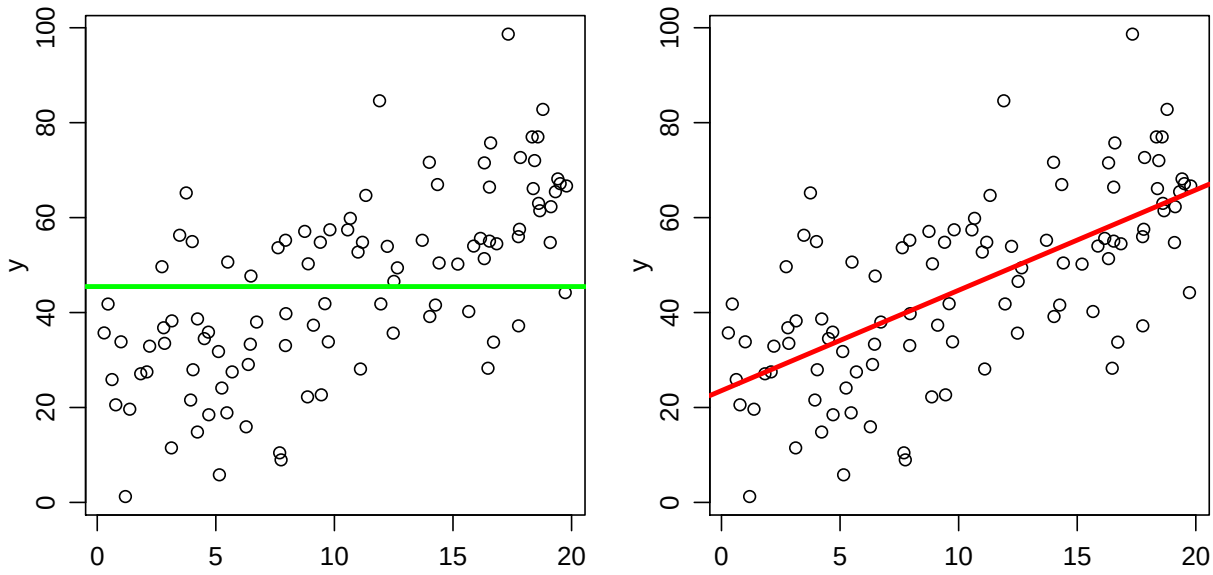
We need a ***relative*** measure of variance about the line. The parameter σ is an absolute measure: we need to scale it somehow so that we can compare all scatterplots on the same scale.

Multiple R^2, or the percentage of variance explained

We seek a relative measure of variance about the fitted line which assesses the strength or quality of the relationship between X and Y. An obvious choice is to ***compare the variance around the fitted line with the variance in the null model.***

This tells us how ***useful*** the predictor X has been in removing the variance or scatter in Y.

In an ideal world, we would be able to explain Y perfectly without any scatter at all. The ***reduction in scatter due to X*** is a good measure of what we have achieved by attempting to explain Y using X, relative to what we could have achieved with no predictor variables.



Let's generate some data and fit both the slope model, `lm(y~x)`, and the null model, `lm(y~1)`. Note that the syntax `lm(y~x)` is actually an abbreviation for `lm(y~1+x)`, meaning we fit a linear model with $\mathbb{E}(Y \mid X = x) = \alpha \times 1 + \beta \times x$.

To fit the null model by itself, we clarify the intercept term explicitly using `lm(y~1)`. This means the null model is: $\mathbb{E}(Y \mid X = x) = \alpha \times 1$, ***with no slope term.***

```
mydat <- data.frame(x=runif(100, 0, 20))
mydat$y <- 25 + 2*mydat$x + rnorm(100, 0, 15)
## Fit the linear model including the slope term, slope.fit.
slope.fit <- lm(y~x, mydat)

## Fit the null model: null.fit, using the notation lm(y~1).
null.fit <- lm(y~1, mydat)
```

The relative amount of scatter ***still left*** in the full model, compared with the null model, is:

$$\text{scatter remaining} \;=\; \frac{\text{sum}\left\{\text{residuals(slope.fit)}^2\right\}}{\text{sum}\left\{\text{residuals(null.fit)}^2\right\}}.$$

So the relative amount of scatter that has been ***removed*** or ***explained*** by including variable X in the model is:

$$\textit{proportion of variance explained} \;=\; 1 - \frac{\textit{sum}\left\{\textit{residuals(slope.fit)}^2\right\}}{\textit{sum}\left\{\textit{residuals(null.fit)}^2\right\}}.$$

This idea of variable X *explaining* a proportion of the variance or scatter in Y is why we often use the term ***explanatory variable*** for X. We call X a *predictor* or *explanatory* variable interchangeably.

Let's check out the expression above with the model output for model `slope.fit`:

```
summary(slope.fit)
   Coefficients:
               Estimate Std. Error t value Pr(>|t|)
   (Intercept)  23.5453     2.8970   8.127 1.36e-12 ***
   x             2.1147     0.2416   8.755 6.07e-14 ***
   ---
   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

   Residual standard error: 14.56 on 98 degrees of freedom
   Multiple R-squared:  0.4389, Adjusted R-squared:  0.4331
   F-statistic: 76.64 on 1 and 98 DF,  p-value: 6.071e-14
```

What does the expression above give us for proportion of variance explained?

```
   1 - sum(residuals(slope.fit)^2) / sum(residuals(null.fit)^2)
      [1] 0.4388579
```

This matches the R output for `Multiple R-squared.` R-squared is also called the *coefficient of determination* or the *squared correlation coefficient.* However, the best way to think about R^2 is the ***proportion of variance explained.***

R^2 is the proportion of variance explained by the fitted model, compared with the null model.

(Aside: R^2 is called the *squared correlation coefficient* because it can also be obtained by `cor(mydat$y, fitted(slope.fit))^2` .)

The *residual standard error* is the estimated $\widehat{\sigma}$, from `summary(slope.fit)$sigma`. The computation R performs for $\widehat{\sigma}$ is: `sqrt(sum(residuals(slope.fit)^2/98))`.

The adjusted R^2 can be ignored: it just makes a small adjustment for the number of parameters. Here, we have 98 df in the `slope.fit` model, and 99 in the `null.fit` model, so the adjusted R^2 is: `1 - sum(residuals(slope.fit)^2/98) / sum(residuals(null.fit)^2/99)`.

Difference between R^2 and the p-value

R^2 is interpreted as a measure of the *quality* of our linear fit, or how effective X is as a predictor of Y. It measures the proportion of variance in Y that is removed or explained by X. ***R^2 is always a number between 0 and 1, and it can be compared for any two scatterplots.***

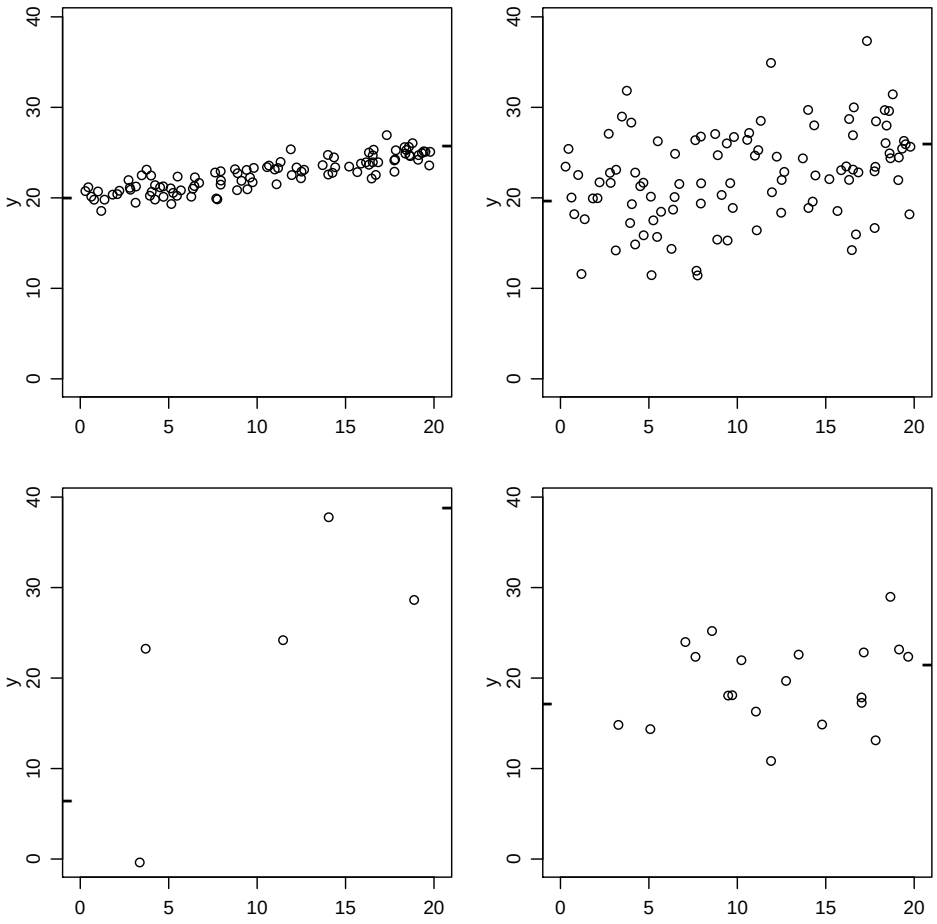
R^2 aims to measure the relationship between X and Y in the population at large. If we increased or reduced our sample size from a population, we would not expect R^2 to change much. (Try generating `mydat` using 1000 data points instead of 100, and see if R^2 changes much.)

The p-value is a measure of ***our evidence, in our sample, that a linear relationship between X and Y exists.***

In other words: the p-value measures our evidence against $H_0 : \beta = 0$ in favour of $H_1 : \beta \neq 0$. The p-value is heavily dependent upon sample size — because it measures ***the evidence in OUR SAMPLE.***

The p-value is also affected by the strength of the relationship, because strong relationships are easy to detect even in small samples. However it is important to recognise that the p-value is not primarily measuring a population quantity: it is measuring ***evidence in OUR SAMPLE.***

Have a guess at the ball-park p-values and R^2 values in the scatterplots below.



1.6 Writing Methods & Assumption Checks and the Executive Summary

Up to now, we have used the notation $Y_i = \alpha + \beta x_i + \epsilon_i$. When writing up your results, you should use the alternative notation $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$. The first notation is simpler, which is why we have used it for introducing the modelling framework, but the second notation is more extendable, which will be useful for models we will see soon with more than one predictor variable.

Methods and Assumption Checks

What you write in this section should allow another statistician to repeat your analysis and get the same results.

1. Say **what sort of model** you fitted, and why. *'The data appear to follow a linear trend, so we fitted a simple linear regression model.'*

2. Show you have thought about whether your **sample** is appropriate for this model. Are the data independent and representative of the population? If you have doubts, mention them here. Examples: *'The dataset is a random sample of students, so we can assume it is an independent and representative sample.'* *'We are not told how the sample was obtained, so we cannot comment on whether the data are representative of the intended population.'*

3. Mention results from the **three assumption plots.** If everything is fine: *'The residual plot showed patternless residuals with approximately constant scatter, and there were no concerns about normality or influential points.'*
   Or just: *'There were no problems with any of the assumptions.'*

   If there are doubts about constant scatter, these should be mentioned but might not be very serious. Similarly if the normcheck plot has short or long tails but is reasonably symmetric. *'There was a hint of non-constant scatter in the residuals, but this was not of great concern.'* Or: *'Residual scatter had an increasing appearence, so confidence interval coverage might not be reliable.'*
   If the residual plot has clear curvature in the central red trend line, a different model will be needed: see the next chapters.

4. If you **removed any data**, say what you removed and why. *'Two data-points had much larger x-values than the others, so they were removed because we cannot assume they follow the same pattern as the rest of the data.'* Or: *'One data-point with an excessively large y-value was removed as a presumed error.'*

5. Give the **model formula.**
   Our model is: $Weight_i = \beta_0 + \beta_1 Height_i + \epsilon_i$, *where* $\epsilon_i \sim \text{iid } N(0, \sigma^2)$.

6. Give the **percentage of variance explained** and comment on its size.

Executive Summary

The purpose of the Executive Summary is to describe your findings to a ***non-technical audience***. You should include the following items, which are closely linked to the Key Questions we posed at the beginning of this handout.

1. **What is the purpose of the analysis?** You can reproduce the purpose given in the question. Your introduction should also indicate the population of interest, and mention if you excluded any data-points and why.

   Example: *We aimed to model the relationship between the height and weight of NZ adults, and assess how useful the model is for prediction.*

2. **Do we have evidence that a relationship exists? If so, what does the relationship look like?** You can combine the answers to these two key points concisely in your write-up.

   - For evidence of a relationship, look at the p-value for the β_1 parameter. If this is significant, there is evidence of a relationship.
   - In the Executive Summary, use plain English to describe the **strength of evidence**. You may quote the p-value in brackets if you wish.
   - If there is a relationship, describe what it looks like using simple language like 'positive', 'increasing', 'negative', or 'decreasing'.

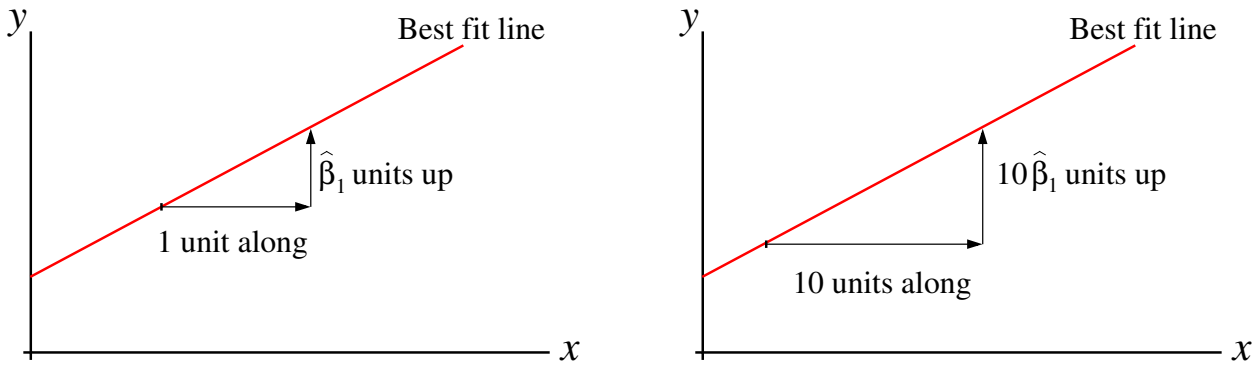
   We found evidence of a positive relationship between weight and height ($p<0.05$).
   There was strong evidence of a decreasing relationship between exercise time and heart rate ($p = 0$).
   There was only weak evidence that yield increased with fertilizer dose ($p = 0.07$).

3. **How can we quantify the relationship numerically?** Quantification of the relationship is based on the principle that for every one unit increase in x, the estimated increase in the mean of y is $\hat{\beta}_1$. However, you need to apply this principle intelligently:

   - When you are talking about the best-fit line, you are talking about the estimated ***mean*** or ***average*** of y. Ensure your wording makes this clear.
   - Quote your answers in terms of ***confidence intervals*** rather than point estimates: use `confint(fit)` and read off the lower and upper confidence limits for β_1. The basic template for your answer is then something like, *'We estimate that every unit increase in x is associated with an increase in the mean y of between* `CIlow` *and* `CIhi` *units.'*
   - Use natural language for variables and units: e.g. *'We estimate that for every extra hour a person spends exercising each week...'*. **Do not** use variable names like `exerTime` or mathematical notation like x and y.

- Use appropriate rounding: e.g. quoting a change in the mean heart rate of 4.1 beats per minute is informative, whereas quoting 4.128715 obscures the key message and looks like you are not thinking about what you write.
- ***Important!*** Choosing a 1-unit increase in x is not always appropriate. Sometimes 1 unit is far too small or far too large. For example, if all your data fall between $x = 0.3$ km and $x = 0.8$ km, then you have no justification for specifying the result of a 1 km increase in x. Equally, if your data fall between $x =$ \$3 million and $x =$ \$8 million, then quoting the results of a \$1 increase in x is meaningless. If the data are input to R in units of kilometres or dollars, then R will use those same units for reporting $\hat{\beta}_1$ and confidence intervals.

  To adjust the scale appropriately, first decide how many of the units that R is using you want to quote for an increase in x: e.g. 0.1 km or \$1 million dollars. Then just multiply the lower and upper confidence limits by this same amount. If one unit increase in x corresponds to an average increase of between `CIlow` and `CIhi` units in y, then the average increase generated by a 0.1 unit increase in x is between 0.1 `CIlow` and 0.1 `CIhi` units; and by a 1 million unit increase in x is between 1e6×`CIlow` and 1e6×`CIhi` units.



4. **Predictions.** If asked for predictions, check whether the question is asking about predicting the ***mean***, or predicting the ***outcome for a subject***.

   (a) If the question is about the ***mean***, then it should include the word 'mean' or 'average'. Your answer should then be based on ***confidence intervals for the mean***, and must clearly use the word 'mean' or 'average'. Examples:

   — *Predict the average weight for a person of height 175 cm.* Use:
   `predict(fit, newdata=data.frame(height=c(175)), interval="conf")`
   'We estimate that the average weight for people of height 175 cm is between. . . '

   — *Find the average heart rate for a person who does 5 hours exercise per week, and say if your estimate is consistent with the value of 72 beats per minute.* Use:
   `predict(fit, newdata=data.frame(exerTime=c(5)), interval="conf")`
   and see if the confidence interval contains the value 72.

(b) More commonly, the question is about generating a prediction for a subject. In this case, it will not mention the word 'mean' or 'average'. Your answer should then be based on ***prediction intervals***, and your wording should indicate individual subjects rather than population means. Examples:

— *Predict the weight of a person of height 175 cm.* Use:
`predict(fit,newdata=data.frame(height=175), interval="prediction")`
'We estimate that a typical person of height 175 cm will weigh between ... and ... kg.'

— *What heart rate do you predict for a person who does 5 hours exercise per week?* Use:
`predict(fit,newdata=data.frame(exerTime=5), interval="prediction")`

5. **Is the model useful for prediction?** You should only mention this in the Executive Summary if prediction is an explicit aim of the analysis. There are two ways you can comment about usefulness.

(a) **Percentage of variance explained.** Quote R^2 as a percentage to the nearest whole number, and describe it as 'percentage of variance explained'. Do not use the notation R^2 in the Executive Summary. Comment on the size of R^2 in terms of usefulness. Examples:

The model explained 82% of the variability in heart rate, so it is reasonably useful for prediction.

The model explained only 46% of the variability in weight, so we cannot expect predictions to be very precise relative to the overall range of the data.

The model explained only 32% of the variability in outcome, so although we found evidence of a trend, the usefulness for prediction purposes is limited.

The model explained only 16% of the variability in the data, so it is not very useful for prediction.

(b) **Width of the prediction interval, relative to the overall range of y.** If the prediction interval has a width of 10 units of y, is this useful or not? The answer depends upon the original range of y in the data.

If the original range of y was 100 units, then a prediction interval of width 10 units has narrowed it down to 10% of the original, so we have made an important gain. If the original range of y was 12 units, we have barely made any gain by narrowing it down to 10 units.

Use wording like: *'The prediction interval for heart rate spans a range of 10 beats/minute, which is a quarter of the original data range of 40 beats/minute. The prediction is therefore quite useful.'*

Handout 2: Linear Model Variations

Chapters 1 and 2: introduction to the simple linear model

$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

Fit with `lm(y~x)` or `lm(y~1+x)`. To deliberately omit the intercept, `lm(y~x-1)`.

`summary(fit)` gives p-values for t-tests of $H_0 : \beta_0 = 0$ (intercept is zero), and $H_0 : \beta_1 = 0$ (slope is zero: this is the test of *no relationship* between X and Y).

To describe the fitted model, use a plain language equivalent of *'the fitted value increases by $\widehat{\beta}_1$ units for every $x = 1$ unit increase.'* Use an appropriate x-unit increase for the context, e.g. not $x = 1$ if the x-axis spans a range of millions.

Chapter 3: null model, also useful as a one-sample *t*-test

$Y_i = \beta_0 + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

Fit: `lm(y~1)`. Use as null model for baseline comparisons, or for paired t-test.

The estimate $\widehat{\beta}_0$ is the sample mean: $\widehat{\beta}_0 = \overline{y} =$ `mean(y)`.
The estimate $\widehat{\sigma}$ is the sample standard deviation: $\widehat{\sigma} =$ `sd(y)`=`sqrt(var(y))`.
To do a t-test by hand of $H_0 : \beta_0 = \mu_0$ vs $H_1 : \beta_0 \neq \mu_0$:

$$\text{Test statistic} = \frac{\text{Estimate} - \text{Hypothesized value}}{\text{Standard error of estimate}} = \frac{\texttt{mean(y)} - \mu_0}{\texttt{sd(y)/sqrt(n)}}.$$

`summary(fit)` gives the p-value for the t-test of $H_0 : \beta_0 = 0$: same as the test above but for $\mu_0 = 0$ only. This is most useful for a **paired *t*-test**, where each observation y_i is the difference between two items in a pair. For example:

- y_i is the difference between a husband's age and a wife's age, when testing H_0 that the average age difference within a married couple is zero;
- y_i is the difference between birthweights of a first-born sibling and a second-born sibling, when testing H_0 that birth order has no effect on birthweight.

Chapter 4: quadratic model

$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

Fit: `lm(y ~ x + I(x^2))`. Enables a curved quadratic-shaped fitted line.

`summary(fit)` gives test of $H_0 : \beta_2 = 0$ as well as the other coefficients. Use the β_2 test to decide if the quadratic term is needed.

To describe the fitted model, say what *shape* the curve is, e.g. *'always increasing, curving upwards'*, or *'decreases to a minimum at $x = \ldots$ and then increases'.*

To plot the fitted line over the data (replacing `xvals` by values, e.g. `(1:20)`):

```
plot(y~x, dat)
pred <- predict(fit, data.frame(x=xvals), interval="conf")
lines(xvals, pred[,"fit"], col="red")
```

Chapter 5: simple linear model when X is a two-level categorical variable

Suppose the predictor variable X is a categorical (label) variable, taking two levels: e.g. ***Yes and No.*** For example, X might represent whether a subject is a NZ Citizen or not.

Because we can't use words like 'Yes' and 'No' in equations, we introduce what we call a ***dummy variable*** that converts the different word-responses into the numbers 1 and 0. In the Citizen case, the dummy variable is

$$\texttt{Citizen}_i = \begin{cases} 1 & \text{if subject } i \text{ answers Yes,} \\ 0 & \text{if subject } i \text{ answers No.} \end{cases}$$

The model is:

$$Y_i = \beta_0 + \beta_1 \texttt{Citizen}_i + \epsilon_i, \quad \text{where } \epsilon_i \sim \text{iid Normal}(0, \sigma^2).$$

This has the following effect:

$$\textbf{\textit{When subject } i \textit{ answers Yes:}} \qquad Y_i = \beta_0 + \beta_1 + \epsilon_i$$
$$\textbf{\textit{When subject } i \textit{ answers No:}} \qquad Y_i = \beta_0 + \epsilon_i.$$

Thus this model *only* specifies ***two different intercepts: one for the Yes group, and one for the No group.***

The parameter β_1 specifies ***the difference between the Yes and the No group means.***

This gives us an easy way of doing a *two-sample t-test*: in other words, a t-test when there are two independent samples (the Yes group and the No group). We can use `summary(fit)` to test $H_0 : \beta_1 = 0$. If β_1 really is zero, it means ***there is no evidence of a difference between the two groups.***

To fit a model with a categorical (label) predictor:

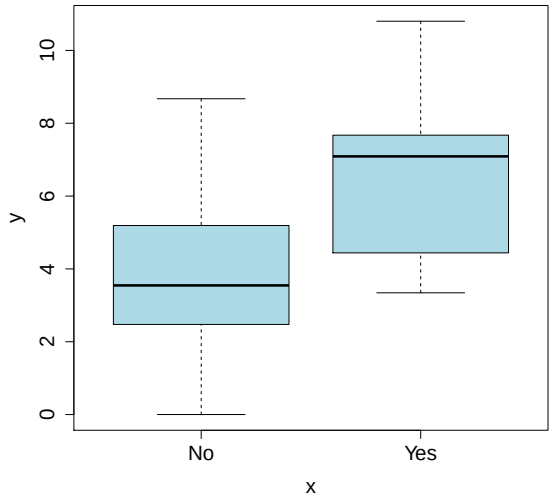
- If variable X is coded in the dataframe as character labels (e.g. Yes and No), R will automatically understand how it should be treated. You can then fit the model using `lm(y~x)` as usual. In the Citizen case, `lm(y ~ Citizen)`.
- Sometimes, variable X might be coded as numeric labels (e.g. 1 and 0). R will treat this as a numeric variable by default, which isn't what you want. To force R to treat the variable as a label variable, use ***as.factor(x)*** in the model formula: `lm(y ~ as.factor(x))`. For example, `lm(y ~ as.factor(Citizen))`. The term ***factor*** is an alternative name for a categorical variable.

Demonstration of the two-level factor model

```
mydat <- data.frame(x=c(rep("Yes", 15), rep("No", 15)))
mydat$y <- 5 + 2*as.numeric(mydat$x=="Yes") + rnorm(30, 0, 3)
```

```
mydat
     x     y
  1  Yes 4.114200
  2  Yes 6.122423
  3  Yes 7.776365
  :   :       :
  16  No 4.077031
  17  No 2.140948
  18  No 3.055272
```

```
plot(y~x, mydat)
```



The plot automatically comes out as a boxplot. You can see that it does indeed look like a two-sample t-test.

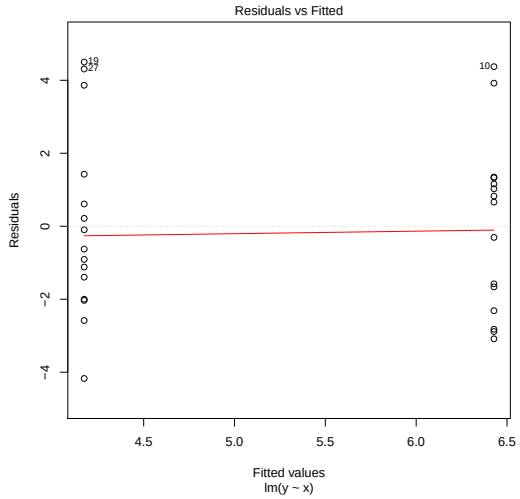
Fit the linear model:

```
myfit <- lm(y~x, mydat)
summary(myfit)
```

```
  Call:
  lm(formula = y ~ x, data = mydat)

  Coefficients:
              Estimate Std. Error t value Pr(>|t|)
  (Intercept)   4.1725     0.6380    6.54 4.33e-07 ***
  xYes          2.2557     0.9022    2.50   0.0185 *

  Residual standard error: 2.471 on 28 degrees of freedom
  Multiple R-squared:  0.1825, Adjusted R-squared:  0.1533
  F-statistic:  6.25 on 1 and 28 DF,  p-value: 0.01855
```

You can do all the usual model checks (`normcheck(myfit)`) and residual plots:

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



Writing the model formula for a categorical model:

Use notation like this: $\texttt{Income}_i = \beta_0 + \beta_1 \texttt{Citizen}_i + \epsilon_i$, where $\texttt{Citizen}_i$ takes the value 1 if person i is a Citizen and 0 if not, and $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

Changing the baseline group

R orders the factor categories in alphabetical order, so the alphabetically-first level will be chosen as the baseline, which is given group mean equal to β_0. Here, the baseline is the 'No' category. You can change this behaviour using the R command `relevel(x, ref="Yes")`. Let's see what happens.

```
mydat2 <- data.frame(x=mydat$x, xnew=relevel(mydat$x, ref="Yes"), y=mydat$y)
mydat2
      x xnew        y
  1 Yes  Yes 4.114200
  2 Yes  Yes 6.122423
  :   :       :
summary(mydat2$x)                    summary(mydat2$xnew)
   No Yes                               Yes No
   15  15                                15 15

summary(lm(y ~ xnew, mydat2))
     Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
     (Intercept)   6.4282     0.6380   10.08 8.15e-11 ***
     xnewNo       -2.2557     0.9022   -2.50   0.0185 *
     ---
     Multiple R-squared:  0.1825, Adjusted R-squared:  0.1533

summary(lm(y ~ x, mydat2))
     Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
     (Intercept)   4.1725     0.6380    6.54 4.33e-07 ***
     xYes          2.2557     0.9022    2.50   0.0185 *
     ---
     Multiple R-squared:  0.1825, Adjusted R-squared:  0.1533
```

You can also turn off the intercept altogether, using `lm(y~x-1)`. However, this will render R^2 meaningless. Why? ***Because it reports %variance explained compared with the variance in the zero-intercept model!***

(We don't expect that either group has mean zero, so the zero-intercept model is not of interest.)

```
summary(lm(y ~ x-1, mydat2))
     Coefficients:
          Estimate Std. Error t value Pr(>|t|)
     xNo     4.173      0.638    6.54 4.33e-07 ***
     xYes    6.428      0.638   10.08 8.15e-11 ***
     ---
     Multiple R-squared:  0.8375,  Adjusted R-squared:  0.8259
```

Chapter 6: log-transformed responses, y

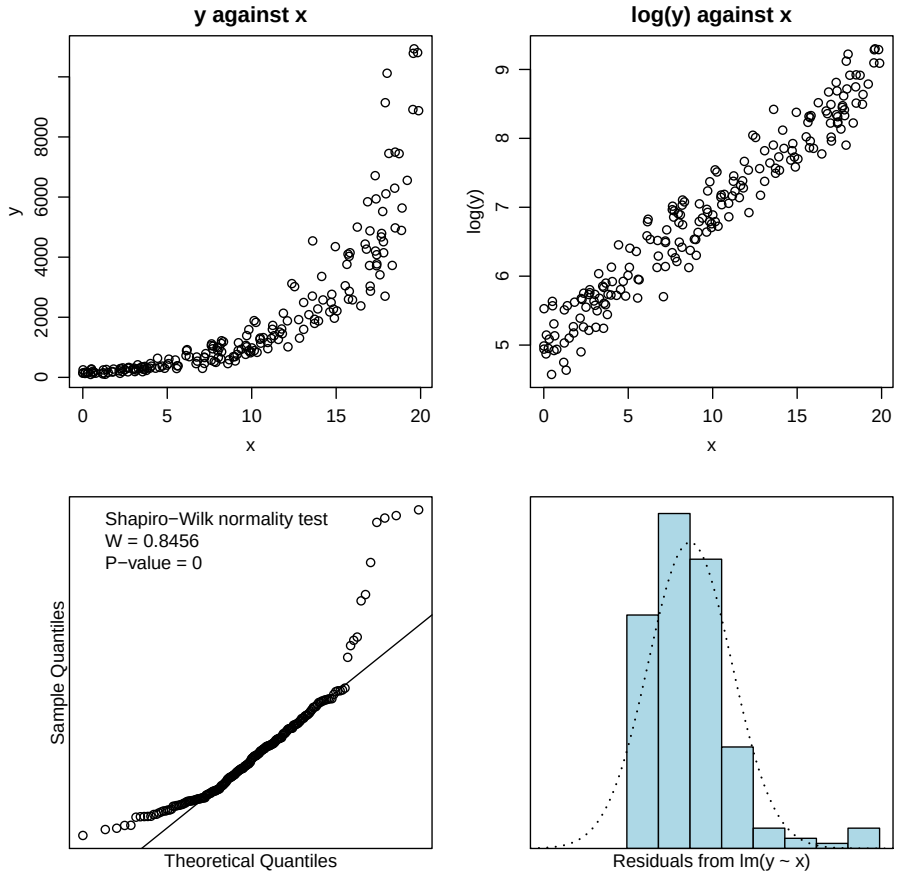
If the response Y shows increasing scatter at higher values, it can sometimes be appropriate to use a ***log-transformation:***

$\log(Y_i) = \beta_0 + \beta_1 x_i + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

Similarly, you can use any of the other models we've seen, such as $\log(Y_i) = \beta_0 + \epsilon_i$, or $\log(Y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$, etc.

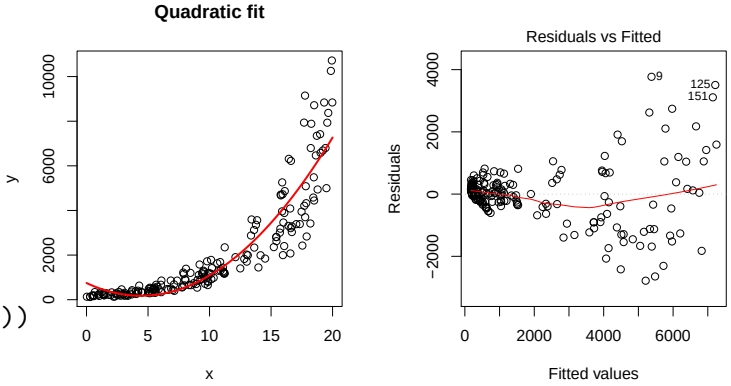
Fit with `lm(log(y)~x)`, or `lm(log(y)~x+I(x^2))`, etc.

The figures show typical data patterns that indicate a log-transformation may be appropriate. ***Note there is both curvature and increasing scatter.***



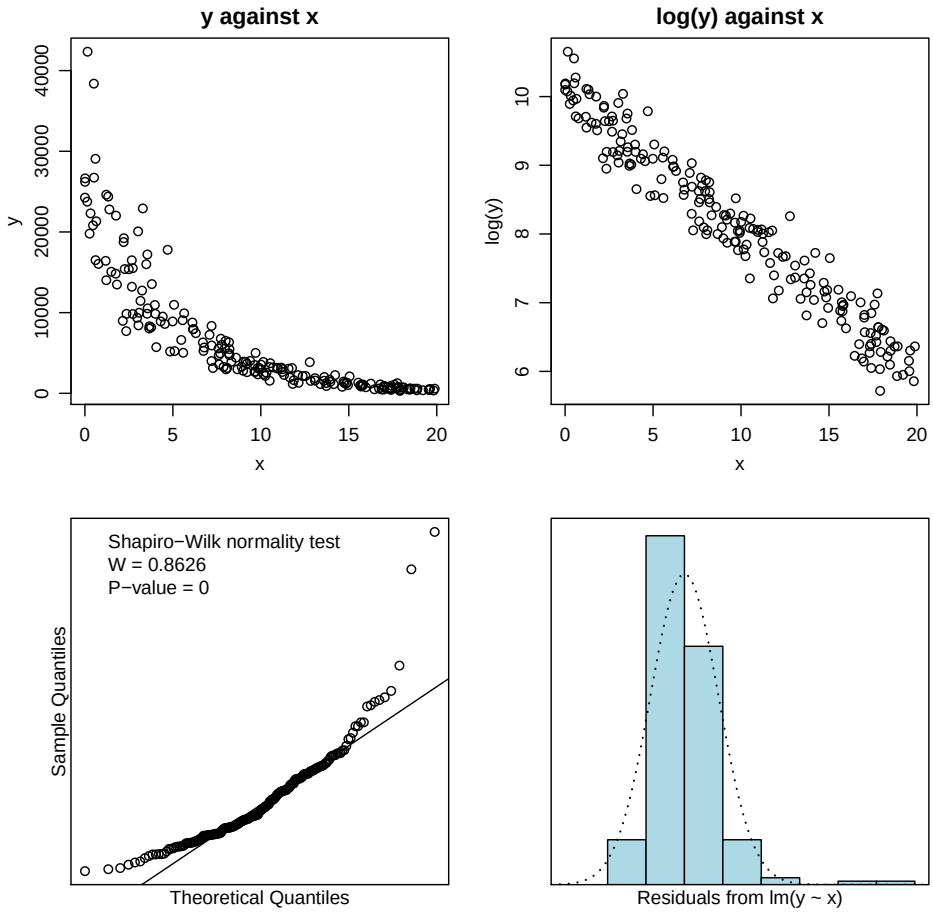
What if we try to model the curvature using a quadratic model?

```
myfit <- lm(y ~ x + I(x^2), mydat)
plot(y~x, mydat)
lines(0:20,
 predict(myfit, data.frame(x=0:20)))
plot(myfit, which=1)
```



The residual plot is awful!

The example below has a decreasing trend, but the same problem of more scatter at higher y.



What you need to know about logs:

- If $y = e^z$, then $z = \log(y)$.
- $e^{\log y} = y$, and $\log\left(e^z\right) = z$.
- $\log(ab) = \log(a) + \log(b)$.
- $e^{a+b} = e^a\, e^b$.
- $\log\left(y^a\right) = a \log y$. So: $\exp(a \log y) = y^a$.

Important: back-transforming mean{log(Y)} DOES NOT give mean(Y)

When we model $\log(Y)$ in a linear model, we estimate the ***mean of log(Y).***

We can exponentiate this, but ***we will NOT get back the mean of Y.***

The problem is that $e^{(\text{mean log Y})}$ does NOT equal mean$(e^{\log Y})$.

Instead, to convert back to the Y-scale, we must focus on the ***median of*** Y***.***

Using the median instead of the mean

Try these commands:

```
> y <- runif(100, 0, 1)  ## 100 random numbers between 0 and 1
> mean(y)
[1] 0.5030
> exp(mean(log(y)))
[1] 0.3698       ## exp(mean of log y) is very different from mean(y)

> median(y)
[1] 0.5086
> exp(median(log(y)))
[1] 0.5086      ## exp(median of log y) is the same as median(y)
```

When we fit the model $\log(Y_i) = \beta_0 + \beta_1 x_i + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$, we make the assumption that $\log(Y)$ ***is Normal.***

For the Normal distribution, the median equals the mean. So when we talk about the ***mean*** in a simple linear model, we are simultaneously talking about the ***median.***

The point here is that if we back-transform a fitted value (i.e. a mean) for $\log(Y)$ by exponentiating, we will ***not*** get back the mean of Y. But, if we back-transform the ***median*** of $\log(Y)$, we ***will*** get back the median of Y.

And, since the mean of $\log(Y)$ is the same as the median, due to our Normal assumption for $\log(Y)$, we can back-transform any fitted value for $\log(Y)$ as long as we ***DESCRIBE*** it as the ***fitted MEDIAN for*** Y***.***

- You need to know that when we fit a simple linear model to log-transformed responses ($\log Y$), then ***we only talk about the MEDIAN of*** Y ***after back-transforming. We don't talk about the MEAN of*** Y***.***

Multiplicative model

If $\log(Y_i) = \beta_0 + \beta_1 x_i + \epsilon_i$, then $Y_i = \exp(\beta_0)\exp(\beta_1 x_i)\exp(\epsilon_i)$.

The same multiplicative effect applies to the fitted medians. The fitted value on the log scale can be described as either the mean or the median:

$$\widehat{\mu}_i = \text{median}\,\{\log(Y_i)\} = \widehat{\beta}_0 + \widehat{\beta}_1 x_i.$$

On the Y-scale, the back-transformed value can only be called the median:

$$\text{median}(Y_i) = \exp(\widehat{\beta}_0)\exp(\widehat{\beta}_1 x_i).$$

On the Y-scale, ***terms multiply rather than add.***

Quantifying the model in the Executive Summary

When we fit a simple linear model, we quantify it in the Executive Summary using wording like,

> *'We estimate that every unit increase in x is associated with an increase in the mean Y of between* `CIlow` *and* `CIhi` *units.'*

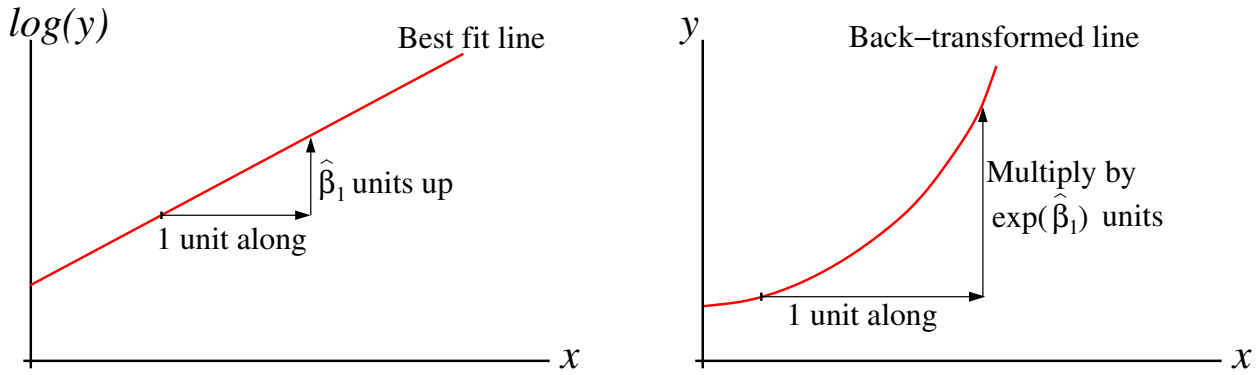
Here, `CIlow` and `CIhi` are the lower and upper confidence limits for β_1 obtained from `confint(fit)`.

When we use a log-transformation for Y, we use a similar formulation, ***except:***

1. we need to back-transform `CIlow` and `CIhi` by exponentiation;
2. we should use the word ***MEDIAN*** instead of mean;
3. terms ***MULTIPLY*** rather than add.

Your summary should therefore follow a template like this:

> *'We estimate that for every unit increase in x, we MULTIPLY the MEDIAN of Y by between* `exp(CIlow)` *and* `exp(CIhi)` *units.'*



This works because a 1-unit increase in x takes the estimated median of Y ***from*** $\exp(\widehat{\beta}_0)\exp\left\{\widehat{\beta}_1 x\right\}$ ***to*** $\exp(\widehat{\beta}_0)\exp\left\{\widehat{\beta}_1(x+1)\right\} = \exp(\widehat{\beta}_0)\exp\left\{\widehat{\beta}_1 x\right\}\exp(\widehat{\beta}_1)$, so we have multiplied our original value by $\exp(\widehat{\beta}_1)$.

Why is this useful? ***Because the SAME expression holds for ALL x.***

We could also describe the increase in the median of Y for other increases in x:

- We estimate that for every 10 units increase in x, we multiply the ***median*** of Y by between $\exp(10 * CIlow)$ ***and*** $\exp(10 * CIhi)$ ***units.***
- We estimate that for every 0.1 units increase in x, we multiply the median of Y by between `exp(0.1*CIlow)` and `exp(0.1*CIhi)` units.

To describe the fitted model, we first extract the confidence interval for β_1 using `confint(fit)[2,]`, and then back-transform it using `exp(confint(fit)[2,])`.

Percentage growth and Percentage change

We have seen that every 1-unit increase in x corresponds to ***MULTIPLYING the MEDIAN of Y by $e^{\widehat{\beta}_1}$.***

We could also express this as a ***percentage growth*** or a ***percentage change.*** For example:

- If each unit increase in x multiplies the median of Y by 1.2, the final value is ***120% of the original value.***

  The ***percentage change*** is ***a 20% increase.***

  The result has been increased TO 120% of its previous value, but it has been increased BY 20%.

- If each unit increase in x multiplies the median of Y by 0.3, the final value is ***30% of the original value.***

  The ***percentage change*** is ***a 70% decrease.***

  The result has been decreased TO 30% of its previous value, but it has decreased BY 70%.

In general:

- To convert a multiplier $\exp(\widehat{\beta}_1)$ into a ***percentage***, just ***multiply by 100.***

  Then write that each unit increase in x increases the median of Y ***TO*** the specified percentage of its original value.

  For example: *'Every unit increase in x increases the median of Y TO 120% of its previous value.'*

- To convert a multiplier $\exp(\widehat{\beta}_1)$ into a ***percentage change***, ***subtract 1 and then multiply by 100.***

  Or equivalently, multiply by 100 and then subtract 100.

  Then write that each unit increase in x changes the median of Y ***BY*** the specified percentage of its original value.

  For example: *'Every unit increase in x increases the median of Y BY 20%.'*

  If the change is a decrease rather than an increase, the percentage change will be negative. In that case you could also discard the negative sign and describe it as a percentage decrease. For example, if $\exp(\widehat{\beta}_1) = 0.3$, then:

  - the percentage ***change*** is $100 \times (0.3 - 1) = 100 \times (-0.7) = -70$***%,***
  - the percentage ***decrease*** is ***70%.***

In practice, you will do these calculations using ***confidence intervals.***

- Convert multiplier to a percentage and use the word TO:

  `100 * exp(confint(fit)[2,])`.

- Convert multiplier to a ***percentage change*** and use the word BY:

  `100 * ( exp(confint(fit)[2,]) - 1 )`.

If you are scaling the units of x, e.g. to report percentage change for every 5 units increase in x, where should you put the multiplier 5?

Directly before the 'confint'. We scale the confidence interval on the linear scale, and then back-transform it.

So the ***percentage change*** in the median of Y for every 5 units increase in x is:

`100 * ( exp( 5*confint(fit)[2,] ) - 1 )`.

It is very important to have all the brackets and terms in the right places.

Examples

Fill in the numbers to make these statements equivalent.

1. Every unit increase in x is associated with a multiplicative increase in the median of Y:
   a) of between 1.5 and 1.8 units;
   b) TO between ***150% and 180% of its previous value;***
   c) BY between ***50% and 80%.***

2. Every 10-unit increase in x has the following effect on the median of Y:
   a) multiplied by between 3 and 4 units;
   b) ***increased TO between 300% and 400% of its previous value;***
   c) ***increased BY between 200% and 300%.***

3. Every unit increase in x has the following effect on the median of Y:
   a) multiplied by between 0.1 and 0.2 units;
   b) ***decreased TO between 10% and 20% of its previous value;***
   c) ***decreased BY between 80% and 90%.***

4. Every 100-unit increase in x has the following effect on the median of Y:
   a) decreased by between 60% and 70%.
   b) decreased to between ***30% and 40% of its previous value;***
   c) multiplied by ***between 0.3 and 0.4 units.***

Back-transforming confidence and prediction intervals for fitted values

As before, you can create confidence and prediction intervals for individual fitted values ***on the log scale*** using `predict(fit, newdata, interval="conf")` and `predict(fit, newdata, interval="prediction")`.

You need to back-transform these intervals by exponentiation to report them on the Y-scale.

- For ***confidence intervals***, exponentiate and then report as the MEDIAN.

  'With 95% confidence, we estimate that the median house price in a suburb located 20km from the city centre lies between $800,000 and $900,000.'

- For ***prediction intervals***, exponentiate. There is no change in interpretation from the simple linear model.

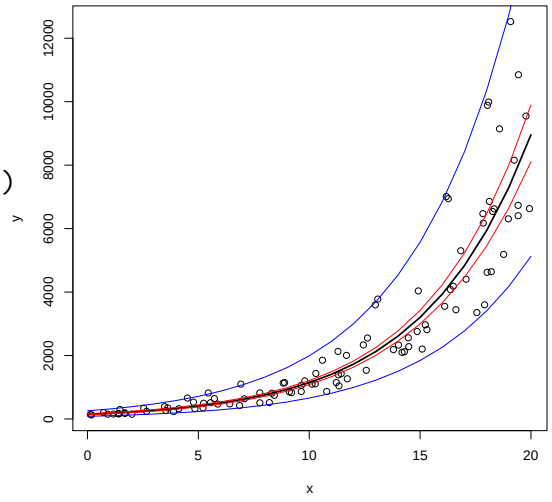
  'With 95% confidence, we estimate that the price of an individual house in a suburb located 20km from the city centre will be between $600,000 and $1,200,000.'

To obtain these intervals, exponentiate the usual intervals as follows:

```
exp( predict(fit, data.frame(x=20), interval="conf") )
exp( predict(fit, data.frame(x=20), interval="prediction") )
```

Example

```
## Generate simulated data where log(Y)
## follows the simple linear model:
mydat <- data.frame(x=runif(100, 0, 20))
mydat$log.y <- 5+0.2*mydat$x+rnorm(20,0,0.3)
mydat$y <- exp(mydat$log.y)

head(mydat)
       x log.y    y
    19.6   8.5 4773
    13.0   6.7  823
    16.1   7.8 2493
    :  :   :
## Fit the simple linear model to LOG Y:
fit.log <- lm(log.y~x, mydat)
```




```
## Backtransform for confidence intervals and prediction intervals ON THE Y-SCALE:
conf.res <- exp( predict(fit.log, data.frame(x=0:20), interval="conf") )
pred.res <- exp( predict(fit.log, data.frame(x=0:20), interval="prediction") )
```

Extract the CIs using `conf.res[,"lwr"]`, `conf.res[,"upr"]`, etc.
The confidence intervals are for the ***median*** of Y. The prediction intervals are intervals in which we expect about 95% of Y data-points to fall.

Chapter 7: log-log models (both x and y are log-transformed)

In some models, we log-transform not only Y, but also X. We call these ***log-log models or power-law models.*** The model structure is:

$\log(Y_i) = \beta_0 + \beta_1 \log(x_i) + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

This model is similar to the previous model, except now ***the predictor, X, is log-transformed, as well as the response, Y.***

If we exponentiate both sides, we get:

$$\begin{aligned} \text{median}(Y_i) &= \exp(\beta_0) \times \exp\{\beta_1 \log(x_i)\} \\ &= e^{\beta_0} \times x_i^{\beta_1} \\ &= \kappa\, x_i^{\beta_1}, \end{aligned}$$

where we use the notation $\kappa = e^{\beta_0}$ for convenience.

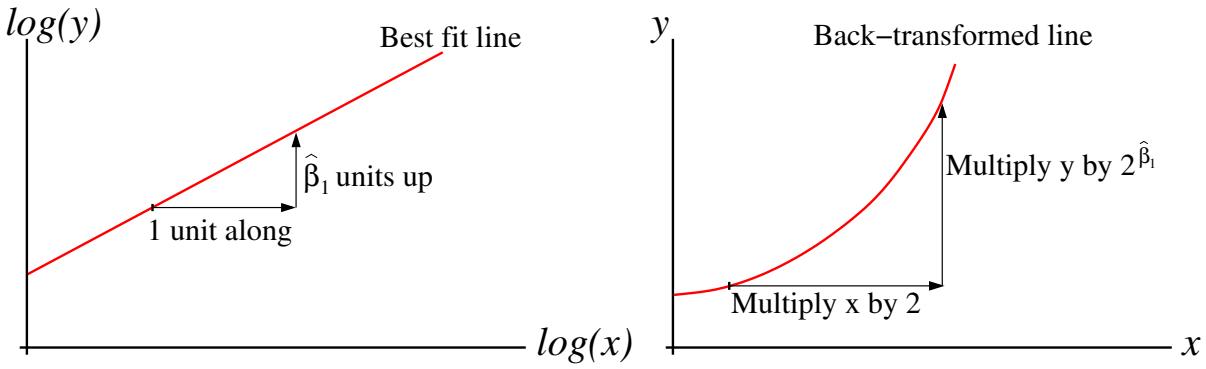
Our model therefore says that the median or central value of Y is some unknown power of x:

$$\text{median}(Y_i) = \kappa\, x_i^{\beta_1}.$$

This is why we call the model a ***power-law model.*** The aim is to ***estimate the power, β_1: e.g. does Y increase as X^3, or $X^{2.5}$, or . . . ?***

Describing the log-log model in the Executive Summary

When we describe the log-log model in the Executive Summary, we need to use ***multiplicative changes in BOTH x AND Y.***



What you need to remember is:

- Every time we multiply x by M, we multiply the median of Y by $M^{\widehat{\beta}_1}$.

For example, if we double x:

$$(\text{new median of } Y) = \kappa\,(2x)^{\widehat{\beta}_1} = 2^{\widehat{\beta}_1}\left(\kappa\, x^{\widehat{\beta}_1}\right) = 2^{\widehat{\beta}_1} \times (\text{old median of } Y).$$

When should we use a log-log model?

There are two main reasons for using a log-log or power-law model:

1. We have external knowledge that the relationship is a power-law, e.g. due to the laws of physics or geometry.
2. We want to be able to express both variables multiplicatively or in terms of ***percentage changes.*** For example:
   - *We estimate that every time we double the value of the predictor, x, we multiply the median of Y by between 3 and 5.*
   - *We estimate that every time we increase the predictor x by 10%, we increase the median of Y by between 25% and 30%.*

Use the same rules as for Chapter 6:

- To convert a multiplier $M^{\widehat{\beta}_1}$ into a ***percentage***, just multiply by 100.

  Then write that every time we multiply x by M, we increase the median of Y ***TO*** the specified percentage of its original value.

  For example: *'Every time we double x, we increase the median of Y TO between 120% and 130% of its previous value.'*

  Because the multiplier is $M = 2$, use `100 * 2^confint(fit)[2,]`.

- To convert a multiplier $M^{\widehat{\beta}_1}$ into a ***percentage change***, subtract 1 and then multiply by 100. (Or equivalently, multiply by 100 and then subtract 100.)

  At the same time, convert the multiplier itself into a percentage change in x, by subtracting 1 and multiplying by 100. Multiplying x by M is equivalent to increasing x by $100\,(M-1)\%$.

  Then write that every time we increase x by $100(M-1)\%$, we change the median of Y ***BY*** $100\left(M^{\widehat{\beta}_1} - 1\right)\%$.

  For example: *'Every 10% increase in x changes the median of Y BY between 20% and 30%.'*

  What is the value of M in this case? ***For a 10% increase in x, we need $M = 1.10$.***

  Because the multiplier is $M = 1.10$, use `100*( 1.10^confint(fit)[2,] - 1 )`.

  Note: if the percentage change in x is very small, e.g. 1%, then the calculation for percentage change, `100*(1.01^confint-1)`, is approximately equal to `confint`, so using `confint` suffices. However, it's complicated to remember this exception, so it's better just to remember the general rules above.

Chapter 6 summary

Try log-transforming Y if the residual plot, `plot(lm.fit, which=1)`, shows ***increasing and right-skewed scatter.*** Conduct the usual `normcheck` plots to check the suitability of the log-transformation.

$\log(Y_i) = \beta_0 + \beta_1 x_i + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$, or alternative model.

Fit with `lm(log(y)~x)`, or `lm(log(y)~x+I(x^2))`, etc.

Back-transform confidence and prediction intervals to the Y scale using exp. Make sure you only talk about the ***median*** of Y, where you usually talk about the mean. For fitted medians, and their confidence and prediction intervals, use `exp(fitted(fit))`, `exp(predict(fit, newdata, interval="conf"))`, and `exp(predict(fit, newdata, interval="prediction"))`.

To describe the model in the Executive Summary, if using the straight-line model $\log(Y_i) = \beta_0 + \beta_1 x_i + \epsilon_i$, we estimate each unit ***additive*** increase in x:

— multiplies the median of Y by $\exp(\widehat{\beta}_1)$ units;

— changes the median of Y ***TO*** $100\exp(\widehat{\beta}_1)\%$ of its current value;

— changes the median of Y ***BY*** $100\left\{\exp(\widehat{\beta}_1) - 1\right\}\%$.

In practice, report these numbers using confidence intervals:

— multiplies by `exp(confint(fit)[2,])` units;

— changes median of Y to `100*exp(confint(fit)[2,])`% of current value;

— changes median of Y by `100*( exp(confint(fit)[2,]) - 1 )`%.

Chapter 7 summary

$\log(Y_i) = \beta_0 + \beta_1 \log(x_i) + \epsilon_i$, where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

Fit with `lm(log(y)~log(x))`.

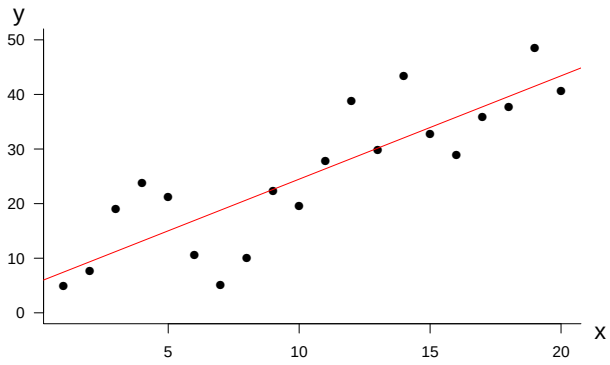
The equivalent power-law description is: median$(Y_i) = \kappa\, x_i^{\beta_1}$, where $\kappa = e^{\beta_0}$.

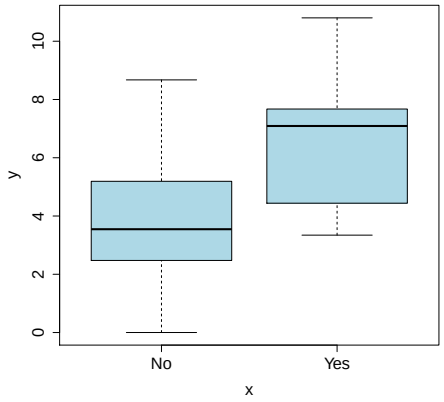
For fitted values, back-transform using exp, and refer to the ***MEDIAN*** of Y. To obtain the fitted medians, and their confidence and prediction intervals, use `exp(fitted(fit))`, `exp(predict(fit, newdata, interval="conf"))`, and `exp(predict(fit, newdata, interval="prediction"))`.

To describe the model in the Exec Summary, use multiplicative or percentage changes in ***BOTH*** x and Y. For example, using $M = 1.50$ for a 50% increase in x: *We estimate that for every 50% increase in x, the percentage change in the median of Y is between* `100*( 1.50^confint(fit)[2,] - 1 )`%.

Stats 20x Handout 3: Multiple predictors

Chapter 8: different fitted lines for two different groups





We have seen models where Y is a response to a numeric variable, X (Chapters 1-2), and where Y has two different means for two different groups (Chapter 5).

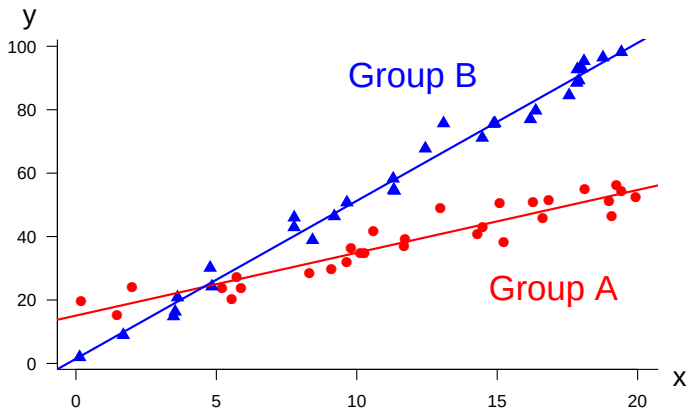
Here, we combine these two models. Suppose we have data that include ***both*** a categorical group variable with levels A and B, ***and*** a numeric variable, X:

gp	x	y
A	0.2	19.57
A	1.5	15.38
A	2.0	24.17
:	:	:
B	0.1	1.97
B	1.7	8.95
:	:	:

We might wish to fit ***a different straight line for each group:***

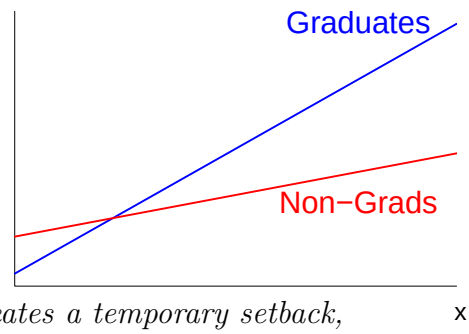
For Group A: $Y_i = \alpha_1 + \beta_1 x_i + \epsilon_i$ where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$,

For Group B: $Y_i = \alpha_2 + \beta_2 x_i + \epsilon_i$ where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$.

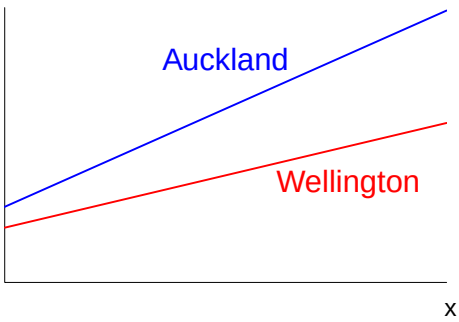


Why would we want to fit two different lines?

- X is Age (21-65), Y is income after debts (\$). We want to compare how net income differs for university graduates and non-graduates.
  Studying at university creates a temporary loss of income. By what age is this neutralized, and how much advantage does it bring in the long term?
  — *Many similar examples where an initial investment creates a temporary setback, but a faster rate of return in the long run.*



- House prices in Auckland and Wellington. X is year, Y is house sale price. We might suspect prices are more expensive in Auckland, but are they also increasing at a higher rate?
  — *Many similar examples where X is year, and we want to test growth under two conditions. For example, Y is kiwi population size under two different predator control regimes.*



Our interest in fitting these models is to investigate:

1. the relationship between X and Y in each group;
2. ***whether this relationship DIFFERS between the two groups.***

Similar to the set-up for Chapter 5 (categorical variables), the default way that R handles this type of model is:

- Fit an intercept, α_1, and a slope, β_1, for the ***baseline group.***
  (Unless requested otherwise, the baseline group is decided alphabetically.)
- Fit a ***top-up*** intercept parameter, α_{extra}, for the second group. The final intercept for the second group is $\alpha_2 = \alpha_1 + \alpha_{\text{extra}}$. We can test if $\alpha_{extra} = 0$.
- Also fit a ***top-up*** slope parameter, β_{extra}, so the second group has slope $\beta_2 = \beta_1 + \beta_{\text{extra}}$. We can test if $\beta_{extra} = 0$.
  (If $\beta_{\text{extra}} \neq 0$, this might force the intercepts apart, so the previous test is of less interest.)

Writing the model formula in the Methods and Assumptions section

When writing the model formula, use notation $\beta_0, \beta_1, \beta_2, \ldots$ for parameters, and define ***dummy variables*** for categorical predictors as in Ch 5 (Handout 2). For the house price example, Auckland will automatically be baseline, so define dummy variable `Wellington` which takes value 1 if house i is in Wellington and 0 otherwise. To give Wellington ***both*** a top-up intercept ***and*** a top-up slope:

$$\texttt{Price}_i = \beta_0 + \beta_1 \times \texttt{Year}_i + \beta_2 \times \texttt{Wellington}_i + \beta_3 \times \texttt{Year}_i \times \texttt{Wellington}_i + \epsilon_i$$

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

Interaction terms

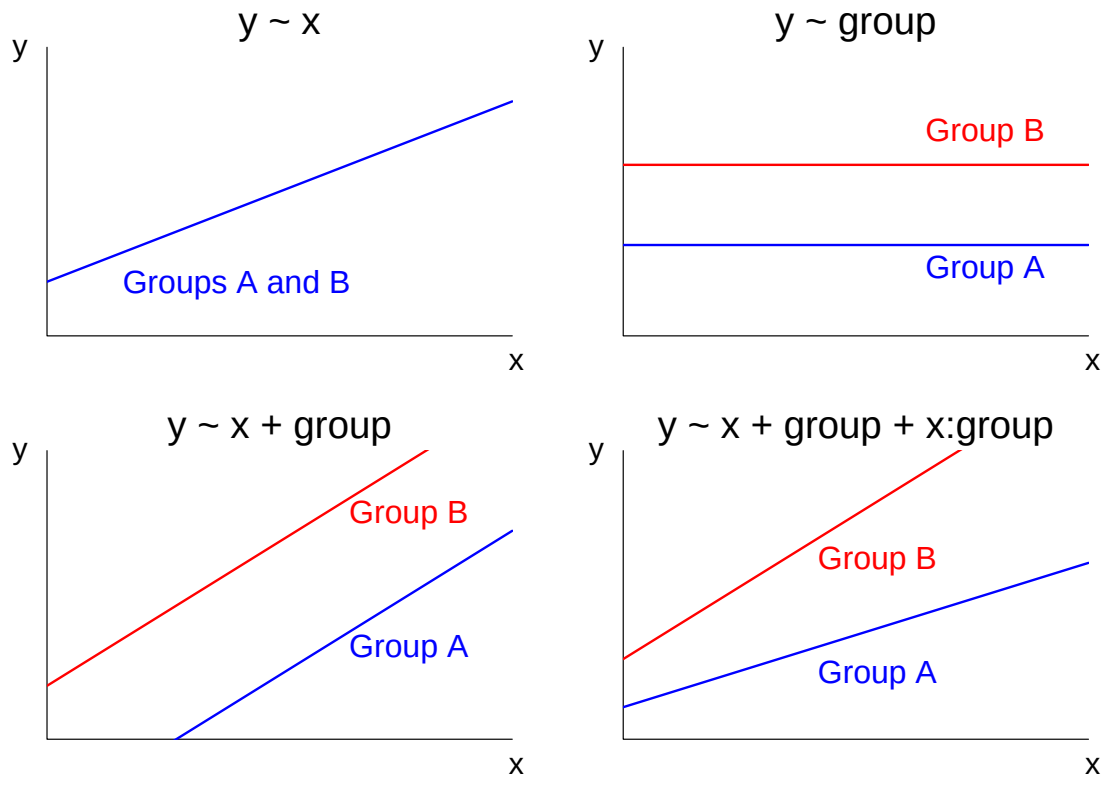
When a model includes a different slope for each group, this is called an ***interaction*** effect.

- Y is related to X through a sloping relationship. We could say that X *acts* on Y: a change in X brings about a change in the mean of Y.
- The 'Group' variable also changes the mean of Y: it shifts the mean up or down depending on which group the subject is in. We can say that the Group variable also *acts* on Y to change its mean.
- When the *slope of the line also depends upon Group,* then we say that the two predictors ***interact*** in their effect on Y.

The X variable acts on Y via a slope, while the Group variable acts on Y by generating different intercepts. An ***interaction*** between X and Group means that each group has a different *slope* as well as a different intercept.

> An ***interaction*** between a numeric variable and a factor means that ***we fit a different slope for every different group in the factor.***

Different ways for X and Group to act on Y



Fitting the interaction model

We build up the model term by term, and also interpret the output in the same way. First generate some data such that there are two groups, A and B, with two different straight-line relationships between X and Y.

```
## Group A:   Y = 15 + 2*x + Normal(0, sigma=4)
datA <- data.frame(group=rep("A", 30), x=runif(30, 0, 20))
datA$y <- 15 + 2*datA$x + rnorm(30, 0, 4)

## Group B: Y = 2 + 5*x + Normal(0, sigma=4)
datB <- data.frame(group=rep("B", 30), x=runif(30, 0, 20))
datB$y <- 2 + 5*datB$x + rnorm(30, 0, 4)

## Join datA and datB using row-bind (rbind) to make a single data frame, mydat:
mydat <- rbind(datA, datB)
```

group	x	y
A	0.2	19.57
A	1.5	15.38
:	:	:
B	0.1	1.97
B	1.7	8.95
:	:	:

```
## Fit the interaction model:
myfit <- lm(y ~ x + group + x:group, data=mydat)
summary(myfit)
  Call:
  lm(formula = y ~ x + group + x:group, data = mydat)

  Coefficients:
              Estimate Std. Error t value Pr(>|t|)
  (Intercept)  15.0795     1.5405   9.789 9.85e-14 ***
  x             1.9809     0.1178  16.816  < 2e-16 ***
  groupB      -13.6050     2.1457  -6.340 4.22e-08 ***
  x:groupB      2.9994     0.1651  18.162  < 2e-16 ***
  ---
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

  Residual standard error: 3.645 on 56 degrees of freedom
  Multiple R-squared:  0.979,Adjusted R-squared:  0.9779
  F-statistic: 870.7 on 3 and 56 DF,  p-value: < 2.2e-16
```

By default, R has taken the baseline group to be ***Group A.*** The fitted line for the baseline group is: ***Group A:*** $\hat{y} = 15.1 + 1.98x$***:*** very close to the true relationship of $\mathbb{E}(Y|x) = 15 + 2x$.

The intercept for Group B is the baseline intercept minus 13.6, and the slope for Group B is the baseline slope plus 3.0 (or 2.9994). So the fitted line for Group B is: ***Group B:*** $\hat{y} = (15.1 - 13.6) + (1.98 + 3.0)x$***, i.e.*** $\hat{y} = 1.5 + 5.0x$. Again this is close to the true relationship of $\mathbb{E}(Y|x) = 2 + 5x$.

Testing for an interaction

In general, we fit the interaction model using the shorthand `lm(y ~ x * group)`.

The notations lm(y ∼ x + group + x:group), and lm(y ∼ x ∗ group), mean the same thing.

The way that R sets up the fit, as a baseline intercept and slope, plus contrast intercepts and slopes for the other group levels, enables us to test whether these contrasts are significant in the model. The most important of these tests is ***the test for different slopes.***

If the test $H_0 : \beta_{\text{extra}} = 0$ is ***non-significant,*** then our sample data ***do not support the need for two different slopes.***

Thus, if we see no $*$'s in the R output for the `x:groupB` row, we have no evidence against H_0. In this case, ***we should generally drop the interaction term and retreat to the simpler model y ∼ x + group. (Refit this model.)*** This simpler model is called a ***'main effects only' model.***

By contrast, if we do see $*$'s in the R output for `x:groupB`, then we do have evidence against H_0 and ***we should retain the interaction effect.*** In this case, we have evidence that $\beta_{\text{extra}} \neq 0$, so group B has a different slope from group A.

If the interaction term is significant, we usually retain all its terms — in other words we keep all three terms `x + group + x:group`, even if the row for `x` or `group` is not marked as significant. This means that we can always interpret the parameters in terms of baseline group plus top-ups or contrasts. While you could try omitting the main effect terms, this would be unusual unless you had a good reason to do so. For example, if there was some physical reason why both groups must have the same intercept, you could fit the model `x + x:group`, which would give each group a different slope but the same intercept. However, as a strong guideline, ***look only at the ouput for the interaction row. If it is significant, keep the whole interaction term, x * group.***

Notes: 1. The model with an interaction between a numeric and a categorical variable is also called an ANCOVA model. ANCOVA stands for 'Analysis of Covariance', but it is probably less confusing to think of it as 'analysis of covariables', because of the interaction between the two variables X and Group.

2. The R command `model.matrix(myfit)` can be useful to see exactly what terms R is adding together to get the fitted value for every observation.

Chapter 9: ANOVA for factors with more than two levels

When a categorical variable contains ***more than two groups***, we can no longer rely on the t-tests we get from the `summary(lm.fit)` output to assess the significance of the factor. Instead, we need to use the ***anova*** command.

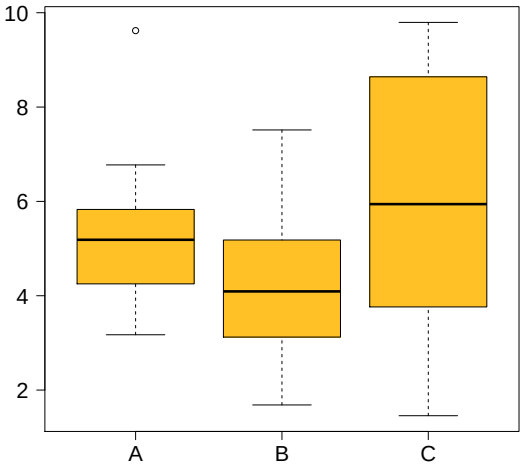
There are two reasons we can no longer use the `summary(lm.fit)` output alone.

1. The `summary(lm.fit)` output only compares all groups ***against the single baseline group.*** It is possible that there is no significant difference between Groups A and B, and between Groups A and C, but there might be a significant difference between Groups B and C. If this is the case, we will not notice it if we only look at `summary(lm.fit)`.

   Note: we can use the `relevel` command to change the baseline group to B or C, and refit the model to see the summary with the new baseline. This is also called 'rotating the factor'. However, this will be very cumbersome for more than 3 groups, and doesn't solve the second problem below.

Example:

```
## Create a categorical variable with
## three different levels, A, B, and C:
mydat <- data.frame(group=c(rep("A", 20),
                rep("B", 20), rep("C", 20)))

## Generate three different means for
## mydat$y in the three groups:
mydat$y <- c(5+rnorm(20, 0, 2),
             4+rnorm(20, 0, 2),
             6+rnorm(20, 0, 2))
```




```
plot(y~group, mydat)  ## Plots the boxplot above
myfit <- lm(y~group, mydat) ## Fits the linear model with "group" as a factor
summary(myfit)  ## Using this, it appears nothing is significant:

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.2157     0.4339  12.020   <2e-16 ***
groupB       -1.0308     0.6137  -1.680   0.0985 .
groupC        0.7479     0.6137   1.219   0.2280

## We can use 'relevel' to rotate the Group factor:
## we now see that groups B and C differ significantly, but this is cumbersome.
summary(lm(y~relevel(group, ref="B"), mydat))

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                  4.1850     0.4339   9.644 1.39e-13 ***
relevel(group, ref = "B")A   1.0308     0.6137   1.680  0.09850 .
relevel(group, ref = "B")C   1.7786     0.6137   2.898  0.00532 **
```

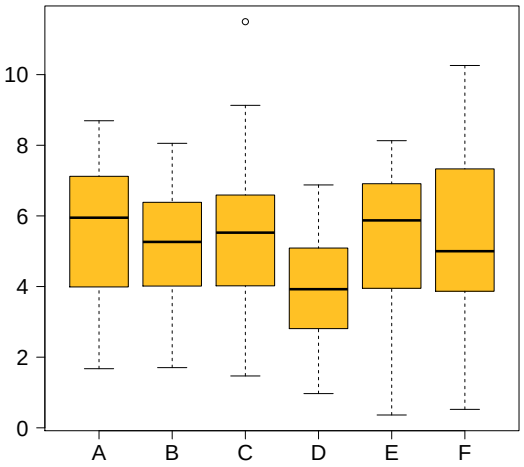
```
## Using anova, we only need one command to show the group factor is significant:
anova(myfit)

Analysis of Variance Table
Response: y
          Df  Sum Sq Mean Sq F value  Pr(>F)
group      2  31.902 15.9510  4.2355 0.01928 *
Residuals 57 214.666  3.7661
```

2. The second reason why we do not use `summary(lm.fit)` to test for inclusion of the whole factor is the problem of ***multiple testing.*** If we use 5% as our threshold of statistical significance, then we have a 5% chance of generating a 'false positive' or Type I error for every hypothesis test we conduct. This false positive rate expands rapidly if we conduct multiple tests. For example, if we conduct 5 tests in which H_0 is true, the probability that ***at least one test*** will generate a false positive (falsely significant result) is $1-(1-0.05)^5 = 0.23$. This is way higher than our ideal of 0.05 for the false positive rate.

   If we fit a factor with k levels, this will generate $k-1$ pairwise t-tests against the baseline reference group in the `summary(lm.fit)` output. We then have an unacceptably high false positive rate.

Example:

```
## Create a categorical variable with
## six different levels, A, B, C, D, E, F:
mydat <- data.frame(group=c(rep("A", 20),
                rep("B", 20), rep("C", 20),
                rep("D", 20), rep("E", 20),
                rep("F", 20)))

## Generate mydat$y with the SAME mean
## in each of the six groups:
mydat$y <- 5 + rnorm(120, 0, 2)

plot(y~group, mydat)
myfit <- lm(y~group, mydat)

summary(myfit) ## One of the 5 pairwise tests is significant (false positive):

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.49017    0.46256  11.869   <2e-16 ***
groupB      -0.34032    0.65416  -0.520   0.6039
groupC       0.06894    0.65416   0.105   0.9163
groupD      -1.34140    0.65416  -2.051   0.0426 *
groupE      -0.24224    0.65416  -0.370   0.7118
groupF       0.02415    0.65416   0.037   0.9706
```



```
## anova(myfit) correctly deduces there is no difference between the groups:
anova(myfit)

Analysis of Variance Table
Response: y
           Df Sum Sq Mean Sq F value Pr(>F)
group       5  28.41  5.6821  1.3278 0.2574
Residuals 114 487.83  4.2792
```

ANOVA

ANOVA stands for ***analysis of variance.*** Instead of looking at a series of comparisons of each group with a baseline group, it treats the whole categorical variable in one go.

ANOVA tests whether or not we should include the WHOLE VARIABLE in the model.

It does this by looking at the ***decrease in residual variance due to this variable,*** and comparing it with the overall residual variance. This is why it is called 'analysis of variance.' There are limits to how much decrease in variance can be obtained by a 'lucky fluke', under the null hypothesis that the variable has no impact on the response. If inclusion of the variable in our model generates a greater decrease in variance than would be expected by lucky fluke, ANOVA will give a significant result and we can conclude that the variable is fulfilling a useful purpose and should be retained in the model.

The broad idea behind this is a fundamental principle of modelling: the ***principle of parsimony.*** This principle specifies that we should always aim to use ***the smallest adequate model*** for our data. Think of variables in a model like employees in a business. Every employee costs the company money, because they need salary. An employee is only productive if they earn the business more than they cost. The same is true of variables. A variable *costs* uncertainty in a model: it requires a number of parameters to be estimated, each of which uses up information and creates uncertainty in the form of standard errors. However, a variable also creates an opportunity to *reduce* uncertainty, in the form of the residual variance, by 'explaining' some of the variance in the way we described in Chapter 1. The idea of ANOVA is that we want to keep variables in the model if they 'earn more than they cost': that is, if the work that they do in reducing the residual variance is beyond lucky chance. The more parameters that the variable introduces into the model, the higher the reduction in residual variance must be for the variable to 'earn its keep'.

The ANOVA table

Just for the record, here is how the ANOVA table is computed. We will use the table from the student language example in Chapter 9 of the coursebook. The response, `lang`, measures student language scores after they have been treated with one of three different teaching methods (predictor `method`). There is an additional numeric predictor of student `IQ`. The overall model is: `lm( lang ~ IQ * method )`, so we are fitting a different straight-line response of language score to IQ, for each of the three different teaching methods.

```
teach.fit <- lm(lang ~ IQ * as.factor(method), data=teach.dat)
anova(teach.fit)
Analysis of Variance Table

Response: lang

                       A       B       C=B/A        C/D     p-val: df from B & 24

                      Df     Sum Sq    Mean Sq    F value       Pr(>F)
IQ                     1    1004.42    1004.42    26.1416    3.124e-05 ***
as.factor(method)      2    2901.83    1450.91    37.7625    3.867e-08 ***
IQ:as.factor(method)   2      78.82      39.41     1.0257       0.3737
Residuals             24     922.13      38.42

               resid df=24             D=38.42
```

Column A shows how many parameters each variable requires: the more parameters, the better the reduction in variance the variable will need to produce to 'earn its keep'. Thus column A is connected with the 'cost' of each variable.

Column B displays the reduction in residual variance that each variable has contributed to the model.

Column C compares the useful contribution in column B, with the 'cost' of the variable in column A. This value needs to be sufficiently high for the variable to be retained.

Value D provides the baseline variance against which column C is compared. Our test statistic for whether the variable has 'earned its keep' is given in the '`F value`' column, and is obtained from C/D.

Roughly speaking, the second row of the table shows that the variable `method` costs 2 parameters, but those parameters contribute 1450 each to reducing the residual variance. This compares very favourably with the final residual variance of $D = 38.42$ per observation, so we conclude that these two parameters have been very successful and we will retain this variable in the model.

By contrast, the interaction term `IQ:method` also costs 2 parameters, but these parameters only make a contribution of size 39.4 each. This is not a large contribution compared with the residual variance of 38.4 for each residual observation. In fact, each parameter should make a significantly larger contribution than the residual variance per observation to justify its inclusion. The result is that the interaction term is non-significant and we do not retain it in the model.

Chapter 10: models with multiple predictors of different formats

In Chapter 10 we see that we can add multiple predictors to a single model, including multiple numeric and categorical variables, together with interactions.

In general, choosing an optimal set of predictors is not easy, and is treated in more detail in Stats 330. However, there is one key point to note now. This is the risk of ***multi-collinearity,*** which occurs when ***two predictors are linearly related to each other.***

For example, if two predictors X_1 and X_2 are repeating almost the same information as each other, the model cannot distinguish which of the response it should attribute to X_1 and which to X_2. Pairs of predictors that have a closely linear relationship can be identified visually using the `pairs20x` function.

The classic symptoms of multi-collinearity in a fitted model are that adding one of the predictors: (a) makes little difference to R^2; (b) suddenly switches off the significance of one or more previous predictor; and (c) inflates the standard errors of all the predictors involved. See what happens in the example below.

```
mydat <- data.frame(x1=runif(100, 0, 20))
## Make predictor x2 VERY closely related to predictor x1:
mydat$x2 <- mydat$x1 + rnorm(100, 0, 0.1)
## y is a response to x1, but will also be very closely related to x2:
mydat$y <- 5 + 2*mydat$x1 + rnorm(100, 0, 3)

## If we fit the model with x1 only, all is well:
summary(lm(y~x1, mydat))     ## Multiple R-squared:  0.9299
   Coefficients:
               Estimate Std. Error t value Pr(>|t|)
   (Intercept)  4.45960    0.65486    6.81  7.9e-10 ***
   x1           2.10132    0.05829   36.05  < 2e-16 ***

## Fit the model with x1+x2 and we see the symptoms of collinearity:
summary(lm(y~x1+x2, mydat))   ## Multiple R-squared:  0.9303
   Coefficients:
               Estimate Std. Error t value Pr(>|t|)
   (Intercept)   4.4502     0.6561   6.783  9.3e-10 ***
   x1            5.0569     3.6391   1.390    0.168
   x2           -2.9553     3.6382  -0.812    0.419
```

Stats 20x Handout 4: *n*-way ANOVA

ANOVA for testing inclusion of categorical variables in a model

In Chapters 9 and 10, we saw the R command `anova(lm.fit)` for testing whether an entire variable should be kept in a model. Recall that 'anova' stands for ***'analysis of variance'***. The `anova()` command is needed when the model includes ***factors with three or more levels.*** For other cases, it will tend to deliver the same results as the `summary(lm.fit)` command. The key points to remember about the `anova` command are:

1. `anova` considers the contribution of the variable ***as a whole*** to reducing the residual variance in the model.

2. It solves the problem in `summary(lm.fit)` that all groups are compared only with the single baseline group, so differences between two other groups might go unnoticed.
   (E.g. if neither group B nor C differs from group A, but B and C do differ from each other).

3. Importantly, it solves the problem of ***multiple testing*** in `summary(lm.fit)`. Conducting numerous pairwise tests between each group and the baseline group greatly inflates the chance of seeing ***false positives.*** Typically we consider results 'significant' if they return a p-value ≤ 0.05, and this same value 0.05 defines the false-positive rate of the test. If H_0 is true and we conduct k different tests, then the chance that ***at least one*** will return a false-positive result is about $1-(1-0.05)^k$, which is unacceptably high even for k as small as 3 or 4. The `anova` command rethinks the whole question, and returns an output based on whether the variable as a whole has reduced the residual variance more than could reasonably be 'fluked' under H_0.

One-way ANOVA and two-way ANOVA

The word 'ANOVA' is strictly speaking a type of test, as described above. However, it is also used colloquially to describe a ***type of model for which this test is appropriate.*** This colloquial usage has become very widespread and entrenched, so it is important to understand both meanings of the word.

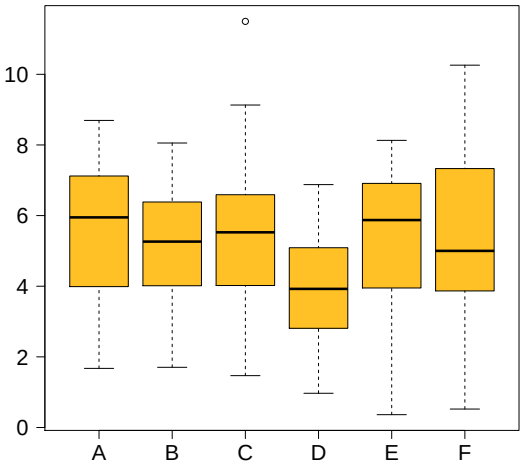
- A ***one-way ANOVA*** describes a model with a single predictor consisting of ***one categorical variable with 3 or more levels.***

  lm(y ~ x), where y is numeric and x is a factor with ≥ 3 levels.

  We are interested in testing whether ***all*** groups have the same mean.

  This is equivalent to testing whether the variable needs to be included in the model at all: because if all groups have the same mean, the null model `lm(y ~ 1)` applies. Thus, the command ***anova(lm(y~x)) will simultaneously answer both questions.***

A ***one-way ANOVA*** simply tests if ***all groups have the same mean.*** This can be viewed as testing whether the categorical variable makes a significant contribution to reducing residual variance, over and above the null model `lm(y~1)` in which all groups do have the same mean.



'Do the groups have the same mean?', and 'Is the variable justified?', can be seen as the same question posed in different ways. The single command `anova(lm(y~x))` addresses both questions.

- We will also see ***two-way ANOVA*** models. These are models with ***two predictors, both of which are categorical variables.*** We are interested in the main effects, and also in the interaction between the two predictors: `lm(y ~ x1 * x2)`. The command `anova(lm(y ~ x1 * x2))` displays whether the main effects and the interaction are significant.

We have already covered most of the material needed to understand one-way and two-way ANOVA models. The new material in this handout is:

- 'Means and effects' notation for writing out the model formula.
- Commands `summary1way` and `multipleComp` for assessing one-way ANOVA models and the significance of pairwise comparisons, adjusted for the multiple-testing problem described above.
- Commands `interactionPlots` and `summary2way` for exploring two-way ANOVA data and assessing model fit.

All four of the functions mentioned above are in the `s20x` library: `library(s20x)`.

Chapter 11: One-way ANOVA

This example comes from the Fruitfly data in Chapter 11 of the coursebook. There are 5 groups of male fruitflies, placed under different conditions based on the number of receptive females nearby. The response is lifespan, measured in `days`. The idea is that the potential for reproductive success might influence a male fruitfly's lifespan, in terms of what (if anything) he has to live for.

```
head(Fruitfly.df): days group
                     40    G1
                     37    G1
                     44    G1
```

```
tail(Fruitfly.df): days group
                     56    G5
                     60    G5
                     44    G5
```

Our model has the following formulation: `lm ( days ~ group )`. Because `group` is a categorical variable, this means: ***fit a different intercept parameter for every different level of the variable 'group'.***

Up to now, we have written out the model formula in Methods & Assumptions like this:

$$\texttt{days}_i = \beta_0 + \beta_1\,\texttt{group2}_i + \beta_2\,\texttt{group3}_i + \beta_3\,\texttt{group4}_i + \beta_4\,\texttt{group5}_i + \epsilon_i\,,$$

where $\epsilon_i \sim$ iid Normal$(0, \sigma^2)$, and where:
$\texttt{group2}_i$ is 1 if the fruitfly is in group 2 and otherwise 0;
$\texttt{group3}_i$ is 1 if the fruitfly is in group 3 and otherwise 0;
and so on to group 5.

This is a tedious notation, so we now introduce another notation called ***means and effects notation.*** We can rewrite the model formula like this.

Let subject$_{ij}$ be the jth subject in group i, for groups $i = 1, 2, 3, 4, 5$. Then:

$$\texttt{days}_{ij} = \mu + \alpha_i + \epsilon_{ij}\,,$$

where μ is the overall mean lifespan across all groups;
α_i is the ***effect*** of being in group i: α_i ***is a top-up intercept for group i;***
and where $\epsilon_{ij} \sim$ iid Normal$(0, \sigma^2)$ as usual.

Thus, we reconsider the model to be composed of:

- a ***grand mean,*** μ (the mean of Y across all groups);
- an ***effect of being in group i***, α_i, which describes the difference of group i from the grand mean, so it is easily interpreted as a positive effect or a negative effect;
- and the usual scatter term ϵ_{ij}, which is now given a double subscript $_{ij}$ to emphasize that we are looking at an individual subject (j) coming from group i. That is, every different fly in group i has its own scatter value, ϵ_{ij}. We cannot just use a single subscript ϵ_i because that would imply that every member of group i has exactly the same value for its random scatter.

We will come back to the means and effects notation later. For now we will continue with the one-way ANOVA example.

The ANOVA table

We will use the following model fit:

```
Fruitfly.fit <- lm(days ~ group, data=Fruitfly.df)
```

```
> anova(Fruitfly.fit)
Analysis of Variance Table

Response: days
           Df Sum Sq Mean Sq F value    Pr(>F)
group       4  11939 2984.82  13.612 3.516e-09 ***
Residuals 120  26314  219.28
```

The significance of the `group` variable, as a whole, is shown in the first row. In the example above, we learn that the `group` variable adds *4* parameters to the model, and on average the amout of work each of these 4 parameters does to reduce the residual variance is `Mean Sq`$= 11939/4 = 2984.8$.

Compare this with an average of $26314/120 = 219.28$ residual variance left ***per observation*** in the Residuals line, after model parameters have been deducted from the degrees of freedom. We can think of this as a reference requirement for a parameter to be included in the model.

- One observation tells us ***one thing*** about the population.
- One parameter *also* tells us ***one thing*** about the population. For example, it tells us that Y has a population mean of μ.
- Thus there is a duality between ***parameters*** and ***observations.*** We can think of each parameter as 'using up' the information from one observation.
- We use the term ***degrees of freedom*** to count both observations and parameters. If there are n observations and p parameters, then the degrees of freedom left after the parameters have been deducted is $n - p$.
- This makes sense: for example, if we have n observations (n d.f.), and specify *one* thing about them (e.g. their mean is μ), then we can only *freely* choose $n - 1$ observations in order to satisfy this constraint. Once we have chosen $n - 1$, the last one is determined by the knowledge that they must all have mean μ. So we say that the mean μ has 'used up' 1 degree of freedom, leaving us with only $n - 1$ d.f. for the residuals.

If parameters somehow 'equate' to observations, then it makes sense to use the amount of *residual variance per observation* as a benchmark against which to compare the useful work a parameter is doing in the model:

- An observation *adds* residual variance to the model (via its scatter). We can view this as the 'cost' of each degree of freedom in the model.
- A parameter *reduces* the residual variance. The more a parameter reduces the residual variance, the more useful it is.
- A parameter is deemed to be making a *useful contribution* if it reduces the residual variance by a lot more than the 'cost' of one degree of freedom.

Going back to the ANOVA table:

```
           Df Sum Sq Mean Sq F value    Pr(>F)
group       4  11939 2984.82  13.612 3.516e-09 ***
Residuals 120  26314  219.28
```

The ***cost*** of one d.f. in this model is $26314/120 = 219.28$: the residual variance per residual d.f.

The ***contribution*** per d.f. made by the `group` variable is $11939/4 = 2984.82$, which is 13.6 times larger than the cost of each d.f.: $2984.82/219.28 = 13.6$.

It is vanishingly unlikely that a variable could 'fluke' such a staggering result by chance ($p = 3.5 \times 10^{-9}$). Thus the `group` variable is contributing much more than simply paying back the 4 d.f. that it costs. This is displayed in the highly significant result, and we conclude we should retain this variable in the model.

As a rough rule, the F-ratio can be compared with the value 1, which creates an equivalence between the amount a parameter earns, and the cost of its d.f. Ratios substantially higher than 1 will deliver significant F tests. The value required for significance depends upon the two degrees of freedom (4 and 120 here), but roughly speaking F-values that are a few integers higher than 1 are likely to be significant, and those close to 1 are likely to be non-significant.

Link between the ANOVA table and R^2

As we have seen, the `Sum Sq` column reveals how much residual variance has been removed by the `group` variable, relative to the null model. The amount that still remains is reported in the `Residuals` row. Thus, we can use these two numbers to calculate the ***proportion of variance explained,*** which is exactly the same as the definition of R^2 we have been using since Chapter 1.

To calculate R^2, we take the ratio of 'variance explained' to 'total':

$$R^2 = \frac{11939}{11939 + 26314} = 0.3121.$$

This is the same value as we shall see in the output for `summary(Fruitfly.fit)`.

Links between `summary(fit)`, `anova(fit)`, and `summary1way(fit)`

For ANOVA-type models, we do not use `summary(fit)` much, but primarily use `anova(fit)` and a new command, `summary1way(fit)` instead. Here are the connections between the three.

1. `summary()`

The `summary()` command is the familiar model output. However, it suffers from the two problems of ***single baseline group and multiple testing*** mentioned on page 1. For 1-way ANOVA models, it is only really useful for reading off R^2: however even that can be achieved by using `anova` instead of `summary`.

```
> summary(Fruitfly.fit)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   63.560      2.962  21.461  < 2e-16 ***
groupG2        1.240      4.188   0.296    0.768
groupG3       -0.200      4.188  -0.048    0.962
groupG4       -6.800      4.188  -1.624    0.107
groupG5      -24.840      4.188  -5.931 2.98e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.81 on 120 degrees of freedom
Multiple R-squared:  0.3121,Adjusted R-squared:  0.2892
F-statistic: 13.61 on 4 and 120 DF,  p-value: 3.516e-09
```

2. `anova()`

`anova` is the core command for analysing these models. ***IMPORTANT: Use the anova(fit) command first of all, to decide upon the global significance of the categorical variable.***

```
> anova(Fruitfly.fit)
Analysis of Variance Table

Response: days
           Df Sum Sq Mean Sq F value    Pr(>F)
group       4  11939 2984.82  13.612 3.516e-09 ***
Residuals 120  26314  219.28
```

The significance of the `group` variable is shown in the first row. Additionally:

- We can calculate R^2 by $11939/(11939 + 26314) = 0.3121$.
- The F-statistic and p-value from the table match the ones reported in `summary(fit)` above. This is because `summary(fit)` uses the same F-test idea to report on the global significance of ***all*** variables in the model. Here we only have one variable, `group`, so the two tests match.

3. summary1way()

summary1way is a s20x library command that conveniently summarizes the output from anova as well as producing summary-statistics for each group, and also reporting the 'means and effects' formulation of the estimated model.

- summary1way(Fruitfly.fit) combines the outputs from summaryStats(y~group, Fruitfly.df) and from anova(Fruitfly.fit).
- If each of these commands have already been used, then there might be no need for summary1way. We often use summaryStats in a previous exploratory phase.

```
> summary1way(Fruitfly.fit)
ANOVA Table:
                Df  Sum Squares  Mean Square  F-statistic  p-value
Between Groups  4   11939.28     2984.82      13.61195     0
Within Groups   120 26313.52     219.27933
Total           124 38252.8

Numeric Summary:
          Sample size     Mean  Median  Std Dev  Midspread
All Data          125    57.44      58 17.56389         24
G1                 25    63.56      62 16.45215         28
G2                 25    64.80      65 15.65248         22
G3                 25    63.36      65 14.53983         21
G4                 25    56.76      56 14.92838         20
G5                 25    38.72      40 12.10207         15

Table of Effects: (GrandMean and deviations from GM)
typ.val      G1      G2      G3      G4      G5
  57.44    6.12    7.36    5.92   -0.68  -18.72
```

Things to look for:

- Are the standard deviations within each group *reasonably* similar?
  As a rule of thumb, accept up to a factor of two difference.

  Similarly for the midspreads. The midspread is the interquartile range within the group, i.e. the difference between the 75% and the 25% quantiles.

- The ***Grand Mean*** and ***Effects*** output matches the mean and effects notation $y_{ij} = \mu + \alpha_i + \epsilon_{ij}$. Here, the 'grand mean' is μ and is (obscurely) given by typ.val= 57.44 in the table of effects. The effects G1 to G5 specify $\alpha_1, \ldots, \alpha_5$ and give us a quick insight into which groups have high means (positive effects) and which have low means (negative effects).

  Here we see group G5 has a large negative effect (very low mean relative to the overall mean μ).

Pairwise comparisons, adjusted for multiple testing

So far, we have used `anova` to decide that the categorical variable is significant: so the factor must contain groups whose means differ on a pairwise basis. We have specified that we can ***not*** use the output from `summary(fit)` to decide *which* pairs of group means differ — however, we still wish to answer this question.

How can we specify *which* groups differ from each other, without falling into the multiple-testing trap?

The answer is that we need to use a ***different test:*** one that is specifically designed to be a global test that does not fall foul of multiple-testing. The test we use is called ***Tukey's adjusted test***, named after American statistician John Tukey (1915-2000). Another name is ***Tukey's Honest Significant Difference method,*** or Tukey's HSD test.

The broad idea behind Tukey's adjustment is that we deliver *all* the pairwise differences between groups as a single 'package' or 'parcel'. Tukey's test is designed such that ***the probability of getting at least one false positive in the WHOLE package is restricted to 0.05.***

That is, if H_0 is true, then 5% of 'parcels' will contain at least one falsely significant result just by chance. This means that 95% of parcels will contain *no* falsely significant results. So our standard understanding of what we mean by a significance test is preserved.

Tukey's adjusted test is designed to create an accurate false-positive rate across ALL p-values simultaneously.

For this reason it is sometimes called a 'simultaneous' test.

The more specific idea behind Tukey's test is that it poses the null hypothesis of (say) k samples *all* drawn from the same population: for example we have $k = 5$ groups in our fruitfly example. Tukey derived test statistics for the *MAXIMUM pairwise difference among ANY of the k groups.* We would expect this to be larger than the pairwise difference between just two groups, which is what our `summary(fit)` pairwise t-tests deliver. The pairwise differences among groups in our sample are then referenced against this global-pairwise-difference distribution. The native R function for this calculation is `TukeyHSD`, and the name of the global reference distribution is the *studentized range distribution.* However, the Stats 20x team have wrapped up a more easily-accessible version of the Tukey HSD test in an `s20x` library function called `multipleComp`, where 'multipleComp' stands for ***multiple comparisons.***

Here is the output from `multipleComp` for the fruitfly ANOVA model:

```
> multipleComp(Fruitfly.fit)
          Estimate  Tukey.L Tukey.U Tukey.p
G1  -  G2    -1.24 -12.8405 10.3605  0.9983
G1  -  G3     0.20 -11.4005 11.8005  1.0000
G1  -  G4     6.80  -4.8005 18.4005  0.4855
G1  -  G5    24.84  13.2395 36.4405  0.0000
G2  -  G3     1.44 -10.1605 13.0405  0.9970
G2  -  G4     8.04  -3.5605 19.6405  0.3127
G2  -  G5    26.08  14.4795 37.6805  0.0000
G3  -  G4     6.60  -5.0005 18.2005  0.5158
G3  -  G5    24.64  13.0395 36.2405  0.0000
G4  -  G5    18.04   6.4395 29.6405  0.0003
```

You can see that the function gives ***all possible pairwise comparisons*** between groups G1 to G5. There are 10 comparisons in total: $10 = \binom{5}{2} = {}^5C_2$. Compare this with the output from `summary`, which only gives the 4 pairwise comparisons of groups G2 to G5 with the baseline group, G1.

The final column, `Tukey.p`, gives the Tukey-adjusted p-values for all 10 pairwise comparisons. These p-values have been adjusted legitimately for multiple testing, so we can be confident in their validity.

It is legitimate to quote results like, 'The mean of group G5 differs significantly from the mean of every other group G1 to G4.'

We know that the overall false-positive rate (i.e. Type I error rate) of our whole 'parcel' of comparisons is 5%, if we take 'significance' to mean the Tukey adjusted p-value is 0.05 or less. So we have confidence in our 'parcel' of conclusions.

We can use the `subset` command to extract only those pairwise comparisons that are significant at the 5% level. Note that R returns the `multipleComp` results in matrix format, so we must first convert the matrix to a dataframe to apply the `subset` command.

```
> Fruitfly.mc <- multipleComp(Fruitfly.fit)
> subset(data.frame(Fruitfly.mc), Tukey.p < 0.05)
          Estimate Tukey.L Tukey.U Tukey.p
G1  -  G5    24.84 13.2395 36.4405   0e+00
G2  -  G5    26.08 14.4795 37.6805   0e+00
G3  -  G5    24.64 13.0395 36.2405   0e+00
G4  -  G5    18.04  6.4395 29.6405   3e-04
```

Like any significance test, the Tukey-adjusted test can also be converted into Tukey-adjusted 95% confidence intervals for each pairwise mean difference. The CIs are reported in the `Tukey.L` and `Tukey.U` columns. The confidence intervals ***are wider than those obtained from "summary".***

An example conclusion is: 'We estimate that the mean lifetime for Group 5 flies is ***between 13.2 and 36.4 days lower than that for Group 1 flies'.***

Summary of one-way ANOVA models

Here is a set of steps for conducting a one-way ANOVA analysis.

1. Plot the data using `plot(y ~ group)`, which defaults to a boxplot when `group` is categorical.
2. Generate numerical summaries within each group using `summaryStats(y ~ group)`. Check that standard deviations and midspreads within each group are reasonably similar, e.g. ***within a factor of two from smallest to largest.***
3. Fit the one-way ANOVA model using `fit <- lm(y ~ group)`.
4. Check the fit using `plot(fit, which=1)`, `normcheck(fit)`, and `cooks20x(fit)`.
5. View the fit using `anova(fit)`. If the `group` factor is not significant, this is the final conclusion and probably the end of the analysis.
6. If the `group` factor is significant, then use `multipleComp(fit)` to identify which pairs of groups differ significantly, and read off 95% confidence intervals for these differences.
7. Use `summary(fit)` ONLY to read off R^2 once the final model is decided.

Suitable wording for the Executive Summary: *We have very strong evidence ($p < 0.001$) that the lifespan of male fruitflies differs according to treatment group. Inspection of pairwise differences, controlled for multiple testing, reveals that males in Group 5 have significantly lower lifespans than males in every other group. We estimate that the mean lifespan for Group 5 males ranges from 6 to 38 days less than that of males in the other groups.*

Chapter 12: Two-way ANOVA

A two-way ANOVA model is simply a model with ***a numeric response, y, and TWO categorical predictors.***

If the first predictor has k levels, and the second has ℓ levels, the number of possible combinations of the predictors is $k \times \ell$. Even if $k = \ell = 2$, there are already 4 possible combinations. Thus, for any two-way ANOVA, however small the individual factors are, we need to use the global testing method `anova` to test for inclusion of variables and interactions. We also need to use a multiple-test adjustment when identifying combinations of the predictors with a significant effect.

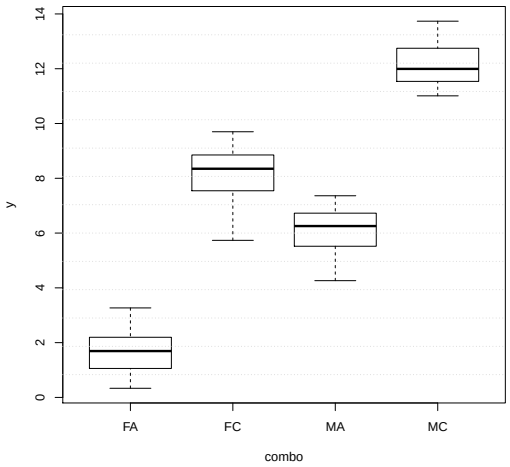
We will use the `s20x` function `interactionPlots` for initial exploratory plots. After the model has been fitted and viewed with `anova`, we will use `summary2way` to generate concise summaries of significant effects.

Build-your-own two-way ANOVA

Two-way ANOVAs can be tricky to comprehend, so we will use simulated data to clarify the ideas. We will simulate a numeric response y to two factors, `Sex` (male or female) and `Age` (child or adult).

First we will simulate a response ***without interaction*** between Sex and Age.

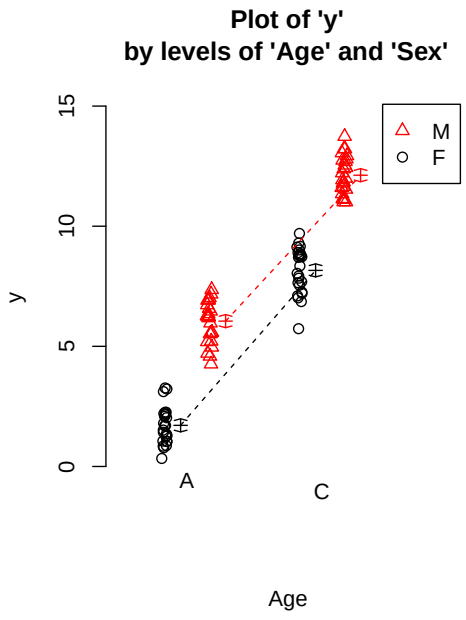
```
## Generate data with 50 observations in each of FA, FC, MA, MC combinations:
mydat <- data.frame(Sex=c(rep("F", 50), rep("M", 50)),
                    Age=rep(c(rep("A", 25), rep("C", 25)), 2))

## Generate y with baseline mean 2 for FA, add 6 for C, and add 4 for M:
mydat$y<-2+as.numeric(mydat$Age=="C")*6+as.numeric(mydat$Sex=="M")*4+rnorm(100,0,1)

## Cheat and create a "combo" factor with levels FA, FC, MA, MC:
mydat$combo <- factor(paste0(mydat$Sex, mydat$Age))

## Plot the cheat factor, combo:
plot(y ~ combo, mydat)
```



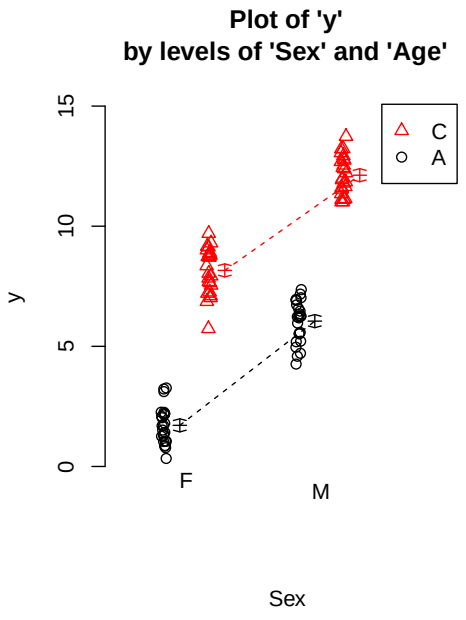
We can see that going from Adult to Child creates a jump of 6 units, and going from Female to Male creates a jump of 4 units, and there is no interaction between the two factors.

Look at `interactionPlots` for this dataset:

```
interactionPlots(y~Age * Sex, mydat)
```




```
interactionPlots(y~Sex * Age, mydat)
```



Whichever way around the interaction plots are plotted, the lines between the two groups are ***parallel.***

Check the ANOVA analysis on the data without interaction:

```
> myfit <- lm(y ~ Sex * Age, mydat)
> anova(myfit)
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq   F value Pr(>F)
Sex        1 430.54  430.54  595.5998 <2e-16 ***
Age        1 980.24  980.24 1356.0245 <2e-16 ***
Sex:Age    1   0.88    0.88    1.2216 0.2718
Residuals 96  69.40    0.72
```

The results are exactly what we would hope for: ***the main effects are both significant, but the interaction term is not significant.***

To finish off the analysis of the no-interaction data, we fit the simpler model without interaction. We then use `summary2way` for Tukey-adjusted pairwise comparisons, specifying the argument `page="nointeraction"`.

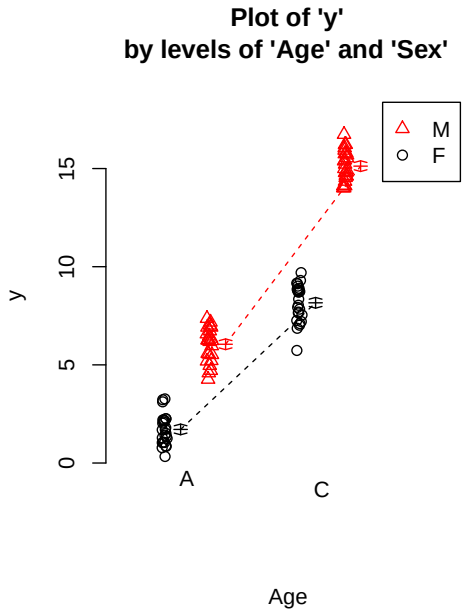
```
> myfit.final <- lm(y ~ Sex + Age, mydat)  ## Main effects only
> summary2way(myfit2, page="nointeraction")
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = fit)

$Sex
        diff      lwr      upr p adj
M-F 4.149915 3.812039 4.487791     0

$Age
        diff      lwr      upr p adj
C-A 6.261749 5.923873 6.599625     0
```

The results are what we would expect. The pairwise differences for (Male - Female) are about 4 units, and those for (Child - Adult) are about 6 units.

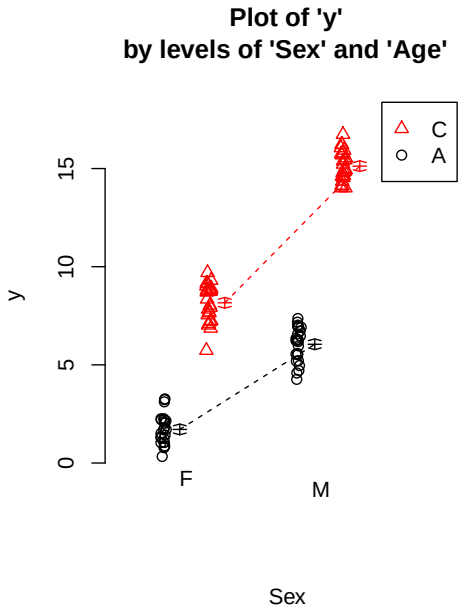
Now let's see what happens when we have a model ***with interaction.*** To create an interaction, we need to add an extra top-up mean to observations in one of the four combo-categories. Let's give Male Children an extra top-up of 3 units.

```
mydat.int <- mydat
mydat.int$y[mydat.int$combo=="MC"] <-  mydat.int$y[mydat.int$combo=="MC"] + 3
myfit.int <- lm(y ~ Sex * Age, mydat.int)
```

```
interactionPlots(y~Age * Sex, mydat)
```




```
interactionPlots(y~Sex * Age, mydat)
```



The lines are no longer parallel. ***Use interactionPlots(model) to get a quick idea of whether there is an interaction between the two factors. An interaction is signalled by non-parallel lines.***

Use `anova(fit)` as usual to test formally for the interaction.

```
anova(myfit.int)
          Df  Sum Sq Mean Sq  F value    Pr(>F)
Sex        1  798.04  798.04 1103.977 < 2.2e-16 ***
Age        1 1506.12 1506.12 2083.509 < 2.2e-16 ***
Sex:Age    1   43.04   43.04   59.536 1.128e-11 ***
Residuals 96   69.40    0.72
```

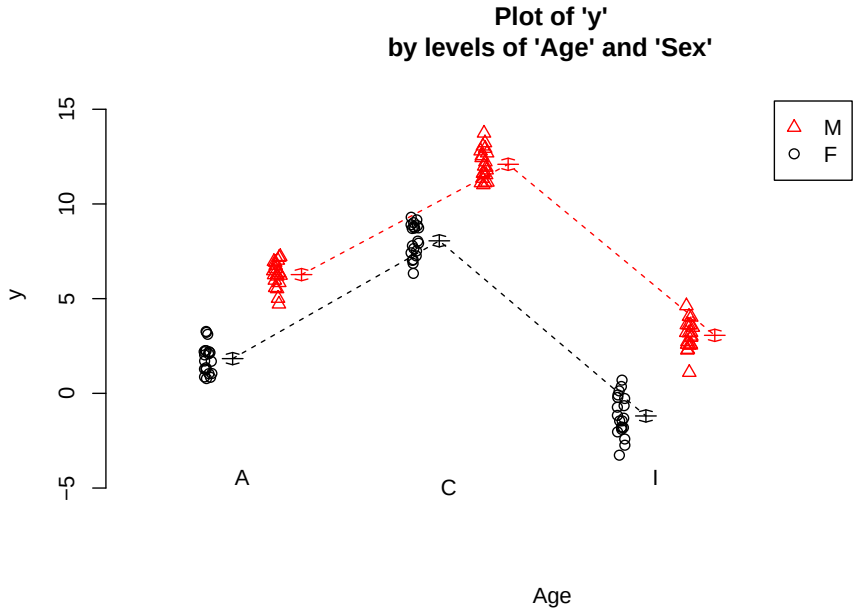
The interaction is significant. Now use `summary2way` with `page="interaction"` to investigate pairwise effects:

```
summary2way(myfit.int, page="interaction")
  Tukey multiple comparisons of means
    95% family-wise confidence level

$`Comparisons within Sex`
            diff      lwr      upr p adj
F:C-F:A 6.449696 5.820939 7.078454     0
M:C-M:A 9.073802 8.445045 9.702560     0

$`Comparisons between Sex`
            diff      lwr      upr p adj
M:A-F:A 4.337862 3.709104 4.966620     0
M:C-F:C 6.961968 6.333210 7.590726     0
```

Note: If you use `page="nointeraction"` on an interaction model, the results give overall within-group effects while ignoring the interaction effect, and are hard to interpret.

Interaction or not?

Do these plots represent an interaction between Age and Sex, or not?
A third level has been added to the Age variable: level 'I' for Infant.




```
interactionPlots(y~Age * Sex)
```

No: there is no interaction here. The coloured lines are parallel to each other, even though they are not parallel from one level of Age to another.

Check with ANOVA:

```
Analysis of Variance Table

Response: y
           Df  Sum Sq Mean Sq   F value Pr(>F)
Age         2 1726.74  863.37 1212.8885 <2e-16 ***
Sex         1  540.48  540.48  759.2812 <2e-16 ***
Age:Sex     2    0.79    0.39    0.5544  0.576
Residuals 114   81.15    0.71
```

As expected, there is no significant interaction from the ANOVA analysis. We should now ***refit the main-effects only model for our final analysis: lm(y ~ Age + Sex).***

Summary of two-way ANOVA models

1. A two-way ANOVA has a numeric response variable, y, and two categorical predictors: x_1 and x_2.

2. First use `interactionPlots( y ~ x1 * x2 )` for a visual impression of whether there is an interaction between the two predictors. An interaction is signalled by non-parallel lines among the different colours on the plot.

3. Fit the model `lm(y ~ x1 * x2)`. Do the usual three assumption checks.

4. Check for an interaction term using `anova(fit)`.

5. If the interaction is significant, use `summary2way(fit, page="interaction")` to explore pairwise comparisons. The model formula is:

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk},$$

where: μ is the grand mean;
α_i is the effect for factor x_1;
β_j is the effect for factor x_2;
γ_{ij} is the interaction effect for the combination of factors x_1 and x_2;
and $\epsilon_{ijk} \sim$ iid Normal$(0, \sigma^2)$ as usual.

6. If the interaction is not significant, revert to the simpler model with main-effects only: `lm(y ~ x1 + x2)`. Check for significance of the main effects using `anova(fit)`. If they are both significant, use `summary2way(fit, page="nointeraction")` to explore pairwise comparisons.

7. If one of the main effects (say, x_2) is still not significant, revert to the single-effect model, `lm(y ~ x1)`. This is now a one-way ANOVA model and can be analysed as shown in Chapter 11.

8. Use `summary(fit)` ONLY to read off R^2 once the final model is decided.

Means and effects notation

Means and effects notation is a compact notation for describing a model formula. It is particularly useful for models involving factors with multiple combinations.

1. First count the number of categorical variables (factors). If there are v factors, you will need $v+1$ different subscripts. For example: ***1-way ANOVA:*** $v = 1$ ***so we need*** $v + 1 = 2$ ***subscripts: use*** i ***and*** j***.***
2-way ANOVA: $v = 2$ ***so we need*** $v + 1 = 3$ ***subscripts: use*** i***,*** j***,*** k***.***

2. The ***response*** is easy: it will use all subscripts. ***E.g.*** y_{ij} ***or*** y_{ijk}***.*** The final subscript enables all individuals to be unique.

3. Start off with a ***grand mean,*** μ***.***

4. For ***main effects,*** use a different Greek letter for each ***factor.*** The *levels* within a factor are taken care of by the subscripts, i, j etc.

- α_i means 'use a different top-up intercept for every group in factor 1.' For example, if factor 1 is Sex, then α_i denotes a top-up intercept for level i of the Sex variable. Here, i could be Male or Female: so α_1 is a top-up intercept for Males, and α_2 is a top-up intercept for Females. In general, α_i is a top-up intercept for whichever group i is relevant to observation y_{ijk}.
- Likewise, β_j signals a different top-up intercept for each group in factor 2. Remember to keep the subscripts separate for the different factors: factor 1 gets subscripts i, factor 2 gets subscripts j. For example, if factor 2 is Age, then levels j could be either Child or Adult: so β_1 is a top-up intercept for Children, β_2 is a top-up intercept for Adults, and β_j is a top-up intercept for ***whichever group j is relevant to observation y_{ijk}.***

5. If there are ***interaction terms,*** these will involve two or more variables, so they have ***two or more subscripts.*** Use another Greek letter to describe the interaction, but keep the subscripts attached to the original variables.

   For example, an interaction between Sex and Age would be described by γ_{ij}. This means ***a different top-up intercept for every combination of Sex and Age.***

   This term deals with the extra parameters needed to deal with non-parallel lines in the interaction-plots.

6. The ***error term*** will use all subscripts, just like the response. ***E.g.*** ϵ_{ij} ***or*** ϵ_{ijk}***.***

Note: the notation is a way of ***understanding*** the terms in the model. We still fit the model exactly as before. For example, although there are 4 combinations for γ_{ij} in the example above, we still only fit 1 extra parameter, because we have already fitted the other 3 parameters in the main effects.

Question: How would you write the ANCOVA model of Chapter 8 in means and effects notation?

Answer: There is a different intercept and a different slope for each level of a single factor variable. We only have one factor, so we need 2 subscripts, i and j, where the second subscript is unique to each subject. The numeric predictor is called x. For the jth subject in group i, we can write the group-specific top-up intercept as α_i, and the group-specific slope as β_i. Writing x_{ij} for the observation of x for this subject:

$$y_{ij} = \mu + \alpha_i + \beta_i\, x_{ij} + \epsilon_{ij}.$$

Stats 20x Handout 5: Non-Normal Models

Rethinking the scatter model

We have seen numerous examples of models with the general formulation

$$Y_i = \alpha + \beta x_i + \epsilon_i \quad \text{where } \epsilon_i \sim \text{iid } N(0, \sigma^2).$$

We can read this as:

- the mean of Y_i is $\alpha + \beta x_i$, and the variance is σ^2.

We could view this model a slightly different way from usual. Instead of viewing Y_i as 'mean plus Normal scatter', we could describe it as 'Normal scatter about a mean'. That is, we could rewrite our model for Y_i as follows:

$$Y_i \sim N(\alpha + \beta x_i \, , \sigma^2).$$

The notation $Y_i \sim N(\alpha + \beta x_i \, , \sigma^2)$ simply means that Y_i ***has a Normal (bell-curve) distribution, with mean*** $\alpha + \beta x_i$***, and variance*** σ^2***.***

This is exactly the same as saying that Y_i is generated via a mean term $\alpha + \beta x_i$, with add-on Normal scatter ϵ_i that has mean 0 and variance σ^2. In other words:

$$\alpha + \beta x_i + N(0, \sigma^2) \quad \equiv \quad N(\alpha + \beta x_i \, , \, \sigma^2).$$

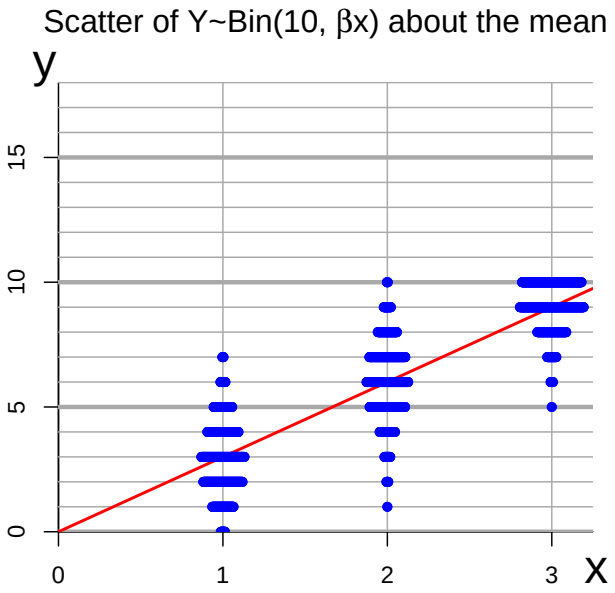
In itself, this seems a fairly obvious and uninteresting statement. However, it opens up a whole new way of thinking about modelling. If we can formulate models using statements like $Y_i \sim N(\alpha + \beta x_i \, , \sigma^2)$, then ***why do we need to stick to the Normal distribution?***

We could propose models where Y_i is Binomial, or Poisson, or Gamma, or Exponential ... in fact there is a whole world of distributions with different properties, suitable for different situations. Here are some examples.

Binomial distribution

The ***Binomial distribution*** counts the number of successes in n independent trials: e.g. the number of heads out of 10 tosses of a coin. Because it is a discrete ('counting') distribution, it can only take values $0, 1, 2, \ldots, n$. This creates a very different scatter pattern from Normal scatter, which is always symmetric on both sides of the mean. For example, if $Y_i \sim \text{Binomial}(10, p = 0.9)$, then it has mean $10 \times 0.9 = 9$. In that case, the *only* allowed scatter above the mean would be the value 10, whereas below the mean we could have scatter 8, 7, 6, and so on.

Here is a picture of a response Y with a Binomial$(10, p)$ distribution, where the p parameter denotes the probability of 'success' in each of the 10 trials.



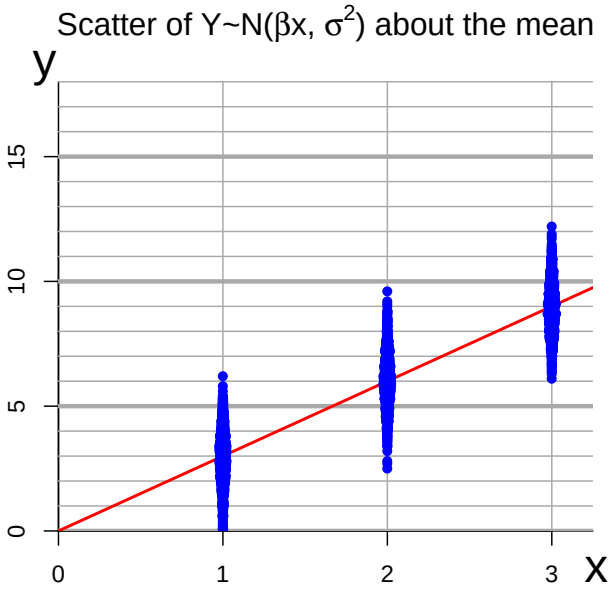
We have a numeric predictor variable, x, and the p parameter depends upon x via $p = \beta x$, where β is our parameter to be estimated.

You can see that the Binomial response Y when there are 10 trials has ***a hard upper limit at*** $Y = 10$***.***

You can see why we might want to use this distribution for modelling. For example, we might be interested in children's scores on a reading test (marked out of 10) as they progress through Year 1, 2, and 3 of school. Each point on the scatterplot corresponds to one child in year x, and that child's mark out of 10 on the reading test.

The Binomial scatter model is very different from our previous models with constant Normal scatter. Firstly, the scatter is not constant about the fitted line. Secondly, Y values are restricted to integers $0, \ldots, 10$; and thirdly, there are hard limits at 0 and 10. The hard limits are particularly problematic: as we can already see from this plot, ***any estimate of*** β ***that sends the fitted line below 0 or above 10 will give an impossible distribution for*** Y***.***

By comparison, the Normal model allows any values for Y — positive or negative, and not just integers — and it has no lower or upper limits. Here is the Normal scatter model for comparison. Note how it goes below 0 and above 10 in this instance. Also notice the constant scatter.



There are two problems to solve if we want to use the Binomial model. Firstly, we can no longer use least-squares for estimating parameters. Remember from Handout 1 that least squares is ***only*** appropriate for the constant-scatter model. However, we also mentioned there that least-squares is just a special case of a much more general estimation method called ***maximum***

likelihood estimation. So this problem is easily solved: as long as we know the equation for probabilities in the Binomial distribution, we can calculate the likelihood and maximize it to generate parameter estimates $\widehat{\alpha}$ and $\widehat{\beta}$. See Stats 210 for details. This is all dealt with by R behind the scenes.

The second problem is the problem of ***hard limits*** that we saw overleaf, where the Binomial fitted line has to be confined between 0 and 10 if the response Y has a Binomial(10, p) distribution. This problem will have an effect on how we formulate the model. We will need to transform the fitted line using a function that maps the whole real line onto the restricted range. We will say much more about this soon, but here is a quick idea. Let's say we have a linear expression, like

$$\ell = \alpha + \beta x \, .$$

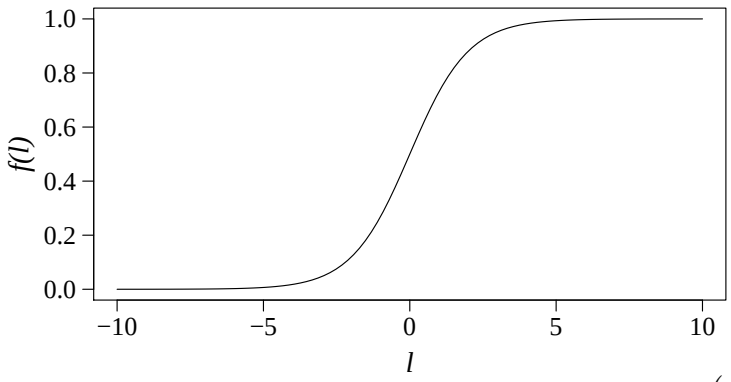
The parameters α and β can take any values in the real numbers, so ℓ can also be any real number, positive or negative.

Now consider: $\quad f(\ell) \quad = \quad \dfrac{\exp(\ell)}{\exp(\ell)+1} \quad = \quad \dfrac{\exp(\alpha+\beta x)}{\exp(\alpha+\beta x)+1} \, .$

We know that $\exp(\ell)$ is always positive, regardless of whether ℓ is positive or negative. And if we take any positive number and divide it by itself plus one, we will always get a fraction.

$$\ell = -\infty \quad \Rightarrow \quad \exp(\ell) = e^{-\infty} \simeq 0 : \qquad \text{so} \qquad \frac{\exp(\ell)}{\exp(\ell)+1} \simeq \frac{0}{0+1} = 0.$$

$$\ell = +\infty \quad \Rightarrow \quad \exp(\ell) = e^{+\infty} \simeq \infty : \qquad \text{so} \qquad \frac{\exp(\ell)}{\exp(\ell)+1} \simeq \frac{\infty}{\infty+1} = 1.$$



The plot shows the shape of the function $f(\ell) = \dfrac{\exp(\ell)}{\exp(\ell)+1}$. Notice how ***the horizontal range is unrestricted between $-\infty$ and ∞, but the vertical range is restricted between 0 and 1, like a probability.***

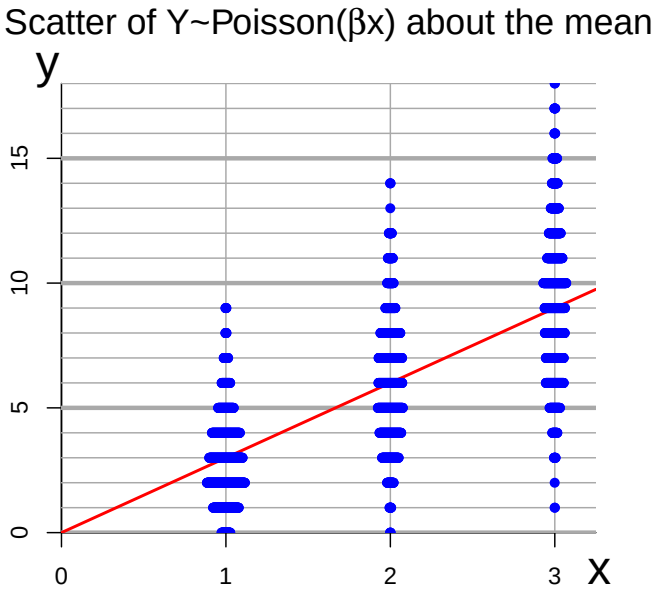
This will be very useful when we want to transform an unrestricted linear predictor ($\ell = \alpha + \beta x$) into a range with both a lower limit and an upper limit.

Poisson distribution

The ***Poisson distribution*** is another counting distribution. It is named after the French mathematician Siméon-Denis Poisson (1781-1840). Formally, the Poisson distribution counts the number of events in a fixed time or space, when events happen at a constant average rate: for example, the number of road accidents in a year, or the number of customers arriving at a supermarket in an hour. However, it is also used as a general-purpose counting distribution, e.g. for the number of kiwi found in a site before and after predator control, or the number of goods bought per day before and after a marketing campaign.

The Poisson distribution has the property that ***variance increases with the mean.*** That is, the uncertainty in an observation increases as its mean gets higher. This is a very familiar phenomenon in everyday life. For instance, imagine you are interested in how long a university assignment will take you. If you estimate it will take 5 hours, you might expect to be an hour wrong either way. On the other hand, if you estimate it will take 100 hours, you might expect to be 10 hours wrong either way. You already have an inbuilt sense that ***variance commonly increases with expected value.***

Here is the scatter model from the Poisson distribution, plotted on the same scale as the previous two plots. The increase in variance with mean is very noticeable. We also notice that the distribution only takes integer values, like the Binomial distribution.



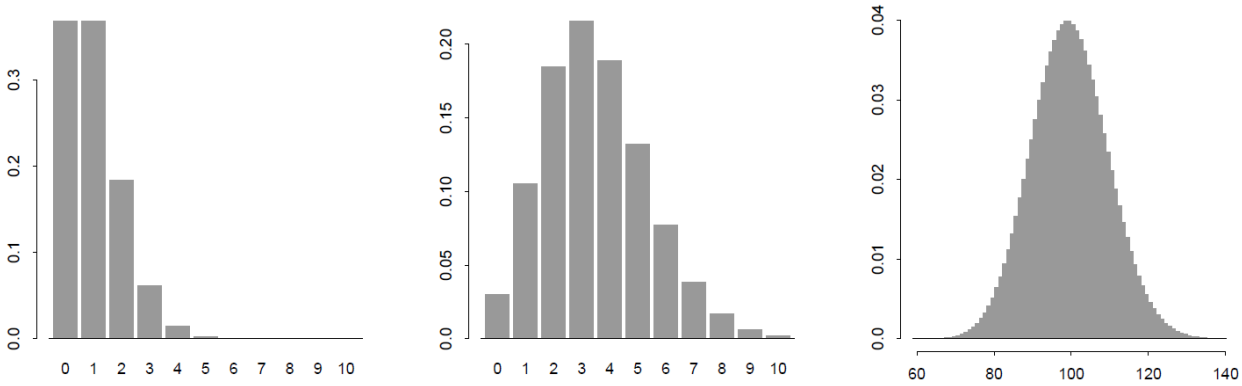
Like the Binomial distribution, the Poisson distribution also has a parameter with a restricted range. We said that the Poisson distribution counts the number of events per unit time, where the number of events per unit time has a constant average. This 'constant average' is given the symbol μ and is the sole parameter of the Poisson distribution. The rate μ represents ***the average number of events per unit time.*** So it is clear that the mean of the Poisson(μ) distribution is μ. It can be shown that the variance of the distribution is equal to μ as well.

$$\text{When } Y \sim \text{Poisson}(\mu), \quad \text{then } \mathbb{E}(Y) = \text{Var}(Y) = \mu.$$

- If road accidents occur at a constant average rate of 400 per year, then the number of accidents in a *particular* year can be modelled as $Y \sim \textbf{\textit{Poisson}}(\mu = 400)$**.** Then the mean and variance of Y are $\mathbb{E}(Y) = \textbf{\textit{Var}}(Y) = 400$**.**
- If earthquakes over a certain magnitude occur at a constant average rate of 5 per week, then the number of earthquakes in a *particular* week can be modelled as $Y \sim \text{Poisson}(5)$, and the mean and variance of Y are both ***5.***

  Here are some Poisson distributions with different parameters μ:

$\mu = 1$ $\mu = 3.5$ $\mu = 100$



The formula for Poisson probabilities is:

$$\mathbb{P}(Y = y) = \frac{\mu^y}{y!} e^{-\mu} \quad \text{for} \quad y = 0, 1, 2, \ldots$$

This formula is derived as a solution to a partial differential equation, so it is far from obvious. All we need to know for Stats 20x is the following.

- The Poisson distribution is a discrete (counting) distribution taking values $0, 1, 2, \ldots$ with no upper limit.
- The mean of the distribution is a parameter μ, and it ***must be positive:*** $\mu > 0$**.** ***Note that*** μ ***will eventually be our 'fitted value' for observation*** Y***: the location of the best-fit line.***
- The variance of the Poisson distribution increases with the mean. ***Scatter increases as the fitted value becomes larger.***

  In fact, strictly speaking, the variance is exactly equal to the mean. However, later we will relax this and consider *'Poisson-like distributions'*, or *quasi-Poisson distributions*, for which the variance is a constant multiple of the mean, rather than being exactly equal.

Because our parameter μ has a ***restricted range,*** $\mu > 0$***,*** we have the same problem mentioned earlier in connection with the Binomial distribution. In the Binomial case, we had a probability parameter p with *two* limits: $0 < p < 1$. For the Poisson case, we only have one limit to contend with: $\mu > 0$. We might still need to use a transformation to ensure that $\mu > 0$, but the selection of suitable transformations is different. By far the most common transformation for Poisson data is the ***exponential / log transformation.***

Again, imagine we have a linear expression, ℓ, which can be any real number, positive or negative:

$$\ell = \alpha + \beta x\,.$$

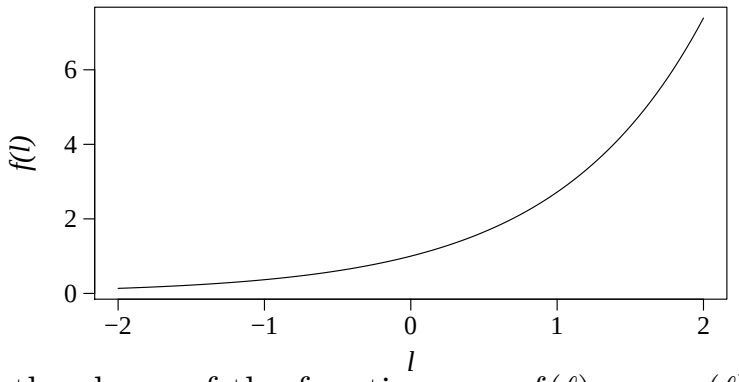
Now consider the exponential transformation, in other words the inverse-log:

$$\mu \;=\; f(\ell) \;=\; \exp(\ell) \;=\; \exp(\alpha + \beta x)\,.$$

Then $\mu = \exp(\ell)$ is always positive, for ℓ either positive or negative.

$$\ell = -\infty \quad \Rightarrow \quad \mu = \exp(\ell) = e^{-\infty} \simeq 0$$

$$\ell = +\infty \quad \Rightarrow \quad \mu = \exp(\ell) = e^{+\infty} \simeq \infty\,.$$



The plot shows the shape of the function $\mu = f(\ell) = \exp(\ell)$. The horizontal range is unrestricted between $-\infty$ and ∞, but the vertical range is restricted between 0 and ∞.

Thus, if we start with a ***linear predictor,*** $\ell = \alpha + \beta x$, and we transform it by the exponential operator to get $\mu = \exp(\ell) = \exp(\alpha + \beta x)$, then our parameters α and β can take any real values but our mean μ is always positive.

Looking at this the other way around: ***if we start with a Poisson variable*** Y ***with mean*** μ***, and model*** $\log(\mu) = \ell = \alpha + \beta x$***, then we can estimate any values of*** α ***and*** β ***without damaging our constraint that*** $\mu > 0$***.***

This is the foundation of a major class of models called ***generalized linear models (GLMs):***

- The log-transformation $g(\mu) = \ell$ is called the ***link function.***
- The distribution of the response, Y, e.g. Poisson, is called the ***error model.***

Generalized Linear Models (GLMs)

Generalized linear models (GLMs) are a very widely-used modelling class. They are useful because they take an important step away from the *simple* linear model that we have seen up to now. That is:

- Generalized linear models allow ***non-Normal error distributions.***

We still use the ***linear predictor***, $\ell = \alpha + \beta x$, as the primary way of modelling the relationship between X and Y. However, as we have seen, when we move away from Normal errors we run into problems with parameters that have restricted ranges. For this reason we need to introduce a ***link function:***

- A ***link function*** is a function $g(\mu)$, that maps a parameter with a restricted range into ***the whole real line.***

Thus there are three components to a generalized linear model:

1. The ***error model:*** e.g. $Y_i \sim$ ***Poisson***(μ_i).
2. The ***link function:*** e.g. $g(\mu_i) = \log(\mu_i)$.
3. The ***linear predictor:*** e.g. $\log(\mu_i) = \alpha + \beta x_i$.

Notes:

1. To qualify as a GLM, only certain error models are allowed, including Normal, Binomial, Poisson, Gamma, and some other distributions. These distributions belong to the so-called ***Exponential family of distributions.*** They have likelihoods of a particularly convenient mathematical form, enabling a very efficient computation for maximum likelihood estimation. In short, ***we get answers quickly.***

   Historically, this computational efficiency made the difference between practicality and impossibility. The ongoing popularity of GLMs today is partly because they were the only practical option in the early days of statistical modelling, and partly because they do offer a wide range of useful modelling options without the options being overwhelmingly diverse. And, they remain very fast.

2. The ***link function*** does not need to be a transformation. We might choose to model $g(\mu_i) = \mu_i$: for example, if we are sure that the fitted line will be far away from 0 and there is no risk of running into problems with negative means. The choice $g(\mu_i) = \mu_i$ is called the ***identity link.*** The choice $g(\mu_i) = \log(\mu_i)$ is called the ***log link.***

3. Much of this course so far has been concerned with doing interesting things to the linear predictor, such as adding quadratic terms, factors, interactions, and so on. All these same operations can still be done with GLMs.

GLM examples

Example 1: Poisson GLM with log link.

The model structure is:

$$Y_i \sim \text{Poisson}(\mu_i) \quad \text{where} \quad \log(\mu_i) = \alpha + \beta x_i \,.$$

Fit with: `glm(y ~ x, family = poisson(link="log"), data = mydat)`

Or, because the log-link is the default for the Poisson family:

```
glm(y ~ x, family = poisson, data = mydat)
```

We back-transform from the linear predictor to the fitted values using

$$\widehat{\mu}_i = \exp\left(\widehat{\alpha} + \widehat{\beta} x_i\right).$$

Crucial point: we are ***NOT transforming Y!***

We are transforming the ***mean*** of Y. Our model statement is fully explicit about the probability distribution of Y: $Y \sim$ ***Poisson***.

The GLM framework enables us to model the mean using the familiar linear predictor, coupled with the link function to keep everything within range, while ***not*** interfering with the probability distribution of Y. Thus, we can choose a probability distribution of Y that is appropriate for the circumstances — such as Poisson — and we don't have to worry about how the properties of this distribution might be messed up by a log-transformation.

Expressing the same idea in two different notations:

$Y_i \sim \text{Poisson}(\mu_i)$	$Y_i \sim \text{Poisson}$
$\log(\mu_i) = \alpha + \beta x_i$	$\log\{\,\mathbb{E}(Y \mid x)\,\} = \alpha + \beta x$
$\mu_i = \exp(\alpha + \beta x_i)$	$\mathbb{E}(Y \mid x) = \exp(\alpha + \beta x)$

Note that ***we do NOT include any extra scatter term, ϵ.***

The scatter model is encompassed in the statement that $Y_i \sim \text{Poisson}(\mu_i)$. It is no longer a simple case of add-on scatter in the way that the Normal scatter model can be viewed. So it would be ***incorrect*** to add an ϵ_i term on to the end of the model statement.

Alternative models: We can use any alternative model formulation for the linear predictor, exactly as we have been doing in Chapters 1-12. For example, $\log(\mu_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$, or $\log\{\mathbb{E}(Y_{ijk})\} = \mu + \alpha_i + \beta_j + \gamma_{ij}$, and so on.

Example 2: Binomial GLM with logit link.

The simplest model structure is:

$$Y_i \sim \text{Binomial}(1, p_i) \quad \text{where} \quad \log\left(\frac{p_i}{1-p_i}\right) = \alpha + \beta x_i \,.$$

This model is suitable for ***binary data*** where Y_i is 0 or 1 depending on whether a single 'trial' was a failure or success. For example, the 'trial' might correspond to a patient given a particular drug treatment; then $Y_i = 1$ if the patient recovered, and $Y_i = 0$ if the patient failed to recover.

Note that for the Binomial(n, p) distribution, the mean is $\mu = n \times p$. So for the Binomial$(1, p)$ distribution, the mean is $\mu = p$. Therefore our linear predictor is

$$g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right) = \log\left(\frac{p_i}{1-p_i}\right) = \alpha + \beta x_i \,.$$

This has the same format as before, where $g(\mu_i) = \ell_i$ is a linear predictor.

Fit with: `glm(y ~ x, family = binomial(link="logit"), data = mydat)`

Or, because the logit-link is the default for the Binomial family:

`glm(y ~ x, family = binomial, data = mydat)`

We back-transform from the linear predictor to the fitted values using

$$\widehat{\mu}_i = \frac{\exp\left(\widehat{\alpha} + \widehat{\beta} x_i\right)}{1 + \exp\left(\widehat{\alpha} + \widehat{\beta} x_i\right)} \,.$$

Crucial point: we are ***NOT transforming*** Y***.***
We are transforming the ***mean*** of Y. Our model statement is fully explicit about the probability distribution of Y: $Y \sim$ Binomial.

Expressing the same idea in two different notations:

$$Y_i \sim \text{Binomial}(1, p_i) \qquad\qquad Y_i \sim \text{Binomial}$$

$$\log\left(\frac{p_i}{1-p_i}\right) = \alpha + \beta x_i \qquad\qquad \log\left(\frac{\mathbb{E}(Y \mid x)}{1 - \mathbb{E}(Y \mid x)}\right) = \alpha + \beta x$$

$$p_i = \frac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)} \qquad\qquad \mathbb{E}(Y \mid x) = \frac{\exp(\alpha + \beta x)}{1 + \exp(\alpha + \beta x)}$$

Connection with log-transformation of Y: the LogNormal distribution

In Chapter 6, we already saw an example where we modelled Y_i with a distribution other than Normal, and used a log-transformation. There, we modelled $\log(Y) \sim$ Normal.

We can now see that we could describe this model a slightly different way. Instead of saying $\log(Y) \sim$ Normal, we could define a new distribution called the 'logNormal distribution', which has the property that its log is Normal — or, seen a different way, it is the exponential of a Normal distribution. Then we could write $Y \sim$ logNormal, and $\log\{\mathbb{E}(Y \mid x)\} = \alpha + \beta x$.

This looks a lot like what we have just done for the Poisson GLM. Indeed, it is essentially the same idea, but with a different scatter model. The only reason why we call this a ***log-transformation of*** Y instead of a GLM is because the logNormal distribution does not happen to fall into the Exponential Family of distributions that defines the GLM class (see page 7). Since this rests on technical details beyond the scope of this course, you can be assured for now that we are not doing anything very different in these two cases.

Most statisticians would prefer to use a GLM rather than a transformation of Y, because Y describes the ***physical phenomenon of interest*** and therefore we want to be clear exactly what our assumptions are about Y. It is rather unintuitive and confusing to try to figure out what might be reasonable probabilistic assumptions about the LOG of Y, instead of Y itself. While we cannot use the logNormal distribution in a GLM itself, there is a very similar distribution that does fall into the GLM family: called the ***Gamma distribution.***

Both the Gamma distribution and the logNormal distribution have the property of being right-skewed: they have long right tails. That is, they allow scatter above the mean to be of higher magnitude than scatter below the mean.

Thus, for the future (beyond Stats 20x), bear in mind that it is more 'aesthetic' to use a GLM with Gamma errors and a log link, rather than log-transforming Y as we did in Chapter 6. Both methods will give very similar answers. The Gamma GLM approach has the advantage that we can draw inference directly on the mean of Y, rather than on its median as we are forced to do in the log-transformation framework of Chapter 6.

```
## Generate some data that genuinely fit the logNormal model:
mydat <- data.frame(x=runif(100, 0, 3))
mydat$y <- exp(-2 + 0.4*mydat$x + rnorm(100, 0, 0.3))
## Fit model using the log-transformation method of Chapter 6:
lm(log(y)~x, mydat)
## Fit again using GLM method with Gamma family: you should get very similar estimates.
glm(y~x, family=Gamma(link="log"), mydat)
```

Stats 20x Handout 6: Poisson GLMs

Generalised Linear Models with the Poisson distribution

We have seen that a Poisson GLM has three components:

1. The ***error model or response distribution:*** $Y_i \sim \text{Poisson}(\mu_i)$.
2. The ***link function:*** $g(\mu_i) = \log(\mu_i)$.
3. The ***linear predictor:*** e.g. $\log(\mu_i) = \alpha + \beta x_i$.

Here, Y is a discrete, numeric variable (***a count of something***). We will use a numeric predictor X for convenience, but it could alternatively be a categorical variable, or there could be several predictors as in previous models that we have seen.

The log link function ensures that we always obtain a Poisson mean μ that is ***greater than 0.*** The log link is not the only function that ensures $\mu > 0$, via its exponential back-transformation, but it is by far the most common choice for Poisson models. This is because it is often reasonable to assume that predictors have a multiplicative effect on the response: $\mu_i = \exp(\alpha + \beta x_i) = e^{\alpha}\, e^{\beta x_i}$.

The `glm` function in R offers three link functions for Poisson models:

- `link="log"`, the default link, favoured for both its practical usefulness and its particularly nice theoretical properties.
- `link="identity"`, meaning no transformation at all. This is fine if the data Y are composed entirely of large counts so we can be sure that the estimated means will be well away from zero, but might generate errors otherwise.
- `link="sqrt"`, which uses the back-transformation $\mu = (\alpha + \beta x)^2$ to ensure positivity. This link is rarely used for modelling.

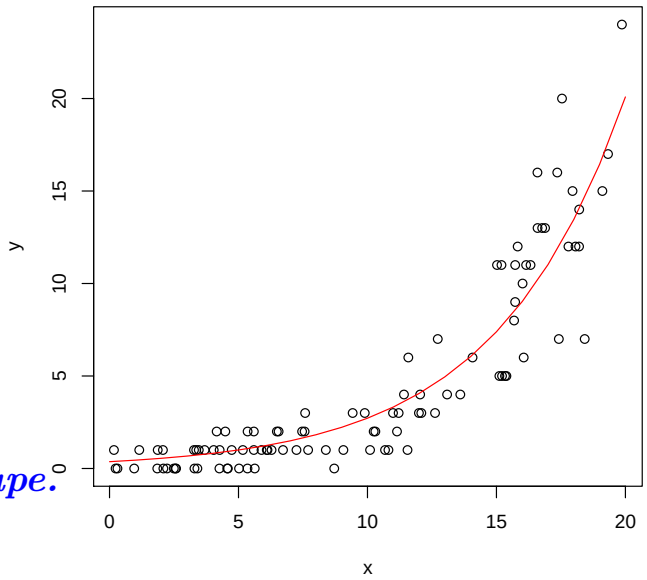
In this handout, we look at the practicalities of fitting a Poisson GLM, focusing on differences from the Normal models we have seen up to now. We will ***only*** use log link functions for Poisson data from now on.

Generating Poisson GLM data

```
## Generate data according to a Poisson GLM with log-link and alpha=(-1), beta=0.2:
mydat <- data.frame(x=runif(100, 0, 20))
mydat$linpred <- (-1) + 0.2*mydat$x
mydat$mu <- exp(mydat$linpred)
mydat$y <- rpois(100, mydat$mu)
## Plot the data and overplot the true mean, mu = exp(-1 + 0.2*x):
plot(y~x, mydat)
lines((0:20), exp(-1+0.2*(0:20)), col="red")
```

Note that we have generated the dataset so that it exactly fits the Poisson model with log-link and linear predictor $\ell_i = -1 + 0.2x_i$.

The plot of y against x is shown here. Note the characteristics:
Y takes integer values only;
scatter increases as the fitted values increase;
the mean line has a curved shape.



Fitting the model

The model-fitting command is very easy: just use `glm` instead of `lm`, and add the argument `family=poisson(link="log")`. In fact, just typing `family=poisson` is enough, but you might find it helpful to use `family=poisson(link="log")` because it includes an explicit reminder that we are working on the log-transformed scale.

```
myfit <- glm(y ~ x, family = poisson(link="log"), data = mydat)
```

Or just: `myfit <- glm(y ~ x, family = poisson, data = mydat)`

Now look at the `summary(myfit)` output. Much of it looks familiar, but there are some differences. There are several mentions of *deviance:* see below.

```
Call:
glm(formula = y ~ x, family = poisson(link = "log"), data = mydat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3908  -0.9024  -0.0726   0.5496   1.9013

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.31856    0.19059  -6.918 4.57e-12 ***
x            0.21986    0.01203  18.270  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 573.274  on 99  degrees of freedom
Residual deviance:  79.106  on 98  degrees of freedom
AIC: 338.56
```

Model checks 1: The residual plot

We are familiar with the three model-checking commands: `plot(myfit, which=1)`, `normcheck(myfit)`, and `cooks20x(myfit)`. Unfortunately, all of these checks are specific to the Normal error model, and must be either modified or discarded for GLMs.

We can still use the first command `plot(myfit, which=1)`, and interpret the output much the same way as we have done previously. However, the plot itself is a little different and needs some explanation. We do not have analogues of the other two commands for non-Normal models.

For GLMs, the only one of these 3 model-checks we use is plot(myfit, which=1). The plot is a little different from what we are used to.

What is a residual?

For Normal models, estimation is conducted by the ***least-squares criterion,*** which we have said is a special case of ***maximum likelihood estimation:***

$$\widehat{\alpha} \text{ and } \widehat{\beta} \text{ minimize } \sum_{i=1}^{n}(y_i - \widehat{\mu}_i)^2 : \text{ the sum of the squared residuals.}$$

In fact, what we are doing here via least-squares is *minimizing* the *negative* of the *log*-likelihood. The log of the likelihood will always be a sum of terms: one term for each observation. Maximizing the likelihood is always the same as maximizing the log-likelihood (because $\log(\mathcal{L})$ is an increasing function of $\mathcal{L}$), which in turn is the same as *minimizing* the *negative* of the log-likelihood. These details are not important. What is important is that the 'thing we want to minimize' to estimate α and β is composed of a *sum of terms,* where *each term corresponds to one observation,* and ***the larger the term is for an observation, the worse is the fit for that observation.***

In other words, for Normal models, we estimate α and β by minimizing

$$\sum_{i=1}^{n}(y_i - \widehat{\mu}_i)^2 = \sum_{i=1}^{n}(\text{residual}_i)^2$$

and if observation i has a *large* term, in other words if $(y_i - \widehat{\mu}_i)^2 = (\text{residual}_i)^2$ is *large,* then the fit for observation i is not very good.

For non-Normal models, the minus-log-likelihood that we minimize is still a sum of terms, one for each observation. And it is still true that ***the larger the term, the worse the fit for observation*** i***.*** All that has changed is that the terms no longer have the form $(y_i - \widehat{\mu}_i)^2$. So why don't we *define* residuals for this model such that they *do* match the term in the log-likelihood?

Up to a couple of details, that is exactly what we do. We ***define*** 'residuals' for a Poisson GLM such that they play the same role in the Poisson log-likelihood as the term $(y_i - \widehat{\mu}_i)$ plays in the Normal log-likelihood. These residuals are called ***deviance residuals,*** and they play the same role as the familiar residuals in a Normal model. We can think of the deviance residuals as measuring the ***deviation*** between each observation and the fitted model. ***A large deviance residual implies a poor fit or large scatter for that observation.***

For the record, the formula for deviance residuals in a Poisson GLM is below, but you are unlikely to see or use this formula in practical modelling.

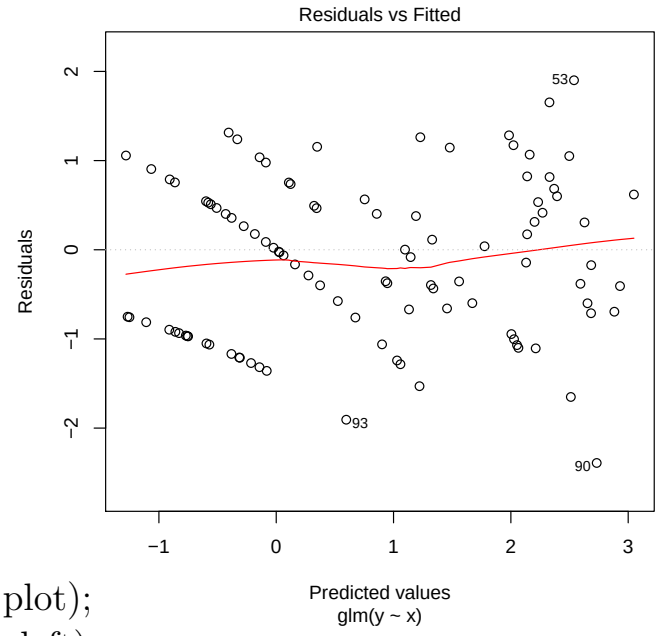
$$(\text{deviance residual})_i = \text{sign}(y_i - \widehat{\mu}_i)\sqrt{2\left(y_i \log \frac{y_i}{\widehat{\mu}_i} - (y_i - \widehat{\mu}_i)\right)}.$$

The other difference in the plot obtained from `plot(myfit, which=1)` is that the horizontal axis displays ***the fitted linear predictor,*** $\widehat{\alpha} + \widehat{\beta}x_i$***: not the fitted value,*** $\widehat{\mu}_i = \exp(\widehat{\alpha} + \widehat{\beta}x_i)$***.***

This stands to reason, because the exponential back-transformation would greatly distort the plot if we plotted the fitted values themselves.

Beyond this, we are looking for the same features as usual in the residual plot for a GLM: ideally the residual plot should be a ***patternless band of scatter of roughly constant width, centred on zero.***

Here is the residual plot from `plot(myfit, which=1)`.
The horizontal axis shows the ***fitted linear predictor,*** $\widehat{\alpha} + \widehat{\beta}x_i$.
The vertical axis shows the deviance residuals, as above.



This plot is satisfactory.
The striking lines of residuals are nothing to worry about. They arise from multiple observations of $y = 0$ for various x values (bottom left of plot); multiple observations of $y = 1$ (next left); and so on.

You can reproduce this plot by: `plot(predict(myfit), residuals(myfit))`.
The command `predict` gives the linear predictor by default, not the fitted values.

Model checks 2: Residual deviance

Because we have a Poisson GLM, the familiar `normcheck` diagnostic is no longer appropriate. Instead, we need some sort of test that ***the Poisson model is satisfactory.*** Such a test is often called a ***goodness-of-fit test.***

This test brings us back to the idea of ***deviance*** again. Recall the bottom part of the output from `summary(myfit)` above:

```
(Dispersion parameter for poisson family taken to be 1)

Null deviance: 573.274  on 99  degrees of freedom
Residual deviance:  79.106  on 98  degrees of freedom
AIC: 338.56
```

The first line, 'Dispersion parameter for poisson family taken to be 1', is a comment about the Poisson assumption that ***variance equals the mean.*** The 'dispersion parameter' is the ratio of variance to mean, and it should equal 1 if the data are genuinely Poisson. However, we often relax this assumption and consider a 'Poisson-like' distribution for which the variance is proportional to the mean, but not necessarily equal to it. Such a model is called a ***quasi-Poisson model*** and we will describe it below.

The test for whether we need to use a quasi-Poisson model instead of a Poisson model uses the penultimate line:

`Residual deviance:  79.106  on 98  degrees of freedom`.

Theory tells us that the residual deviance can be usefully approximated by a Chi-squared distribution, in which case the two numbers printed here should be close to each other. If the first number greatly exceeds the second, our data are said to be ***overdispersed,*** and we should use the quasi-Poisson model instead. We test whether the dispersion is too high for a Poisson model using a p-value from the Chi-squared distribution:

```
> 1 - pchisq(79.106, 98)
[1] 0.9191137
```

In this case, the p-value is ***large, so we have no concern about the Poisson model.***

If this p-value is small, e.g. less than 0.05, it indicates that the assumption of variance being equal to the mean does not hold for this dataset, and ***we should refit the model using the quasi-Poisson distribution, which allows the variance to be larger than the mean.***

Describing the fitted model: confidence intervals for the parameters

Confidence intervals for the estimated parameters α and β can be found using the R function `confint`, as usual. Here is the output for the GLM fit on our simulated data, which had true values $\alpha = (-1)$ and $\beta = 0.2$.

```
> myfit
Call:  glm(formula = y ~ x, family = poisson(link = "log"), data = mydat)
Coefficients:
(Intercept)            x
    -1.3186       0.2199

> confint(myfit)
Waiting for profiling to be done...
                 2.5 %     97.5 %
(Intercept) -1.7042735 -0.9566556
x            0.1968015  0.2440037
```

Both of the confidence intervals contain the respective true values in this example. To describe what the fitted model means in intuitive terms, we need to ***back-transform from the log scale to the response scale using exp(CI).***

This is the same as we did in Chapter 6 for log-transformed Y, except that we can now draw conclusions about the ***mean*** of Y, rather than the ***median.*** (Not having to use this median-fudge is a good reason for using GLMs instead of log-transformation.) We need to remember that when we back-transform through exponentiation, we should report results as ***percentage growth or percentage change.***

Here, our 95% confidence interval for β is ***0.197 to 0.244.***

Back-transforming to the response scale: $\exp(0.197) = 1.22$ ***and*** $\exp(0.244) = 1.28$***.***

Thus every unit increase in x is associated with an increase in the MEAN of Y ***to*** between 122% and 128% of its current value. Expressed as a percentage change, the mean of Y increases ***BY*** between 22% and 28% of its current value.

So our wording might be: ***We estimate that the mean of*** Y ***increases by between 22% and 28% for every every unit increase in*** x***.***

If we need to use multiple units of x, e.g. kx, then multiply the confidence interval by k before exponentiating: ***e.g. for*** $k = 5$***,***
100*(exp(5*confint(myfit)[2,]) - 1) gives percentage change of 168% to 239%.
We estimate that the mean of Y *increases by between 168% and 239% for every FIVE units increase in* x*.*

Fitted values for specified values of x, and their confidence intervals

We need to be careful using the function `predict` for GLMs, because ***by default, it returns the fitted LINEAR PREDICTOR (i.e. fitted values on the log scale), rather than the fitted values $\widehat{\mu}$ on the response scale.***

To extract fitted values, we have three choices:

- `fitted(myfit)` will deliver fitted values $\widehat{\mu}_i$ ONLY for the x_i values in the original data set. It has no other functionality.
- `predict(myfit, type="response", newdata=my.newdat)` is the core R way of generating fitted values ***on the response scale*** (i.e. back-transformed to the scale of Y). You can also enter new data and generate predictions of Y for new x values. Unfortunately, the `predict()` function no longer has the argument `interval="conf"` or `interval="prediction"` for GLMs.
- The `s20x` library has a convenient function `predictCount`, which does the same as `predict(myfit, type="response", newdata=my.newdat)`, ***and*** delivers confidence intervals. This is the best option, but it only works for Poisson or quasiPoisson GLMs.

```
> predictCount(myfit, newdata=data.frame(x=c(1,10,20)))
  Predicted  Conf.lower Conf.upper
1     0.333       0.235      0.473
2     2.411       2.062      2.819
3    21.728      18.829     25.074
```

ANOVA analyses

The command `anova(fit)` is suitable for GLMs in the same scenarios as it is used for ordinary linear models: for example, when we have a single categorical predictor with more than two levels, or when we have two or more categorical predictors or a mix of categorical and numeric variables and their interactions.

There is one key difference. As the name suggests, ANOVA is about ***analysis of variance.*** However, you'll notice that we have replaced many of our ideas about 'variance' for simple linear models with the notion of 'deviance' for GLMs. Therefore, ANOVA is now interpreted as an ***analysis of deviance*** rather than an analysis of variance.

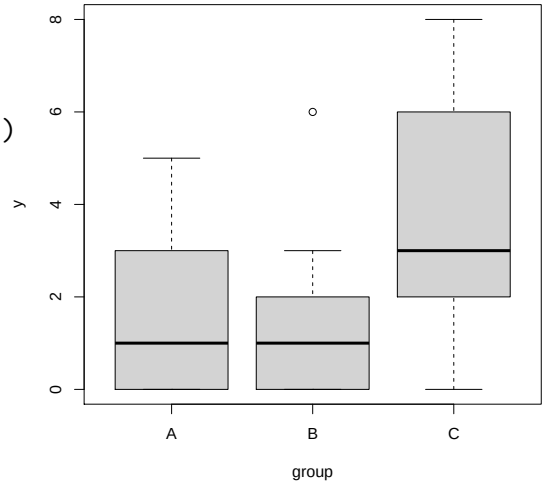
The statistical theory for deviance is less well-developed than that for ANOVA in Normal models. However, it has been shown that the Chi-squared distribution provides a good reference distribution for whether or not a variable contributes sufficient 'reduction in deviance' to be retained in the model.

For practical purposes, all you need to remember is:

- ***When conducting ANOVA with a GLM model, we need to specify anova(fit, test="Chisq").***

Try it out by creating a one-way ANOVA style analysis, but with log-link and Poisson errors. Instead of `anova(fit)`, we use `anova(fit, test="Chisq")`, and read results off an ***Analysis of Deviance table*** instead of an ***Analysis of Variance table***.

```
mydat <- data.frame(group=c(rep("A", 10), rep("B", 10), rep("C", 10)))
## Create three intercepts on the log-scale, with small but noticeable differences:
mydat$lin.pred <- c(rep(0.5, 10), rep(0.8, 10), rep(1, 10))
## Create the mean on the response scale for each group:
mydat$mu <- exp(mydat$lin.pred)
## Generate response, mydat$y, using Y~Poisson(mu):
mydat$y <- rpois(30, mydat$mu)
## Plot:
plot(y~group, mydat)

## Fit the model:
myfit <- glm(y~group, family=poisson, mydat)

## ANOVA using test="Chisq":
anova(myfit, test="Chisq")

Analysis of Deviance Table
Model: poisson, link: log
Response: y
Terms added sequentially (first to last)
```




```
      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                     29     68.513
group  2   13.906        27     54.607 0.0009558 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The conclusion is that the `group` factor is significant, so the variable `group` is contributing sufficient reduction in deviance to be retained in the model.

Quasi-Poisson models

We said earlier that we need to check Poisson GLM fits for the suitability of the Poisson model, in particular the requirement that ***variance = mean.*** If this assumption appears untenable, we should use a ***quasi-Poisson model***.

The quasi-Poisson fit relaxes the precise assumption that variance = mean, and replaces it with a looser assumption that ***variance is a constant multiple of the mean,*** where this multiple (called the *dispersion parameter*) will be estimated along with the other model coefficients.

When we enter the realm of quasi-GLMs, we are sacrificing the strong statistical theoretical foundation of proper GLMs in the interests of pragmatism. The quasi-Poisson fit is a bit of a fudge, and does not have the strong theoretical support that other models have. However, so-called *overdispersed* count data are very common in practice, so the quasi-Poisson GLM is very commonly used.

There are only three things you need to know about quasi-Poisson GLMs:

- Fit the quasi-Poisson model using `family = quasipoisson` instead of `family = poisson`. Everything else about the model formula stays the same.
- When you use `summary(fit)`, the quasipoisson family will typically give larger standard errors; similarly, using `confint` or `predictCount` will typically give wider confidence intervals than the standard Poisson fit.
- When using quasi-Poisson models (or any quasi-GLM), there is no longer the same theoretical justification for using `anova(fit, test="Chisq")`. Instead, ***for any quasi-GLM fit,*** change to `anova(fit, test="F")`.

Let's look at an example. We will generate genuinely Poisson data, as we did before, but pick a data-set that just happens to show a bit of overdispersion. Then we will look at the difference due to using the quasi-Poisson model.

```
## Generate data according to a Poisson GLM with log-link and alpha=(-1), beta=0.2:
mydat <- data.frame(x=runif(100, 0, 20))
mydat$linpred <- (-1) + 0.2*mydat$x
mydat$mu <- exp(mydat$linpred)
mydat$y <- rpois(100, mydat$mu)

## Fit standard Poisson GLM:
fit1 <- glm(y~x, family=poisson, mydat)
summary(fit1)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.0653     0.1723  -6.184 6.24e-10 ***
x             0.2054     0.0114  18.019  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 531.48  on 99  degrees of freedom
Residual deviance: 119.39  on 98  degrees of freedom
```

Just by chance, this data-set has quite a large residual deviance (119.39) relative to the residual df (98). (Remember that 5% of genuine Poisson data-sets will give a residual deviance in the top 5%, and will be falsely declared as significant overdispersion.)

Check out the p-value for overdispersion from this data-set:

```
1 - pchisq(119.39, 98)
[1] 0.06999373
```

The p-value is not quite significant, so we do ***not*** have to use a quasi-Poisson fit in this case. Let's plough on and use the quasi-Poisson fit anyway, to see what changes:

```
fitQ <- glm(y~x, family=quasipoisson, mydat)
summary(fitQ)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.0653     0.1829  -5.825 7.26e-08 ***
x             0.2054     0.0121  16.972  < 2e-16 ***

(Dispersion parameter for quasipoisson family taken to be 1.127065)

Null deviance: 531.48  on 99  degrees of freedom
Residual deviance: 119.39  on 98  degrees of freedom
```

All that has happened is that ***the standard errors for the quasi-Poisson fit are larger, and the dispersion parameter is now specified to be greater than 1.***

We just have to remember to use `anova(fit, test="F")` where appropriate.

Summary of modelling with Poisson GLMs

1. Fit using `myfit <- glm(y ~ x, family=poisson(link="log"), data=mydat)`.
2. Use `plot(myfit, which=1)` to check the residuals. These are now ***deviance residuals.*** The plot is interpreted as usual.
3. Use `summary(myfit)` to read off the residual deviance (R) and the residual df (D). Check that the Poisson model is appropriate using `1 - pchisq(R, D)`. If this is non-significant, continue with the Poisson fit. If it is significant, refit the model using `glm(y ~ x, family=quasipoisson(link="log"), data=mydat)`.
4. Extract confidence intervals for parameters using `confint(myfit)`. Back-transform using exponentiation, and quote results as percentage growth or percentage change for the ***mean*** of Y (not the median).
5. Extract confidence intervals for fitted values using `predictCount(myfit, newdata)`, or `predict(myfit, type="response", newdata)`. (Do ***not*** exponentiate these!)
6. Use the `anova` command in the same situations as usual. For regular Poisson GLMs, use `anova(myfit, test="Chisq")`. For quasi-Poisson GLMs, use `anova(myfit, test="F")`.

Stats 20x Handout 7: Binomial GLMs

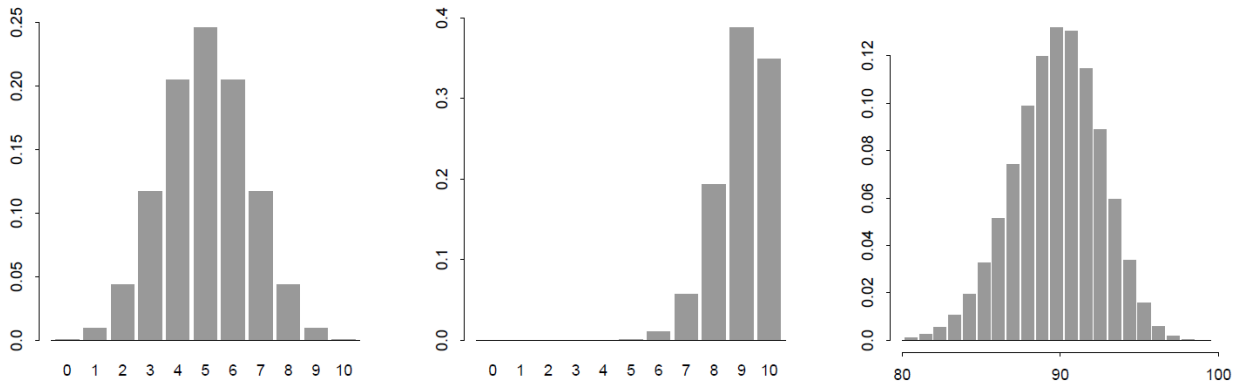
The Binomial distribution

The Binomial distribution is another discrete (counting) distribution. The Binomial(n, p) distribution counts the ***number of successes in*** n ***independent trials, where all trials have the same probability of success*** p***.***

The Binomial distribution therefore has both a lower limit ***and*** an upper limit. ***If*** $Y \sim$ ***Binomial***(n, p)***, then*** Y ***can only take values*** $0, 1, 2, \ldots, n$.

The mean of the Binomial(n, p) distribution is $\mu = np$. Here are the shapes of some Binomial(n, p) distributions:

$n = 10,\ p = 0.5$ $\qquad$ $n = 10,\ p = 0.9$ $\qquad$ $n = 100,\ p = 0.9$

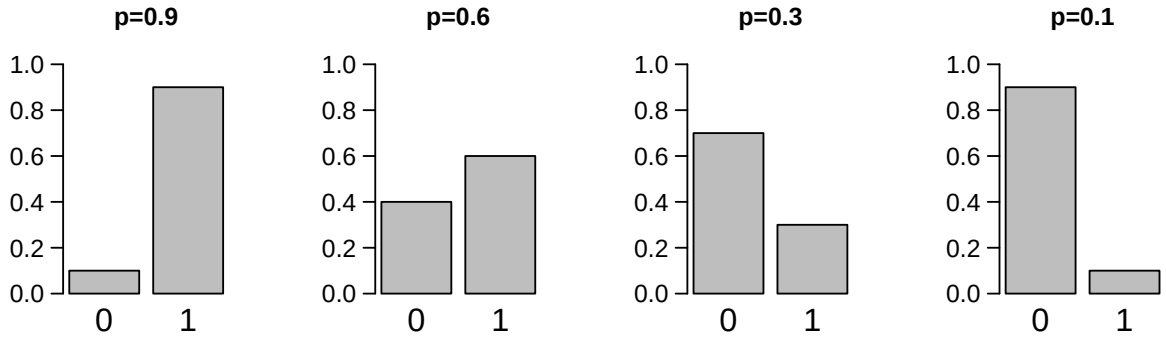


A special case that is particularly useful for modelling is the case where $n = 1$**.**

This corresponds to ***binary data: if*** $Y \sim$ ***Binomial***$(1, p)$***, then*** Y ***can only take values 0 and 1.***

The probability that $Y = 1$ is p, and the probability that $Y = 0$ is $1 - p$**.**

The mean of Y is $\mu = np = p$.



For example, in a medical study, we might have $Y_i = 1$ if patient i has diabetes, and $Y_i = 0$ if they do not. The probability of a patient having diabetes can be modelled in terms of various risk factors. ***Our interest is in determining predictor variables*** $X_1, X_2, \ldots$ ***that govern the risk*** p ***of having diabetes.***

Logit link function

Because the Binomial parameter p (or equivalently, μ), is restricted between 0 and 1, for GLMs we need a link function that maps the interval $[0, 1]$ to the whole real line. We have already discussed a suitable function in Handout 5:

$$p_i = \frac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)} \qquad \Longleftrightarrow \qquad \log\left(\frac{p_i}{1 - p_i}\right) = \alpha + \beta x_i$$

Logistic function ***Logit function***

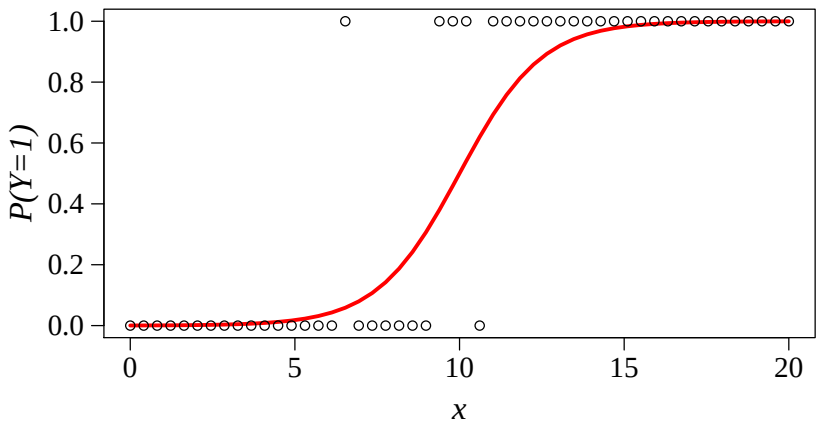
The name of the ***link function*** is the transformation from p to the linear predictor: ***logit link:*** $g(\mu) = g(p) = \log\left(\frac{p}{1-p}\right) = \ell$.

The inverse of this function is called the ***logistic function,*** and is the back-transformation from the linear predictor back to the Binomial probability p. It is easily shown that the inverse-logit function has the logistic form,
$p = \exp(\ell)/\{1 + \exp(\ell)\}$.

What do binary data look like when $p_i = \dfrac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)}$?

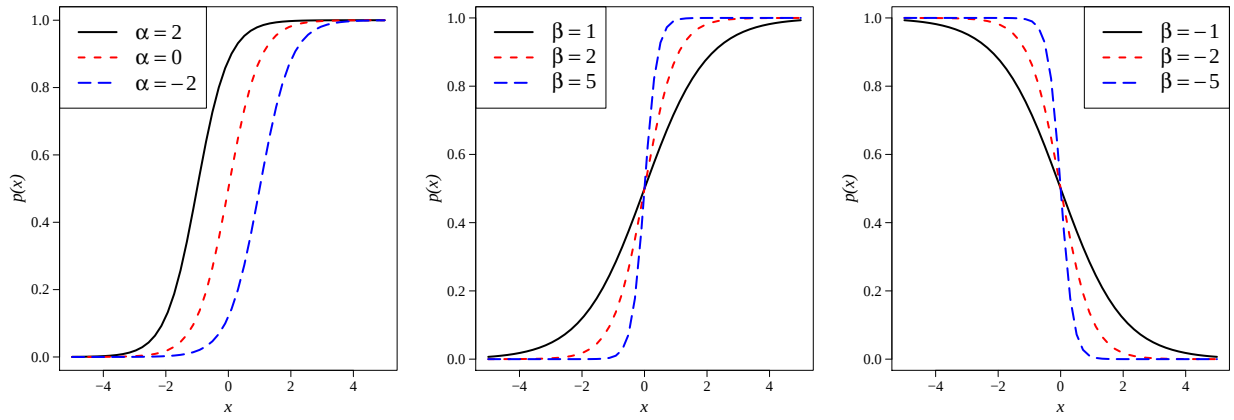
Each Y_i is 0 or 1. When p_i is close to 0, most Y_i will be 0.
When p_i is close to 1, most Y_i will be 1.
For intermediate p_i, Y_i will be a mix of 0s and 1s, in the proportions $1 - p_i$ to p_i.

The graph of p as a function of a numeric variable x is below, using $\alpha = -8$ and $\beta = 0.8$. You can see that, as x ranges from 0 to 20 in this case, the binary outcomes Y_i change from being mostly 0 to mostly 1. In the mid-range of x, there is a mixture of 0 and 1 outcomes.



Different logistic shapes are generated by different values of α and β. Various shapes are shown overleaf.

Various α with $\beta = 2$ *$\alpha = 0$ with various $\beta > 0$* *$\alpha = 0$ with various $\beta < 0$*



Odds and Log-Odds

The logit function $g(p) = \log\left(\frac{p}{1-p}\right)$ can be interpreted as ***log-odds.***

What does this mean?

- The probability that an event of interest ***happens*** is p.
(Here, our event of interest is $Y = 1$.)
- The probability that the event ***does not happen*** is $1 - p$. (Here, $Y = 0$.)
- The event is $\left(\frac{p}{1-p}\right)$ times ***more likely to happen than not to happen.***
- The ratio $\left(\frac{p}{1-p}\right)$ is called ***the ODDS of the event.***
- So $\log\left(\frac{p}{1-p}\right)$ is the ***log-odds.***

You have probably heard people describing chance events in terms of odds: for example saying things like, 'the team is odds-on to win', or 'the team is 10-to-1 on to win'. Language like '10-to-1 on to win' corresponds to ***an odds of*** $10/1 = 10$***.***

In general, the higher the odds, ***the more likely the event is to happen.***

Let's get some practice at thinking in terms of odds:

- Odds of 1: ***as likely to occur as not to occur*** ($p = 0.5$)***.***
- Odds of 10: ***10-to-1, or 10 times more likely to occur than not to occur.*** $p = 10/11$***.***
- Odds of 0.2: ***0.2-to-1, so*** $p = 0.2/1.2 = 0.167$***. Five times more likely NOT to occur than to occur.***
- Lengthening the odds: ***making it more likely that the event will occur.***

Back-transformation to the response scale

As we saw overleaf, when we use the logit link function, we are modelling the ***log-odds*** that $Y = 1$.

When we back-transform the linear predictor $\ell = \alpha + \beta x$, we can either:

1. Use $\dfrac{\exp(\ell)}{1+\exp(\ell)} = p$ to make statements about the probability p that $Y = 1$; or
2. Use $\exp(\ell) = \dfrac{p}{1-p}$ to make statements about the ***odds*** that $Y = 1$.

For other models, we have used back-transformation to make intuitive statements about what the fitted model means. For example, for a Poisson GLM with log link, we make comments like, *'The mean of Y increases by $100\times(e^{\beta}-1)$ percent for every unit increase in x.'*

These expressions are gained from the following calculation: if we start with $x_{\text{old}} = x$, and increase it to $x_{\text{new}} = x + 1$, then the percentage change in $\mathbb{E}Y$ is:

$$\%\text{ change} = \frac{(\mathbb{E}Y \text{ for } x_{\text{new}}) - (\mathbb{E}Y \text{ for } x_{\text{old}})}{(\mathbb{E}Y \text{ for } x_{\text{old}})} \times 100\,.$$

For a log-link function, this fraction simplifies nicely, and we end up with a percentage change of $100\times(e^{\beta}-1)$ for each unit increase in x, or $100\times(e^{10\beta}-1)$ for each 10-unit increase in x, and so forth. In other words, for a model with log-link, the percentage change for a unit-increase in x is ***the same regardless of which x we start at.***

Unfortunately, when we use the logistic back-transformation $p = \dfrac{\exp(\ell)}{1+\exp(\ell)}$, the expression for percentage change does not simplify to something that is the same for all x. For this reason we have to resort to describing the ***percentage change in odds*** when we summarize the model, instead of the percentage change in the fitted value p. Thus we use the back-transformation in point 2 above: $\exp(\ell) = \dfrac{p}{1-p} = (\text{odds that } Y = 1)$.

Percentage change in odds:

We will do the most general case, where we consider a change in x by k ***units.***

The linear predictors are:

$$\begin{aligned} \ell_{\text{new}} = \alpha + \beta x_{\text{new}} &= \alpha + \beta(x+k), \\ \ell_{\text{old}} = \alpha + \beta x_{\text{old}} &= \alpha + \beta x\,. \end{aligned}$$

Then:

$$\begin{aligned}
\%\text{ change in odds} &= \frac{(\text{odds for } x_{\text{new}}) - (\text{odds for } x_{\text{old}})}{(\text{odds for } x_{\text{old}})} \times 100 \\
&= \frac{\exp(\ell_{\text{new}}) - \exp(\ell_{\text{old}})}{\exp(\ell_{\text{old}})} \times 100 \\
&= \frac{\exp(\alpha + \beta x + \beta k) - \exp(\alpha + \beta x)}{\exp(\alpha + \beta x)} \times 100 \\
&= \frac{\exp(\alpha)\exp(\beta x)\exp(\beta k) - \exp(\alpha)\exp(\beta x)}{\exp(\alpha)\exp(\beta x)} \times 100 \\
&= \{\exp(\beta k) - 1\} \times 100\,,
\end{aligned}$$

exactly the same as our usual expression for percentage change with a log-link function. The only thing we have to remember differently is that ***now we report percentage change in the ODDS of*** $Y = 1$***, not in the MEAN of*** Y***.***

In practice, we will use a ***confidence interval*** for β to express this, rather than the point estimate. If our confidence interval is $\widehat{\beta}_{\text{low}}$ to $\widehat{\beta}_{\text{high}}$, then suitable wording for the Executive Summary is:

- *We estimate that the odds of <event Y=1 in plain language> will increase by between* $100 \times \left\{\exp\left(\widehat{\beta}_{low}\right) - 1\right\}\%$ *and* $100 \times \left\{\exp\left(\widehat{\beta}_{high}\right) - 1\right\}\%$ *for every unit increase in* x*.*
- *We estimate that the odds of <event Y=1> will increase by between* $100 \times \left\{\exp\left(k\widehat{\beta}_{low}\right) - 1\right\}\%$ *and* $100 \times \left\{\exp\left(k\widehat{\beta}_{high}\right) - 1\right\}\%$ *for every* k *units increase in* x*.*
- For example: ***We estimate that the odds of a patient getting diabetes increase by between*** 10% ***and*** 20% ***for every 10kg increase in the patient's body fat.***

  What does it mean for ***odds to increase by 10%***?

  - If the odds were 10 at the previous weight, (i.e. 10-to-1 in favour of getting diabetes), they are now $1.1 \times 10 = 11$ ***i.e. 11-to-1 in favour of getting diabetes.***
  - If the odds were 5 at the previous weight, they are now $1.1 \times 5 = 5.5$.
  - If the odds were 1 at the previous weight, they are 1.1 at the new weight.
  - If the odds were 0.2 at the previous weight, they are 0.22 at the new weight.

Binomial GLM for binary data: Logistic Regression

For binary data, $Y \sim \text{Binomial}(1, p)$. Also, $\mathbb{E}(Y) = \mu = p$, so μ and p are interchangeable. The GLM has three components:

1. The ***error model or response distribution:*** $Y_i \sim \text{Binomial}(1, p_i)$, or equivalently, $Y_i \sim \text{Binomial}(1, \mu_i)$.
2. The ***link function:*** $g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right) = \log\left(\frac{p_i}{1-p_i}\right)$.
3. The ***linear predictor:*** $\log\left(\frac{p_i}{1-p_i}\right) = \alpha + \beta x_i$, or another function of predictors as desired.

Fitting the model

Either:

```
myfit <- glm(y ~ x, family = binomial(link="logit"), data = mydat)
```

Or just: `myfit <- glm(y ~ x, family = binomial, data = mydat)`

To describe the model in Methods & Assumptions: *'Our model is* $\log(odds_i) = \beta_0 + \beta_1 x_i$*, where* $odds_i$ *is the odds of a success for subject i.'*

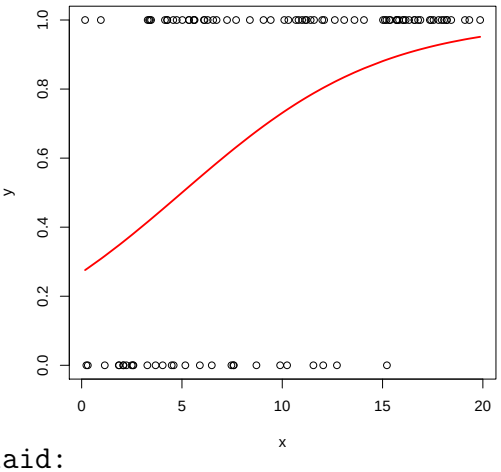
Describing the fitted model: confidence intervals for the parameters

Use the R function `confint`, as usual.

Back-transform the CI for β using `exp(confint(fit)[2,])`, and interpret in terms of a ***percentage change in odds*** as described in the previous section. See the example below.

Example with simulated binary data

```
## Generate data for a Binomial GLM with
## logit-link and alpha=(-1), beta=0.2:
mydat <- data.frame(x=sort(runif(100, 0, 20)))
mydat$linpred <- (-1) + 0.2*mydat$x

## Logistic function for P(Y=1):
mydat$p <- with(mydat,
             exp(linpred) / (1+exp(linpred)))

## Generate response Y: 100 random Binomial
##  numbers with n=1 and p depending on x:
mydat$y <- with(mydat, rbinom(100, 1, p))
```




```
## Plot data y, with the true function p overlaid:
plot(y~x, mydat)
lines(p~x, mydat, col=2, lwd=2)
```

```
## Fit the model:
myfit <- glm(y~x, family=binomial(link="logit"), data=mydat)

## Find confidence intervals:
confint(myfit)
Waiting for profiling to be done...
                 2.5 %     97.5 %
(Intercept) -1.9225462 -0.1143848
x            0.1300101  0.3499907

## Back-transform for percentage change in odds:
100*( exp(confint(myfit)[2,]) - 1 )
   2.5 %   97.5 %
13.88399 41.90544
```

Both of the CIs contain the respective true values, $\alpha = -1$ and $\beta = 0.2$. Our Executive Summary conclusion is: ***We estimate that every 1-unit increase in x increases the odds of $Y = 1$ by between 14% and 42%.***

Fitted values for specified values of x, and their confidence intervals

As with Poisson GLMs, the function `predict` returns the fitted linear predictor by default: that is, the fitted values on the logit scale, rather than the fitted values p on the response scale. Unfortunately, for Binomial GLMs there is no convenient function in the `s20x` library corresponding to `predictCount` for Poisson GLMs. To extract fitted values, we only have two choices:

- `fitted(myfit)` will deliver fitted values p_i ONLY for the x_i values in the original data set. It has no other functionality.
- `predict(myfit, type="response", newdata=my.newdat)` is the core R way of generating fitted values ***on the response scale*** (i.e. the back-transformed probabilities, p_i). You can also enter new data and generate fitted probabilities for new x values.

Unfortunately, there is no built-in way of generating confidence intervals for ***fitted values***, so we have to do it ourselves.

- First generate the CI on the ***logit scale*** using estimate $\pm$ 1.96 $\times$ standard error.
- Then back-transform the CI to the response scale using $\exp(\ell)/(1+\exp(\ell))$. See the example code below.

```
## Create a new data-frame for which we want fitted values:
my.newdat <- data.frame(x=c(0, 10, 20))
## Use predict ON THE LOGIT SCALE to extract fitted linear predictor AND std error:
pred.logit <- predict(myfit, newdata=my.newdat, se.fit=T)

## pred.logit has components "fit" and "se.fit". Use these to calculate CIs
## on the LOGIT scale. Combine the results into the pred.logit dataframe:
pred.logit <- data.frame(within(pred.logit, {
       ci.low <- fit - 1.96 * se.fit
       ci.hi <- fit + 1.96 * se.fit
})[c("fit", "ci.low", "ci.hi")])

## Look at the resulting fitted values and CIs on the LOGIT scale:
pred.logit
         fit     ci.low       ci.hi
1 -0.9888959 -1.8851852 -0.09260657
2  1.3101992  0.7137902  1.90660823
3  3.6092943  2.1027942  5.11579437

## Back-transform all columns at once for fitted probabilities:
pred.p <- exp(pred.logit)/(1+exp(pred.logit))
## Attach the x-values to the beginning:
pred.p <- cbind(my.newdat, pred.p)

## Look at the outcome (with rounded values). These are now FITTED PROBABILITIES:
print(pred.p, digits=3)
   x   fit ci.low ci.hi
1  0 0.271  0.132 0.477
2 10 0.788  0.671 0.871
3 20 0.974  0.891 0.994
```

ANOVA analyses

As for Poisson GLMs, we can use the `anova(myfit)` command in the usual situations for a Binomial GLM. We only need to remember one thing:

- When conducting ANOVA with a GLM model, we need to specify `anova(fit, test="Chisq")`.

No Quasi-fit for Binary data

When the data are ***binary,*** there is no legitimate test of the residual deviance to decide if we should use a quasi-GLM in the same way as we did for the Poisson models. This is because the Chisquared test for residual deviance (`1 - pchisq(dev, df)` for Poisson models) relies upon asymptotic results that do not apply when we only have $n = 1$ trial for each observation.

Therefore we will ***not*** use quasi-Binomial fits ***for binary data.*** The Chisquared test is legitimate for grouped data (see next section) and quasi-Binomial fits are included there.

Grouped Data — three ways to fit logistic regressions

If the predictors are categorical, or numeric with many repeated observations at each value, we can reformat the data as ***grouped data.***
For example, a binary-format data frame like the one on the left could be ***aggregated*** to give the one on the right:

```
 x y
10 0
10 1
10 0
10 0
12 1
12 1
12 0
```

```
 x s n
10 1 4
12 2 3
```

This looks like the traditional notion of a Binomial distribution, where:
n is the ***number of trials at each x-value,***
and s is the ***number of successful trials.***

There are ***three ways to proceed*** with grouped data. Follow the code below.

```
## Generate data with 25 observations for each
## combination of Sex (M/F) and Age (Adult/Child):
mydat <- data.frame(Sex=c(rep("F", 50), rep("M", 50)),
                    Age=rep(c(rep("A", 25), rep("C", 25)), 2))
## For the linear predictor use the model Sex + Age:
mydat$p <- exp(-2+as.numeric(mydat$Age=="C")*1+as.numeric(mydat$Sex=="M")*3)/
           (1+exp(-2+as.numeric(mydat$Age=="C")*1+as.numeric(mydat$Sex=="M")*3))
## Response y in binary format:
mydat$y <- rbinom(100, 1, mydat$p)

## METHOD 1: FIT WITH BINARY (UNGROUPED) DATA AS BEFORE:
fit.bin <- glm(y ~ Sex + Age, family=binomial, data=mydat)
Coefficients:
(Intercept)         SexM         AgeC
     -3.366        3.472        2.249

## REFORMAT THE DATA INTO GROUPED DATA FOR METHODS 2 AND 3:
## Now aggregate the data into grouped format:
gpdat <- aggregate(y ~ Sex * Age, data=mydat, sum)
gpdat$trials <- aggregate(y ~ Sex * Age, data=mydat, length)$y
gpdat$fail <- with(gpdat, trials - y)
names(gpdat)[names(gpdat)=="y"] <- "success"
gpdat
 Sex Age success trials fail
   F   A       1     25   24
   M   A      13     25   12
   F   C       6     25   19
   M   C      23     25    2
```

```
## METHOD 2: FIT TO GROUPED DATA USING SUCCESS-FAIL RESPONSE.
## The response is entered in two columns using cbind(success, fail):
fit.gp <- glm(cbind(success, fail) ~ Sex+Age, family=binomial, data=gpdat)
Coefficients:
(Intercept)         SexM         AgeC
     -3.366        3.472        2.249

## METHOD 3: FIT TO GROUPED DATA USING PROPORTIONS AND WEIGHTS:
## The response is proportion of successes, and "weights" gives number of trials:
gpdat$proportion <- with(gpdat,  success/trials)
fit.prop <- glm(proportion ~ Sex+Age, family=binomial, data=gpdat, weights=trials)
(Intercept)         SexM         AgeC
     -3.366        3.472        2.249
```

We get exactly the same fit regardless of which of the three methods we use.

Quasibinomial fit for grouped data only

Unlike binary data, we ***can*** test for overdispersion with grouped data, using `1-pchisq(resid.deviance, resid.df)`. If this is significant, refit the model using `family=quasibinomial`.
Any ANOVA tests should then use `anova(fit, test="F")`.

Summary of modelling with Binomial GLMs: Logistic Regression

1. Fit with logit link using EITHER:
   Binary data: `glm(y ~ formula, family=binomial, data=mydat)`;
   Grouped data: `glm(cbind(success, fail) ~ formula,family=binomial,data)`
   OR: `glm(proportion ~ formula, family=binomial, data, weights=trials)`.
2. Use `plot(fit, which=1)` to check the residuals. These are now ***deviance residuals.*** The plot is interpreted as usual.
3. For ***grouped data only:*** use `summary(fit)` to read off the residual deviance (R) and the residual df (D). Check that the Binomial model is appropriate using `1 - pchisq(R, D)`. If this is non-significant, continue with the Binomial fit. If it is significant, refit the model using `glm(y ~ formula, family=quasibinomial)`.
4. Extract confidence intervals for parameters using `confint(fit)`. Back-transform using exponentiation, and quote results as percentage growth or percentage change for the ***odds*** of $Y = 1$ (not the mean or median of Y).
5. To obtain fitted ***probabilities,*** use `predict(fit, type="response", newdata)`. Create DIY confidence intervals for fitted probabilities as described on page 8.
6. Use the `anova` command as usual: `anova(fit, test="Chisq")` for regular binomial models; `anova(fit, test="F")` for quasibinomial models.
7. To describe the model in M&A, use *'logistic regression'* or *'binomial GLM'*. If quasibinomial, describe it as overdispersed. Formula: $\log(\text{odds}_i) = \alpha + \beta x_i$.

Stats 20x Handout 8: Tables of Counts

What is a table of counts?

Tables of counts should already be familiar from Stats 10x. Here are some examples familiar from Stats 10x, or soon-to-be familiar in Stats 330.

Question: does snoring affect tendency to have nightmares?

		Nightmares		
		Common	Rare	Total
Snorer?	Yes	11	82	93
	No	12	74	86
	Total	23	156	179

Question: does education affect level of religious practice?

		Religious practice			
		Fundamentalist	Moderate	Liberal	Total
	None	178	138	108	424
Qualification:	High School	570	648	442	1660
	Graduate	138	252	252	642
	Total	886	1038	802	2726

Question: has the childhood weight profile changed over time in NZ?

		Weight			
		Normal	Overweight	Obese	Total
Year:	1989	754	96	21	871
	2000	626	187	81	894
	Total	1380	283	102	1765

In each case, our question is the same: ***Are the rows multiples of each other?***

Clearly, the rows aren't identical, because they have different totals and because of random scatter. Our question is whether we can believe that the table of counts was drawn from a ***population in which the column probabilities are identical for each row.***

This is the same as asking whether there is an ***association*** between the rows and the columns. That is, we are asking whether ***the profile across columns is the same for each row.***

In the framework we have been developing in Stats 20x, you'll notice that the rows correspond to ***the levels of one categorical predictor variable***, and the columns correspond to ***the levels of one categorical response variable.***

This differs from what we have seen before in Stats 20x: up to now we have only seen ***numeric response variables, Y.*** Even if the numeric response was a discrete count, as for Poisson and Binomial GLMs, it was still formulated as a single numeric variable. The closest we have come to formulating the response as categorical was when we fitted the Binomial GLM using grouped data and the `cbind(success, fail)` formulation: e.g. from the end of Handout 7,

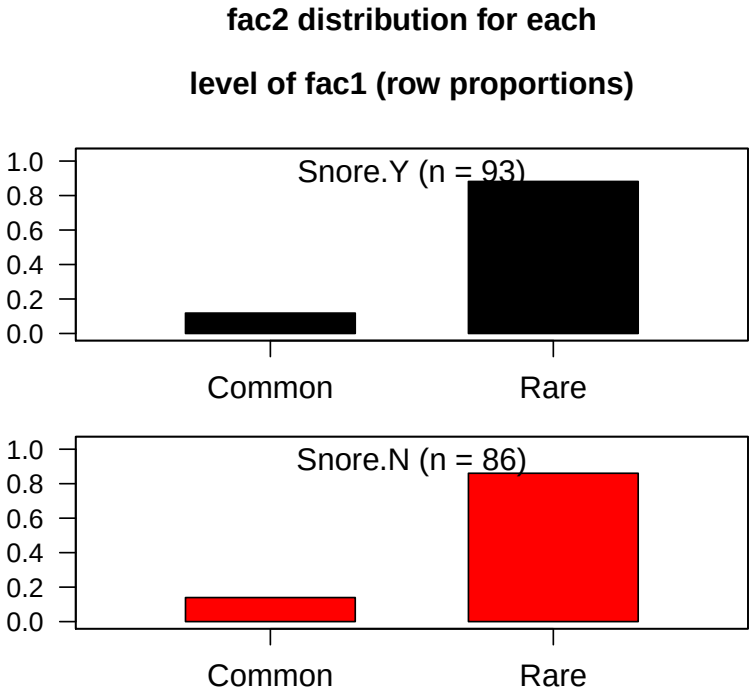
```
glm(cbind(success, fail) ~ Sex + Age, family=binomial, data=gpdat)
```

This handout covers the material from Chapters 16 to 19. We will see that there are several different ways of approaching the table of counts analysis, ***all of which give the same or very similar answers.*** The primary difference in the methods is their ***level of flexibility.*** Some methods are only suitable if the response has two levels; others can deal with multiple levels in the response, but only one categorical predictor; whereas the most flexible method (the Poisson GLM) can deal with any number of predictor variables and any number of response categories, but is also the least intuitive.

Visualizing data from tables of counts

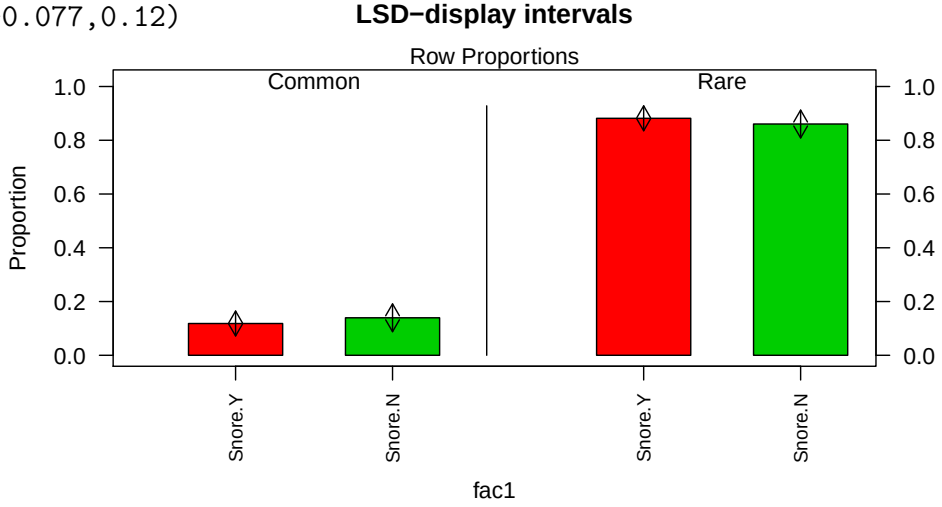
First we show a useful function for visualizing this sort of data. The `s20x` library function `rowdistr` lays out row or column proportions in the same configuration as they appear in the table. The name `rowdistr` indicates that ***we can compare the 'distribution' of the response, according to row.***

```
## Enter the data by hand, as a 2x2 matrix, entering one row at a time:
nightmare.tab <- matrix(c(11, 82, 12, 74), byrow=T, nrow=2)
## Give the matrix some names for convenience:
dimnames(nightmare.tab)<-list(c("Snore.Y", "Snore.N"), c("Common", "Rare"))
## Have a look at the matrix:
nightmare.tab
        Common Rare
Snore.Y     11   82
Snore.N     12   74

## Plot:
rowdistr(nightmare.tab)
Row Proportions
        Common Rare Totals  n
Snore.Y   0.12 0.88      1 93
Snore.N   0.14 0.86      1 86
```



It doesn't look as if there is much to see here: snoring doesn't seem to have an impact on nightmare frequency. We will need to do a formal test to be sure. We can also ask for comparisons between groups, using:

```
rowdistr(nightmare.tab, comp="between")

Row Proportions
        Common Rare Totals  n
Snore.Y   0.12 0.88      1 93
Snore.N   0.14 0.86      1 86

95% CIs for diffs between proportions with fac2 = Common
(rowname-colname)
        Snore.N
Snore.Y (-0.12,0.077)

95% CIs for diffs between proportions with fac2 = Rare
(rowname-colname)
        Snore.N
Snore.Y (-0.077,0.12)
```



The `rowdistr` command also supplies a confidence interval: here it is $(-0.120, 0.077)$. This is obtained using familiar Stats 10x methods for ***comparing proportions from two independent samples.*** Let p_1 be the proportion of ***snorers*** who commonly have nightmares. From the table data, $\widehat{p}_1 = \dfrac{11}{93} = 0.12.$

Let p_2 be the proportion of ***non-snorers*** who commonly have nightmares. Then $\widehat{p}_2 = \dfrac{12}{86} = 0.14.$

In Stats 10x, we saw how to test for the ***difference between proportions, $p_1 - p_2$, from two independent samples:***

$$\text{se}(\widehat{p}_1 - \widehat{p}_2) = \sqrt{\frac{\widehat{p}_1(1-\widehat{p}_1)}{n_1} + \frac{\widehat{p}_2(1-\widehat{p}_2)}{n_2}},$$

and the approximate 95% confidence interval is:

$$\begin{aligned}
\text{Estimate} \;\pm\; & 1.96 \times (\text{standard error}) \\
&= \widehat{p}_1 - \widehat{p}_2 \;\pm\; 1.96\,\text{se}(\widehat{p}_1 - \widehat{p}_2) \\
&= (0.12 - 0.14) \;\pm\; 1.96\sqrt{\frac{0.12(1-0.12)}{93} + \frac{0.14(1-0.14)}{86}} \\
&= (-0.120,\; 0.077)\,.
\end{aligned}$$

This is the same as the confidence interval shown overleaf by `rowdistr`. Here is the calculation in R:

```
p1hat <- 11/93
p2hat <- 12/86
n1 <- 93
n2 <- 86
se.p1.minus.p2 <- sqrt( p1hat*(1-p1hat)/n1 + p2hat*(1-p2hat)/n2 )
p1hat-p2hat + 1.96 * se.p1.minus.p2 * c(-1, 1)
[1] -0.11959764  0.07708701
```

We see that the 95% confidence interval for $p_1 - p_2$ ***contains the value 0.*** This corresponds to a test of the hypothesis $H_0 : p_1 - p_2 = 0$, or equivalently, $H_0 : p_1 = p_2$. Because our CI contains 0, there is no evidence of a difference between p_1 and p_2. That is, ***there is no evidence that the probability of common nightmares differs for snorers and non-snorers.***

We can use `rowdistr` to plot the row profiles and the column profiles for any $m \times n$ table. However, the method above of testing for the equality of two proportions suffers from a familiar problem: ***it is restricted to PAIRWISE tests.*** For more than a 2×2 table, we would need to ***inspect all pairs, AND control for multiple testing.***

The LSD (least significant difference) intervals plotted by `rowdistr` do control for multiple testing, so if we have a table larger than 2×2 they will differ from the simple calculation above. However, it would be preferable to find a ***global test*** for tables of counts. This is similar to the rationale for ANOVA models in Chapters 11 and 12, when we had a continuous numeric response Y, and one or more categorical predictors. There, we used the global test `anova` first, and only after that did we investigate pairwise differences using `multipleComp` or `summary2way`. We also saw that one-way and two-way ANOVA models are simply special cases of simple linear models: `lm(y~group)` or `lm(y~x1*x2)`.

Here, we will take similar steps for tables of counts. The familiar ***Chi-square test*** fulfils our requirement for a global test, but is only available for a single categorical predictor. We will then see how more complicated models can be built using either Binomial or Poisson GLMs.

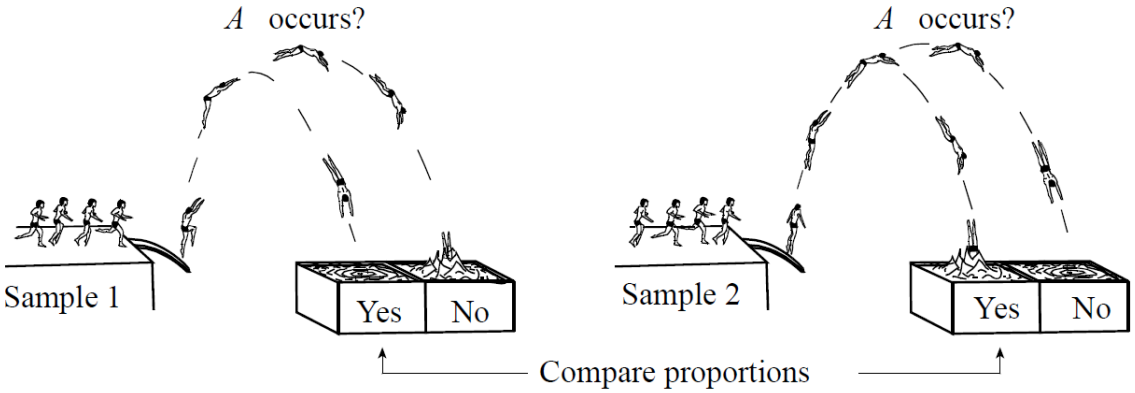
We will go through the various methods below. There are five methods in total:

- All methods are applicable to 2×2 tables of counts.
- Methods 1 and 2 are *only* applicable to 2×2 tables. These constitute the test above for ***comparing proportions from two independent samples,*** and a new method to ***test for a significant odds ratio.***
- Method 3, the ***Chi-square test,*** is applicable for an $m \times n$ table, but it only looks at the relationship between ***two*** categorical variables: in other words, one predictor variable (rows) and one response variable (columns).
- Method 4, the ***Binomial GLM,*** can deal with many predictors, but can only deal with a response with ***two*** levels. That is, the response variable (columns) needs to be in a binary format such as Yes and No, or Success and Failure.
- Method 5, the ***Poisson GLM,*** can deal with as many predictors and as many response levels as desired. However, it is a little less obvious than the other methods, so the flexibility comes at a small price.

We will start with the following table to illustrate the various methods. The numbers are made up to give a convenient illustration, but they are based on reports of a real survey looking at whether young men and women (aged 18-25) aspire to have children. We have two sexes (F/M) and one response question of interest: do you wish to have children some day? Our question is the standard tables-of-counts question: ***does the response (columns) differ by row?***

```
    Y  N
F  70 30
M 160 40
```

Method 1: Comparing proportions from two independent samples (Ch 16)



This method is a revision of the Stats 10x method for comparing proportions from two independent samples. The picture above is from the Stats 10x coursebook. The method was shown on page 4 and is repeated here for completeness.

The data, $\begin{matrix} \\ F \\ M \end{matrix} \begin{matrix} \begin{matrix} Y & N \end{matrix} \\ \begin{bmatrix} 70 & 30 \\ 160 & 40 \end{bmatrix} \end{matrix}$ are reduced to:

1. The estimated probability of 'Yes' for each row: $\widehat{p}_1 = 70/100$ and $\widehat{p}_2 = 160/200$;
2. The two sample sizes, $n_1 = 100$ and $n_2 = 200$.

The standard error of the difference is:

$$\text{se}(\widehat{p}_1 - \widehat{p}_2) = \sqrt{\frac{\widehat{p}_1(1-\widehat{p}_1)}{n_1} + \frac{\widehat{p}_2(1-\widehat{p}_2)}{n_2}},$$

and the approximate 95% confidence interval is:

$$\text{estimate} \pm 1.96 \times (\text{standard error}) \quad = \quad \widehat{p}_1 - \widehat{p}_2 \pm 1.96 \times \text{se}(\widehat{p}_1 - \widehat{p}_2)\,.$$

We conclude that there is evidence that the two samples have different proportions for $\mathbb{P}(\text{Yes})$ if ***the 95% confidence interval does not contain 0.***

Question tested: Is there a difference between $\mathbb{P}(\text{Yes})$ for males and females? This is the same as asking: ***does the response (columns) differ by row?*** Or: ***is there an association between the row variable and the column variable?***

Scope: Only suitable for one pairwise comparison, in other words for a 2×2 ***table.***

For larger tables, we would have to inspect multiple pairwise comparisons, and deal with multiple testing issues.

Method 2: easy confidence interval calculation for the odds ratio (Ch 18)

Given the data, $\begin{matrix} & Y & N \\ F & 70 & 30 \\ M & 160 & 40 \end{matrix}$,

we can easily estimate the ***odds*** of a Yes response for females and males:

$$\textbf{\textit{Odds of Yes for Females}} = \frac{\mathbb{P}(\textbf{\textit{Yes}})}{1 - \mathbb{P}(\textbf{\textit{Yes}})}$$

$$\Rightarrow \quad \textbf{\textit{Estimated Odds of Yes for Females}} = \frac{70/100}{30/100} = \frac{70}{30} \quad (= 2.33)\,.$$

Likewise,

$$\textbf{\textit{Estimated Odds of Yes for Males}} = \frac{160}{40} \quad (= 4.0)\,.$$

Recall that the ***odds*** of an event is ***a number from 0 to ∞, such that the larger the number, the more likely the event is.***

In our data, the odds of response 'Yes' is lower for Females than for Males. This means that females have a lower Yes probability than males do. (Yes, really: young men seem to be more eager to have families than young women.) But is the discrepancy *sufficient* for us to say it's statistically significant, or could it just happen by chance if there is no difference between the opinions of men and women? We need ***a measure of the discrepancy, and its standard error, so we can create a confidence interval.***

Odds ratio and Log odds-ratio

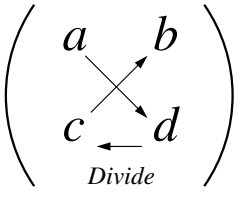
The ***odds ratio*** is the ratio of the two odds above:

$$\text{Odds Ratio} = \frac{\text{odds of Yes for Females}}{\text{odds of Yes for Males}} = \frac{70/30}{160/40} = \frac{70 \times 40}{160 \times 30} \quad (= 0.583)\,.$$

Be careful with the terminology: each of the individual odds is a ratio of Yes to No, but is simply called the ***odds.*** The ***odds ratio*** is the ***ratio of two odds: i.e. the ratio of Yes to No for Females, divided by the ratio of Yes to No for Males.***

It will always have this same format:

- if our table is $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, then the ***odds ratio*** is: $\dfrac{a \times d}{b \times c}$.



Null hypothesis: Under H_0, there is no difference in $\mathbb{P}$(Yes) between Females and Males. This corresponds to ***equal odds of Yes for Females and Males.***

Thus, our null hypothesis is that ***the odds ratio is 1:*** H_0 : $\boldsymbol{OR} = 1$.

Log odds-ratio: Because it is easier to deal with the statistical distribution of a sum rather than a product, we take logs and look at the ***log odds ratio:***

$$\log(\text{OR}) \;=\; \log\left(\frac{a \times d}{b \times c}\right).$$

Under our null hypothesis, the odds ratio is ***1***, so the ***log*** of the odds ratio is $\log(\text{OR}) = \log(1) = 0$.

Our test has become: H_0 : $\log(\boldsymbol{OR}) = 0$.

Remember this is still testing the same null hypothesis: ***does the response (columns) differ by row?***

Standard error for the log odds-ratio: It turns out that the log odds-ratio has an approximate standard error with a very easy formula.

If our table is $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, then:

- The ***log odds ratio*** is $\log\left(\dfrac{a \times d}{b \times c}\right)$;
- The approximate standard error of the ***log*** odds ratio is:

$$\text{se}\,(\log \text{OR}) = \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}\,;$$

- An approximate 95% confidence interval for the ***log*** odds ratio is:

$$\text{estimate} \pm 1.96\,(\text{standard error}) \;=\; \log\left(\frac{a \times d}{b \times c}\right) \;\pm\; 1.96 \times \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}\,.$$

- To test H_0 : $\log(\text{OR}) = 0$, ***calculate the 95% confidence interval and see if it contains 0.***

  If the 95% CI contains 0, then ***there is no evidence that the odds of a Yes response differ between rows.***

  Put another way, if the 95% CI contains 0, then ***there is no evidence of an association between the row variable and the column variable.***

Notes: 1. You are expected to know this formula and procedure for testing H_0 using the log odds-ratio. It is included in the course because it is easy to use.

2. The formula for the standard error is obtained by a method called the ***delta method:*** see Stats 730.

3. We can use the same ideas to derive a p-value for the test against H_0, using $2\,\mathbb{P}\left(Z > \frac{|\log \text{OR}|}{\text{se}(\log \text{OR})}\right)$ where $Z \sim N(0,1)$: i.e., `2*(1-pnorm(abs(logOR/se)))`.

Back-transformation: To back-transform the CI for the log odds-ratio back to the identity scale, simply ***exponentiate:***

'We estimate the odds of Yes for females is between $\exp($***lower CI for log OR***$)$ *and* $\exp($***upper CI for log OR***$)$ *times that of males.*
This is/is not significantly different from 1.'

Example 1: With the data $\begin{array}{c} \\ F \\ M \end{array}\begin{array}{c} \begin{array}{cc} Y & N \end{array} \\ \begin{bmatrix} 70 & 30 \\ 160 & 40 \end{bmatrix} \end{array}$:

The log odds-ratio is: $\log\left(\dfrac{70\times 40}{160\times 30}\right) = \log 0.583 = -0.539.$

The standard error for the log OR is:

$$\textbf{\textit{se}}\,(\log \textbf{\textit{OR}}) = \sqrt{\frac{1}{70}+\frac{1}{30}+\frac{1}{160}+\frac{1}{40}} = 0.281.$$

Approximate 95% confidence interval for the log OR:

estimate± 1.96 ***(standard error)*** $= -0.539 \pm 1.96\times 0.281 = (-1.09, 0.01)\,.$

This confidence interval contains 0 (just) so we have no evidence at the 5% level of a difference between the odds of Yes for males and females. The p***-value is 2*(1-pnorm(0.539/0.281)) = 0.055.***

Back-transformation: the approximate 95% CI for the OR is:

$$(\exp(-1.09), \exp(0.01)) = (0.34\,,\ 1.01).$$

This CI contains 1, duplicating our conclusion that there is no evidence for an association between the row and column factors.

We estimate that the odds of wanting children for females is between 0.34 and 1.01 times the odds of wanting children for males.

Example 2: Snorers and nightmares (Exercise). The table is $\begin{bmatrix} 11 & 82 \\ 12 & 74 \end{bmatrix}$.

The 95% CI for the log OR is: $(-1.07, 0.69)$: ***no evidence of a difference.***

The p-value is 0.67. If you calculate the p-value from Method 1 analogously, it is also 0.67.

The estimated odds of common nightmares for snorers is between 0.34 and 1.99 times that for non-snorers.

Summary of odds-ratio method (Method 2):

Method: Test for whether the 95% CI for the log odds ratio contains 0. Calculate this CI by hand. The log odds-ratio is $\log\left(\frac{a \times d}{b \times c}\right)$, and the standard error is approximately $\sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}$. The 95% CI is estimate $\pm$ 1.96 (std error).

The p-value is gained from a 2-sided Z-test using test statistic (estimate / se).

Question tested: Do males and females differ with respect to the odds of Yes?
Equivalently: does the response (columns) differ by row?
Or: is there an association between the row variable and the column variable?

Scope: Only suitable for a 2×2 ***table.***

Method 3: Pearson Chi-square test (Chapter 17)

The Chi-square test is another method that is familiar from Stats 10x. It is the easiest way of testing for association of rows and columns in an $m \times n$ table.

Note that an $m \times n$ table means ***we have TWO categorical variables: one with m levels in the rows (the 'predictor'), and one with n levels in the columns (the 'response').***

Let's use the childhood weight distribution from Stats 10x to illustrate:

		Weight			
		Normal	Overweight	Obese	Total
Year:	1989	754	96	21	871
	2000	626	187	81	894
	Total	1380	283	102	1765

The Chi-square test is a ***global test*** that tests for ***any*** association between rows and columns. Under H_0, the rows differ only by their row totals: ***all rows are multiples of each other, due only to the different row totals.***

Some easy algebra shows that if all rows are multiples of each other, then all columns are multiples of each other as well. This means that it doesn't matter which variable we put in the rows and which in the columns: the Chi-square test will return the same answer regardless.

The Chi-square test is based on the theoretical result that, as cell-entries in the table become sufficiently large, the distribution of the following test statistic is approximately Chi-squared under H_0, with degrees of freedom equal to $(\#\text{rows} - 1) \times (\#\text{columns} - 1)$:

$$\text{test statistic} \;=\; \sum_{\text{cells } i,\,j} \frac{\left(\text{observed}_{ij} - \text{expected}_{ij}\right)^2}{\text{expected}_{ij}} = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \,.$$

Here, O_{ij} is the entry ***observed*** in cell (i, j) of the table, and E_{ij} is the entry that would be ***expected*** under H_0, if all rows really were multiples of each other. The entries E_{ij} are gained by allocating the row totals among columns according to the column proportions in the whole table. For example, in the table overleaf, the expected entry for cell $(i, j) = (2, 1)$ under H_0 would be: $894 \times 1380/1765 = 699.0$. Thus the Chi-square test is based on the deviations from these expected frequencies over the whole table. Here, the degrees of freedom for the test are $(2 - 1) \times (3 - 1) = 2$.

The Chi-square test is very easy to run in R, and all the calculations are done for us. You do not need to know the formulas above or apply them by hand. For the childhood weight example:

```
## Enter the data by hand, as a 2x3 matrix, entering one row at a time:
children.tab <- matrix(c(754, 96, 21, 626, 187, 81), byrow=T, nrow=2)
## Give the matrix some names for convenience:
dimnames(children.tab)<-list(c("Yr1989", "Yr2000"), c("Normal", "OW", "Obese"))
## Have a look at the matrix:
children.tab
       Normal  OW Obese
Yr1989    754  96    21
Yr2000    626 187    81

## Conduct the Chi-square test:
chisq.test(children.tab)

Pearson's Chi-squared test

data:  children.tab
X-squared = 76.141, df = 2, p-value < 2.2e-16
```

The conclusion is that ***we have extremely strong evidence that the childhood weight profile differs across rows.***

There is extremely strong evidence of a different profile of childhood weight in 2000 compared with 1989.

For our snorers and nightmare-sufferers:

```
nightmare.tab <- matrix(c(11, 82, 12, 74), byrow=T, nrow=2)
chisq.test(nightmare.tab)
Pearson's Chi-squared test with Yates' continuity correction

data:  nightmare.tab
X-squared = 0.040421, df = 1, p-value = 0.8407
```

The conclusion is the same as the previous two methods: ***no evidence against*** H_0***.***

The p-value is a little different: 0.84 instead of 0.67. The Chi-square test is designed quite differently from the tests in Methods 1 and 2, so the p-values are sometimes a little different. However, these discrepancies tend to vanish as the numbers in the table get larger.

Notes: 1. The Chi-square test relies on reasonably large entries in ***every*** cell of the table. As a rule of thumb, all cell entries should be ≥ 5.

2. If this assumption is violated, an alternative called the Fisher Exact Test can be used: the R command is `fisher.test`. The two tests will give very similar answers if the Chi-square assumptions are met.

Summary of Chi-square test method (Method 3):

Method: Enter the data as a matrix or a data-frame containing only the relevant numeric columns. Use the R command `chisq.test(table)`. The Chi-square test is a global test for association of two variables.

Question tested: Are the rows multiples of each other?
Equivalently: are the columns multiples of each other?
Or: does the response (columns) differ by row?
Or: is there an association between the row variable and the column variable?

These are the same questions asked in Methods 1 and 2, but this time potentially applied to a larger table.

Scope: Suitable for any $m \times n$ table.
Note that this means we are testing association between ***just TWO categorical variables: a row variable with*** m ***levels, and a column variable with*** n ***levels.***

Method 4: Binomial GLM (Chapter 15)

Look again at our 2×2 tables:

$$\begin{array}{c} \\ F \\ M \end{array} \begin{array}{c} \begin{array}{cc} Y & N \end{array} \\ \begin{bmatrix} 70 & 30 \\ 160 & 40 \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ \text{Snore} \\ \text{Not} \end{array} \begin{array}{c} \begin{array}{cc} \text{Common} & \text{Rare} \end{array} \\ \begin{bmatrix} 11 & 82 \\ 12 & 74 \end{bmatrix} \end{array}$$

We could formulate either of these as a ***Binomial GLM on grouped data,*** just as we did at the end of Chapter 15.

The two columns correspond to the outcomes 'success' and 'fail'. The response is a numeric variable Y recording the ***numbers of success and fail for each level of the predictor factor.*** The 'predictor' is the row factor.

Our question of interest is: does the column probability differ according to the row factor?

This is exactly the same as asking, ***do we need the predictor variable in the model?***

If the column probabilities are the same for all levels of the predictor variable in the rows, then we do not need the predictor variable in the model. We would expect a Binomial GLM of `cbind(success, fail) ~ row.factor` to give a ***non-significant coefficient for the row factor.*** By contrast, if there is an association between rows and columns, we would expect a significant coefficient.

Thus, to test for an association between the rows and columns, ***we just look at the significance of the row factor in the summary output for a Binomial GLM of cbind(column1, column2) ~ row.factor.***

It turns out that applying the Binomial GLM on 2×2 tables gives almost identical p-values to Methods 1 and 2, and similar results to the Chi-square test in Method 3. Here it is with the snorers and nightmare example:

```
## Reformat the data into a data frame with columns for Snore, #Success
## responses (Common nightmares), and #Fail responses (Rare nightmares):
nightmare.df <- data.frame(Snore=c("Y", "N"), Common=c(11, 12), Rare=c(82, 74))
## Look at the data frame:
nightmare.df
  Snore Common Rare
1     Y     11   82
2     N     12   74

## Fit the GLM:
bin.glm <- glm(cbind(Common, Rare)~Snore, family=binomial, data=nightmare.df)
```

```
## Look at the summary output:
summary(bin.glm)

Call: glm(formula = cbind(Common, Rare) ~ Snore, family = binomial,
                        data = nightmare.df)
Deviance Residuals:
[1]  0  0

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.8192     0.3112  -5.846 5.05e-09 ***
SnoreY       -0.1897     0.4472  -0.424    0.671
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.8012e-01  on 1  degrees of freedom
Residual deviance: 1.0658e-14  on 0  degrees of freedom
AIC: 12.312
```

The item to look for is ***the p-value for snoreYes: 0.671.***
This says there is no need to include the Snore variable in the model. That is, we have no evidence of an association between Snoring and Nightmares.

The p-value is the same as we received from Methods 1 and 2. We can expect these three methods to give almost identical results every time.

Note that, for a 2×2 table, the Binomial GLM is ***saturated.*** It has fitted two parameters: p_1 for $\mathbb{P}(\text{Common} \,|\, \text{Snorer})$, and p_2 for $\mathbb{P}(\text{Common} \,|\, \text{Not Snorer})$; and it only has two data points: the proportion of Common among the Snorers, and the same among the Non-Snorers. There are therefore no degrees of freedom left and the residual deviance is 0.

However, if we have more than one predictor variable, we can fit various non-saturated models. Because we are looking at grouped data, if all counts are reasonably large we should then do the test for using a quasi-binomial model:

```
1 - pchisq(resid.dev, resid.df)
```

If this is significant, then ***refit the model using family=quasibinomial.*** All else would proceed the same way, except that if you use the `anova` command it should specify `test="F"` instead of `test="Chisq"`, as described in Handout 7.

The advantage of the Binomial GLM method is that it can be used for multiple predictors. By contrast with Method 3 (Chi-square test) which is only suitable for ***one*** categorical predictor and one categorical response variable, the Binomial GLM can have ***any number of categorical or numeric predictors.***

However, it can only have a response variable with ***two categories: 'success' and 'fail'.*** Thus, in the two-predictor case, while the Chi-square test can handle an $m \times n$ table, the Binomial GLM can only handle a $m \times 2$ ***table.***

Summary of Binomial GLM method (Method 4):

Method: Enter the data as a data-frame with one column specifying the levels of the row factor, and two columns for the two levels of the response.

Use `glm(cbind(respY, respN) ~ row.factor, family=binomial)`. If using the `anova` command on the fitted GLM, use `anova(fit, test="Chisq")`.

If counts are large and `1 - pchisq(resid.dev, resid.df)` is significant, refit with `family=quasibinomial`, and use `anova(fit, test="F")` where needed.

To address the question of row and column association, read off the p-value corresponding to the row factor in `summary(fit)`.

Question tested: Does the response (columns) differ by row?
Or: is there an association between the row variable and the column variable?

This is the same question asked in Methods 1, 2, and 3.

Scope: Suitable for any $m \times 2$ table, in which case it gives a test of the association between one m-level categorical predictor and one ***binary*** response variable.

Also suitable for ***any number of predictor variables,*** including factors and numeric variables and their interactions, but only for a ***binary*** (two-column) response variable. Note that models with more than one predictor variable can no longer be easily portrayed as a table of counts.

Method 5: Poisson GLM (Chapter 19)

The Poisson GLM method is the most general, but is also a little less obvious than the other methods. It is suitable for any $m \times n$ table (i.e. one m-level categorical predictor and one n-level categorical response), and more generally for any number of predictors in any combination of numeric and categorical variables. It has a broader scope than the Binomial approach because ***we no longer require that the response variable is binary: it can have any number of levels,*** n***.***

Due to a theoretical link between the Poisson distribution and the Binomial distribution, the Poisson method will give almost identical results to all the previous methods, including the Chi-square test up to minor discrepancies. However, the Poisson GLM is also available for more complicated data structures where none of the other methods is suitable.

To apply the Poisson method, we first need to reformat the data. Here is the nightmare data in both Binomial and Poisson format:

```
nightmare.bin.df:
   Snore Common Rare
       Y     11   82
       N     12   74
```

```
nightmare.pois.df:
   Snore Nightmare Count
       Y    Common    11
       Y      Rare    82
       N    Common    12
       N      Rare    74
```

You can see that the two response columns in the Binomial data-frame have been 'unwrapped' to create ***one column for the levels of the Nightmare factor, and one column for the numeric response, Count.***

The dataframe is therefore formatted such that ***Snore and Nightmare are both predictors, and the response is Count.***

We need to reformat our analysis question accordingly.

- For the Binomial GLM, we asked whether the Nightmare response of (Common, Rare) is affected by the different levels of Snore. This is equivalent to asking whether we need a different parameter for (Nightmare = Common) for the two different levels of Snore.

- For the Poisson GLM, we only have one response column, Count, and two predictors: Snore and Nightmare. The question of whether we need a different parameter for (Nightmare=Rare) for each level of Snore translates to the question, ***is there an interaction between the predictors Snore and Nightmare?***

  In other words, we will fit the model `Count ~ Snore * Nightmare`, and look for evidence of an ***interaction term.***

  - If there is ***no interaction*** between Snore and Nightmare, then the model `Count ~ Snore + Nightmare` will suffice. This means that the model is:

    $$\text{Count}_i \sim \text{Poisson}(\mu_i) \quad \text{where} \quad \mu_i = \exp\left(\beta_0 + \beta_1 \,\text{Snorer}_i + \beta_2 \,\text{Rare}_i\right),$$

    where Snorer_i is 1 for Snorers and 0 for non-Snorers, and Rare_i is 1 for Rare and 0 for Common nightmares. Translating to the Binomial framework:

$$\begin{aligned}
\mathbb{P}(\text{Rare} \,|\, \text{Snorer}) &= \frac{\mathbb{E}\left(\text{number both Rare and Snorer}\right)}{\mathbb{E}\left(\text{number Snorer}\right)} \\
&= \frac{\exp\left(\beta_0 + \beta_1 + \beta_2\right)}{\exp\left(\beta_0 + \beta_1 + \beta_2\right) + \exp\left(\beta_0 + \beta_1\right)} \\
&= \frac{\exp\left(\beta_2\right)}{\exp\left(\beta_2\right) + 1}\,.
\end{aligned}$$

We get exactly the same expression for $\mathbb{P}(\text{Rare} \,|\, \text{Non-Snorer})$:

$$\mathbb{P}(\text{Rare} \,|\, \text{Non-Snorer}) \;=\; \frac{\exp(\beta_0 + \beta_2)}{\exp(\beta_0 + \beta_2) + \exp(\beta_0)} \;=\; \frac{\exp(\beta_2)}{\exp(\beta_2) + 1}\,.$$

So if there is ***no interaction*** between Snorer and Nightmare in the Poisson model, then $\mathbb{P}(\text{Rare} \,|\, \text{Snorer}) = \mathbb{P}(\text{Rare} \,|\, \text{Non-Snorer}) = \exp(\beta_2)/\left\{\exp(\beta_2) + 1\right\}$.

This model is equivalent to a Binomial model in which the probability of rare nightmares is a logistic function of β_2, and Snorer has no influence.

- Conversely, if there ***is an interaction*** between Snore and Nightmare, the model is `Count ~ Snore + Nightmare + Snore:Nightmare`, with

$$\text{Count}_i \sim \text{Poisson}(\mu_i)$$

$$\text{where} \quad \mu_i = \exp\left(\beta_0 + \beta_1\,\text{Snorer}_i + \beta_2\,\text{Rare}_i + \beta_3\,\text{Rare}_i \times \text{Snorer}_i\right).$$

Translating to the Binomial framework:

$$\begin{aligned}
\mathbb{P}(\text{Rare} \,|\, \text{Snorer}) \;&=\; \frac{\mathbb{E}\,(\text{number both Rare and Snorer})}{\mathbb{E}\,(\text{number Snorer})} \\
&=\; \frac{\exp(\beta_0 + \beta_1 + \beta_2 + \beta_3)}{\exp(\beta_0 + \beta_1 + \beta_2 + \beta_3) + \exp(\beta_0 + \beta_1)} \\
&=\; \frac{\exp(\beta_2 + \beta_3)}{\exp(\beta_2 + \beta_3) + 1}\,.
\end{aligned}$$

The effect of Snorer does not vanish, because it is still there in the 'snore-specific' parameter β_3. The expression for $\mathbb{P}(\text{Rare} \,|\, \text{Non-Snorer})$ is not the same:

$$\mathbb{P}(\text{Rare} \,|\, \text{Non-Snorer}) \;=\; \frac{\exp(\beta_0 + \beta_2)}{\exp(\beta_0 + \beta_2) + \exp(\beta_0)} \;=\; \frac{\exp(\beta_2)}{\exp(\beta_2) + 1}\,.$$

This means that an interaction between Snorer and Nightmare in the Poisson model is equivalent to a Binomial model in which the response probabilities for rare nightmares ***do differ*** for the different levels of the predictor factor, Snorer.

Link with the odds ratio for 2 × 2 tables

The odds ratio for Rare nightmares for Snorers versus Non-snorers is the ratio of Rare to Common for Snorers, divided by that for Non-snorers.

```
nightmare.bin.df:
   Snore Common Rare
      Y     11   82
      N     12   74          OR = (82/11) / (74/12) = 1.209
```

In mathematical notation:

$$\begin{aligned}
\text{OR} &= \frac{\mathbb{E}\left(\text{number Rare and Snorer}\right)/\mathbb{E}\left(\text{number Common and Snorer}\right)}{\mathbb{E}\left(\text{number Rare and Non-Snorer}\right)/\mathbb{E}\left(\text{number Common and Non-Snorer}\right)} \\
&= \frac{\exp\left(\beta_0+\beta_1+\beta_2+\beta_3\right)/\exp\left(\beta_0+\beta_1\right)}{\exp\left(\beta_0+\beta_2\right)/\exp\left(\beta_0\right)} \\
&= \frac{\exp\left(\beta_2+\beta_3\right)}{\exp\left(\beta_2\right)} \\
&= \exp(\beta_3)\,.
\end{aligned}$$

Thus, the ***interaction parameter,*** β_3, also gives the ***log odds ratio.***

Summary of Poisson GLM method (Method 5):

You do not need to know or understand the mathematical details above. You only need to know the summary details below.

Method:

- To test for association of two factors using a Poisson GLM, unwrap the data-frame such that each factor has ***one column*** describing its different levels, and there is another column called ***Count*** that will be treated as the response.
- Fit the model `glm(Count ~ factor1 * factor2, family=poisson)`. If all counts are reasonably large, apply the usual test `1 - pchisq(resid.dev, resid.df)` for overdispersion. If this gives a significant result, refit the model using `glm(Count ~ factor1 * factor2, family=quasipoisson)`.
- For a 2×2 table, use `summary(fit)` and look at the row for the ***interaction coefficient.*** If this is significant, we have evidence of an association between `factor1` and `factor2`.

  Furthermore, the estimated interaction parameter gives the ***log odds ratio***. However, you might still favour the manual calculation of log odds ratio to be sure of which way round it is being calculated: e.g. odds of Rare for Snorers divided by odds of Rare for Non-snorers.
- For larger $m \times n$ tables, use `anova(fit, test="Chisq")` or `anova(fit, test="F")`, depending on whether the GLM used `family=poisson` or `family=quasipoisson` respectively. Again, look for the p-value on the ***interaction*** row. If it is significant, we have evidence of an association between the two variables.

Question tested: Is there an association between the row variable and the column variable? This is the same question asked in Methods 1, 2, 3, and 4.

Scope: Suitable for any $m \times n$ table, in which case it gives a test of the association between one m-level categorical predictor and one n-level response variable. The table must be reformatted into the correct dataframe format first.

Also suitable for ***any number of predictor variables,*** including factors and numeric variables and their interactions, with an n-level response variable. Note that models with more than one predictor variable can no longer be easily portrayed as a table of counts, and the test for association is conceptually more complicated if there are several predictors.

Example 1: Nightmare data (2×2 table).

```
## Data in Binomial format:
nightmare.df <- data.frame(Snore=c("Y", "N"), Common=c(11, 12), Rare=c(82, 74))

## Data in Poisson format:
nightmare.pois.df <- data.frame(Snore=c("Y", "Y", "N", "N"),
      Nightmare=c("Common", "Rare", "Common", "Rare"), Count=c(11, 82, 12, 74))
## Fit the Poisson GLM:
pois.fit <- glm( Count ~ Snore * Nightmare, family=poisson, data=nightmare.pois.df)
summary(pois.fit)

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)           2.48491    0.28868   8.608  < 2e-16 ***
SnoreY               -0.08701    0.41742  -0.208    0.835
NightmareRare         1.81916    0.31120   5.846 5.05e-09 ***
SnoreY:NightmareRare  0.18967    0.44716   0.424    0.671
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1.1130e+02  on 3  degrees of freedom
Residual deviance: 6.6613e-16  on 0  degrees of freedom
AIC: 28.978
```

The interaction p-value is ***0.67, exactly as for Methods 1, 2, and 4.*** We conclude, once again, that there is no evidence of an association between Snoring and Nightmare frequency.

The interaction parameter is $\hat{\beta}_3 = 0.18967$. This is the same as the log odds-ratio for Rare nightmares for Snorers versus Non-snorers:

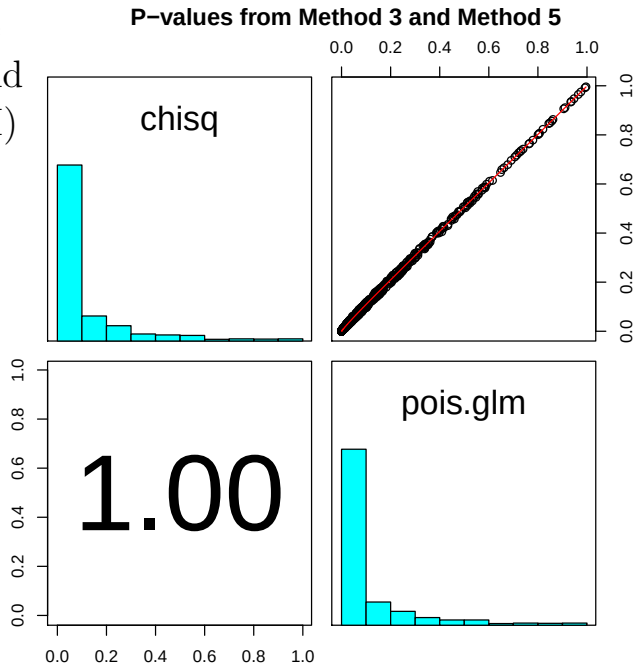
```
Snore Common Rare
    Y     11   82
    N     12   74    log(OR) = log { (82/11) / (74/12) } = log(1.2088) = 0.18967
```

Example 2: Childhood weight data (2×3 table). The Poisson GLM gives the same results as Method 3 (Chi-square test): strong evidence of an association.

```
## ---------------------------------------------------------------------
## Data in Chisq-test format:
children.tab <- matrix(c(754, 96, 21, 626, 187, 81), byrow=T, nrow=2)
dimnames(children.tab)<-list(c("Yr1989", "Yr2000"), c("Normal", "OW", "Obese"))

children.tab
       Normal  OW Obese
Yr1989    754  96    21
Yr2000    626 187    81

chisq.test(children.tab)
X-squared = 76.141, df = 2, p-value < 2.2e-16

## ---------------------------------------------------------------------
## Data in Poisson format:
children.pois.df <- data.frame(Year=c(rep("Yr1989", 3), rep("Yr2000", 3)),
     Weight=rep(c("Normal", "OW", "Obese"), 2), Count=c(754, 96, 21, 626, 187, 81))

children.pois.df
    Year Weight Count
1 Yr1989 Normal  754
2 Yr1989     OW   96
3 Yr1989  Obese   21
4 Yr2000 Normal  626
5 Yr2000     OW  187
6 Yr2000  Obese   81

pois.fit <- glm(Count ~ Year * Weight, family=poisson, data=children.pois.df)

## Use ANOVA because we have more than a 2x2 table: returns same conclusion.
anova(pois.fit, test="Chisq")

Analysis of Deviance Table

Model: poisson, link: log
Response: Count
Terms added sequentially (first to last)

            Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                            5    1660.68
Year         1     0.30         4    1660.38   0.5841
Weight       2  1581.33         2      79.06   <2e-16 ***
Year:Weight  2    79.06         0       0.00   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Examples from many simulations

We can use simulation to investigate the difference between Methods 3 and 5 (Chi-square test and Poisson GLM) on a simulated 2×3 table.
Here is a pairs plot of p-values from the two methods for 1000 simulated tables.



The tables were simulated with row totals of 100 and 200, and column column probability vectors of $\boldsymbol{p_1} = (0.4, 0.5, 0.1)$ and $\boldsymbol{p_2} = (0.3, 0.5, 0.2)$ for the two rows. This constitutes a small but real difference between the two rows.

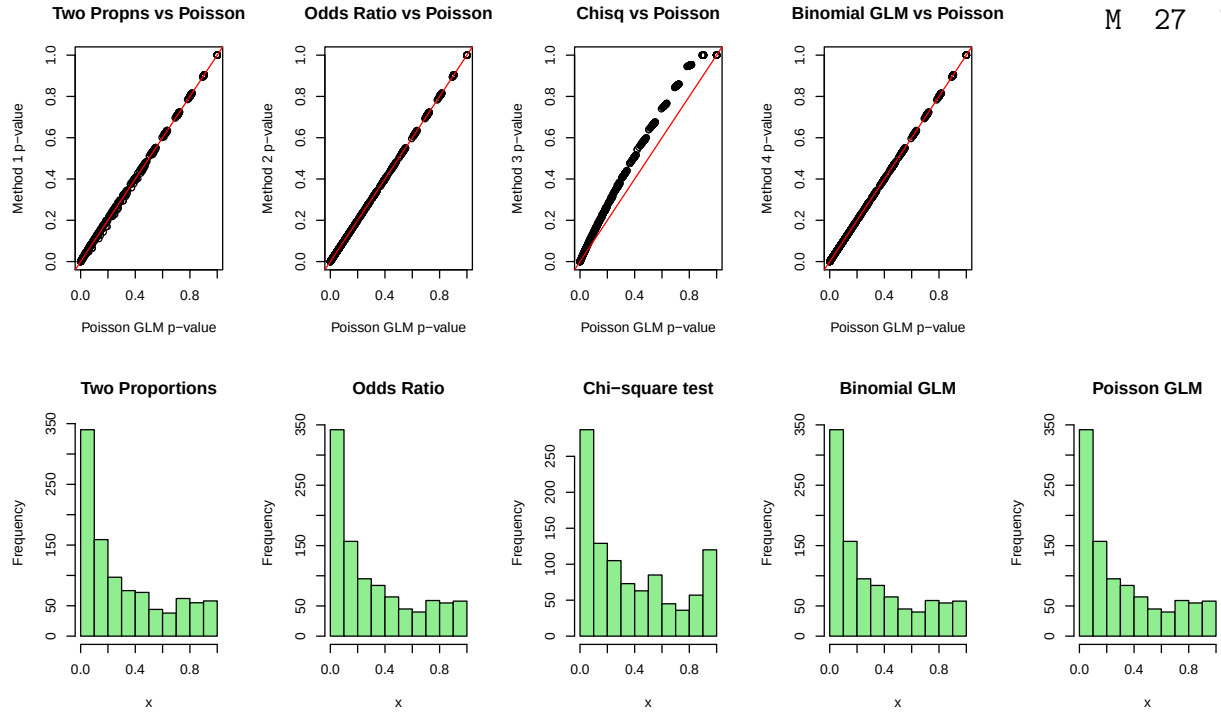
You can see the two methods give ***almost identical*** p***-values.***

An example simulated table in the Chi-square test format is:

	Yes	No	Maybe
F	44	41	15
M	56	103	41

For 2×2 tables, all five methods are applicable. Below are the p-value results for each of the five methods compared against the Poisson GLM method, for a 2×2 table simulated with row totals of 50 and 100, and column 1 probabilities of $p_1 = 0.4$ and $p_2 = 0.3$. All five methods are almost identical, except for some small differences from Method 3 (Chi-square test). Example table:

	Yes	No
F	21	29
M	27	73



Stats 20x Handout 9: Time Series

Time-series data are simply data collected at regularly-spaced time intervals. We are interested in how some phenomenon changes over time. For example, we might be recording air pollution, which changes throughout the seasons and might also have long-term trends over time. Similarly, and of great topical interest, is the amount of polar ice: again, this changes seasonally, but our particular interest is whether there are also long-term trends which could spell disaster as sea levels rise. Time series are also very common in business and commerce, for example to study stock-market fluctuations or demand for some commodity like petrol.

As usual, there is a numeric response variable, Y, and predictor variables that could be numeric, categorical, or both. The distinctive characteristics of time-series data are:

- One of the predictor variables is ***time*** (numeric). So we often call the response Y_t.
- There might also be an additional predictor for ***season*** (categorical).
- Data are typically collected at equally-spaced time-points and presented in order from the first observation to the last.

Our study of time series has four parts. You need to be able to do the following:

1. Know the ***four components of time series:*** trend, cycle, seasonal, random. In Stats 20x we generally combine trend and cycle into a single notion of trend-cycle, so you only need to have a vague idea of what cycles are about.

2. ***Interpret STL decomposition plots.*** These are plots for time-series data that do a quick decomposition of the time series into trend-cycle, seasonal, and random components. They are exploratory plots, so they are more like a raw data plot than a fitted model plot, although R does do some computation behind the scenes in order to produce them. Think of the familiar `trendScatter` or `pairs20x` plots: the STL plot is a similar idea.

   (We call this sort of plot a *non-parametric model.* What we end up with is a useful picture, but we don't have a parametric model which we can interpret in the Executive Summary using notions like 'one unit along means $\widehat{\beta}_1$ units up'.)

3. ***Interpret Holt-Winters forecasting output.*** The word *forecasting* is time-series jargon for 'predicting ahead'. The Holt-Winters method uses a technique called *exponential smoothing* to generate forward predictions. Again, this is a non-parametric technique rather than a fitted model.

4. ***Fit an autoregressive linear model.*** These models are very similar to our familiar linear models using `lm()`, but they have one new component: the *most recent* observation y_{t-1} is used as a predictor for the value of y_t.

9.1 Components of time series: trend, cycle, seasonal, random

1. ***Trend*** describes long-term changes in the mean of the series. It is a smooth, slowly-changing component. It might be appropriate to model the trend using a straight line, either on the identity scale or the log scale.

2. ***Cycle*** describes *irregular but somewhat long-lived* departures from the trend. This component does not repeat: it is irregular in both shape and occurrence. It describes changes of longer duration than seasonal changes, but short relative to the long-term trend. For example, the earth is currently experiencing global warming (trend), but there may still be periods of several years of cooling relative to the long-term trend — for example, these could be caused by polar ice breaking down and drifting north or south. In this case we would say that the trend is warming, but some cooling cycles are evident.

   If we are only using non-parametric descriptions of trend, for example using STL decompositions and Holt-Winters filtering, we can combine cycle and trend together using ***a single trend-cycle term.*** For this reason, we do not consider cycle very much in Stats 20x, except occasionally to point out possible cycles on the STL plots.

3. The ***seasonal component*** is a *regular repeating pattern* which occurs within every time period. For example, polar ice will always be at a maximum in winter, and a minimum in summer, over-and-above any long-term trends.

4. The ***random component*** is the familiar random scatter ϵ that is invariably associated with any observation. It is random and unpredictable, and it might be suitable to describe it as $\epsilon \sim N(0, \sigma^2)$. However, one issue that often arises in time series data is that ***successive ϵ_t terms might not be independent.*** You will need to be able to ***recognise*** the signs of non-independent ϵ_t on STL plots. If errors are not independent, we will no longer be able to assume that $\epsilon_t \sim \text{iid} N(0, \sigma^2)$ when we fit parametric models to time series using `lm`. Thus you will also need to be able to ***deal with the problem of non-independent errors*** by using autoregressive models, as described in Section 9.4.

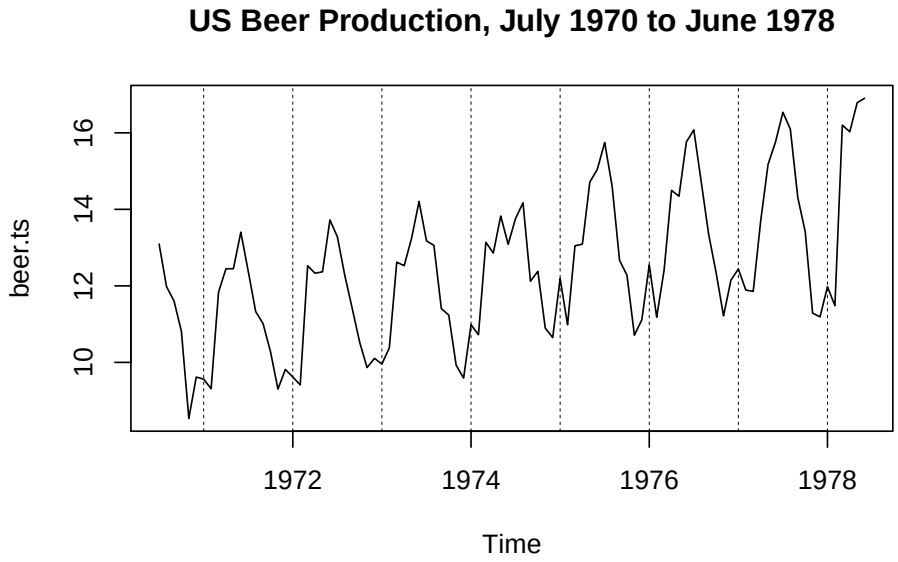
Creating a time series object in R

Because time series have regularly-spaced observations and repeating seasonal effects, we need to use a special command to tell R that our data has these characteristics. We use `ts(data, start, frequency)` to tell R the start-point and repeat frequency of the seasonal component. For example, for the US beer production data in the coursebook, the following command specifies repeating monthly units with 12 observations per year, starting in July 1970:

```
beer.df <- read.table("beer.txt", header=T)
beer.ts <- ts(beer.df$beer, start=c(1970, 7), frequency=12)
```

We can now plot the `ts` object, and R will understand correctly that it should be plotted with lines to join up adjacent observations:

```
plot(beer.ts, main="US Beer Production, July 1970 to June 1978")
abline(v=1971:1978, lty=2)  ## Draw dashed vertical lines to delimit the years
```



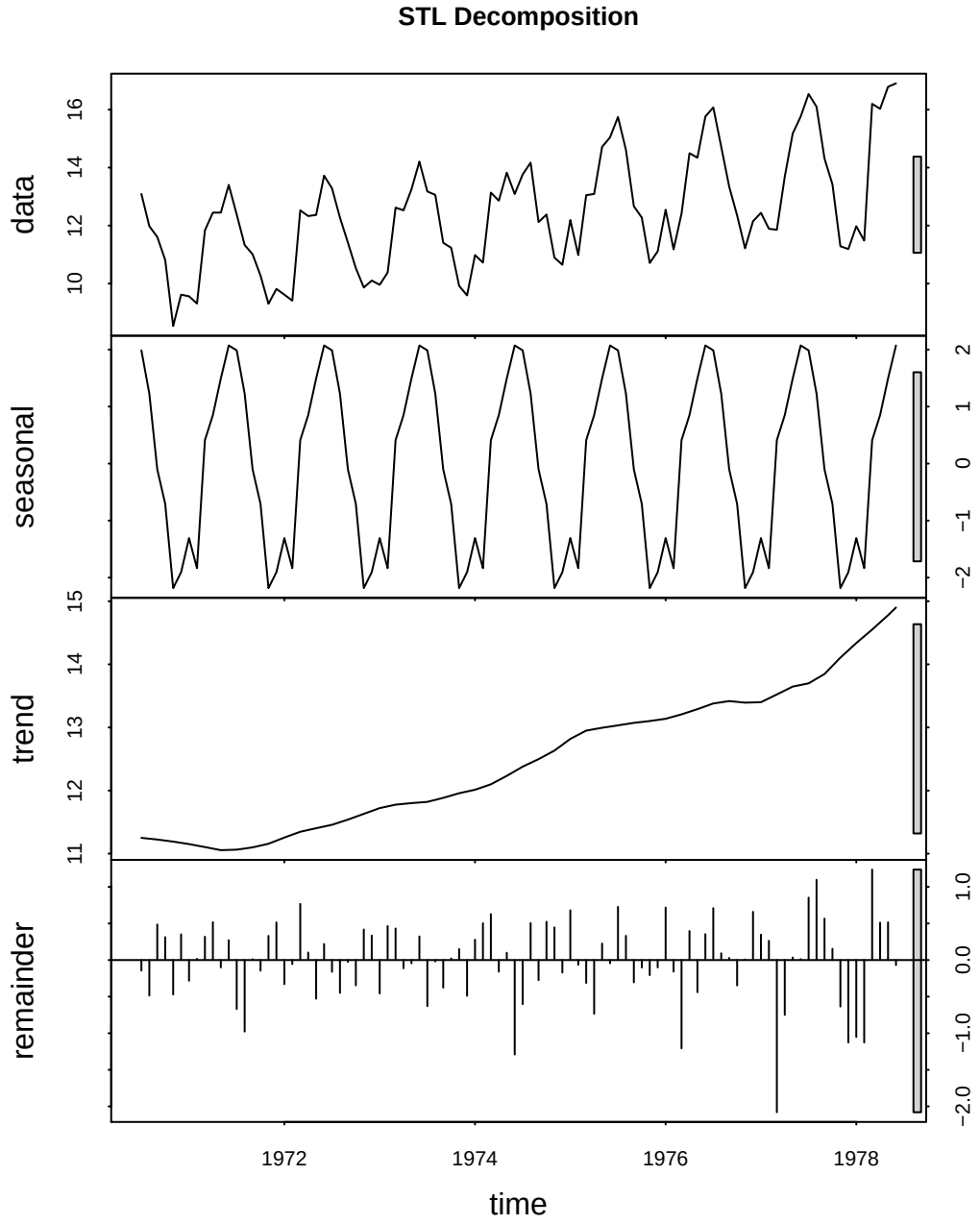
9.2 STL decomposition plots

STL stands for **Seasonal Trend Loess**. The word 'Loess' describes the type of non-parametric smoother that we have previously used in `trendScatter` and `pairs20x`. The rest of the acronym indicates that the plot breaks down the time series into:

1. A long-term **Trend-cycle component,** which is a smooth non-parametric picture of how the mean changes over time.
2. A repeating **Seasonal component,** which repeats with the frequency that you specified when creating the time-series object.
3. The **Random component,** which is what remains after the previous two components are accounted for.

All you need to be able to do with STL output is ***describe the key features of the plot.***

```
beer.ts <- ts(beer.df$beer, start=c(1970, 7), frequency=12)  ## Create ts object
beer.stl <- stl(beer.ts, s.window="periodic")   ## Create the STL breakdown
plot(beer.stl, main="STL Decomposition")  ## Look at the plot (shown overleaf)
beer.stl     ## Look at the numeric output
Components
           seasonal    trend    remainder
Jul 1970  1.9860322 11.25002 -0.144054305
Aug 1970  1.2263050 11.23684 -0.485144696
Sep 1970 -0.1007978 11.22366  0.486140560
Oct 1970 -0.7082518 11.20719  0.313064732
```



Our comments on this plot could be:

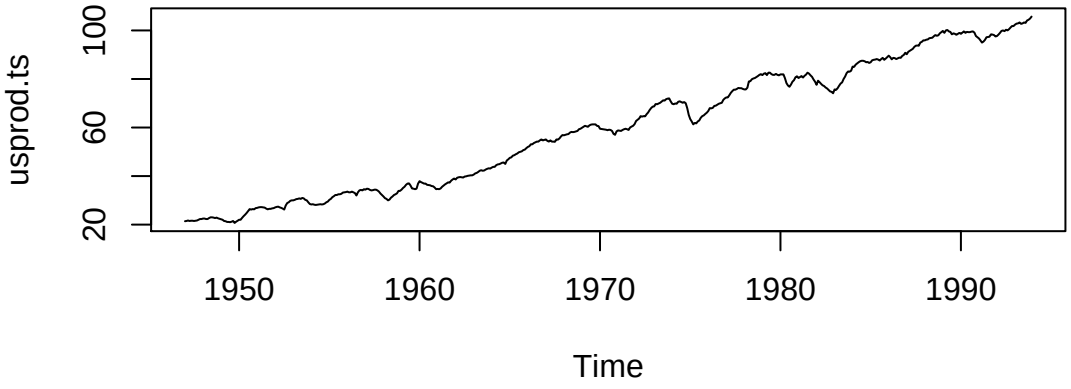
1. ***There is a generally increasing trend in beer production over time. There is a hint of non-linearity, suggesting that the trend may be curving upwards over time.***

2. ***There is a strong seasonal component, which is of high magnitude relative to the overall spread of the data.***

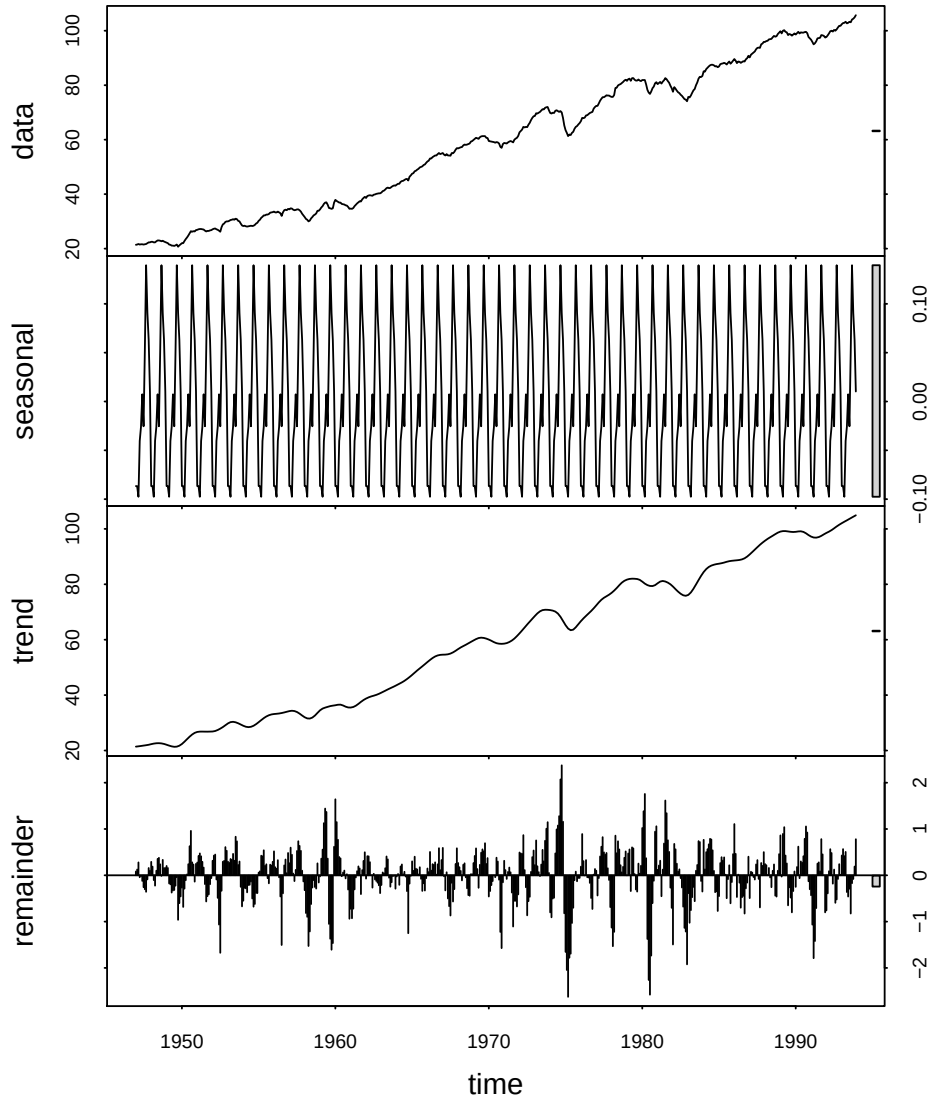
3. ***There is some bunching of the residuals, especially near the end of the data. This suggests there could be some autocorrelation (non-independence) in the residuals.***

Here is the US production index from January 1947 to December 1993.

```
usprod.ts <- ts(usprod.df$ip, frequency=12, start=c(1947, 1))
```




```
plot( stl(usprod.ts, s.window="periodic") )
```



There is a largely increasing trend, with occasional brief downturns. There is no clear seasonal effect — the magnitude of the seasonal effect on the plot is very small in the context of the data as a whole. There is clear bunching of residuals, so we suspect autocorrelation.

9.3 Holt-Winters forecasting

We have seen in previous chapters how we can use a ***parametric*** model to describe the data and predict the response for new values of the predictors. We can do the same with time series, by fitting a linear model as in the next section. However, we might also wish to use ***non-parametric*** methods for describing the time series and predicting forwards. Non-parametric methods rely primarily on the patterns observed in the data, instead of parametric assumptions such as a straight-line trend. The STL decomposition is one non-parametric method for describing the data, but it cannot be used for predicting into the future. The Holt-Winters filter is an alternative method, which describes the data using an alternative non-parametric approach called ***exponential smoothing***, and also has a different formulation for the trend that does enable it to predict forwards into the future. Since the STL method is excellent for plotting time series, we tend to use the Holt-Winters method primarily for ***forecasting.***

The aim of ***forecasting*** is to ***predict forwards in time,*** beyond the end of the data. We must always be cautious when predicting outside of the range of our data. Be aware that all forecasting is based on the strong assumption that ***the observed patterns will continue in like form.***

Exponential smoothing

The idea of exponential smoothing is to create a smooth line that traces the time series data. In its simplest form, the smoothed value for time t is a ***weighted average*** of the smoothed value for time $t-1$, and the current data value at time t. Thus:

$$S_t = \alpha Y_t + (1-\alpha) S_{t-1},$$

where α is a weight between 0 and 1. To start the series, we put $S_1 = Y_1$.

- Small values of α put a larger weight on S_{t-1} and generate a ***smoother series;***
- Large values of α put a larger weight on the data Y_t, which includes its random scatter, so they generate a ***less smooth series.***

If we recursively substitute values for S_{t-1}, S_{t-2}, and so on, we find that:

$$S_t = \alpha Y_t + \alpha(1-\alpha) Y_{t-1} + \alpha(1-\alpha)^2 Y_{t-2} + \ldots + \alpha(1-\alpha)^{t-2} Y_2 + (1-\alpha)^{t-1} Y_1.$$

The method is called ***exponential smoothing*** because each smoothed value uses *all* the previous Y values, but with exponentially decreasing weights.

This simple version of exponential smoothing is suitable for fitting a single trend line to the series. If we use it to predict into the future, all predictions after the end of the series will be given the same value, equal to the final value of the smooth term, S_T.

The Holt-Winters filter uses a more advanced version of exponential smoothing that includes both a ***trend*** and a ***seasonal effect.*** Each of these is obtained by its own version of exponential smoothing: the process is described as ***double exponential smoothing*** and ***triple exponential smoothing*** to add the trend term and the seasonal term respectively. For future predictions, the series is assumed to continue into the future with slope and seasonal effects given by the final values of the trend and seasonal terms.

You do not need to know how this method works, but for interest, here is the formula for double exponential smoothing (trend term only):

$$\begin{aligned} S_t &= \alpha Y_t + (1-\alpha)(S_{t-1} + B_{t-1}) \\ B_t &= \beta(S_t - S_{t-1}) + (1-\beta)B_{t-1}. \end{aligned}$$

The smoothed slope, B_t, is a weighted average of the most recent smoothed slope, B_{t-1}, and a new slope estimate, $S_t - S_{t-1}$. Seasonal effects can be added similarly, but the formula is more complicated. See the Wikipedia page for 'Exponential Smoothing' if you want to find out more.

Applying the Holt-Winters filter in R

Fitting and plotting the Holt-Winters predictions is very easy in R. First you need to convert your data-frame into a time series as before:

```
beer.df <- read.table("beer.txt", header=T)
beer.ts <- ts(beer.df$beer, start=c(1970, 7), frequency=12)
```

Fit the Holt-Winters model directly from the time series object:

```
beer.hw <- HoltWinters(beer.ts)
Holt-Winters exponential smoothing with trend and additive seasonal component.
Smoothing parameters:
 alpha: 0.1176382
 beta : 0.05545409
 gamma: 0.4410278

Coefficients:
a   14.41166884
b    0.05795753
s1   2.73211059
s2   1.90273318
...      ...
s12  2.41794480
```

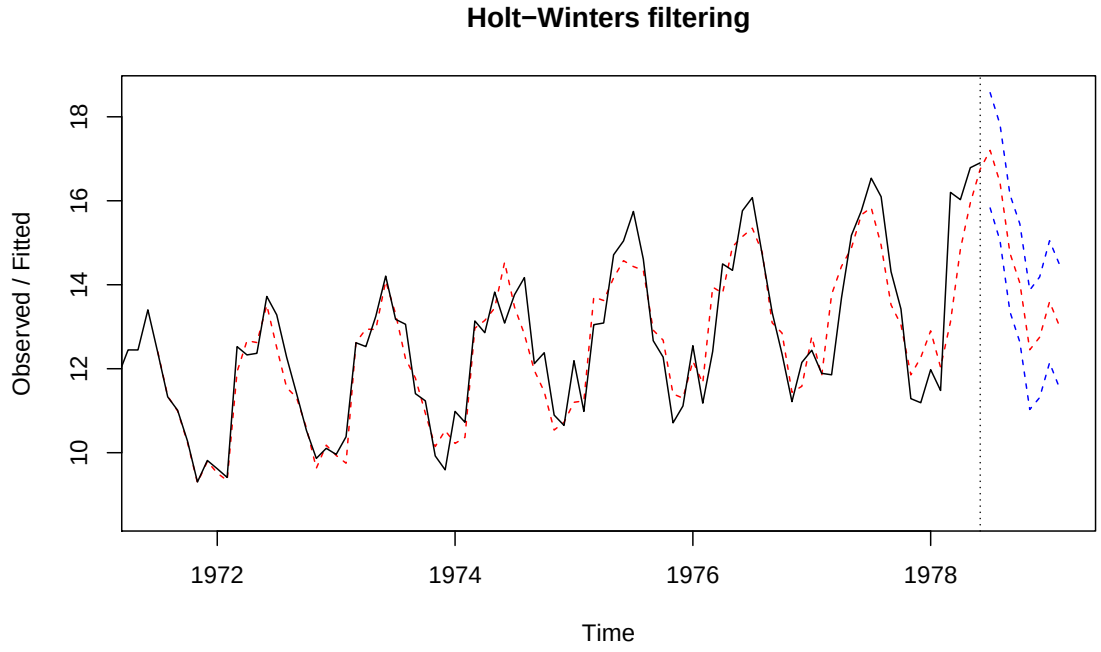
To switch off the seasonal component: `HoltWinters(beer.ts, gamma=FALSE)`.

If you want to switch off both the trend and the seasonal component, use `HoltWinters(beer.ts, beta=FALSE, gamma=FALSE)`.

You can then plot the Holt-Winters model with one very easy command:
`plot(beer.hw)`

However, we usually want to make predictions first before plotting:

```
beer.pred <- predict(beer.hw, n.ahead=8, prediction.interval=T)
beer.pred
               fit      upr      lwr
Jul 1978 17.20174 18.57471 15.82876
Aug 1978 16.43032 17.81384 15.04680
Sep 1978 14.75422 16.14933 13.35912
Oct 1978 14.04175 15.44952 12.63399
Nov 1978 12.45080 13.87233 11.02927
Dec 1978 12.75023 14.18666 11.31381
Jan 1979 13.59643 15.04889 12.14397
Feb 1979 13.01205 14.48172 11.54239

plot(beer.hw, beer.pred, lty.predicted=2, lty.interval=2)
```



Interpretation is very easy. Just remember to interpret the `lwr` and `upr` columns as ***PREDICTION INTERVALS,*** not confidence intervals.

We predict the amount of beer produced in January 1979 would have been between 12.1 and 15.0 million barrels. This prediction spans almost half the range of the data, so it is only moderately useful.

9.4 Autocorrelated regression model

The word *'autocorrelation'* means *self-correlation*. It describes a variable that depends upon itself. This means that the observation at time t, Y_t, depends upon some of the previous observations, $Y_{t-1}, Y_{t-2}, \ldots$.

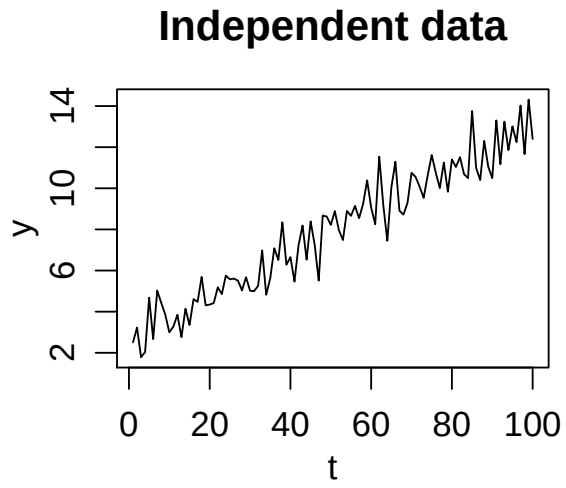
Autocorrelation is a common phenomenon in time series. For example, suppose we are looking at kiwi population numbers over time: Y_t is the number of kiwi at time t. If this year is a very successful breeding year, then we would observe a Y_t value that is higher than expected: ***scatter above the mean.*** However, although this year's high number started as random scatter, its effects will *persist* for future years: more kiwi this year very probably mean more kiwi next year, because this year's large numbers will survive and carry over. Thus, the number of kiwi next year depends in part upon *this year's* random outcome, *as well as* next year's own random scatter. We therefore have ***non-independent errors: our assumption of*** $\epsilon_t \sim$ ***iid Normal***$(0, \sigma^2)$ ***no longer holds.***

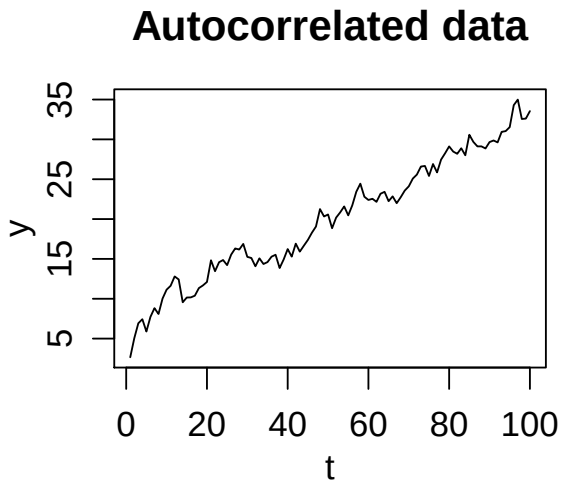
The simplest form of autocorrelation is where Y_t ***depends upon the single previous value,*** Y_{t-1}***.*** This is called ***first-order autocorrelation.***

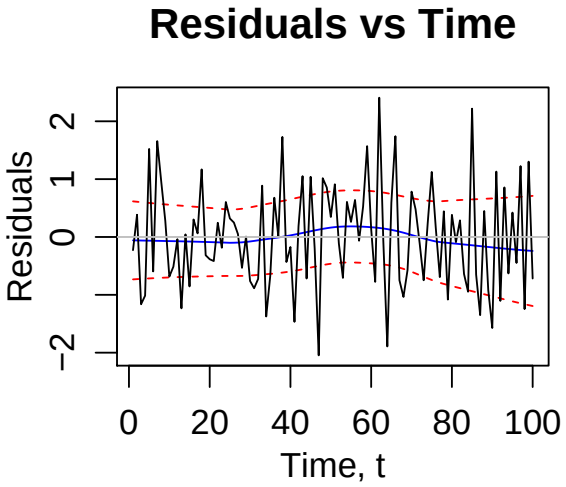
Let's consider how to simulate data with autocorrelation.

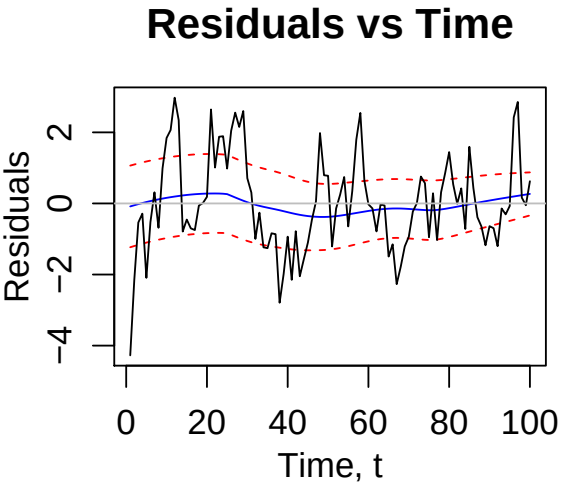
```
## Time, t, is 1 to 100 (measured in days):
t <- 1:100
## Generate response Y with independent errors: Y_t = alpha + beta*t + epsilon_t:
y.ind <- 3 + 0.1*t + rnorm(100, 0, 1)
## Fit the ordinary linear model:
lm.ind <- lm(y.ind ~ t)

## Now generate a new response, Y, with autocorrelation.
## Start by defining a vector of length 100 filled with 0s:
y <- numeric(100)

## The first element, y[1], follows the simple linear model above:
y[1] <- 3 + 0.1*1 + rnorm(1, 0, 1)

## For all subsequent days, the observation depends on what was observed
## the previous day, AS WELL AS the original 3 + 0.1*t + epsilon formula:
for(day in 2:100) y[day] <- 3 + 0.1*day + 0.6*y[day-1] + rnorm(1, 0, 1)

## Fit the ordinary linear model again.
## This is no longer accurate due to non-independent errors:
lm.oops <- lm(y ~ t)
```

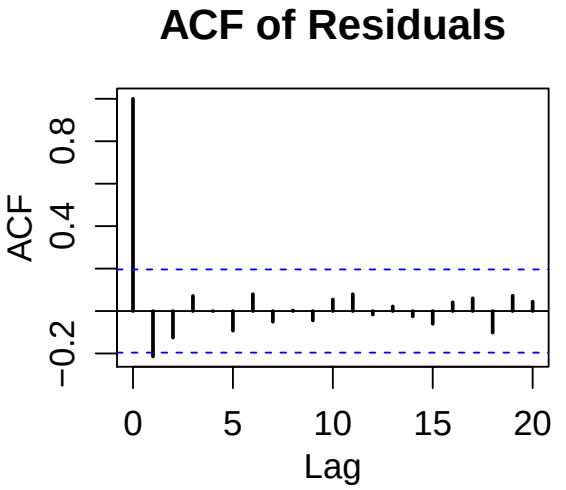
We can now observe how the autocorrelated response Y_t looks different from the independent-error response, $Y_{\text{ind},t}$.

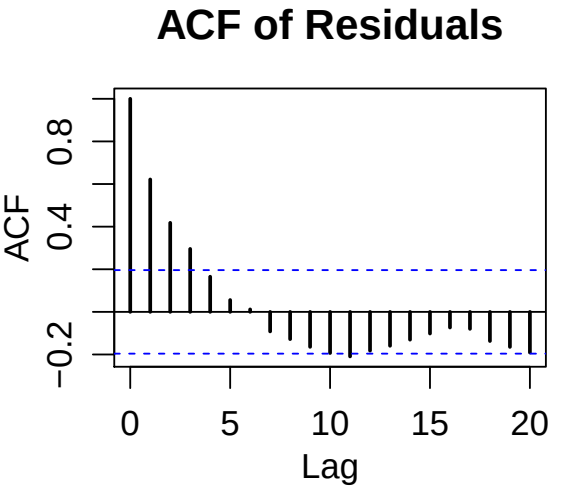












Several features are obvious:

- The autocorrelated response, Y_t, looks ***smoother*** than the independent-error response on the left. (Positive autocorrelation creates a smoothing effect.)
- This same smoothness difference is noticeable in the ***residuals*** of each model, when the residuals are plotted ***against time.***
- The ACF of the independent-errors residuals on the left is ***patternless.*** The ACF of the autocorrelated residuals on the right has ***an oscillating pattern.***

Fitting a model with autocorrelation

Look back to how we simulated autocorrelated data. The model can be written like this:

$$Y_t = \alpha + \beta t + \gamma Y_{t-1} + \epsilon_t, \quad \text{where} \quad \epsilon_t \sim \text{iid Normal}(0, \sigma^2)\,.$$

The errors ϵ_t really are iid. The autocorrelation arises from the introduction of Y_{t-1} ***as a predictor.***

We can easily fit this ***autoregressive model*** in R. We can then estimate the new ***autocorrelation parameter,*** γ***,*** as well as the usual parameters α and β.

Here is the trick. Suppose for a moment that our response y has only ***10 entries.*** We could write these as:

`y:` `y1` `y2` `y3` `y4` `y5` `y6` `y7` `y8` `y9` `y10`

The R syntax `y[-1]`, has the effect of ***omitting the first entry of*** y***:***

`y[-1]:` `y2` `y3` `y4` `y5` `y6` `y7` `y8` `y9` `y10`

Likewise, `y[-10]` ***omits the 10th entry of*** y***:***

`y[-10]:` `y1` `y2` `y3` `y4` `y5` `y6` `y7` `y8` `y9`

So if we use the R syntax `lm(y[-1] ~ y[-10])`, we would obtain the following model:

$$\begin{aligned} y_2 &= \alpha + \beta y_1 + \epsilon_2 \\ y_3 &= \alpha + \beta y_2 + \epsilon_3 \\ &\vdots \qquad\qquad \vdots \\ y_{10} &= \alpha + \beta y_9 + \epsilon_{10}. \end{aligned}$$

This is nearly what we want, but not quite, because we also want the time trend, t. We can include this as usual, but we have to remember to use $t[-1]$ ***in the model, not just*** t***:***

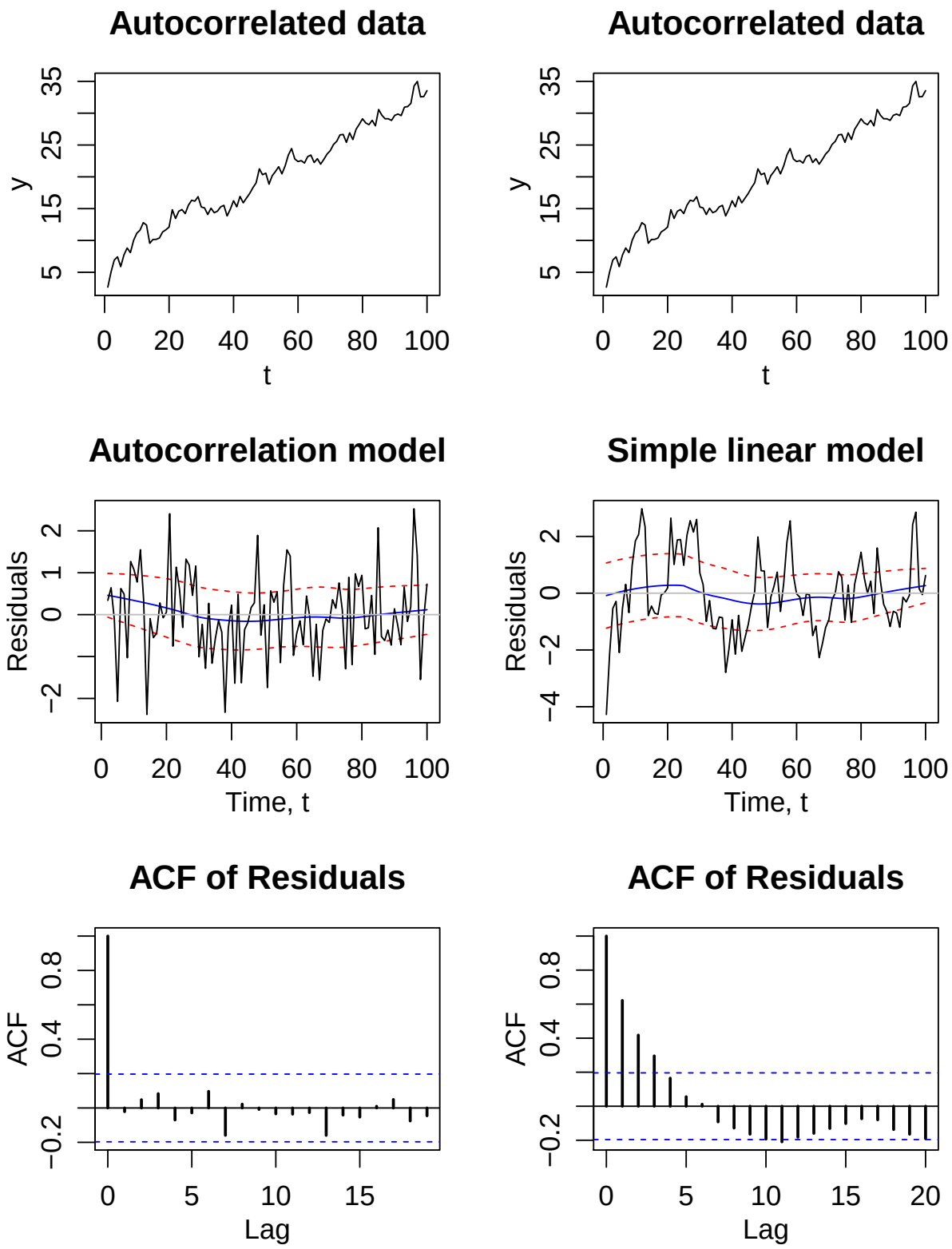
`lm(y[-1] ~ t[-1] + y[-10]):`

$$\begin{cases} y_2 = \alpha + \beta t_2 + \gamma y_1 + \epsilon_2 \\ y_3 = \alpha + \beta t_3 + \gamma y_2 + \epsilon_3 \\ \vdots \qquad\qquad\qquad \vdots \\ y_{10} = \alpha + \beta t_{10} + \gamma y_9 + \epsilon_{10}. \end{cases}$$

In general, if there are n observations in the response, we fit an autoregression model by putting `[-1]` after every predictor and adding a final term `y[-n]`. For example, if the predictors are t and x and the response is y, we would use: `lm(y[-1] ~ t[-1] + x[-1] + y[-n])`.

Observe what happens with the data we simulated earlier. On the left-hand side is output from the new autoregression model: `lm(y[-1] ~ t[-1] + y[-100])`. On the right is output from the naive (incorrect) model we fitted earlier: `lm(y ~ t)`. We see that, with the correct autoregression model on the left, ***the residuals now look random and independent, and the ACF is patternless. The incorrect model on the right is the same as before.***

$\texttt{lm}(y[-1] \sim t[-1] + y[-100])$ $\texttt{lm}(y \sim t)$



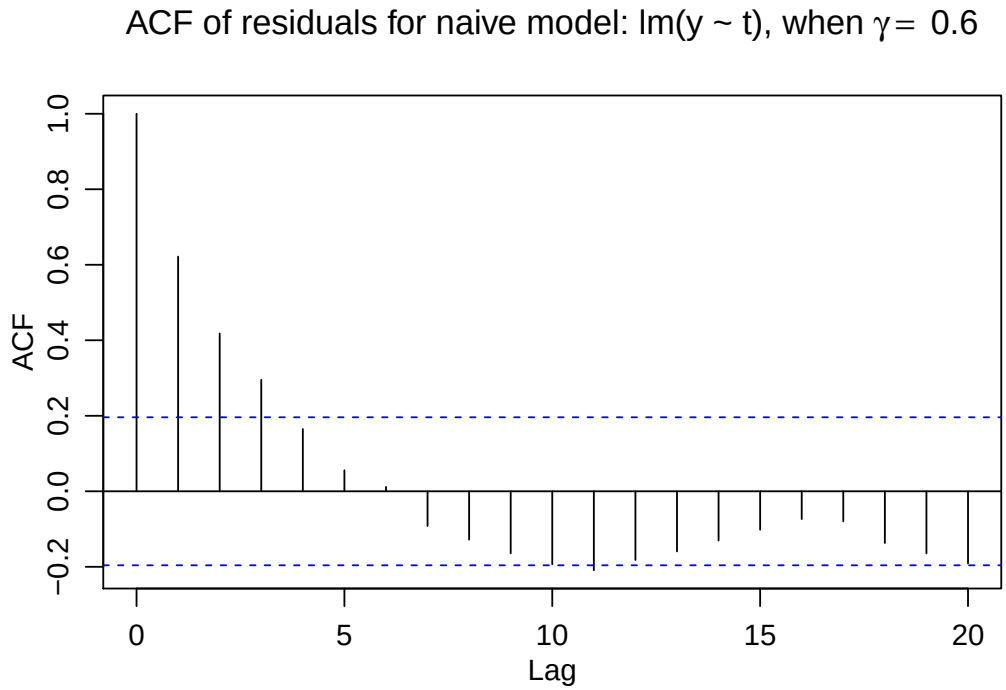
The output from `summary(lm(y[-1]~t[-1]+y[-100]))` shows that all three coefficients are correctly estimated. The true values we used for the simulation on page 9 are $\alpha = 3, \beta = 0.1, \gamma = 0.6$.

```
Call:  lm(formula = y[-1] ~ t[-1] + y[-100])
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.84978    0.51107   5.576 2.26e-07 ***
t[-1]        0.09666    0.01945   4.970 2.92e-06 ***
y[-100]      0.62266    0.07299   8.531 2.12e-13 *** <- Significant autocorrelation
```

- The estimated autocorrelation parameter is ***0.62.***
- The autocorrelation is ***significant (p-value < 0.05). We should retain the autocorrelation term in the model.***

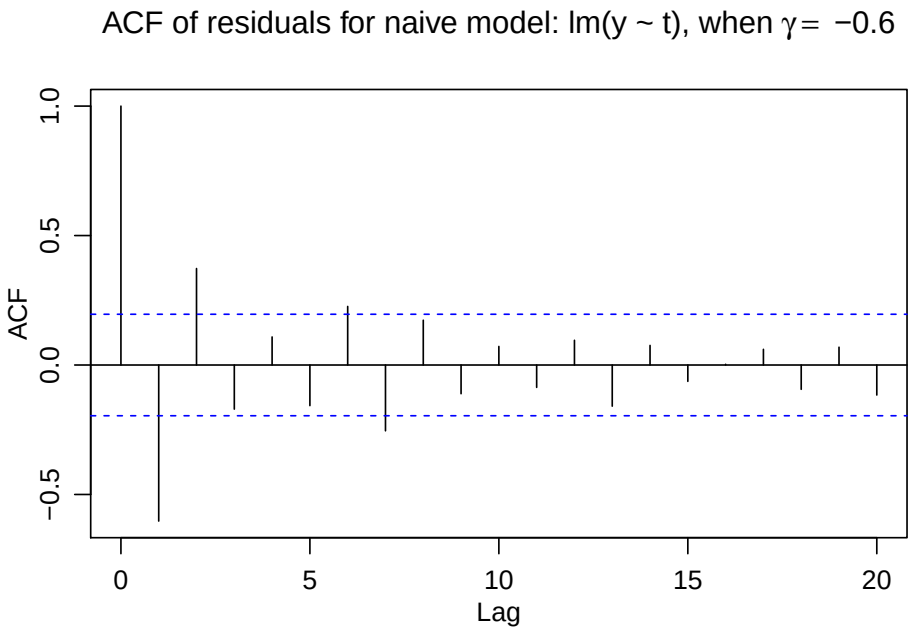
  If the autocorrelation row shows a non-significant result in the summary output, we should ***drop the autocorrelation term and refit the simpler model, lm(y ~ t).***

The ACF plot: AutoCorrelation Function



- The bar at lag 0 shows a perfect association between the residual vector $(r_1, r_2, r_3, \ldots, r_{100})$ and itself: ***bar height = 1.***
- The bar at lag 1 measures the association between $(r_2, r_3, \ldots, r_{100})$ and $(r_1, r_2, \ldots, r_{99})$: ***strong positive association (bar height $\simeq$ 0.6).***
- The bar at lag 2 measures the association between $(r_3, r_4, \ldots, r_{100})$ and $(r_1, r_2, \ldots, r_{98})$: ***a moderate positive association ($\simeq$ 0.4) still carries over at lag 2.***
- The dashed lines show the thresholds for significant autocorrelations.

Here is an ACF chart showing ***negative autocorrelation,*** gained by repeating the simulation on page 9 with $\gamma = -0.6$.

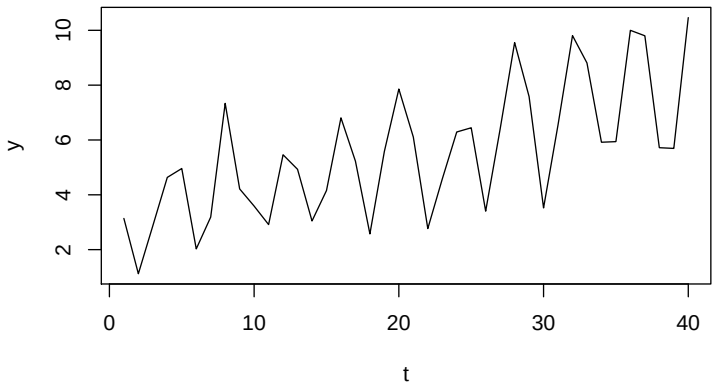


Modelling seasonality

We can include season as a ***factor*** in an autoregressive model. This will enable a different intercept to be fitted for every different season. ***Be careful to convert season number to a factor, e.g. if seasons are numbered as (1, 2, 3, 4), otherwise R will treat it as a numeric variable.***

Here is an example for simulating and fitting a seasonal model.

```
## Generate Y with autocorrelation for t=1, 2, ..., 40:
t <- 1:40
y <- numeric(40)
y[1] <- 3 + 0.1*1 + rnorm(1,0,1)
for(q in 2:40) y[q] <- 3 + 0.1*q
    + 0.1*y[q-1] + rnorm(1,0,1)

## Insert seasonality: R will
## automatically repeat the
## quarterly additions:
y <- y + c(1, -2, -1, 2)
## Define numeric vector "season":
season <- rep(1:4, 40)
```




```
## Fit the autoregressive model using a FACTOR for season:
lm.fit <- lm(y[-1] ~ t[-1] + as.factor(season)[-1] + y[-40])
Coefficients:
                        Estimate Std. Error t value  Pr(>|t|)
(Intercept)              3.48127    0.91139   3.820  0.00056 ***
t[-1]                    0.12162    0.02557   4.757  3.77e-05 ***
as.factor(season)[-1]2 -2.88467    0.45084  -6.398  3.00e-07 ***
as.factor(season)[-1]3 -1.43509    0.81817  -1.754  0.08871 .
as.factor(season)[-1]4  1.39848    0.63249   2.211  0.03407 *
y[-40]                   0.05546    0.17251   0.322  0.74985
```