How good is our straight line?

Bob O'Hara

Contents

How good is my model? **1**
Some Data: Women's times . . . 1

Motivation - is a straight line any good? **3**
A Summary . . . 8

Another View of Regression **8**
How much variation does the model explain? . . . 10
Exercise . . . 11

Regression Assumptions, and how to check them **13**
Residuals . . . 13
Regression Assumptions, Exercise . . . 17
Normal Probability Plots . . . 20
What you can see . . . 25
Normal Probability Plots Exercise . . . 25
Make you own bad data . . . 28
Leverage . . . 32
Leverage Exercise . . . 34
How good is my model? A Summary . . . 37

How can we improve the model? **37**
Box-Cox in R . . . 40
Your Turn . . . 41

Summary **44**

How good is my model?

Begin with the intro video

Click here or watch below

Some Data: Women's times

Here is the data on the winning times from the women's 100m sprint at the Olympics (we will compare this data to the men's times later in the course)

```
Times <- read.csv("https://www.math.ntnu.no/emner/ST2304/2019v/Week5/Times.csv")
WomenMod <- lm(WomenTimes~Year, data=Times)
plot(Times$Year, Times$WomenTimes, lwd=2,
     xlab="Year", ylab="Winning Time")
```

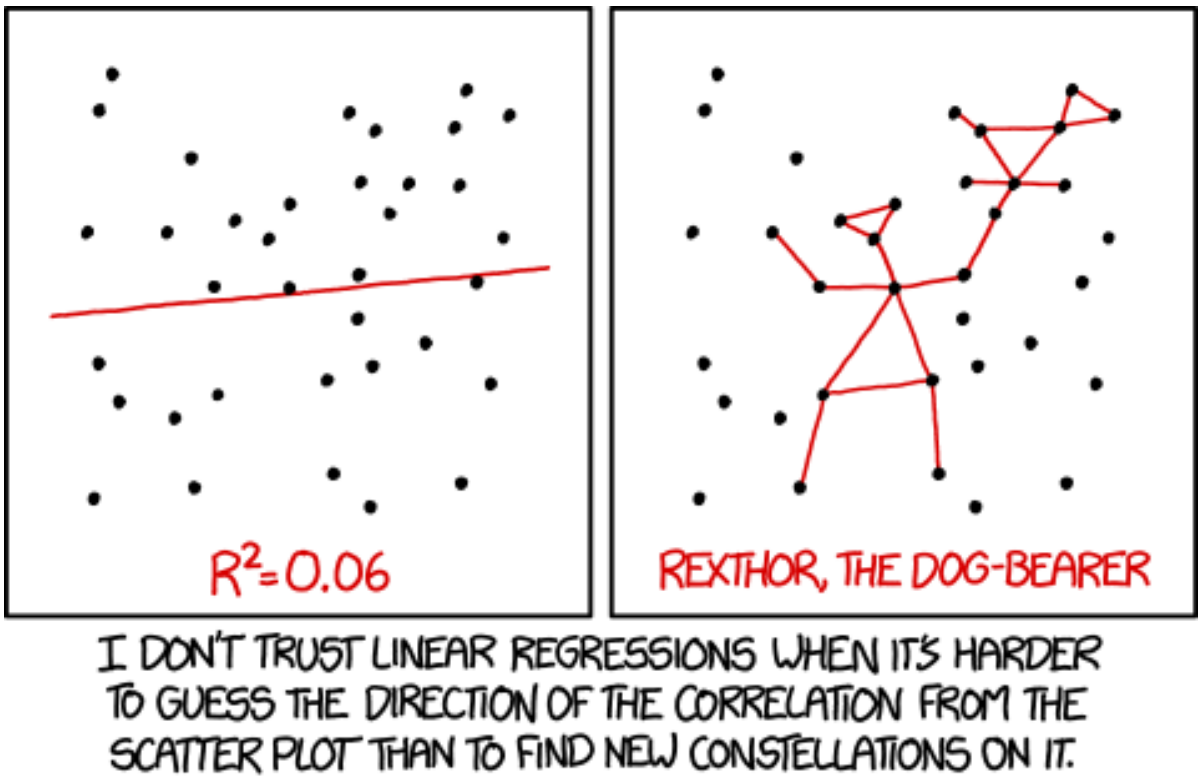
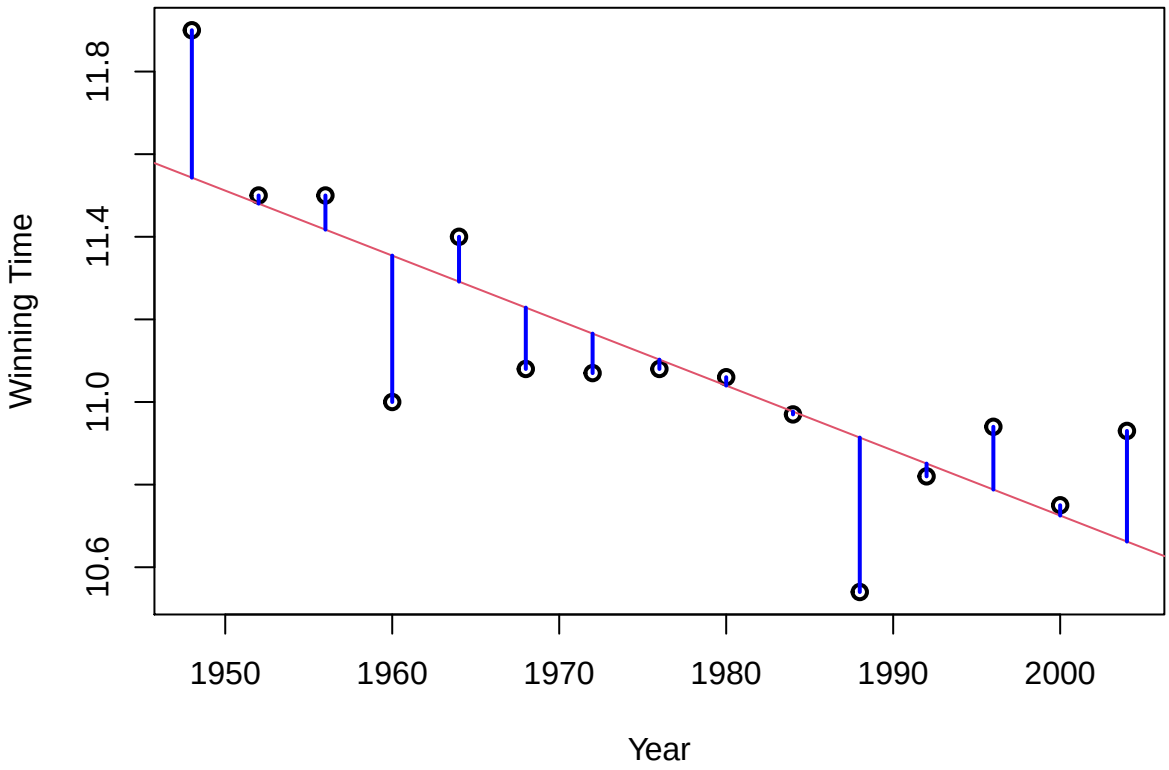


Figure 1: The 95% confidence interval suggests Rexthor's dog could also be a cat, or possibly a teapot.

```
abline(WomenMod, col=2)
segments(Times$Year, fitted(WomenMod), Times$Year, Times$WomenTimes, col="blue", lwd=2)
```

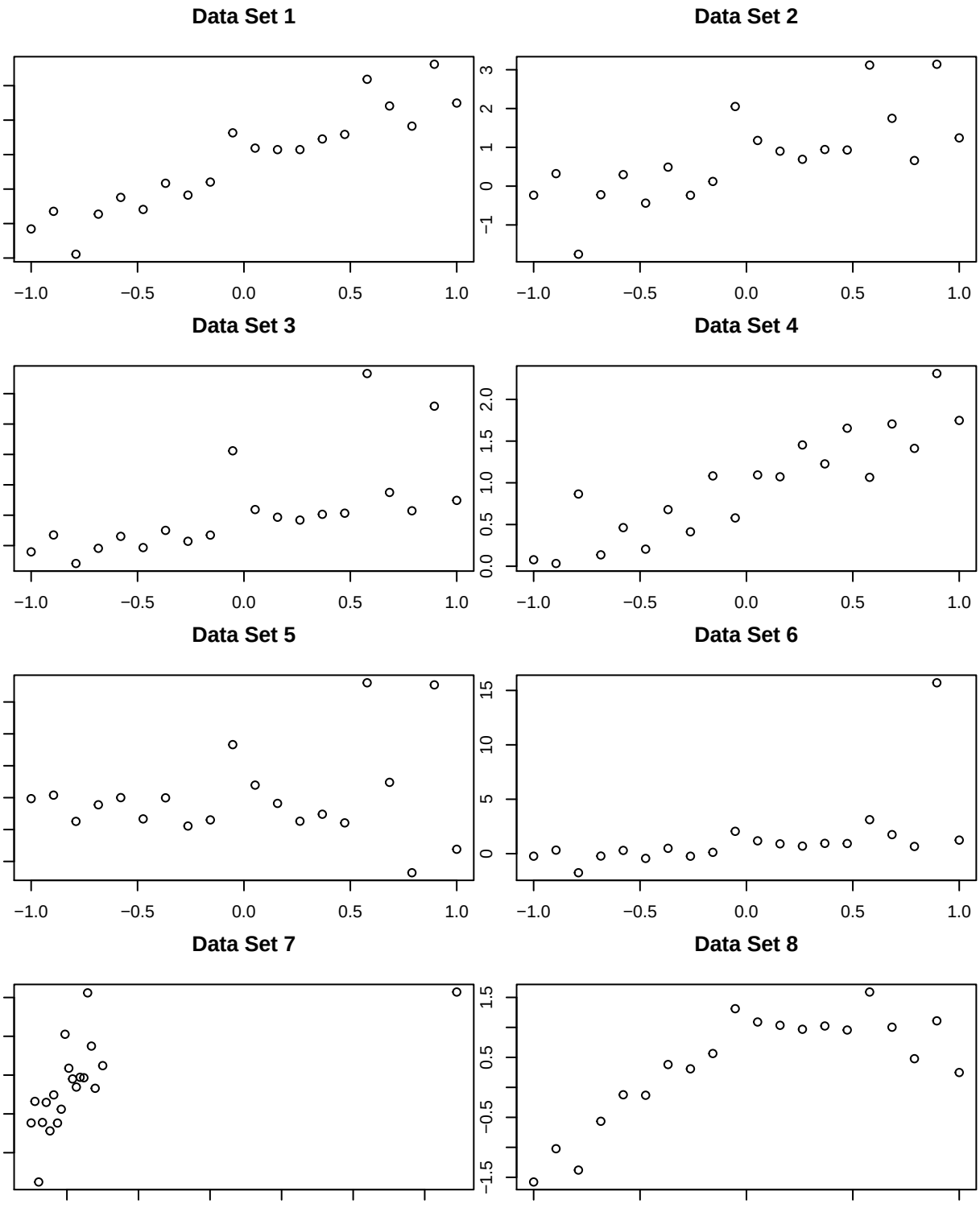


A lot of this module revolves around the fitted and

Motivation - is a straight line any good?

Here are some simulated data sets. For all of them I used the the same errors, but manipulated the data in different ways. For each one, you should decide

- if you think a straight line would be a good fit to the data, and
- if it is not, can you do something simple to improve the fit? (for some you cannot, for some you can)



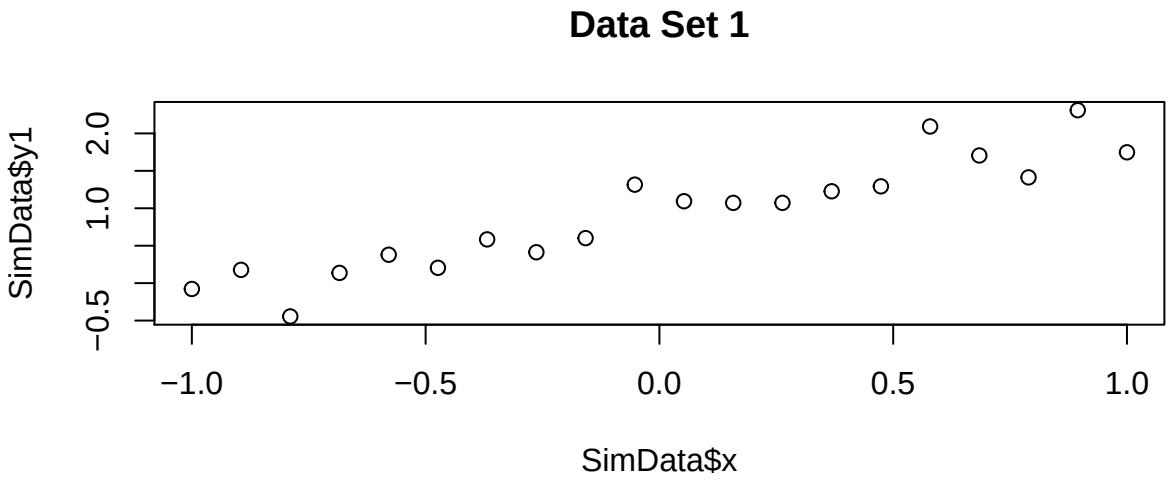
Hint

Try mentally adding a straight line.

For a couple, the straight line is OK, but other aspects are problematic.

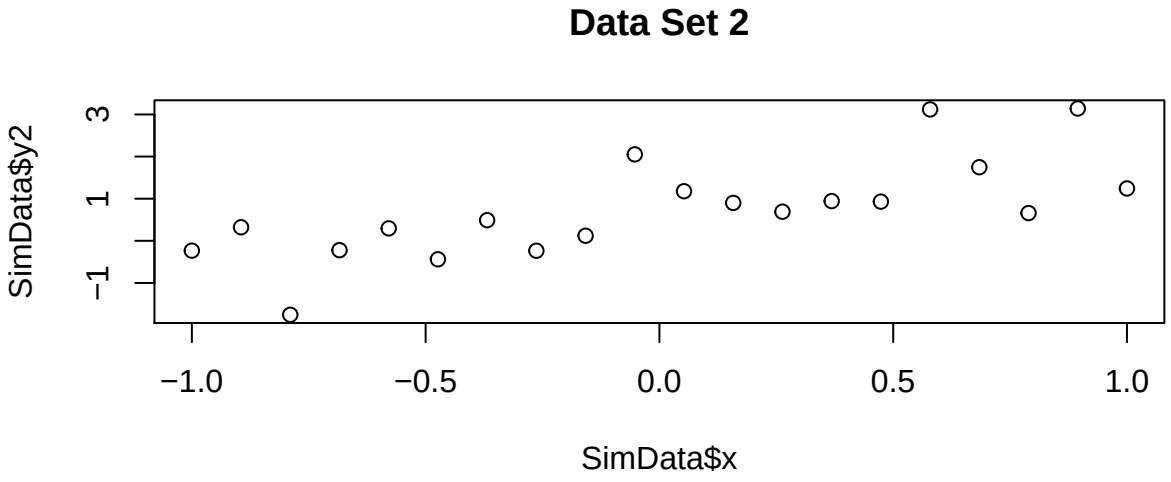
Answers

```
plot(SimData$x, SimData$y1, main="Data Set 1")
```



Data set 1 looks fine: a straight line should work well.

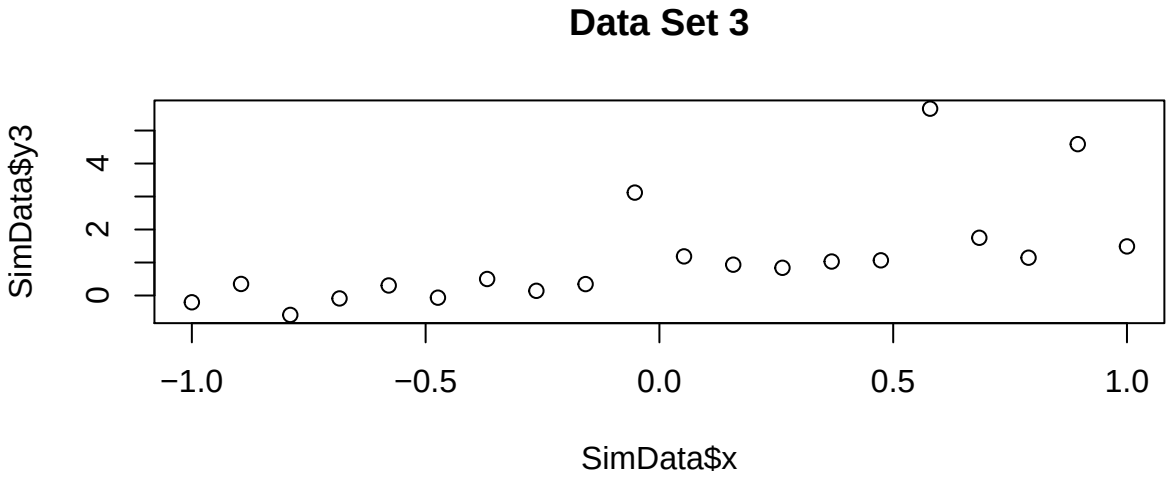
```
plot(SimData$x, SimData$y2, main="Data Set 2")
```



Data set 2 also looks fine: a straight line should work well, but not as well as data set 1.

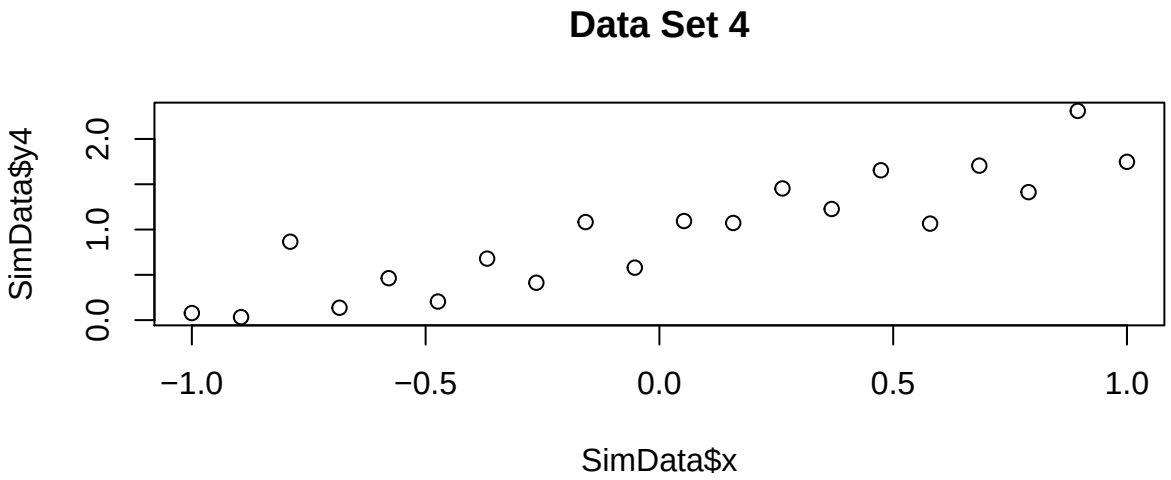
(if you look carefully you might be able to work out how I generated data set 2 from data set 1)

```
plot(SimData$x, SimData$y3, main="Data Set 3")
```



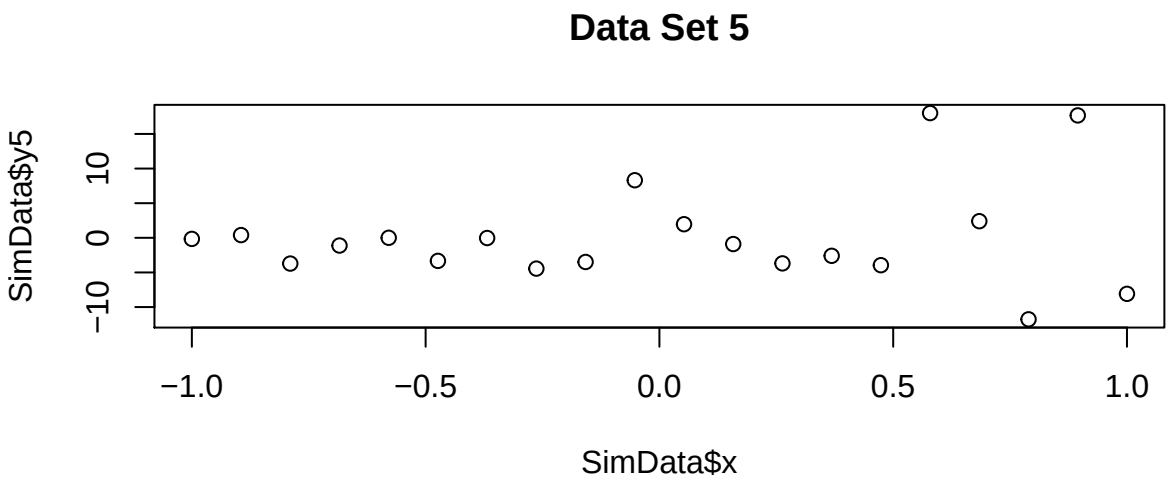
Data set 3 looks OK on the whole, but there are 3 points that are higher than we might expect. We will come back to these later.

```
plot(SimData$x, SimData$y4, main="Data Set 4")
```



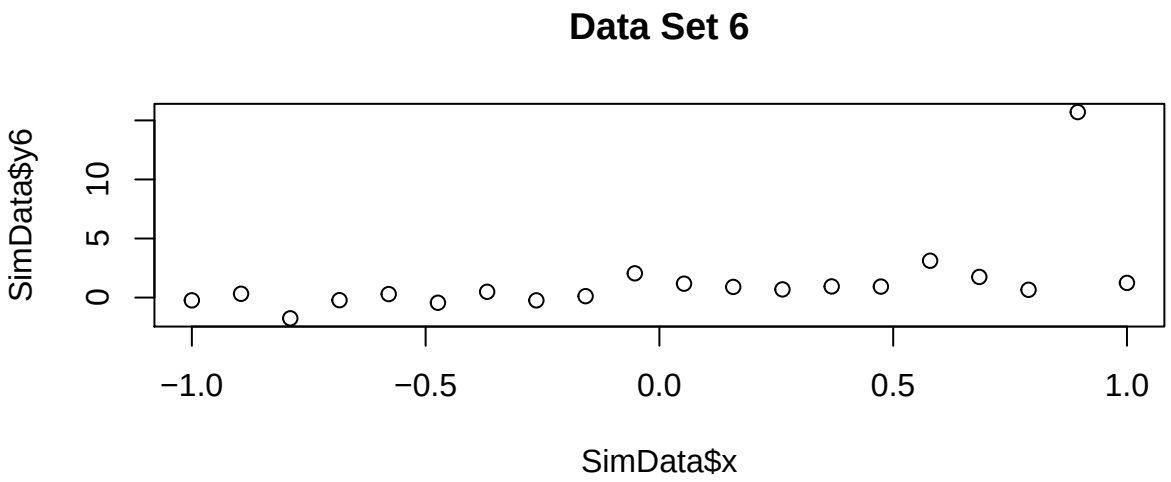
A straight line looks good for Data Set 4. There is something odd about this data, but it might be difficult to spot.

```
plot(SimData$x, SimData$y5, main="Data Set 5")
```



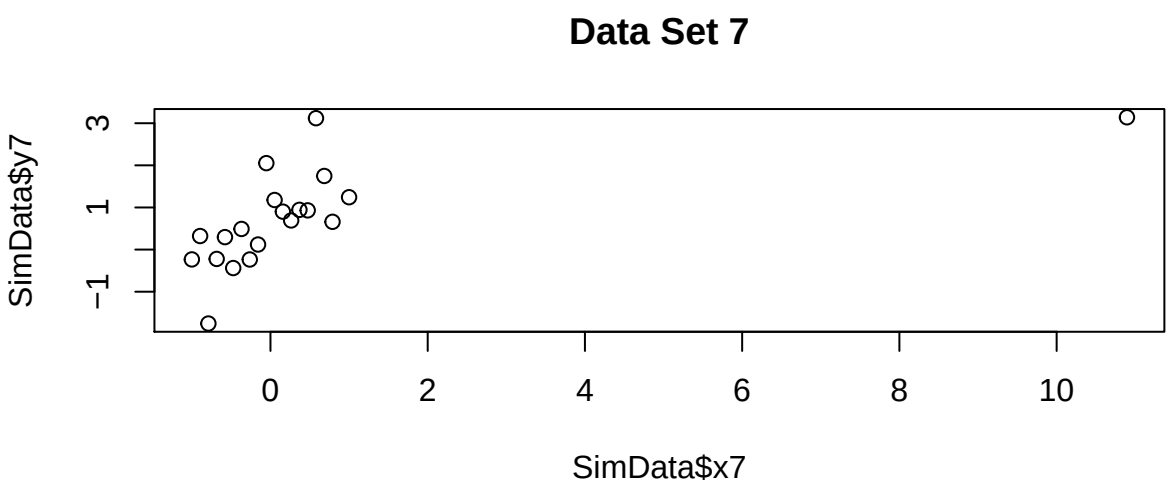
A straight line is OK for data Set 5, but the variance seemt to increase as we move from left to right.

```
plot(SimData$x, SimData$y6, main="Data Set 6")
```



A straight line is fine for most of data set 6, except for that point on the right with a very large y value. We might want to think about removing it to improve the fit (or checking that it isn't a mistake).

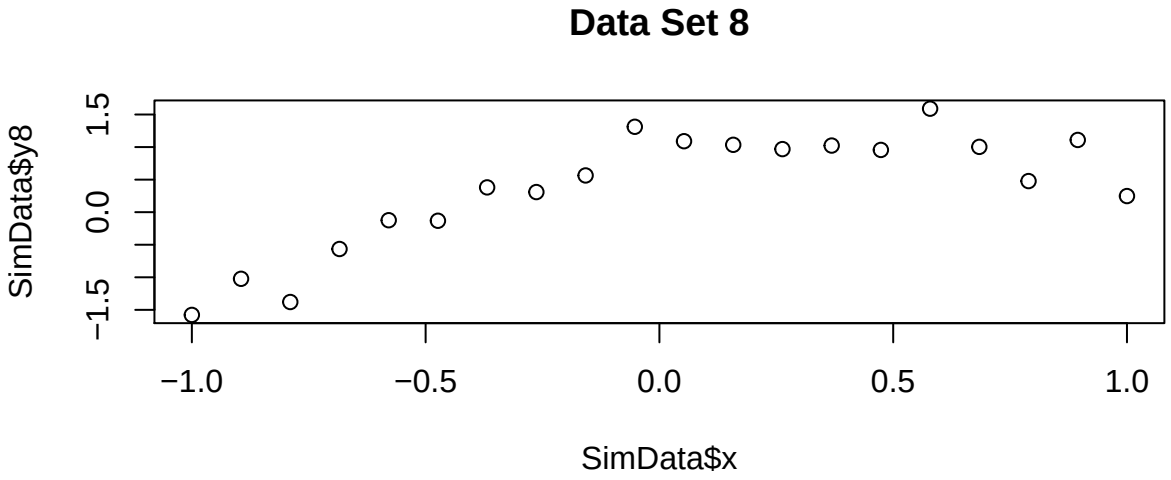
```
plot(SimData$x7, SimData$y7, main="Data Set 7")
```



A straight line is fine for most of data set 7, except for that point on the right with a very large x value. We might want to think about removing it to improve the fit (or checking that it isn't a mistake).

The difference between this and data set 6 means that we might need different tools to find each problem.

```
plot(SimData$x, SimData$y8, main="Data Set 8")
```



A straight line is not a good fit for data set 8. There are a couple of possible solutions: one is not to use x, but x^2 in the regression.

A Summary

The simulated data above suggest that simply fitting a straight line is not always the best idea. The data might not fit it, or there might be strange values, or other oddities in the data. When we have one X and one Y, plotting them is helpful for seeing a lot of problems, but later we will have several X's, and these can hide problems. So we need some tools to pick apart the model and data, so we can find out how well the model is describing the data.

Another View of Regression

We can look on the model as being made up of two parts: the systematic part (which is being explained by covariates - our X's), and the random part (which is the normally distributed error). For a simple regression the two parts look like this:

$$\begin{aligned} y_i &= \mu_i &&+\varepsilon_i \\ &= \alpha + \beta x_i &&+\varepsilon_i \end{aligned}$$

- Systematic part of model: a straight line
- Random part of model: residual error

So, for the Women's Times data, we fit the model like this, and get the coefficients:

```
WomenMod <- lm(WomenTimes~Year, data=Times)
coef(WomenMod)
```

```
## (Intercept)        Year
## 42.18938095 -0.01573214
```

So the model is

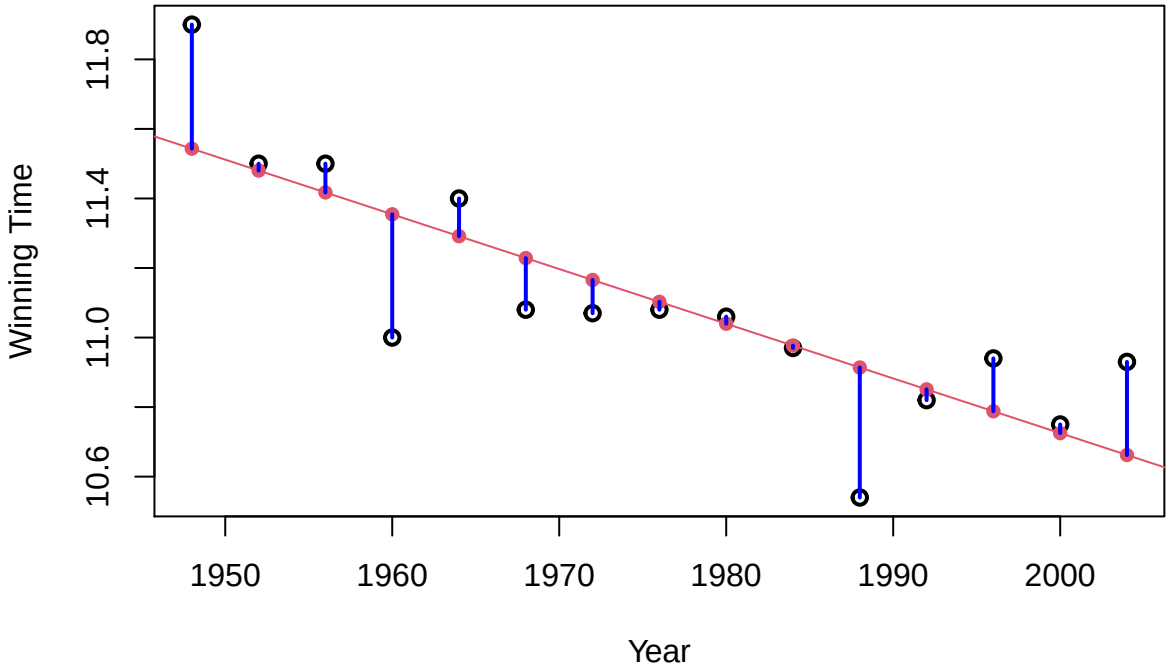
$$y_i = 42.19 - 0.016x + \varepsilon_i$$

The systematic part is

$$E(y_i) = 42.19 - 0.016x$$

Which we call the fitted line, and for the data we can get this in R using `fitted()`. The random part is ε_i.

```
plot(Times$Year, Times$WomenTimes, lwd=2,
     xlab="Year", ylab="Winning Time")
abline(WomenMod, col=2) # add fitted line
points(Times$Year, fitted(WomenMod), col="2", pch=16) # add fitted values
# now add the residuals
segments(Times$Year, fitted(WomenMod), Times$Year, Times$WomenTimes, col="blue", lwd=2)
```



For example, if we look at the year 1964, we see the winning time was 11.4. From the model, the fitted value (i.e. the expected time) is

$$E(y_i) = 42.19 - 0.016 \times 1964 = 11.29$$

The residual is the difference between the data and its fitted value, so is

$$11.4 - 11.29 = 0.11$$

So the winner (Wyomia Tyus, who was 19 at the time) won in a time that was 0.11s slower than the model predicts.

R has functions to calculate these numbers. The functions `fitted()` and `resid()` calculate the fitted values and residuals for all of the data. Here we just extract the values for 1964 from each vector:

```
fitted(WomenMod)[Times$Year==1964]
```

```
##        5
## 11.29145
```

```
resid(WomenMod)[Times$Year==1964]
```

```
##         5
## 0.1085476
```

```
# This should equal the data, 11.4s
fitted(WomenMod)[Times$Year==1964] + resid(WomenMod)[Times$Year==1964]
```

```
##    5
## 11.4
```

So, this is how to calculate fitted values and residuals. All of the models we will see have this general form, but both parts can be much more complicated. For a good model the systematic part, which gives the fitted values, should explain a lot of the variation in the data. The random part of the model - the residuals - should be random. This means there shouldn't be any structure in it.

How much variation does the model explain?

The total variation is

$$\begin{aligned}\text{Var}(y_i) &= \text{Var}(\alpha + \beta x_i) + \text{Var}(\varepsilon_i) \\ &= \beta^2 \text{Var}(x_i) + \sigma^2\end{aligned}$$

- σ^2 is the residual variation. We can ask how much of the total total variation is explained by the model. We do this by calculating the proportion of the total variation explained by the model

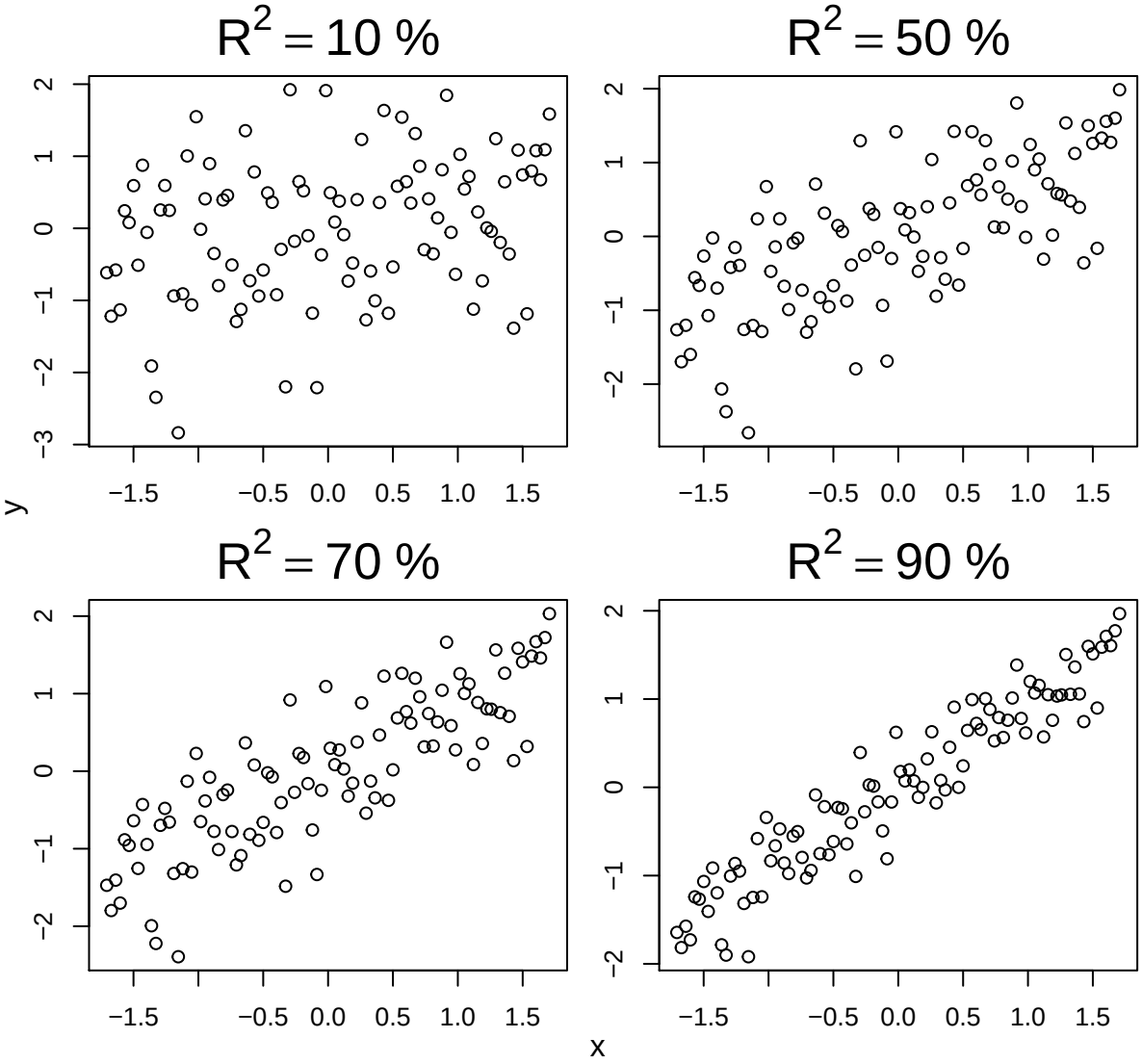
$$R^2 = \frac{\text{Variance Explained}}{\text{Total Variance}} = 1 - \frac{\text{Residual Variance}}{\text{Total Variance}}$$

After a bit of maths, we get

$$R^2 = 1 - \frac{\sum(y_i - \mu_i)^2}{\sum(y_i - \bar{y})^2}$$

where $\sum(y_i - \mu_i)^2$ is the residual variance (i.e. the squared difference from expected value), and $\sum(y_i - \bar{y})^2$ is the total variance (i.e. the squared difference from grand mean). We usually write R^2 as a percentage, so we mutiply this by 100.

An obvious question is what is a good R^2? Unfortunately, there is no simple answer. In general, a value of 4% is bad, and 99% good, but the context is important. 30% is usually not great - most of teh variation is random, and 70% is usually good. But in the laboratory there should be less variation, so 70% might still not be great (especially if you have designed an experiment where there should be a huge effect).



It might be that this is the best we can do, because sometimes data are just very noisy. But it might be that we can do better. R^2 can't tell us whether there is a better model, but we have other tools to do that. It is useful, though, in summarising how well our model is explaining the data.

R^2 is easy to extract from R. It is calculated automatically as part of the `summary()`, so we can get it from this:

```
R2 <- summary(WomenMod)$r.squared
R2
```

```
## [1] 0.6723703
```

```
round(100*R2, 1)
```

```
## [1] 67.2
```

Exercise

Exercise: calculate the R^2 for the 8 plots. Do some seem better than others?

You will need to read in the data, and fit the models. Note that `x` is the same for all `y`'s *except* `y7`. Here is some code to get you started:

```
Data <- read.csv("https://www.math.ntnu.no/emner/ST2304/2019v/Week6/SimRegression.csv")
mod1 <- lm(y1 ~ x, data=Data) # This works for y1, ..., y6, y8
mod7 <- lm(y7 ~ x7, data=Data) # for y7 we need to use x7
```

Hint

```
Data <- read.csv("https://www.math.ntnu.no/emner/ST2304/2019v/Week6/SimRegression.csv")
mod1 <- lm(y1 ~ x, data=Data) # This works for y1, ..., y6, y8
mod7 <- lm(y7 ~ x7, data=Data) # for y7 we need to use x7

summary(mod1)$r.squared
```

```
## [1] 0.8708701
```

Answers

These are the R^2 values. Data set 5 has the worst R^2, which is not surprising if you look at the plots. Even Data set 8, for which the fit is bad, the R^2 is pretty good, at 59.1%

```
round(100*summary(lm(y1 ~ x, data=Data))$r.squared, 1)
```

```
## [1] 87.1
```

```
round(100*summary(lm(y2 ~ x, data=Data))$r.squared, 1)
```

```
## [1] 52.5
```

```
round(100*summary(lm(y3 ~ x, data=Data))$r.squared, 1)
```

```
## [1] 41.5
```

```
round(100*summary(lm(y4 ~ x, data=Data))$r.squared, 1)
```

```
## [1] 80.1
```

```
round(100*summary(lm(y5 ~ x, data=Data))$r.squared, 1)
```

```
## [1] 2.3
```

```
round(100*summary(lm(y6 ~ x, data=Data))$r.squared, 1)
```

```
## [1] 26
```

```
round(100*summary(lm(y7 ~ x7, data=Data))$r.squared, 1)
```

```
## [1] 36.9
```

```
round(100*summary(lm(y8 ~ x, data=Data))$r.squared, 1)
```

```
## [1] 59.1
```

For those who want to be fancy, here's another way of coding this. The advantage is that the models are all in a list, so we can use `lapply()` on the list to extract values like R^2:

```
Models <- list(mod1 = lm(y1 ~ x, data=Data),
               mod2 = lm(y2 ~ x, data=Data),
               mod3 = lm(y3 ~ x, data=Data),
               mod4 = lm(y4 ~ x, data=Data),
               mod5 = lm(y5 ~ x, data=Data),
               mod6 = lm(y6 ~ x, data=Data),
               mod7 = lm(y7 ~ x7, data=Data),
```

```
                mod8 = lm(y8 ~ x, data=Data))
Rsq <- round(100*unlist(lapply(Models, function(mod) summary(mod)$r.squared)), 1)
```

For this course, there is no need to use this level of computing.

Regression Assumptions, and how to check them

Regression model make a number of assumptions:

- straight line
- errors are independent
- errors have the same variance
- errors are normally distributed
- errors have zero mean

The final assumption is forced by the maximum likelihood estimation, so we don't need to worry about it.

Model is systematic part + random part

$$\begin{aligned} y_i &= \mu_i &+\varepsilon_i \\ &= \alpha + \beta x_i &+\varepsilon_i \end{aligned}$$

We want to randomness to be be in the random part, and for this to follow a normal distribution, and for the systematic part to contain the non-random structure in the data. If we are to check these assumptions, we need some tools.

Residuals

Click here or watch below

The most basic tools for looking at how well the model is meeting the assumptions is the concept of the residual. The model for the data is $y_i = \alpha + \beta x_i + \varepsilon_i$, and we estimate this with the fitted model:

$$y_i = \hat{\alpha} + \hat{\beta} x_i + e_i$$

$\hat{\alpha}$ and $\hat{\beta}$ are the parameter estimates, so $\hat{\alpha} + \hat{\beta} x_i$ is the fitted value for y_i, i.e. what we would expect if there was no randomness. e_i are the ***residuals***: these are estimates of ε_i. The data can be looked on as being

$$\text{Data} = \text{Fitted} + \text{Residual}$$

We sometimes standardise the residuals, to make it easier to interpret their sizes: $t_i = \frac{r_i}{\sqrt{Var(r_i)}}$. We can extract residuals from R using the `residuals()`function:

```
Women.res <- residuals(WomenMod)
round(Women.res, 2)[1:5]
```

```
##     1     2     3     4     5
##  0.36  0.02  0.08 -0.35  0.11
```

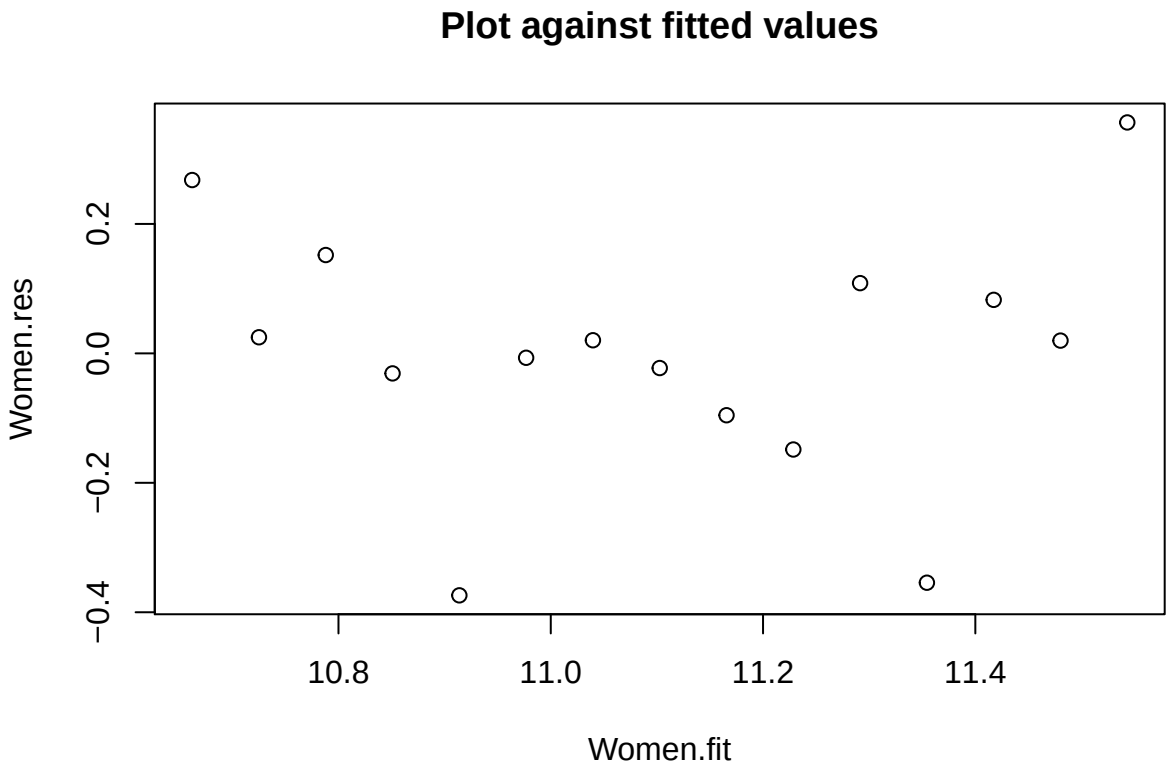
We want to know if they are normally distributed, and also that there is no structure in them that should be in the fitted values. Although we can stare at them or do some tests, it is more informative if we to plot them in different ways. One helpful plot is to plot them against the fitted values (i.e. $\mu_i = \hat{\alpha} + \hat{\beta} x_i$). We can get the fitted values from a model in R using the `fitted()`function:

```
Women.fit <- fitted(WomenMod)
round(Women.fit, 2)[1:5]
```

```
##     1     2     3     4     5
## 11.54 11.48 11.42 11.35 11.29
```

Now we can plot the residuals against the fitted values:

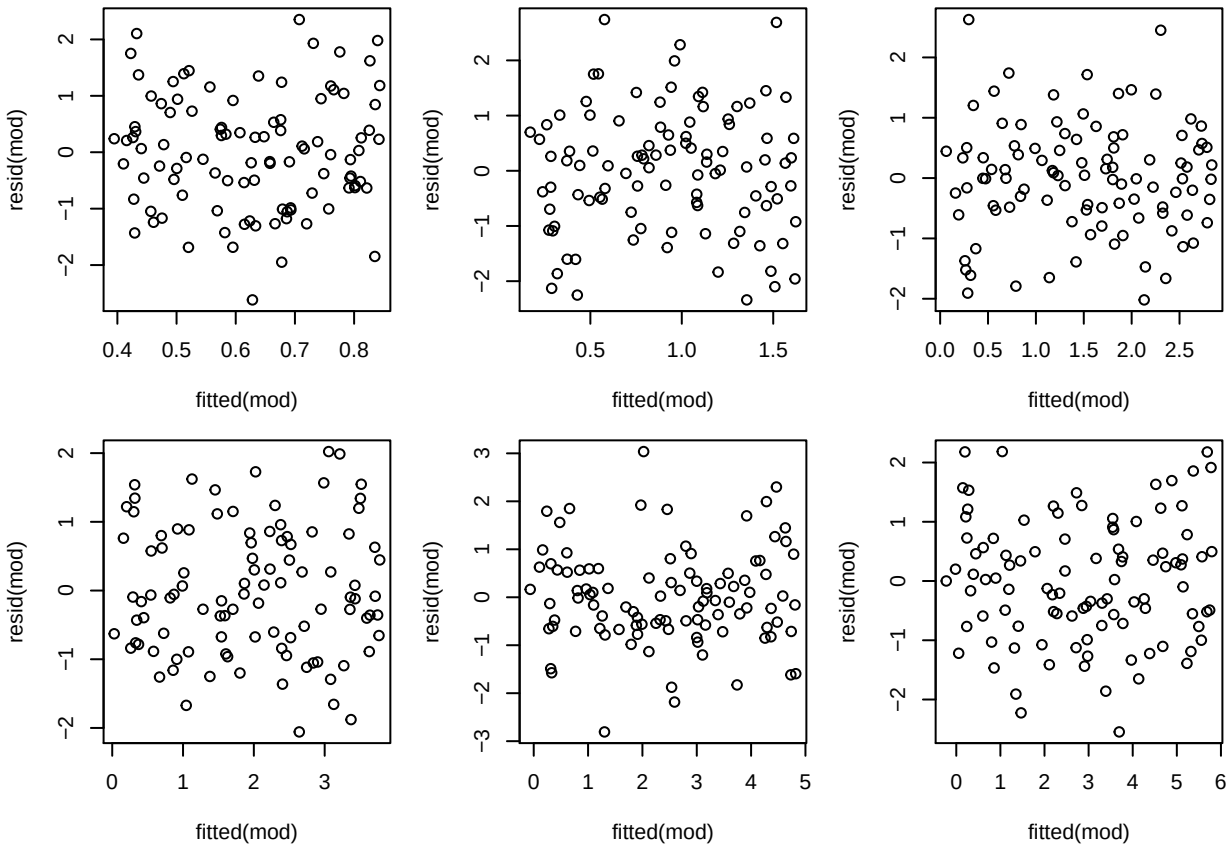
```
plot(Women.fit, Women.res, main = "Plot against fitted values")
```



Residuals should have no structure, beyond being normally distributed[1]. So, we can check for structure to check if the residuals are good. Here are some "good" residual plots:

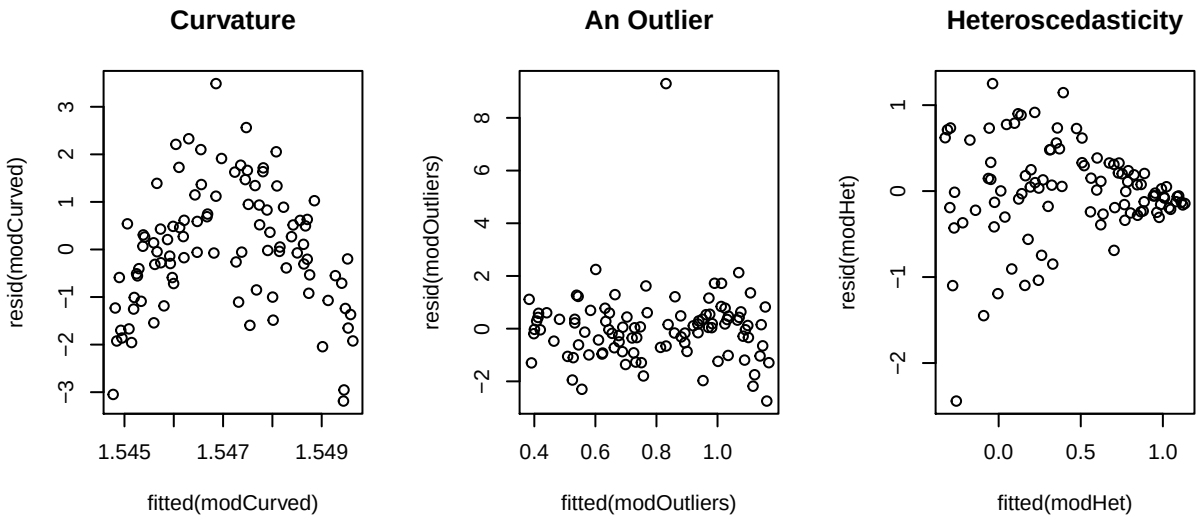
```
N <- 100; NSims=6
X <- runif(N)
GoodY <- X%o%c(1:NSims) + matrix(rnorm(NSims*N), ncol=NSims)
mods <- apply(GoodY, 2, function(y, x) lm(y~x), x=X)

par(mfrow=c(2,NSims/2), mar=c(4.1,4.1,1,1))
lapply(mods, function(mod) plot(fitted(mod), resid(mod)))
```

[1] this is not quite true: because we are estimating them, the estimation induces a slight correlation. But this is almost universally ignored in practice



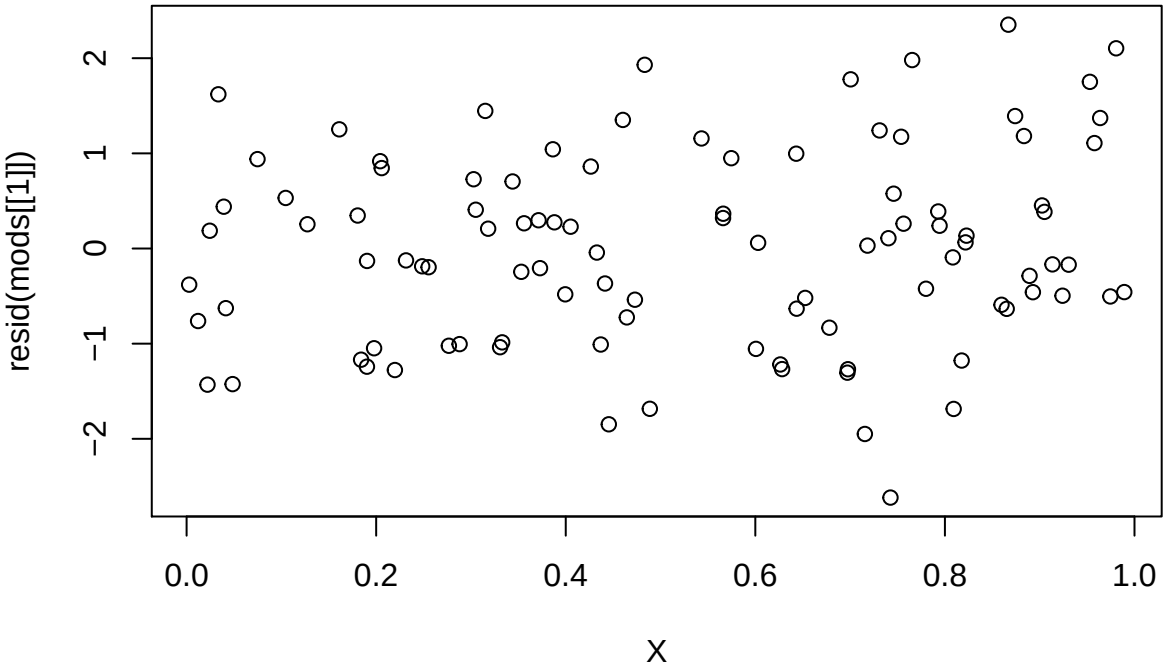
Here are sopme that are not so good: wecan see structures such as curvature, outliers and heteroscedasticity (variance changing).

```
N <- 100
X <- runif(N)
CurvedY <- rnorm(N, 10*X - 10*X^2, 1)
OutliersY <- rnorm(N, X, 1); OutliersY[5] <- OutliersY[5]+10
HetY <- rnorm(N, X, 1.01-X)

modCurved <- lm(CurvedY ~ X)
modOutliers <- lm(OutliersY ~ X)
modHet <- lm(HetY ~ X)

par(mfrow=c(1,3))
plot(fitted(modCurved), resid(modCurved), main="Curvature")
plot(fitted(modOutliers), resid(modOutliers), main="An Outlier")
plot(fitted(modHet), resid(modHet), main="Heteroscedasticity")
```



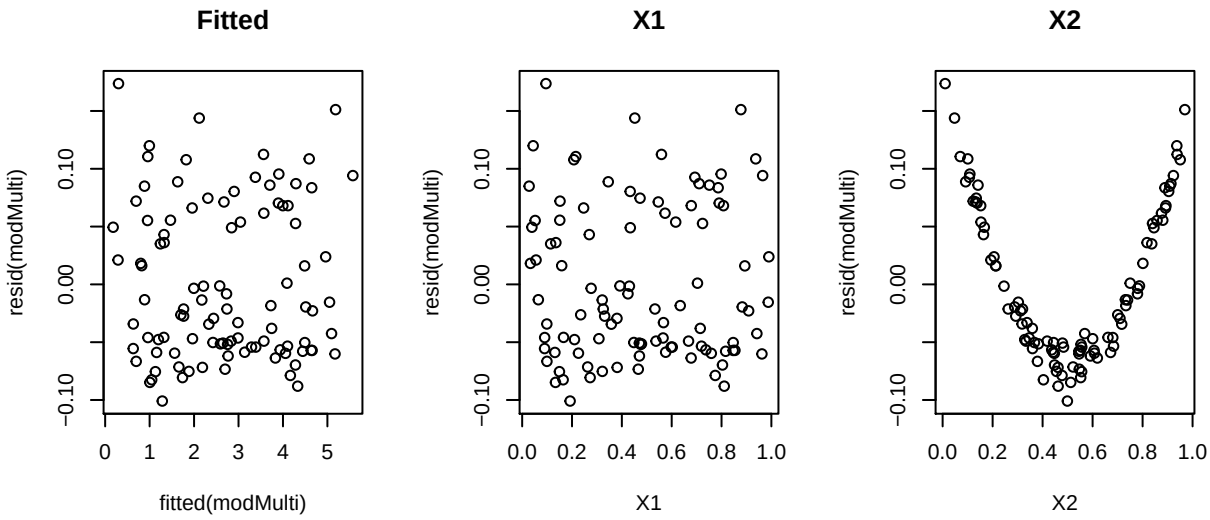
We can also plot the residuals against the covariate:

```
plot(X, resid(mods[[1]]))
```



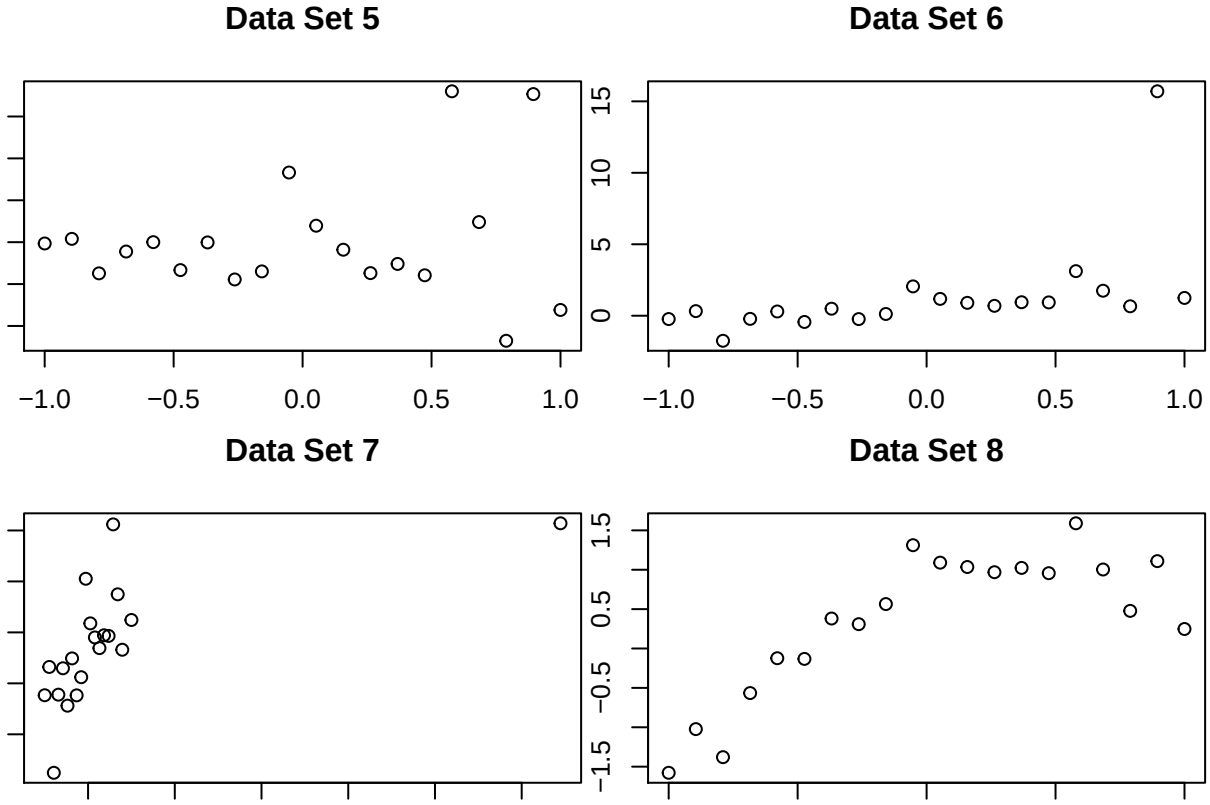
When we only have one covariate this does not look much different to plotting against the fitted valiues, but it does makea difference when we look at multiple regression later. As a taster, here's an example when plotting agaisnt the fitted values looks OK, but one of the covaraites has a non-linear effect.

```
X1 <- runif(N)
X2 <- runif(N)
Ym <- rnorm(N, 5*X1 + X2^2, 0.01)
modMulti <- lm(Ym ~ X1 + X2)

par(mfrow=c(1,3))
plot(fitted(modMulti), resid(modMulti), main="Fitted")
plot(X1, resid(modMulti), main="X1")
plot(X2, resid(modMulti), main="X2")
```



Regression Assumptions, Exercise

For data sets 5 - 8, which assumption is wrong? Here are those data sets again:



Plot the residuals against the fitted values for all 8 plots.

- For which data do they suggest a problem?
- What is the problem?
- Can you think of ways to improve these models?
  - no, you haven't been given the tools yet! So you can be creative

Hint

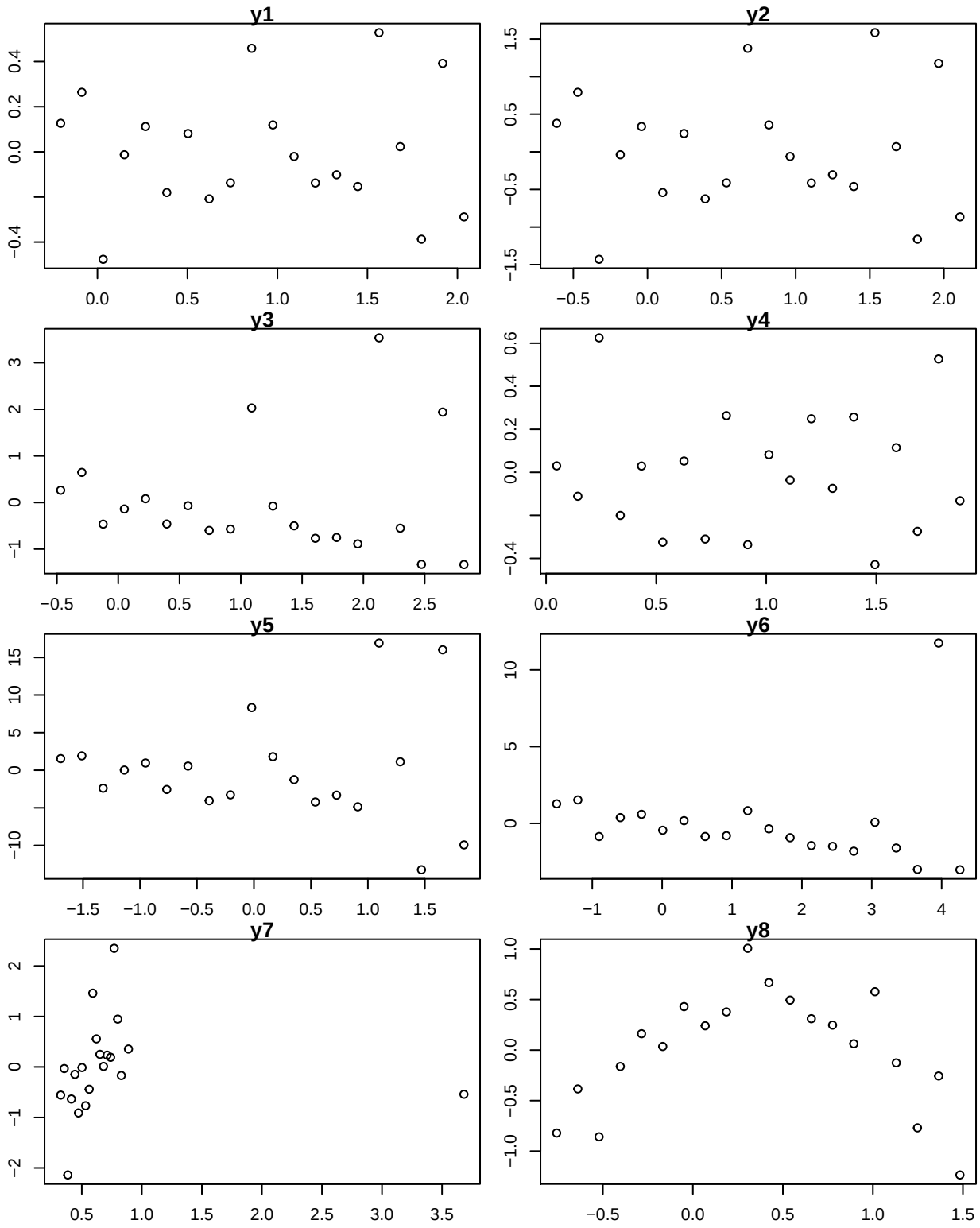
The code should look like this (but not with `mod`, instead with the object you got from your `lm()` function):

```
plot(fitted(mod), resid(mod))
```

Answers

So, here I used `lapply()` to shorten the code.

```
par(mfrow=c(4,2), mar=c(2,2,1,1))
lapply(Models, function(mod) {
  plot(fitted(mod), resid(mod), main=names(mod$model)[1])
})
```



We can see that models 1, 2 and 4 look fine.

Models 3 and 5 might be heteroscedastic (5 is, 3 has 3 large positive residuals). There are a few things that could be done, including fitting a model where we model the varaince. Or read on.

Model 6 has an obvious outlier. And model 7 has that annoying point on the right. These could both be

removed with extreme prejudice.

The residuals for Model 8 look unhappy: they are obviously curved. So fitting a curved line of some sort is a good idea.

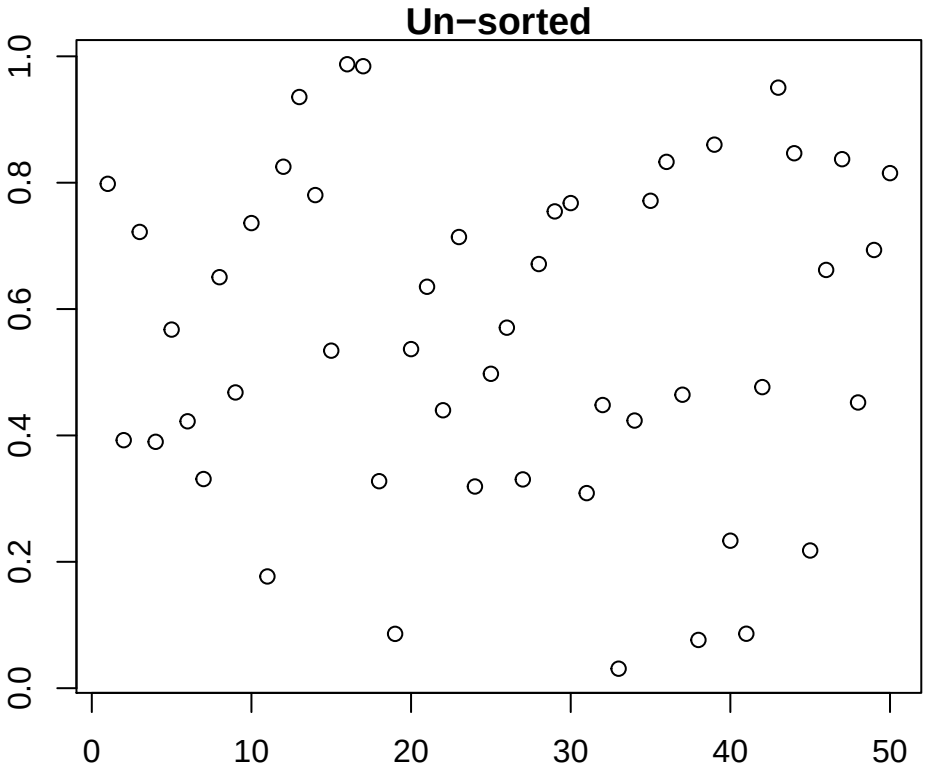
Normal Probability Plots

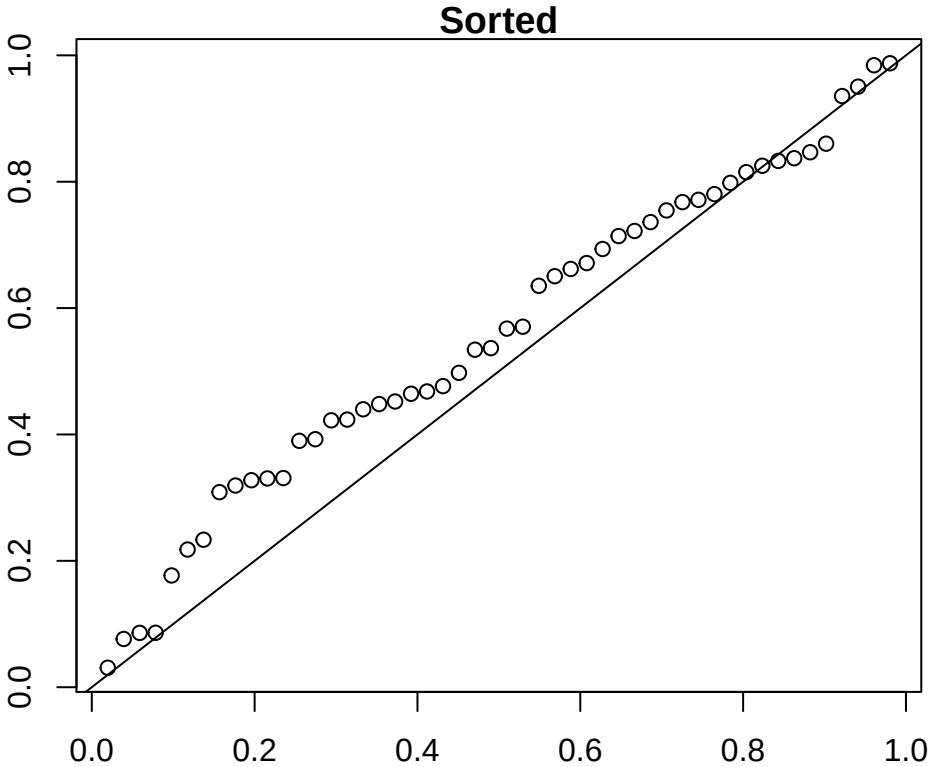
Click here or watch below

Residual plots can show some deviant patterns, but they are poor as a test of normality. There are formal tests of normality, but they generally don't work too well. Instead we can use normal probability plots. These are a particular sort of 'Q-Q plot' (Q-Q = quantile-quantile). The idea of a Q-Q- plot is that if we have N samples from a distribution, then the smallest should be approximately the same as the $1/(N+1)$th quantile, the next smallest approximately the $2/(N+1)$th quantile etc.

So, for example, for a uniform distribution the quantiles should be evenly spaced:

```
N <- 50
# NNorm <- scale(rnorm(N))
NUnif <- runif(N, 0, 1)

par(mfrow=c(2,1), mar=c(2,2,1,1))
plot(1:N, NUnif, xlab="Order drawn", ylab="Value", main="Un-sorted")
plot((1:N)/(N+1), sort(NUnif), xlab="Rank", ylab="Value", main="Sorted")
abline(0,1)
```

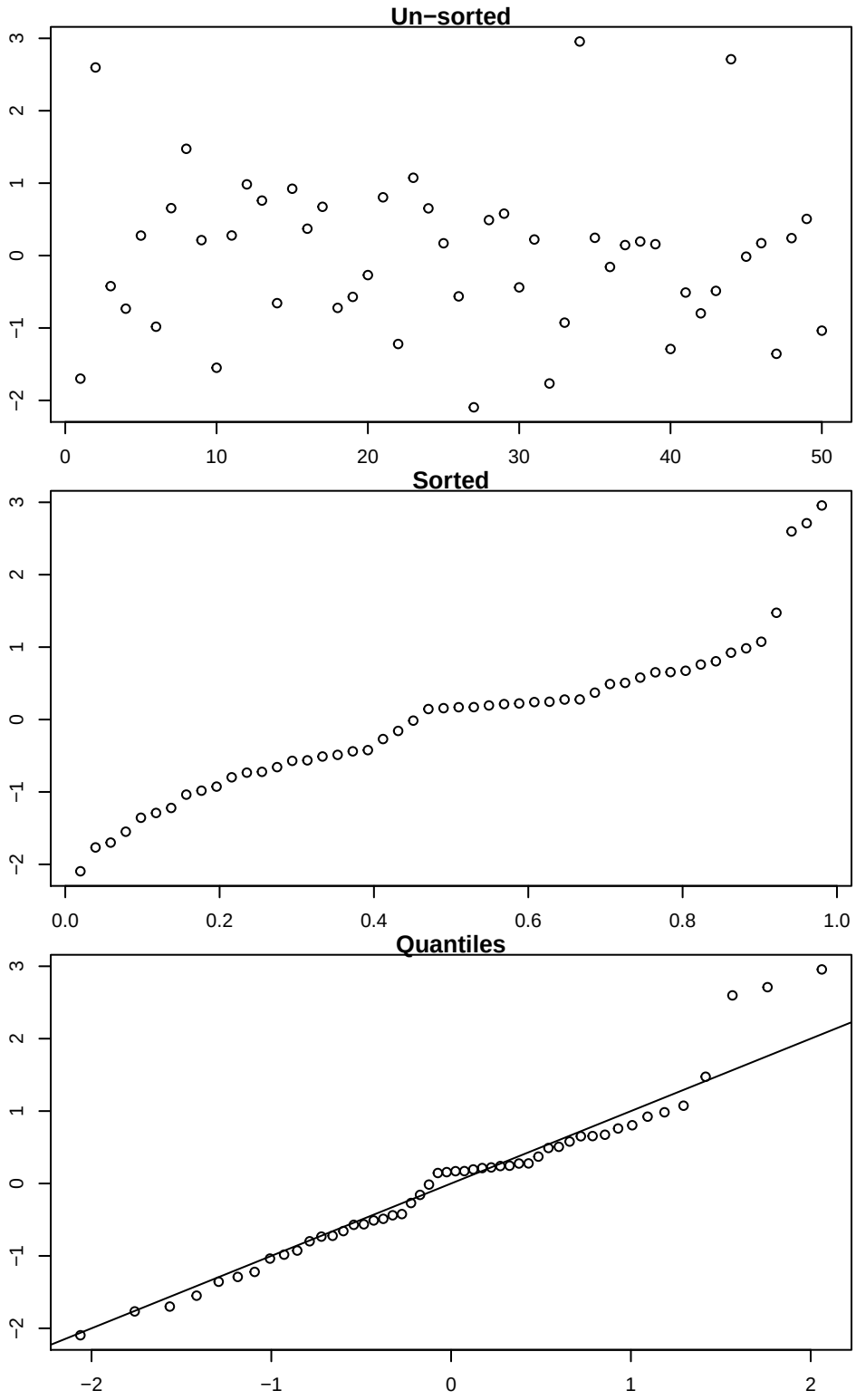




The bottom plot is the QQ plot. On the y axis is the data (or the residuals, below), and the x axis is the expected value for that data points.

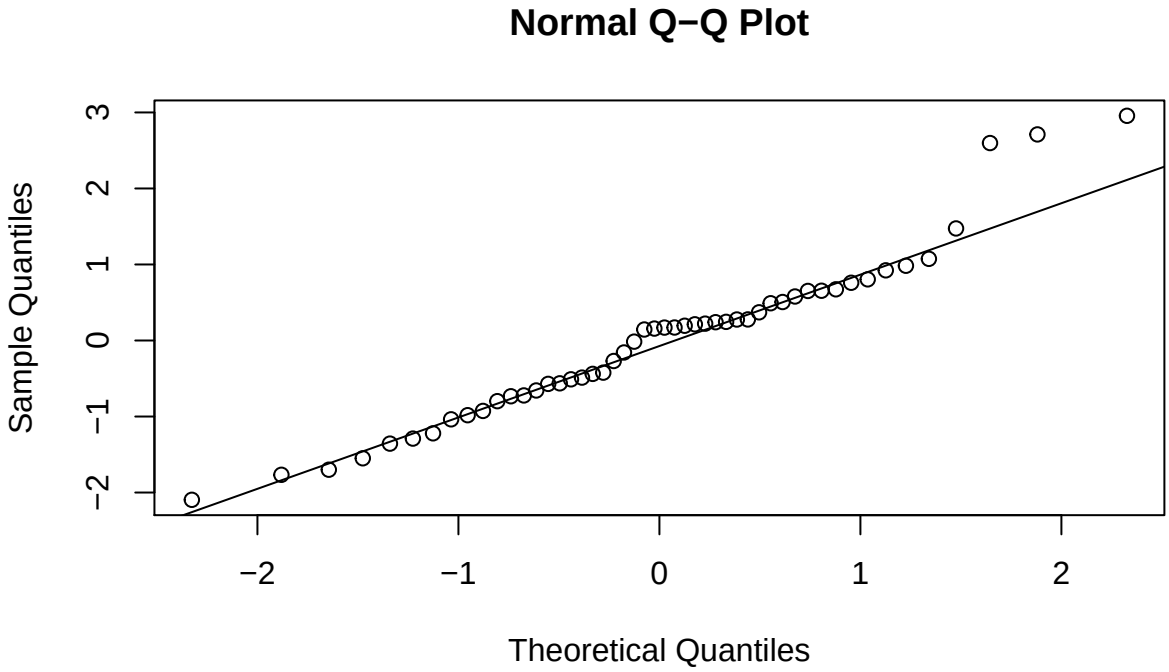
For a normal distribution the spacing is uneven, but we can calculate it

```r
N <- 50
# NNorm <- scale(rnorm(N))
NNorm <- rnorm(N, 0, 1)

par(mfrow=c(3,1), mar=c(2,2,1,1))
plot(1:N, NNorm, xlab="Order drawn", ylab="Value", main="Un-sorted")
plot((1:N)/(N+1), sort(NNorm), xlab="Rank", ylab="Value", main="Sorted")
plot(qnorm((1:N)/(N+1)), sort(NNorm), xlab="Rank", ylab="Value", main="Quantiles")
abline(0,1)
```



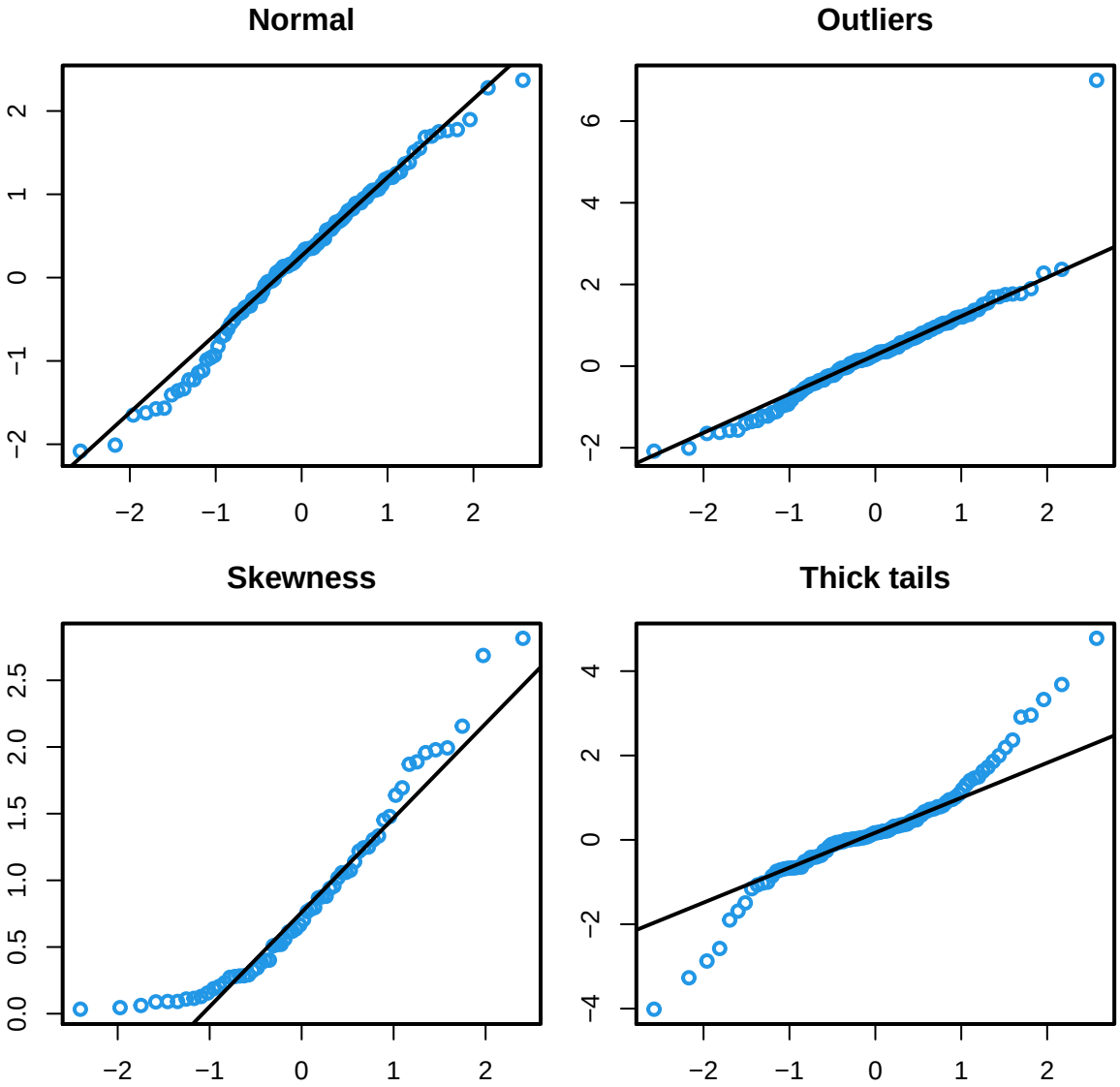
R even has code specifically to make these plots, so everything's easier. The line that is drawn is the expected line, which the points should fall along.

```
qqnorm(NNorm)
qqline(NNorm)
```



So, again, the y axis will have the residuals, and the x axis has the quantiles, their expected values.

What you can see



For the Normal, the points lie along the straight line (there is always a bit of variation around it, though).

For the outlier, one point has a much larger value than we would expect. So it sits a long way above the line.

For skewness, values get more and more positive (compared to what we would expect) as they get larger, so the points curve upwards. This is positive skew: it they were skewed the other way hte points on teh QQ plot would curve the other way.

Thick tails ("leptokurtosis". We don't expect you to remember this term) are like having lots of outliers: there are several points which are bigger and smaller than we would expect. So the points at both ends of the QQ plot are further away than we would expect.

Normal Probability Plots Exercise

- Draw normal probability plots for the 8 data sets. Do any suggest problems? You will need to extract the residuals and draw their normal probability plots.
- Try to draw normal probability plots that are normal, and then have outliers, skewness and thick tails
  - you will need to simulate data (e.g. with `rnorm()`), and then add points, or transform the data

You can simulate and change your data like this:

```
# simulate a normal distirbtion with 50 points, a mean of 0 and variance 1
SimNormData <- rnorm(50, 0, 1)
# Change the 13th point
SimNormDataChange <- SimNormData; SimNormDataChange[13] <- 50
# Transform the data
SimNormDataTransform <- SimNormData^2
```

And play around with this code and see how the normal probability plots change. You should experiment with adding outliers and transforming the data to see what effect this has. (there is a general point to make here: experimenting and looking at what happens can help you understand what can happen. And it can be fun too).

Hint

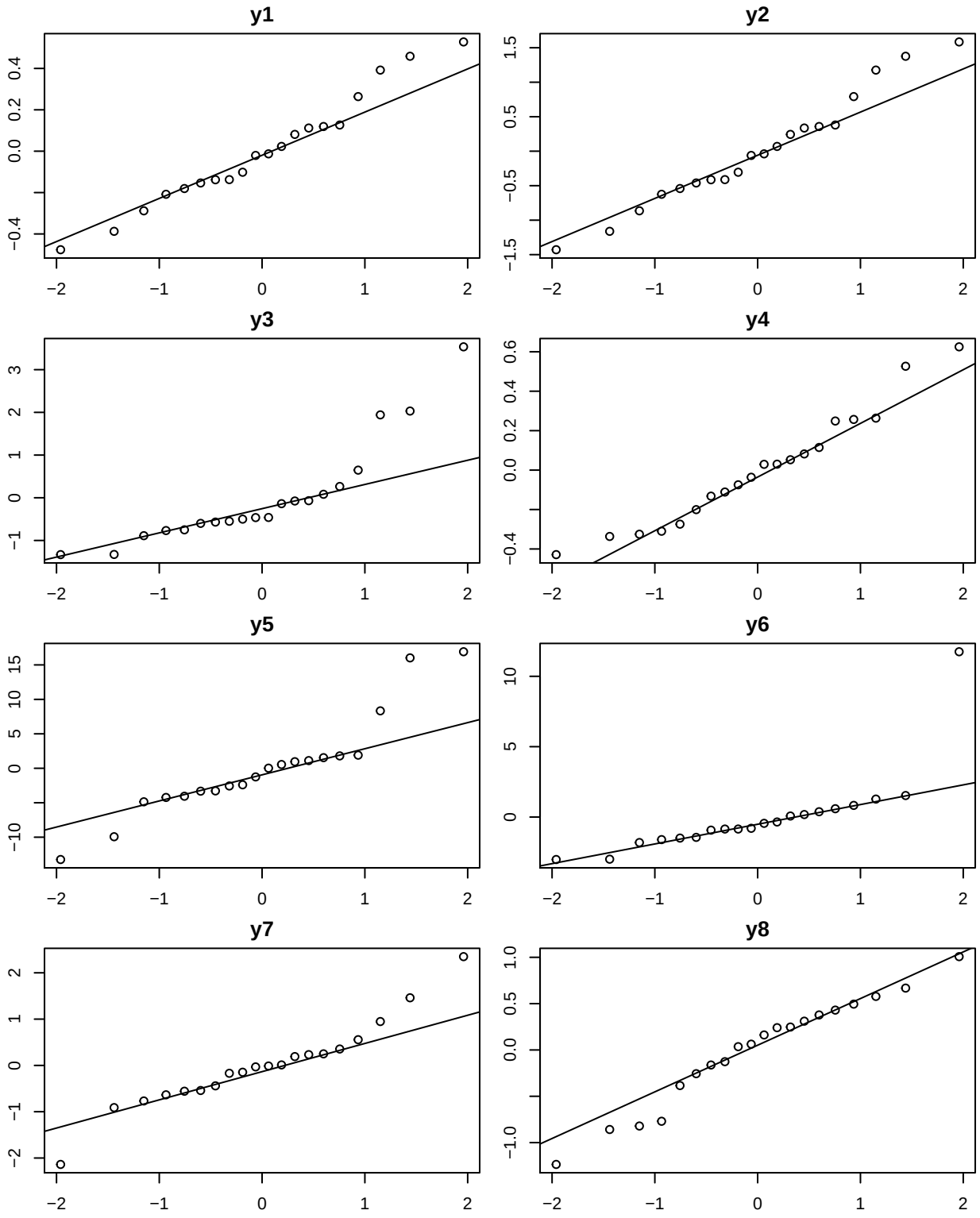
For each model, you just need to use `qqnorm()`. `qqline()` isn't necessary, but it can help.

```
mod <- lm(...) # hopefully you have already done this
qqnorm(resid(mod)) # remember to plot the *residuals* from the model!
qqline(resid(mod))
```

Answers

Here are all of the plots:

```
par(mfrow=c(4,2), mar=c(2,2,2,1))
lapply(Models, function(mod) {
 qqnorm(resid(mod), main=names(mod$model)[1])
 qqline(resid(mod))
})
```



Data set 1 looks good *Data set 2* looks good *Data set 3* looks like it has positive curvature, i.e. skewness in the data (or there might be outliers: with so few data points it can be difficult to decide which) *Data set 4* looks good *Data set 5* looks like it has thick tails (or a few too many outliers) *Data set 6* has a great big outlier *Data set 7* might have thicker tails (this is the data set with a huge x) *Data set 8* looks good (this is the one with a curve in the residuals)

We can see a few problems in the data, e.g. data set 3 is positively skewed. But, for example, data set 7 doesn't show the obvious problem.

Make you own bad data

For each question below, create the data using the code, plot the graphs, and try to interpret them!

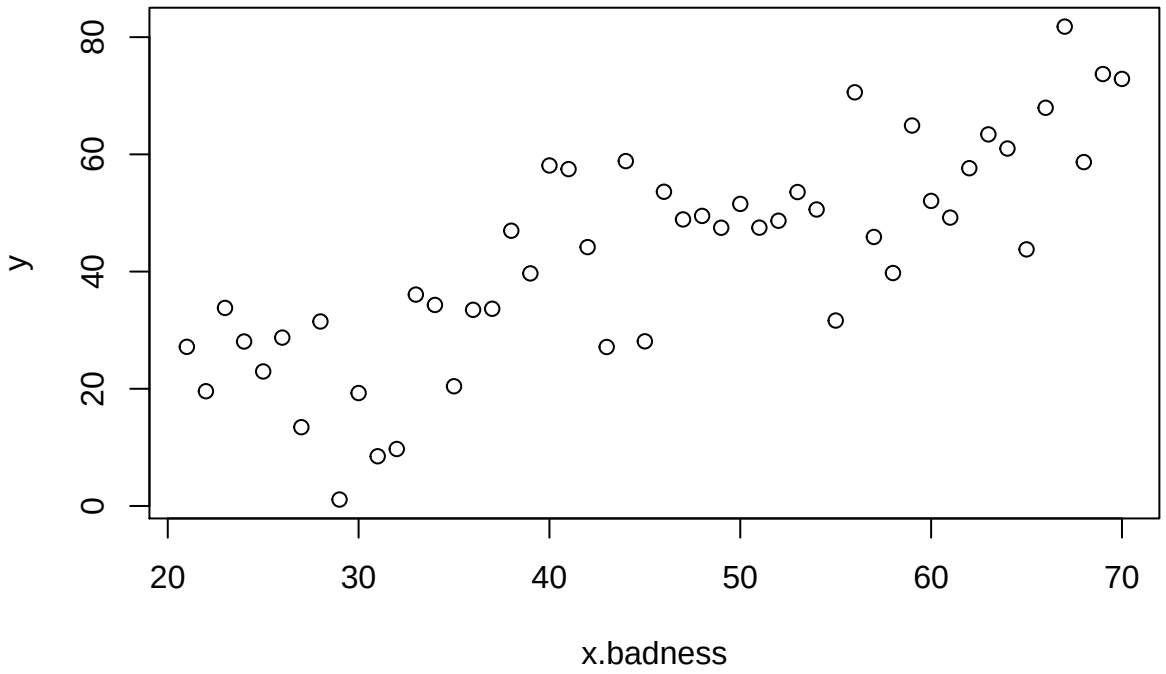
Start off with creating an outlier using the code below.

This is easy: add a "bad" point

Remember indexing with [] (some notes in this week's exercise)

You will need to remove the comments on the plots. You can also play around with different types of badness (e.g. different ways of trying to get skew). Explore, and see what happens!

```
# make good data
x.badness <- 21:70
y <- rnorm(length(x.badness), x.badness, 10)
plot(x.badness,y)
```




```
# add an outlier
y.outlier <- y
y.outlier[20] <- 200
mod.outlier <- lm(y.outlier~x.badness)
#qqnorm(resid(mod.outlier))
#qqline(resid(mod.outlier))
```

Now try some skewness.

```
# add skew
err.skew <- rnorm(length(x.badness), 4, 1)^2
y.skew <- x.badness + err.skew
mod.skew <- lm(y.skew~x.badness)
par(mfrow=c(1,2))
#plot(x.badness, y.skew)
```

```
#qqnorm(resid(mod.skew))
#qqline(resid(mod.skew))
```

Finally some thick tails.

```
# make tails thick
err.tailed <- rnorm(length(x.badness), 4, c(1,5)) # repeat std dev 1, 5
y.tailed <- x.badness + err.tailed
mod.tailed <- lm(y.tailed~x.badness)
par(mfrow=c(1,2))
#plot(x.badness, y.tailed)
#qqnorm(resid(mod.tailed)); qqline(resid(mod.tailed))
```

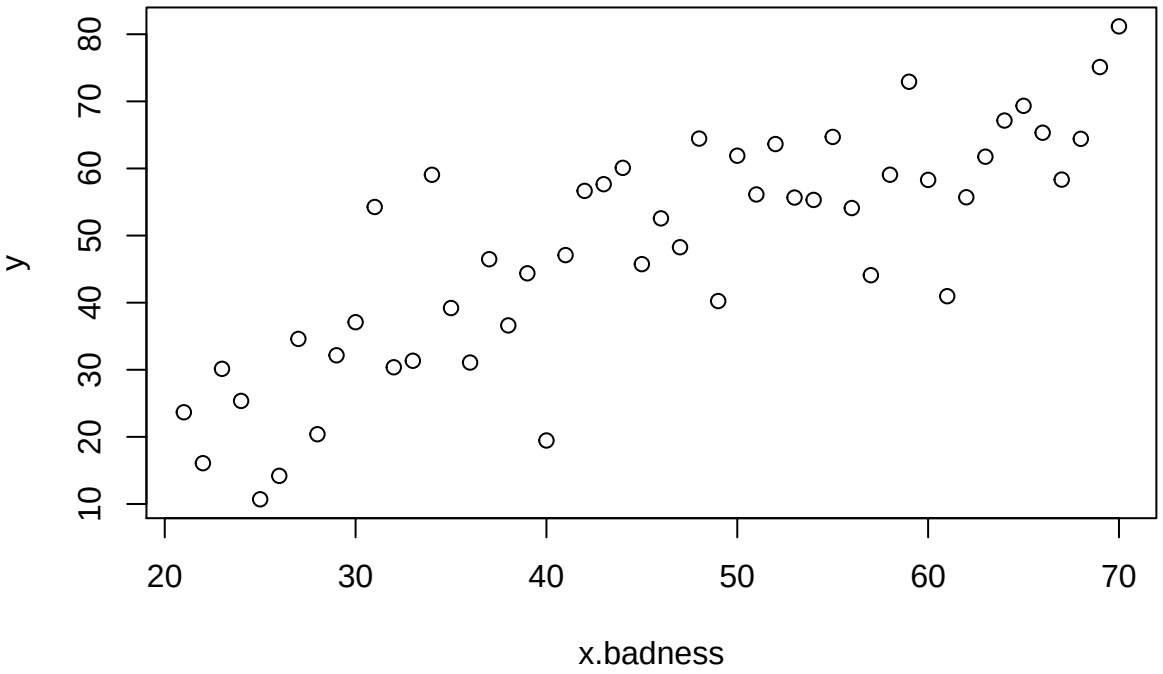
Hint

Have a look at the plots, nad compare those to the ones above.

Answers

The good data should look good. Your data might look slightly different, because we are generating this randomly.

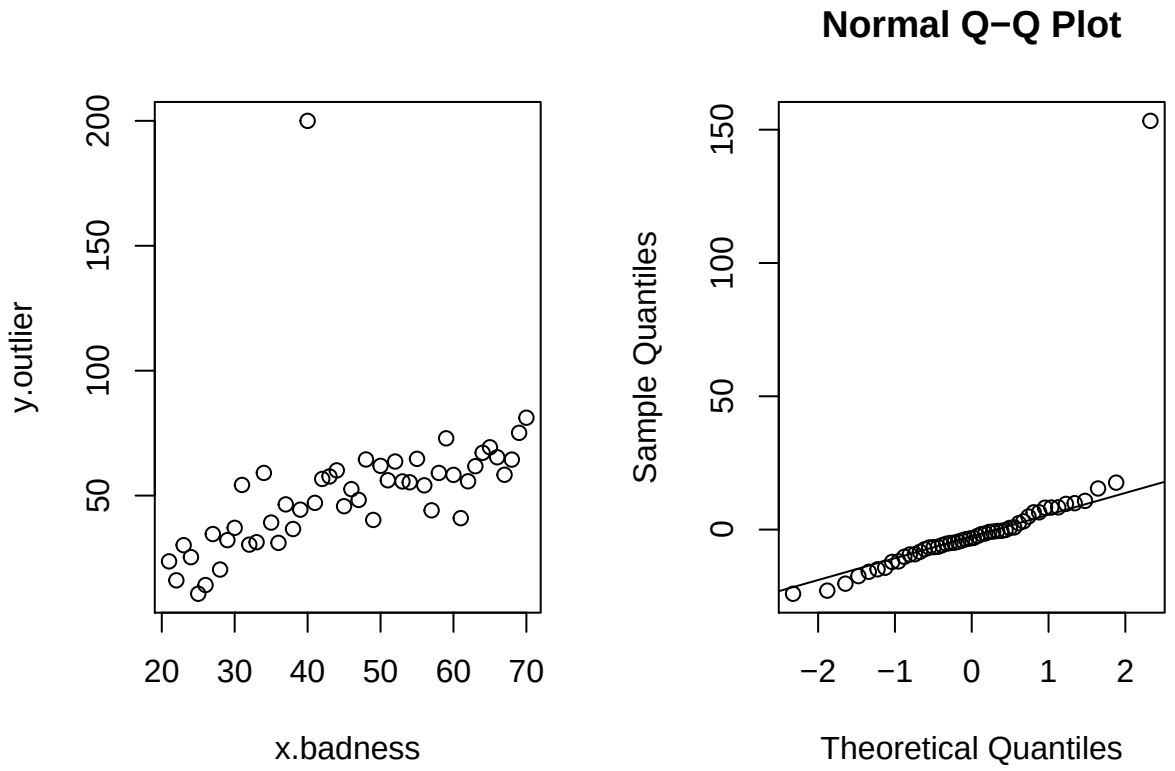
```
# make good data
x.badness <- 21:70
y <- rnorm(length(x.badness), x.badness, 10)
plot(x.badness,y)
```



The outlier adds an outlier:

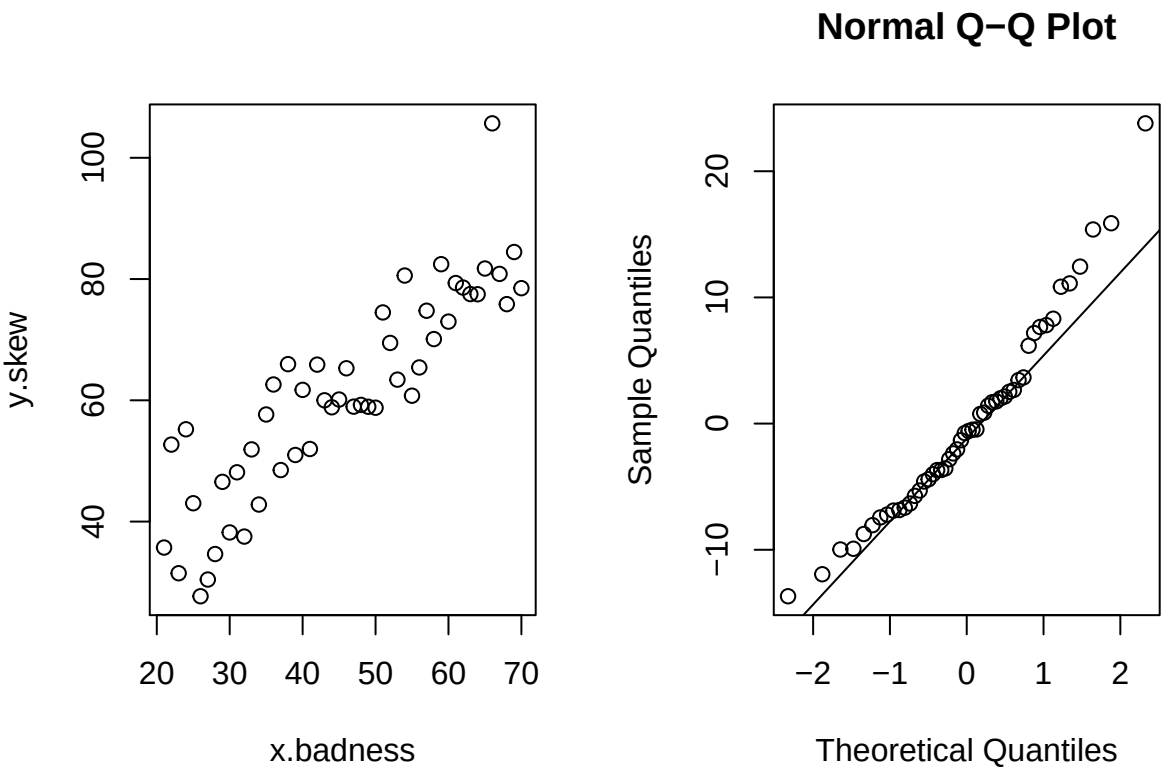
```
# add an outlier
y.outlier <- y
y.outlier[20] <- 200
mod.outlier <- lm(y.outlier~x.badness)
par(mfrow=c(1,2))
plot(x.badness, y.outlier)
qqnorm(resid(mod.outlier))
```

```
qqline(resid(mod.outlier))
```



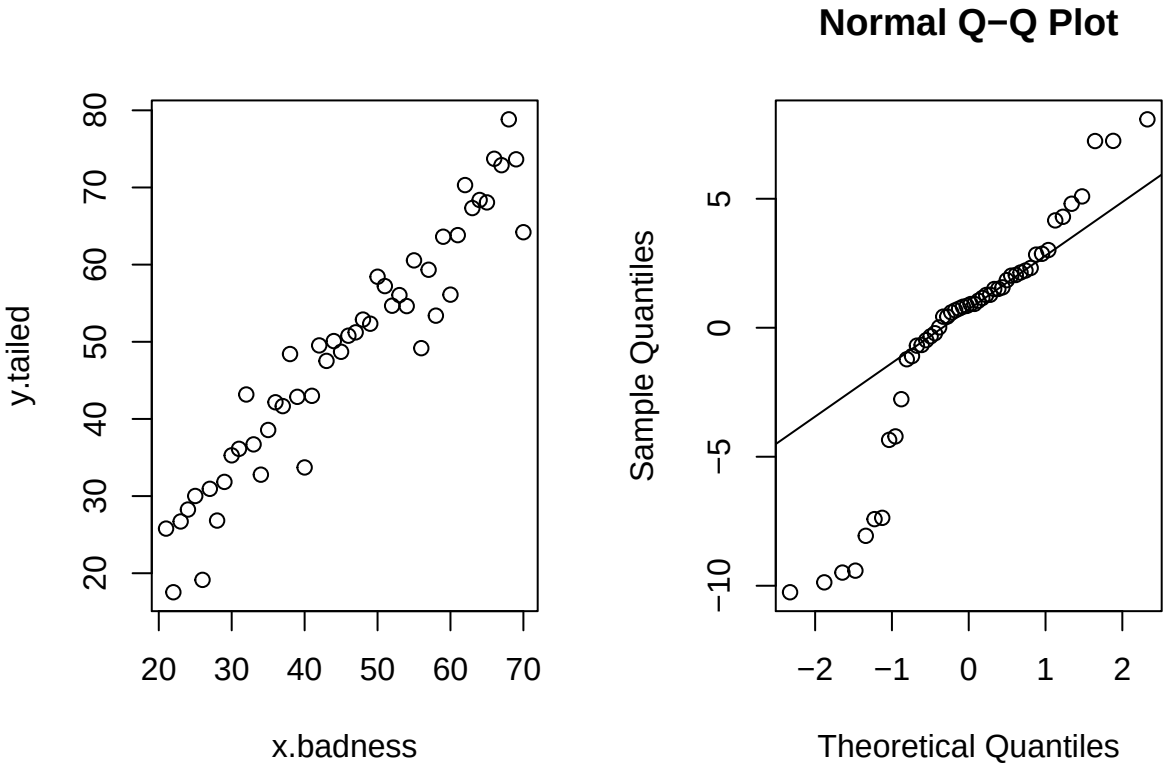
Now try some skewness. You could try `rnorm(length(x.badness), 4, 1)^4` to get more skew, or `rnorm(length(x.badness), 4, 1)^0.5` to get negative skew.

```
# add skew
err.skew <- rnorm(length(x.badness), 4, 1)^2
y.skew <- x.badness + err.skew
mod.skew <- lm(y.skew~x.badness)
par(mfrow=c(1,2))
plot(x.badness, y.skew)
qqnorm(resid(mod.skew))
qqline(resid(mod.skew))
```



Finally some thick tails.

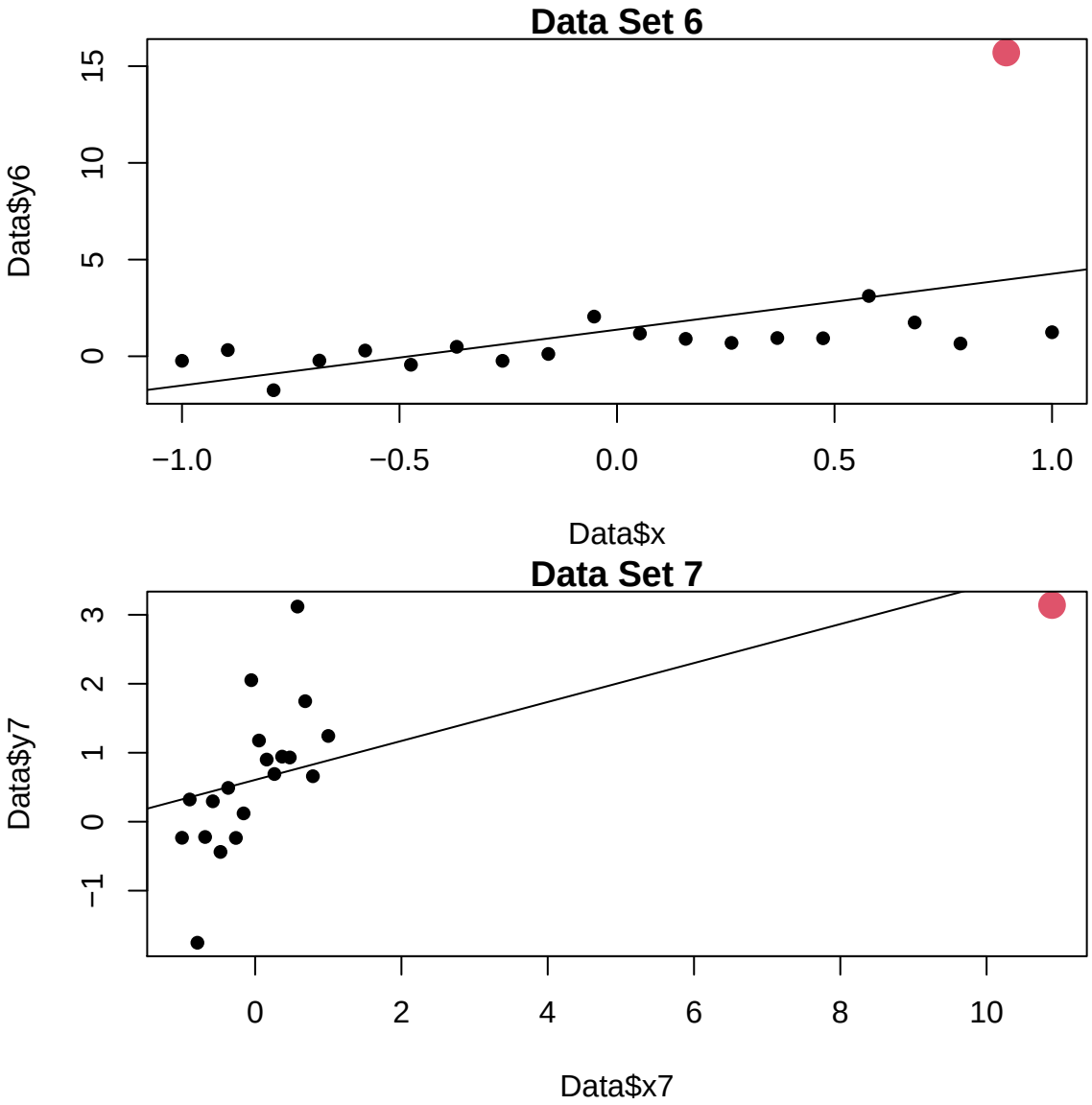
```
# make tails thick
err.tailed <- rnorm(length(x.badness), 4, c(1,5)) # repeat std dev 1, 5
y.tailed <- x.badness + err.tailed
mod.tailed <- lm(y.tailed~x.badness)
par(mfrow=c(1,2))
plot(x.badness, y.tailed)
qqnorm(resid(mod.tailed)); qqline(resid(mod.tailed))
```



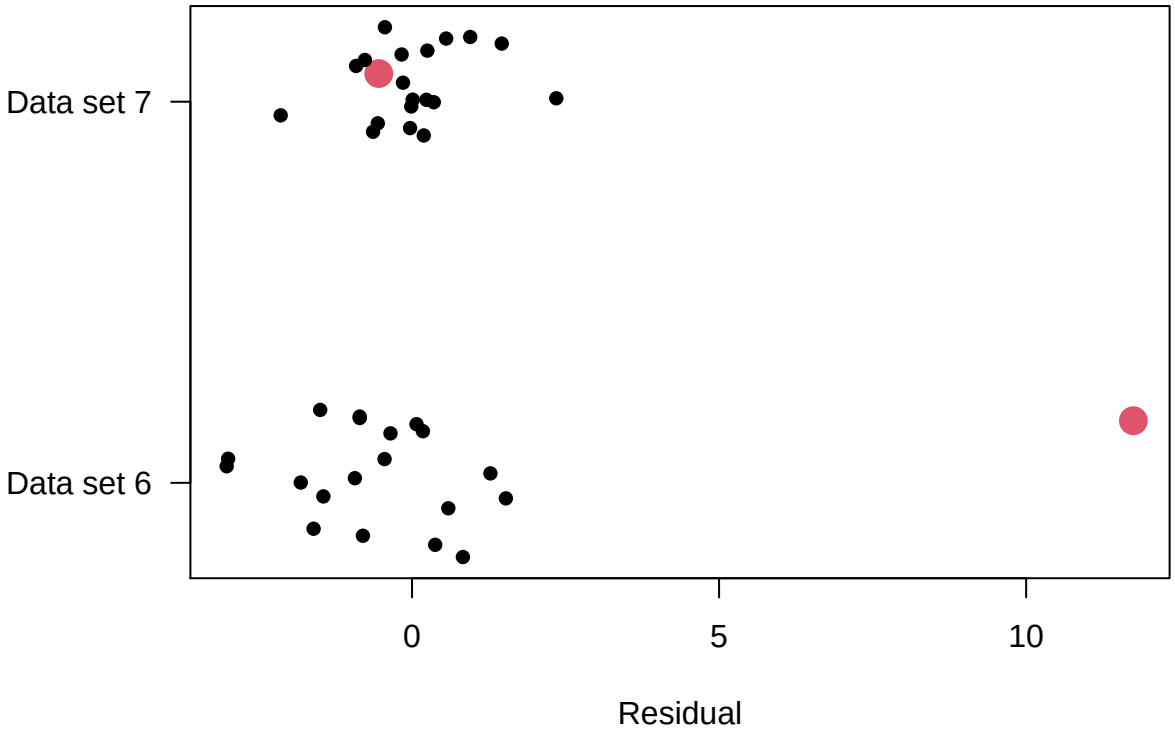
Leverage

Click here or watch below

This is less well know, but can be a problem, and can't always be seen with residuals. We can use data sets 6 and 7 to see the problem. First. here are the data sets, with the troublesome points highlighted.



We can fit a model and plot the residuals (the y-axis is jittered: the points has a small amount added to them, so they are easier to see):



For data set 6 the outlier is obvious in the residuals. But for data set 7 the obviously weird point has a residual that's almost 0. If we look at the regresson line, we can see that the weird point pulls the line towards it, so the residual isn't that small. We can tak about it as an influential point, because it has a large influence on the fitted line.

We can't use residuals to find influential points like these, instead we ask how much the fitted values for the other points change if we remove a data point. For each point we predict the other points with the full model, and call each one of these $\hat{y}_j$, and also predict them with a model where point i has been removed (and call these $\hat{y}_{j(-i)}$). So a large difference, $\hat{y}_j - \hat{y}_{j(-i)}$, would suggest a large effect. This will also depend on the total variation, so we calculate this, which we call **Cook's D**:

$$D_i = \frac{\sum_{j=1}^{n}(\hat{y}_j - \hat{y}_{j(-i)})^2}{s^2}$$

where s^2 is the residual variance. Large values of D_i mean a large influence, so if $D_i > 1$, or $D_i > 4/n$ (depending on who you ask!).

Leverage Exercise

Fit the model to data set 7 with and without the weird point - You can remove the point and fit the model like this:

```
DataNotWeird <- Data[Data$x7<10,]
ModelNotWeird <- lm(y7 ~ x7, data=DataNotWeird)
```

Look at the fitted models. How similar are they?

- How similar are the parameter estimates
- plot the fitted lines on the data (with `abline()`)
- Calculate Cook's D for the different data sets, and plot them against `x`. Do you see any influential points?

```
cooks.distance(WomenMod)[1:5]
```

```
##           1           2           3           4           5
## 0.643742510 0.001416355 0.018007524 0.243659099 0.017246109
```

Hint

Use `coef()` to compare the models. You can use `abline(mod)` to plot the model on top of a plot of the data.

Answers

First, how similar are the parameter estimates? We can fit the model to all the data, and without the weird point, and look at the coefficients:

```
ModelAll7 <- lm(y7~x7, data=Data)
ModelMost7 <- lm(y7~x7, data=DataNotWeird)

coef(ModelAll7)
```

```
## (Intercept)          x7
##    0.606328    0.282340
```

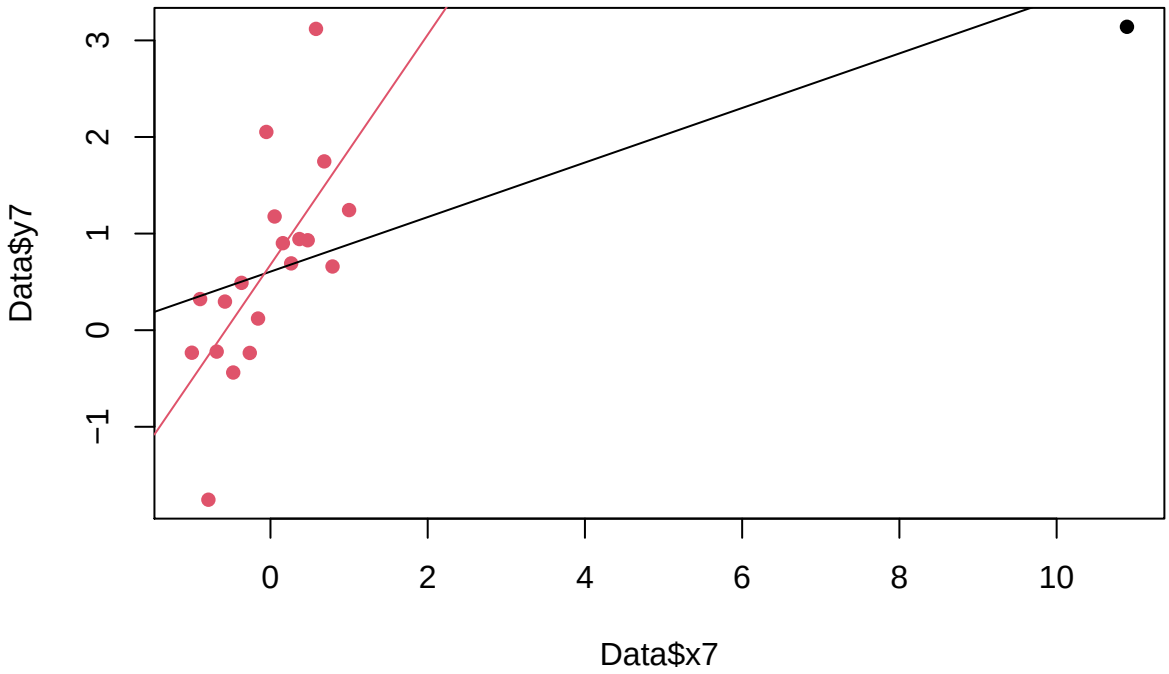
```
coef(ModelMost7)
```

```
## (Intercept)          x7
##   0.6776091   1.1905997
```

We can see a slight change in the intercept, but more importantly is the large change in the slope, from 0.28 to 1.19.

Plotting the lines shows the difference clearly:

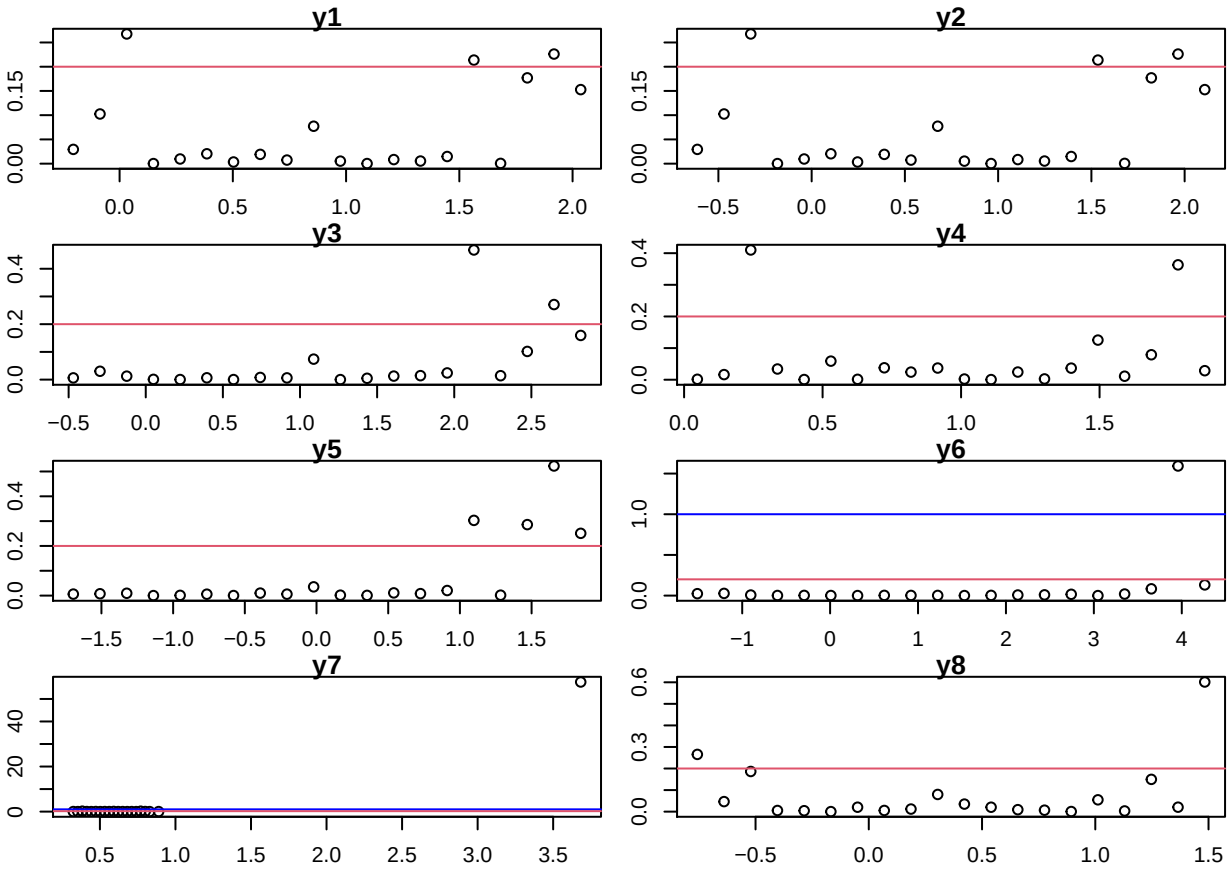
```
plot(Data$x7, Data$y7, col=1+(Data$x7<10), pch=16)
abline(ModelAll7)
abline(ModelMost7, col=2)
```



The line fitted to hte full data is pulled a long way down from where it would be otherwise.

Now we can look at Cook's D for all of the data. You can probably guess what data set 7 will look like. This is all of them:

```
par(mfrow=c(4,2), mar=c(2,2,1,1))
lapply(Models, function(mod) {
  plot(fitted(mod), cooks.distance(mod), main=names(mod$model)[1])
  abline(h=1, col="blue"); abline(h=4/length(mod$residuals), col=2)
})
```




```
## $mod1
## NULL
##
## $mod2
## NULL
##
## $mod3
## NULL
##
## $mod4
## NULL
##
## $mod5
## NULL
##
## $mod6
## NULL
##
```

```
## $mod7
## NULL
##
## $mod8
## NULL
```

The blue line shows $D = 1$, but is ot visible for many of the plots. So for most there is little or no evidence for influential data. Data set 6 (with the outlier) and data set 7 (with the influential point) are the exceptions: for both we can see the extremely large values.

How good is my model? A Summary

Before looking at how to improve models, it is worth summarising how to check a model.

The model can be read as a sum of the fit and the residuals. The proportion of the total variation explained by the model is summarised with R^2.

The residuals should have no pattern. Here we try look at them graphically, because a couple of graphs can reveal a lot of problems:

- Residual plots can show curvature, outliers and heteroscedasticity
- Normal Probability Plots can show a lack of normality (e.g. skewness), and also outliers
- Cook's D can be used to identify influential points.

How can we improve the model?

Click here or watch below

If we have identified some problems, what do we do? The first thing to do it to check the data and model for silly mistakes, like typos. Once that has been done, we should start to think about what the mis-fit means, and whether it will cause any real problems. For example, does it change our conclusions? Or if we are making predictions, we want to know if the mis-fit will change the predictions.

If we find individual data points are a potential problem (e.g. they are outliers or influential), we first need to check that they are not typos, or other errors (e.g. computers reading them in incorrectly). If they are genuine, then we can remove them & see if that makes a big difference. If it does, we have to think hard. It is possible that they really should be 'in' the data, so that we would be inflating the fit of the model by removing them. And if there are more than one or two points that we could remove, it might be that the problem is not that they are outliers. For example, it could be that the distribution genuinely has more large values than expected from a normal distribution (the technical term for this is that the data are leptokurtic. This term will not be on the exam).

Other problems are due to the model, as a whole, not fitting. In particular, we can sometimes see curvature or skewness in the residuals. In these cases there are a few solutions, based on transformations. One possibility, if we only have curature or if the curvature is only apparent in the plot against one covariate (but not others) is to transform the covariate, e.g. $\sqrt{x_i}$, x_i^2, $\log(x_i)$, so the model becomes

$$y_i = \alpha + \beta x_i^p + \varepsilon_i$$

where, for example, $p = 2$ for the squared transformation. The transformation can sometimes make biological sense. An alternative approach is to add more terms, for example a quadratic term:

$$y_i = \alpha + \beta x_i + \gamma x_i^2 + \varepsilon_i$$

There will be more about this later, when we get on to multiple regression.

An alternative approach is to transform the response, e.g. $\sqrt{x_i}$, x_i^2, $\log(x_i)$, to give a model like this:

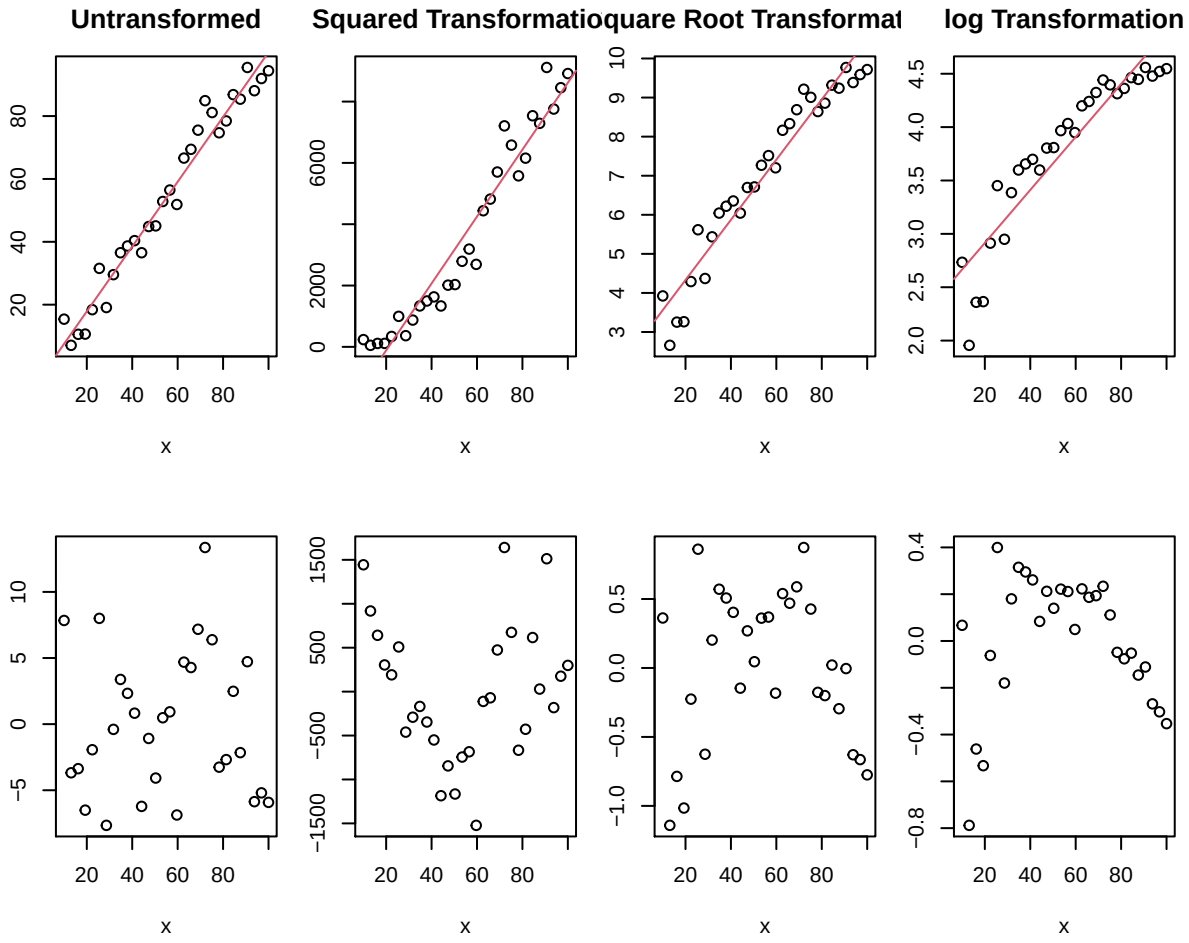
$$y_i^p = \alpha + \beta x_i + \varepsilon_i$$

Doing this can change both the linearity and the skewness, so it can be a bit more trickly. But there are times when it is the clear thing to do. There is a general class of transformations that can be used, the Box-Cox family:

$$y_i \to y_i^p$$

e.g. if we want a square root transformation, we can use $p = 1/2$. If we decide $p = 0$ is appropriate, we use $log(y_i)$. The transformation has two effects: on the curvature of the response, and on how the variance changes with the mean. We can look at the effect on curvature first:

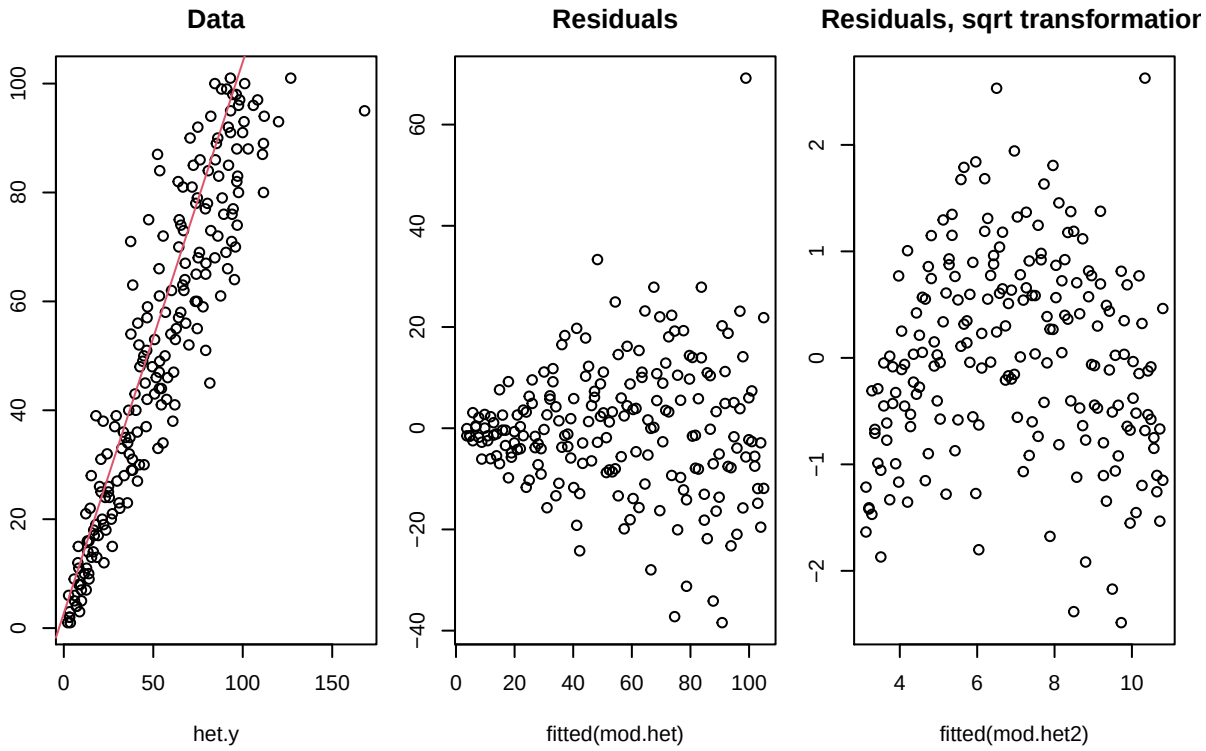
```
x <- seq(10,100, length=30)
y <- rnorm(length(x), x, 5)
ySq <- y^2
ySqrt <- sqrt(y)
ylog <- log(y)

par(mfrow=c(2,4), mar=c(4.1,2.1,3,1), oma=c(0,2,0,0))
plot(x, y, main="Untransformed")
 abline(lm(y~x, data = Times), col=2)
plot(x, ySq, main="Squared Transformation")
 abline(lm(ySq~x, data = Times), col=2)
plot(x, ySqrt, main="Square Root Transformation")
 abline(lm(ySqrt~x, data = Times), col=2)
plot(x, ylog, main="log Transformation")
 abline(lm(ylog~x, data = Times), col=2)

plot(x, resid(lm(y~x, data = Times)))
plot(x, resid(lm(ySq~x, data = Times)))
plot(x, resid(lm(ySqrt~x, data = Times)))
plot(x, resid(lm(ylog~x, data = Times)))
```



And then how the variance changes with the mean. If the variance is constant, we say the data are **homoscedastic**. If the variance changes with the mean, it is **heteroscedastic**. These are two terms that will also not be on the exam (especially not an oral exam).

The Box-Cox transformation can change this relationship

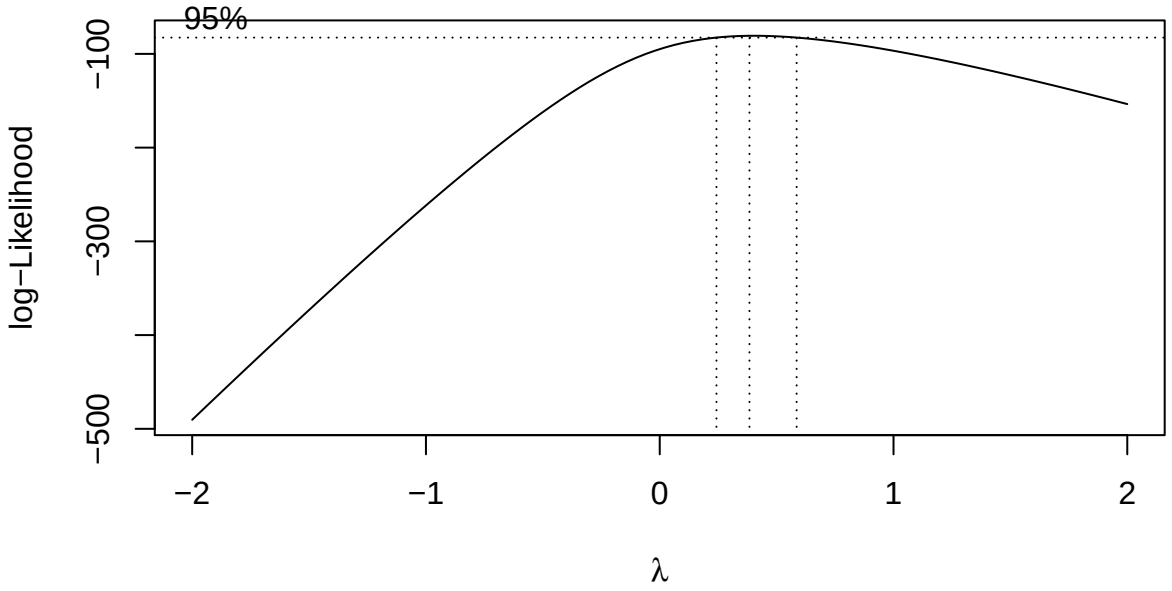
```
het.x <- rep(1:101,2)
a=2
b = 1
sigma2 = het.x^1.3
eps = rnorm(length(het.x),mean=0,sd=sqrt(sigma2))
het.y=a+b*het.x + eps
mod.het <- lm(het.y ~ het.x)
mod.het2 <- lm(sqrt(het.y) ~ het.x)

par(mfrow=c(1,3), mar=c(4.1,2.1,3,1), oma=c(0,2,0,0))
plot(het.y, het.x, main="Data")
abline(mod.het, col=2)

plot(fitted(mod.het), resid(mod.het), main="Residuals")
plot(fitted(mod.het2), resid(mod.het2), main="Residuals, sqrt transformation")
```



Box-Cox in R

We could do the transformation by trying several different transformations. This is a good way of getting a feel for what it's doing, and the transformation doesn't usually have to be precise. But R has a function to find the best Box-Cox transformation, which is found in the MASS package:

```
library(MASS)
x <- 1:50
y <- rnorm(50,0.1*x, 1)^2
boxcox(lm(y ~ x)) # 0.5 is true transformation
```



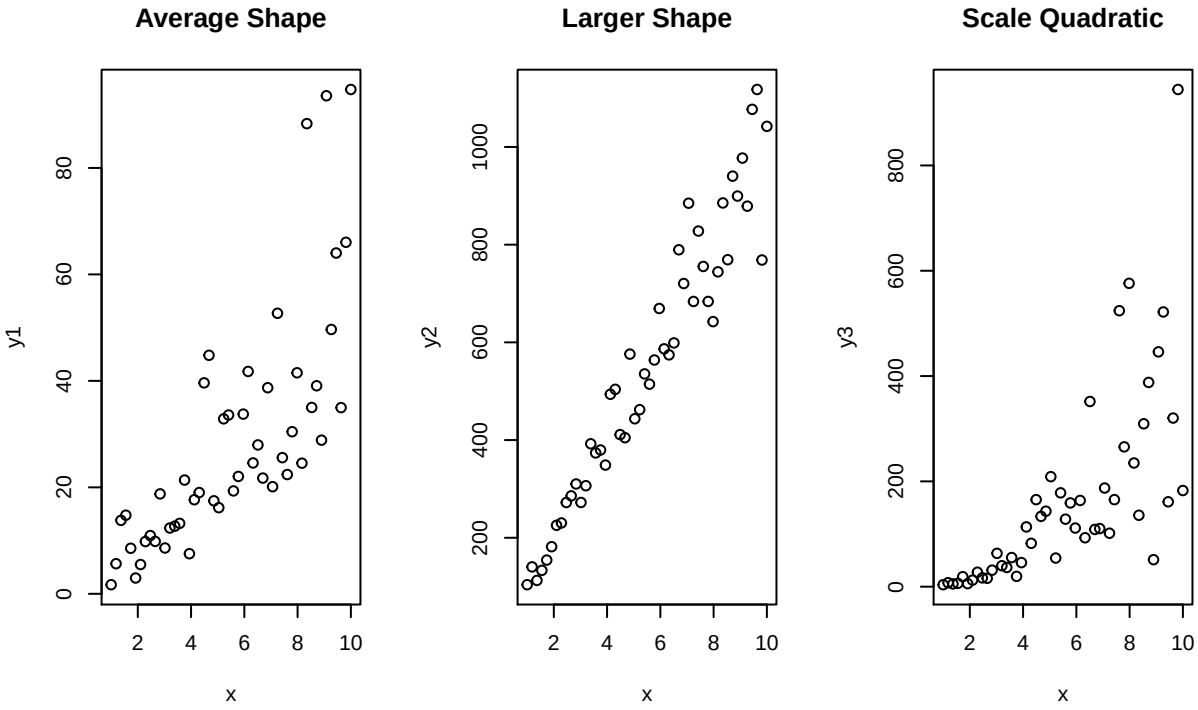
Your Turn

Unfortunately `boxcox()` needs positive responses, so we can't use the data we have already been using. Instead we can create data from a Gamma distribution with `rgamma()`. It has two parameters (after the number of points to simulate):

- Shape: controls skew: the higher, the more symmetrical
- Scale: this controls the mean (mean = shape*scale)

We can create a (bad) model by keeping the shape constant but letting the scale vary with x:

```
x <- seq(1,10, length=50)
y1 <- rgamma(length(x), shape=5, scale=x)
y2 <- rgamma(length(x), shape=100, scale=x)
y3 <- rgamma(length(x), shape=5, scale=x^2)

par(mfrow=c(1,3))
plot(x,y1, main="Average Shape")
plot(x,y2, main="Larger Shape")
plot(x,y3, main="Scale Quadratic")
```



- Look at the curves, and describe what it looks like
- Regress each y (i.e. y1, y2, y3) against X
- Check the residuals.
- See if a transformation helps, e.g.

```
gam.mod <- lm(y1 ~ x)
library(MASS)
boxcox(gam.mod)
```

- if a transformation is suggested, try it, and check the residuals again

Hint

You can use `boxcox()` to get what is an optimal model, but it still might not be a good model. So it is worth

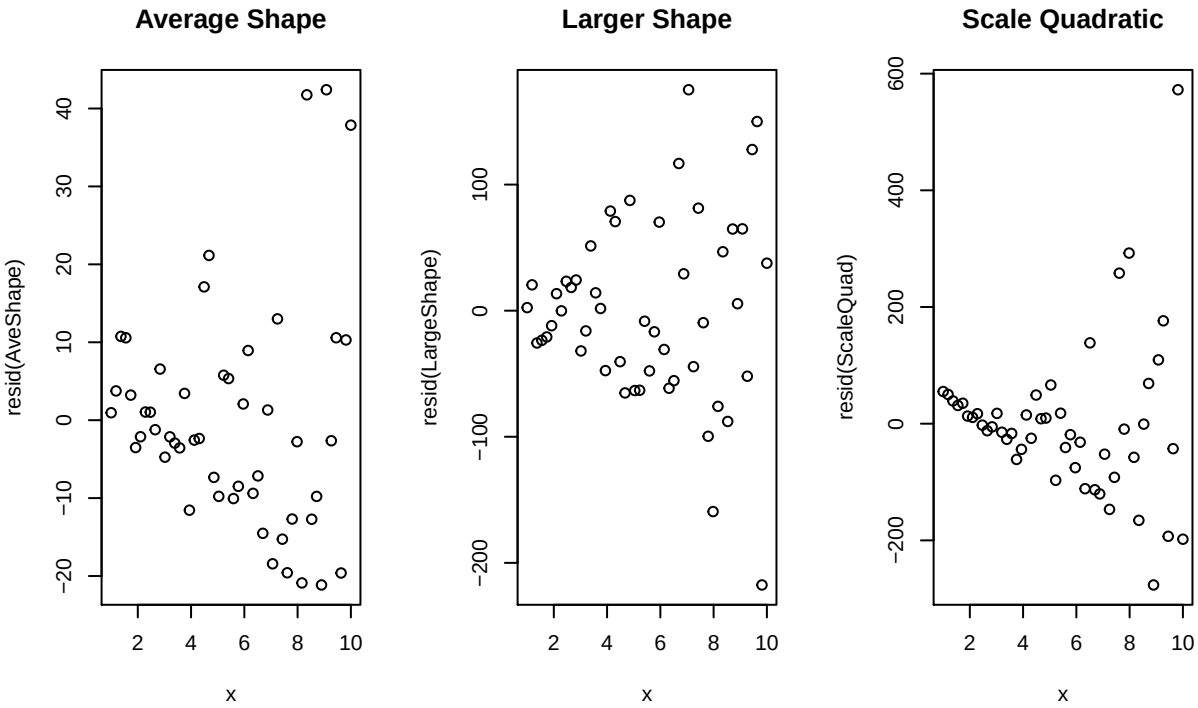
plotting the residuals, even after you have what you think is a good model.

This problem is as much about interpreting the outout as about getting it.

Answers

First fit the models, and plot the residuals. This time I've plotted against x.

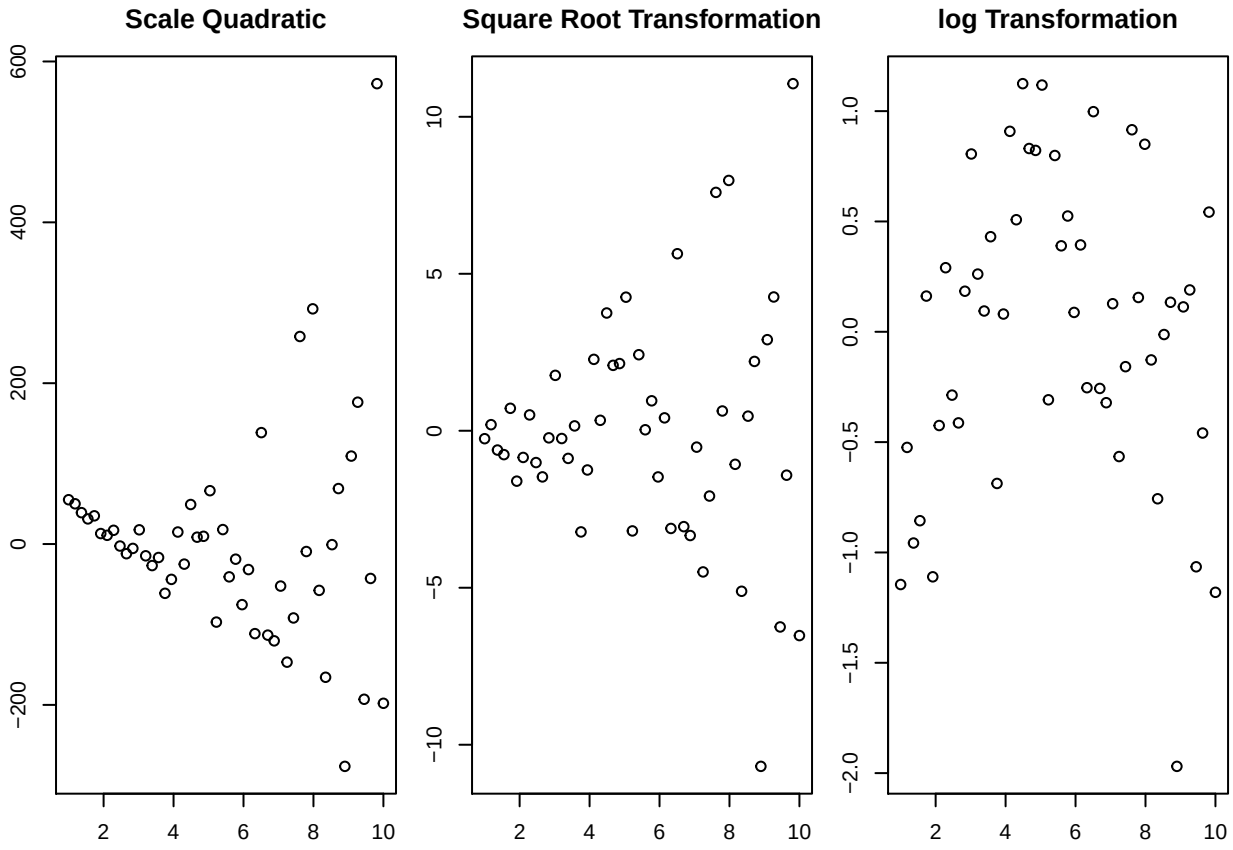
```
AveShape <- lm(y1 ~ x)
LargeShape <- lm(y2 ~ x)
ScaleQuad <- lm(y3 ~ x)

par(mfrow=c(1,3))
plot(x, resid(AveShape), main="Average Shape")
plot(x, resid(LargeShape), main="Larger Shape")
plot(x, resid(ScaleQuad), main="Scale Quadratic")
```



We can see that in all cases there is a lot of heteroscedasticity (which is the way the gamma distribution works). For the Scale Quadratic model particularly, there are far fewer high residuals, which might suggest there is also positive skewness.

Now to look for a better model. First, for the Scale Quadratic model we can try the square root and log transformations, and look at the residuals:

```
ScaleQuad.sqrt <- lm(sqrt(y3) ~ x)
ScaleQuad.log <- lm(log(y3) ~ x)

par(mfrow=c(1,3), mar=c(2,2,3,1))
plot(x, resid(ScaleQuad), main="Scale Quadratic")
plot(x, resid(ScaleQuad.sqrt), main="Square Root Transformation")
plot(x, resid(ScaleQuad.log), main="log Transformation")
```



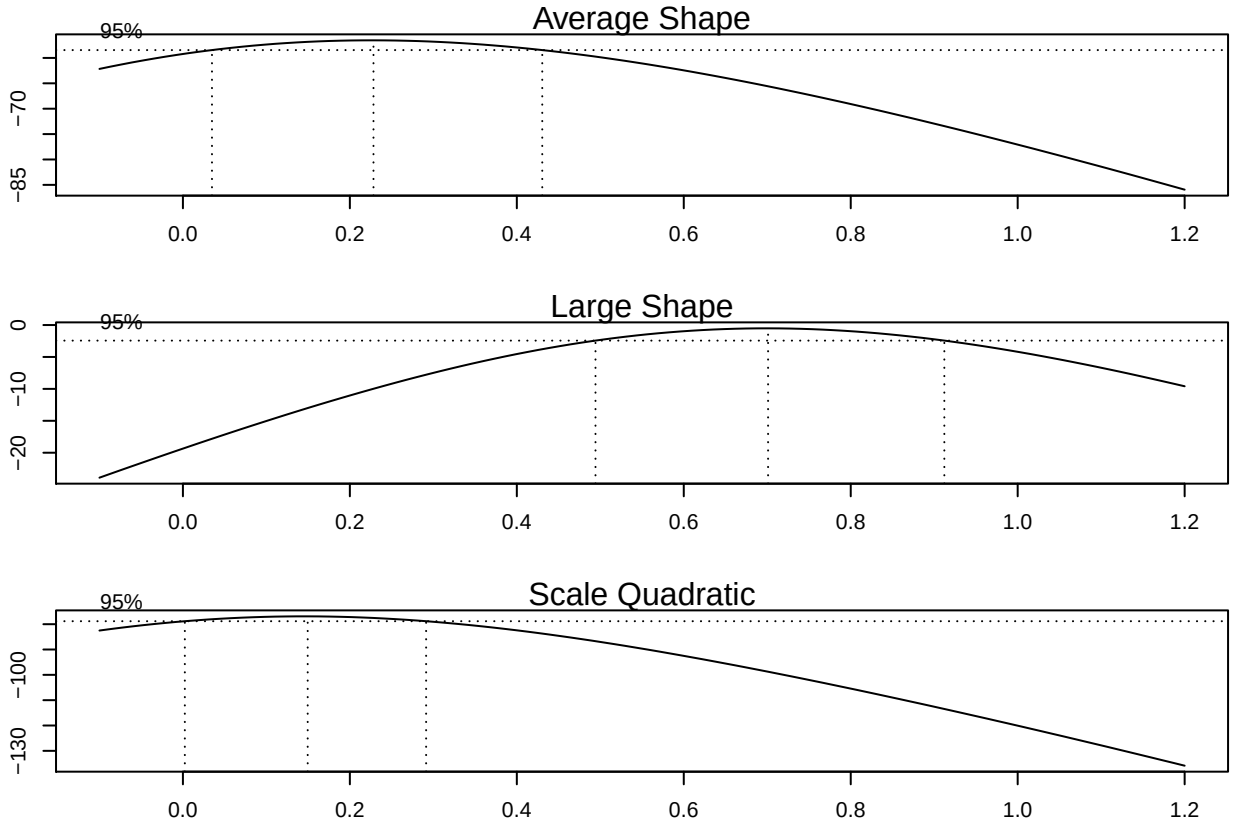
The square root transformation is not enough to make the residual homoscedastic: the log transformation is much better, but might have made the relationship curved. This suggests a log transformation, and possibly adding a quadratic term to the effect of x will help.

We can use the `boxcox()` function to look at all three data sets. I have restricted the range of the power (the default is -2 to 2, but usually 0 to 1 is OK).

```
par(mfrow=c(3,1), mar=c(3,2,2,1))
boxcox(AveShape, lambda=seq(-0.1,1.2,0.1))
mtext("Average Shape", 3)
boxcox(LargeShape, lambda=seq(-0.1,1.2,0.1))
mtext("Large Shape", 3)
boxcox(ScaleQuad, main="Scale Quadratic", lambda=seq(-0.1,1.2,0.1))
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'main' will be disregarded
```

```
mtext("Scale Quadratic", 3)
```



Depending on the simulation, the result might be bit different. But a transformation lower than 1 is called for, and possibly as low as 0 (a log transformation). The Large Shape data seems to want a larger value of lambda.

You should check the residuals for the other models, as we did for the scale quadratic model above. There may not be a perfect transformation.

Summary

We now know how to asses the model fit

- R^2 show how much variation the model explains
- Residual plots and Normal Probability Plots can show curvature, outliers, and varying variance
- Influential Points can be detected using Cook's D. These may not be large outliers!
- We should check outliers & other odd points - are they typos?
- We can try to transform the response to get a better model

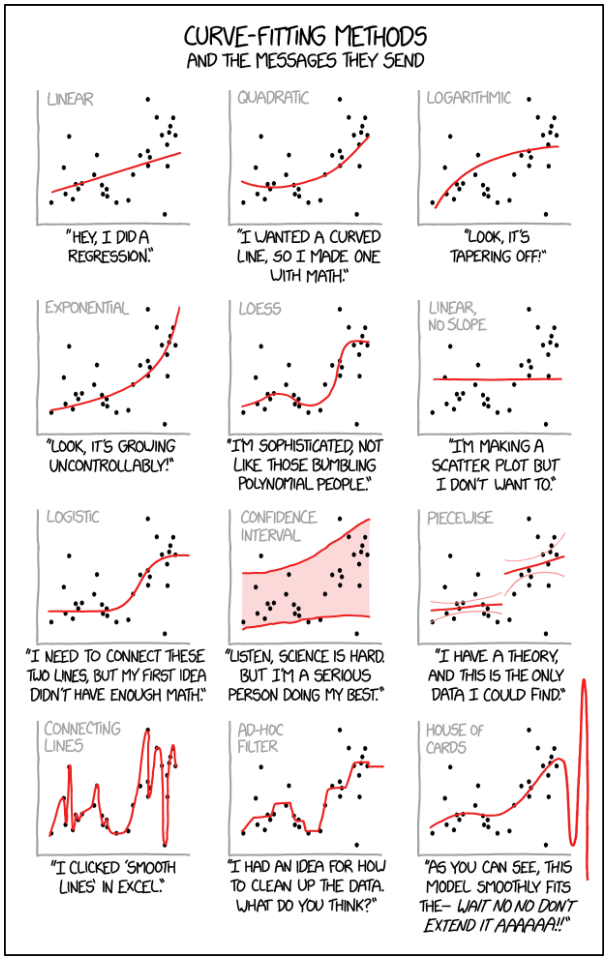


Figure 2: Don't overinterpret