Week 4: The Normal Distribution

Recap

Begin with the intro video

Click here or watch below

So far we have

- learned about maximising the likelihood
- estimated confidence intervals and standard errors

These are the basic tools we will use to fit and understand our models

More than one datum

So far we have only used one data point. But what if we have more? The standard approach starts by making one additional assumption: that each data point is collected independently of the rest. This would mean that each time data is collected, it doesn't depend on what the last data point was.

If we assume that the data are independent, then from the definition of independence:

$Pr(X_1 \& X_2) = Pr(X_1)Pr(X_2)$

So, because the likelihood is the probability of each data point given the parameters, we can calculate the likelihood for the parameters (θ) given all of the data (X_i's) by multiplying the probabilities:

$$Pr(X_1, X_2, ..., X_n|\theta) = Pr(X_1|\theta)Pr(X_2|\theta)...Pr(X_n|\theta) = \prod_{i=1}^{n} Pr(X_i\theta)$$

But we usually work on the log-likelihood scale, where we have:

$$l(\theta|X_1, X_2, ..., X_n) = \sum_{i=1}^{n} \log(Pr(X_i|\theta))$$

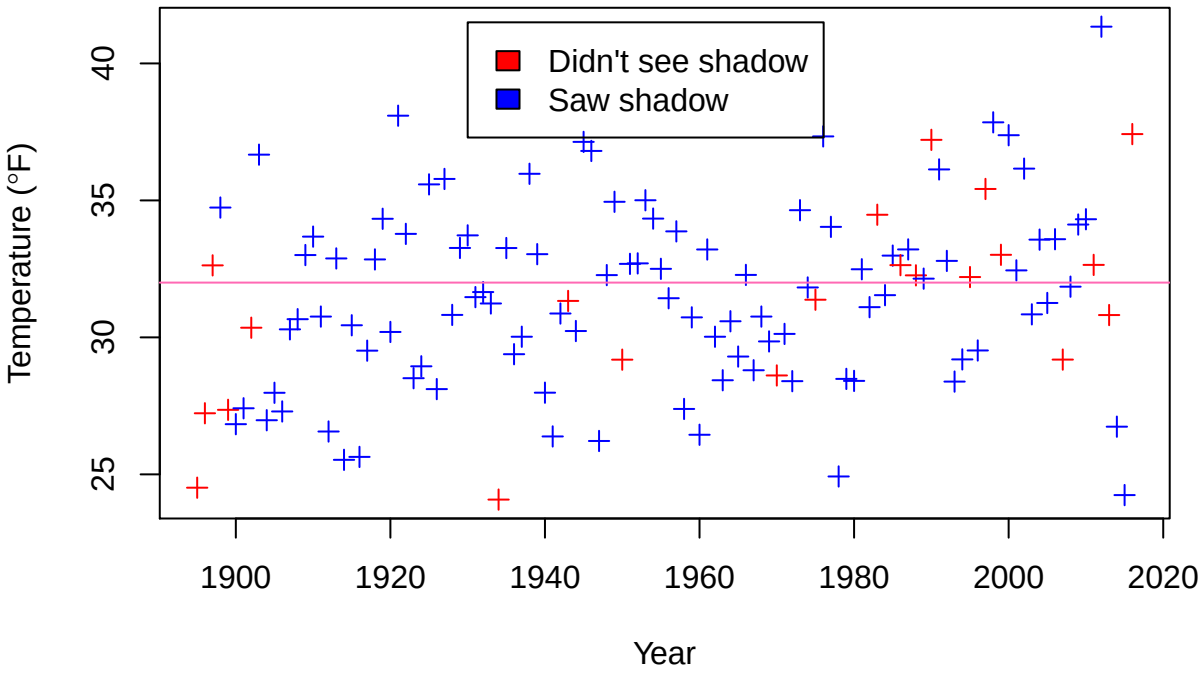
So, we just add the log-likelihoods together, which is easier than multiplying, and makes a lot of the maths easier.

Punxsutawney Phil & Groundhog Day

Every 2nd of February, the town of Punxsutawney in the US has a ritual where they go to Gobbler's Knob, drag a groundhog out of the ground, call it Phil, and ask it if it sees ~~Bill Murray's~~ its shadow. If Punxsutawney Phil sees his shadow, he'll decide to retreat, because there will be another 6 weeks of winter.

The data can be found here: https://www.math.ntnu.no/emner/ST2304/2021v/Week04/GroundhogDayta.csv It is a .csv file and can be downloaded AND imported using `read.csv()` and the full url above.

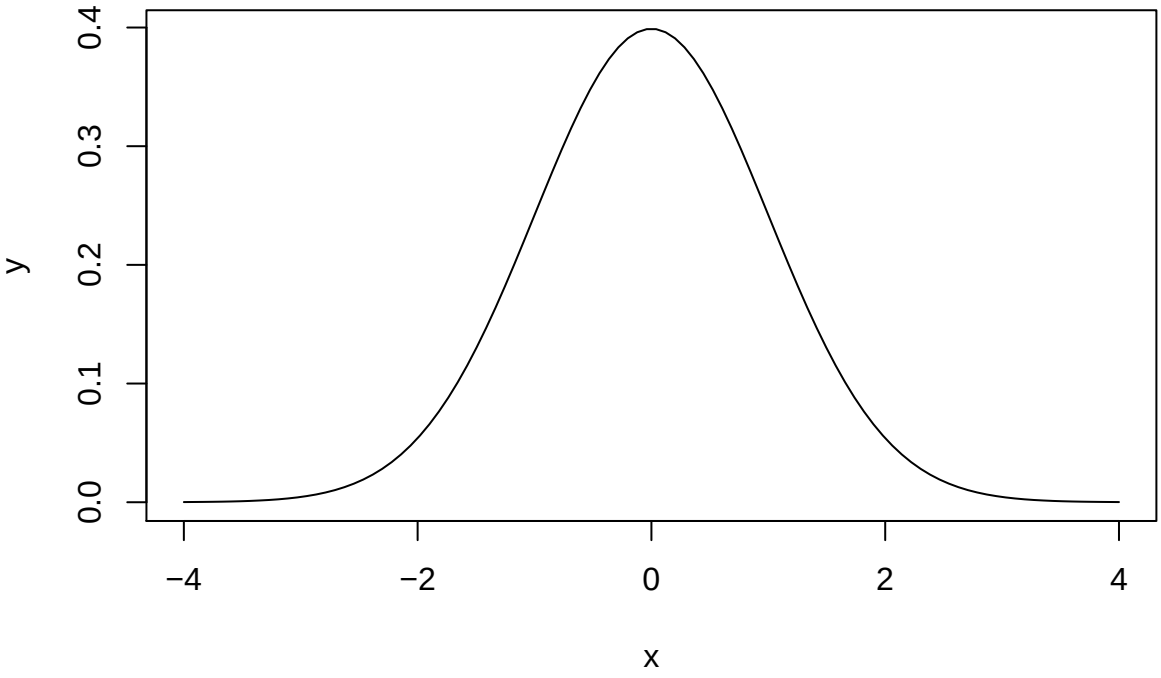
The question is whether Punxsutawney Phil is actually any good at predicting the weather. We will answer that by looking at whether he sees his shadow, and the temperature for the following 6 weeks.



Before getting to the question of whether Phil is any good as a weatherman, we'll look at the distribution of the temperature. Of the 122 observations, 56 are above freezing (above 32 degrees Fahrenheit as it is the USA). But we want to summarise the data in more detail, partly so we can ask more complicated questions and also so we can develop the theory. We will do this with the normal distribution.

The Normal Distribution

The normal distribution looks like this:



Or, for those who like equations, like this:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

x is the thing that follows a normal distribution (e.g. temperature). $f(x|\mu, \sigma^2)$ is the density: a higher density means that value of x is more likely[1]. We call $f(x|\mu, \sigma^2)$ the **probability density function**, and use it to calculate the probability that x is between two values (e.g. $Pr(-1.96 < x < 1.96|\mu = 0, \sigma^2 = 1) = 0.95$) The parameters are μ and σ^2: the mean and variance (sometikmes we use the standard deviation, σ, rather than σ^2, or even $1/\sigma^2$, the precision).

We can split the function up into bits. The first part is

$$\frac{1}{\sqrt{2\pi\sigma^2}}$$

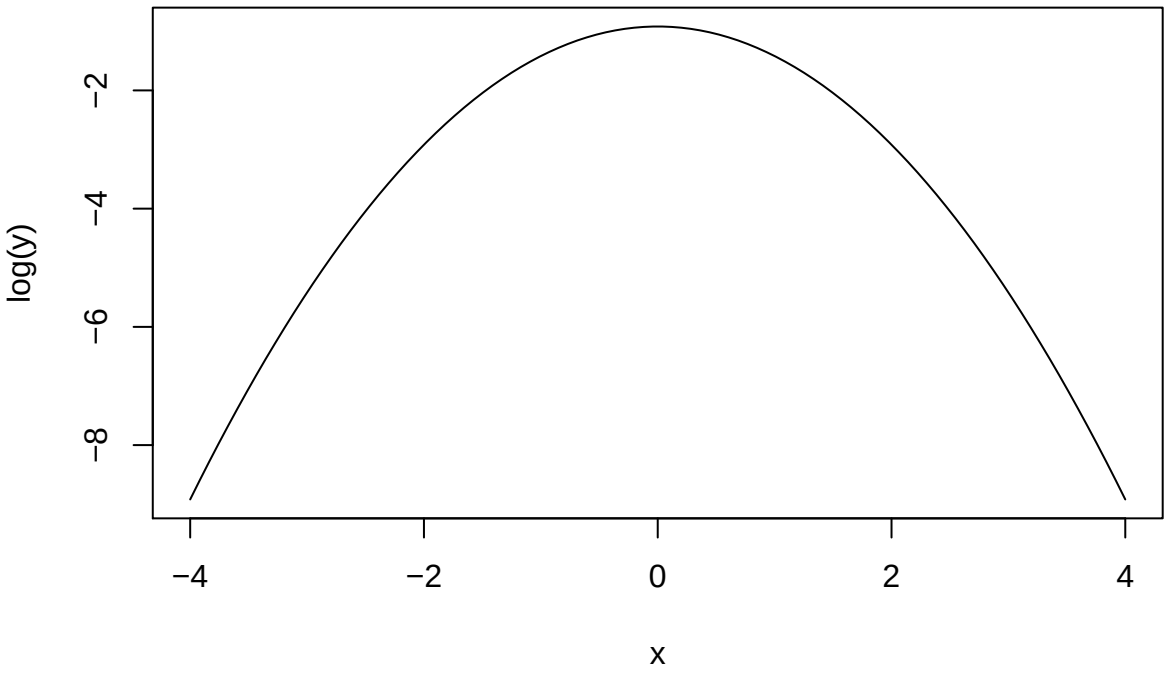
which does not depend on x, so is a constant. It is there because the area under the whole curve has to equal 1. More important is the second part

$$e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

If we take the log transformation (so we have the log probability, or later the log likelihood) we see that it is this

$$-\frac{(x-\mu)^2}{2\sigma^2}$$

So is a quadratic curve, and looks like this:



[1] this is not strictly correct, as $Pr(X = x) = 0$ for a continuous density. More strictly a higher density means that an interval around that value is more likely to contain x

The Normal Distribution: Exercises

For these exercises you will need a few extra functions. You can find them by using `source()` to import and run a script found here https://www.math.ntnu.no/emner/ST2304/2021v/Week04/NormalDistFunctions.R

Let's look at the normal distribution, and some R functions we can use to play with it. In particular:

- `rnorm()` - this simulates from a normal distribution. Takes arguments: `n` = number of observations to simulate, `mean` = mean of the normal distribution you simulate from, `sd` = standard deviation of the normal distribution you simulation from.
- `dnorm()` - this calculates the density of a normal distribution for values `x`. First argument is `x`, then `mean`. `sd`, and `log` which can = TRUE or FALSE.
- `pnorm()` - this calculates the cumulative density of a normal distribution for values `x`. This is the probability that a random draw from the distribution would be less than `x`. First argument is `q` = quantiles you want the probability for, then `mean` and `sd`.
- `qnorm()` - this calculates the values of `x` that would be the `q`th quantile. First argument is `p` = probabilities you want `x` for, then `mean` and `sd`.

Simulate 1000 data points from a normal distribution with the same mean and standard deviation as the temperature data, using `rnorm()` and plot with `hist()`

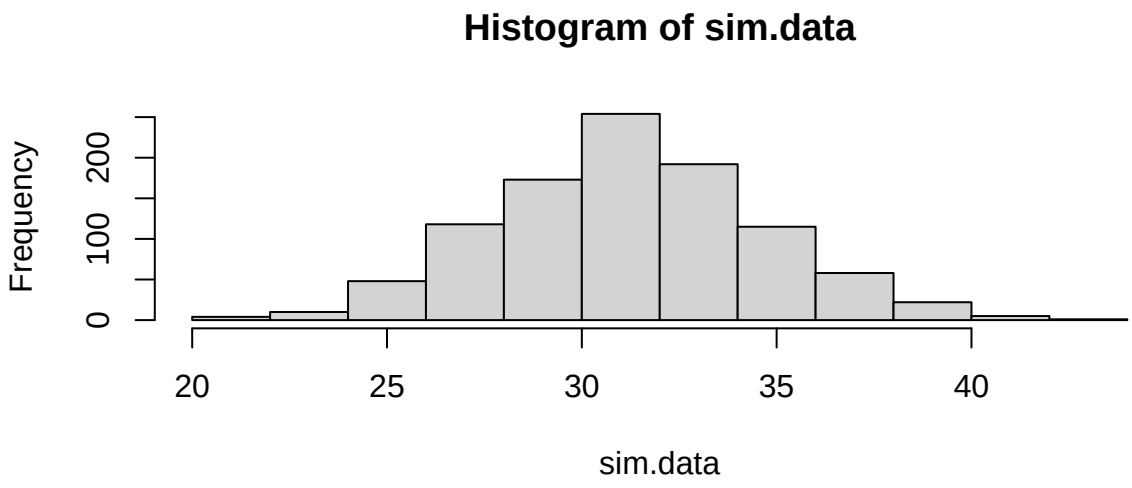
R help

You will need to edit this code.

```
MeanTemp <- mean(YOURDATA$Temperature)
sdTemp <- sd(YOURDATA$Temperature)
sim.data <- rnorm(n=5, mean = MeanTemp, sd = sdTemp)
hist(sim.data)
```

ANSWER

```
MeanTemp <- mean(GDay$Temperature)
sdTemp <- sd(GDay$Temperature)
sim.data <- rnorm(n=1000, mean = MeanTemp, sd = sdTemp)
hist(sim.data)
```



Calculate the density data from the distribution for values using `dnorm()`, but for more than 4 points and plot it with `plot()` - why is the density different from the probability (hint: the normal is continuous)

You might also want to use `seq()` to create the input for `dnorm()`. This will mean that the numbers are in ascending order rather than mixed like the raw data from the question above.

R help if you need it

```
# First, create a sequence of numbers to calculate the density on
At.data <- seq(min(YOURDATA$Temperature), # takes the minimum temperature value
               max(YOURDATA$Temperature), # takes the maximum temperature value
               length = 4) # tells R how many to have in the sequence

# create the density data
dens.data <- dnorm(At.data, mean = MeanTemp, sd = sdTemp)

# plot it
plot(x = At.data, y = dens.data, type="l")
```

ANSWER

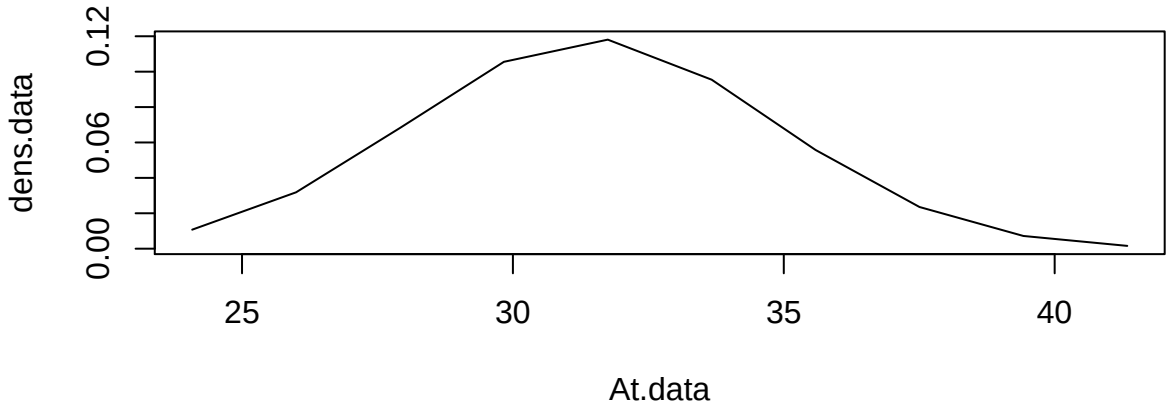
The density is different to the probability for the normal distribution because it is a continuous distribution. Therefore, the likelihood of observing any combination of observed data points for a given parameter value is 0 because there are infinite possibilities of observed data. So, instead we take the probability density of observed data points being between two values (often very close together).

R answers:

```
# First, create a sequence of numbers to calculate the density on
At.data <- seq(min(GDay$Temperature), # takes the minimum temperature value
               max(GDay$Temperature), # takes the maximum temperature value
               length = 10) # tells R how many to have in the sequence

# create the density data
dens.data <- dnorm(At.data, mean = MeanTemp, sd = sdTemp)

# plot it
plot(x = At.data, y = dens.data, type="l")
```



Calculate the cumulative density data from the distribution for values using `pnorm()` and plot this.

R code:

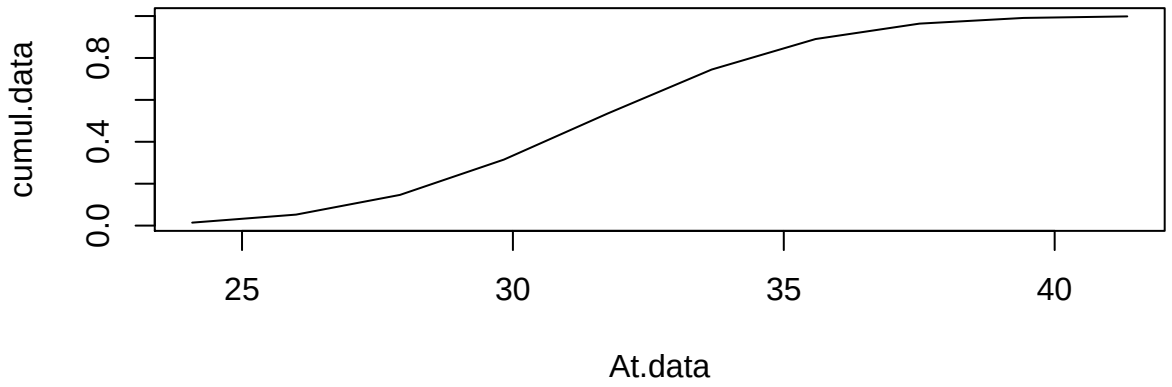
```
# At.data = a sequence from min to max temperature values
cumul.data <- pnorm(At.data,
                    mean = MeanTemp,
                    sd = sdTemp)

plot(At.data, cumul.data, type="l")
```

ANSWER

```r
# At.data = a sequence from min to max temperature values
cumul.data <- pnorm(At.data,
                    mean = MeanTemp,
                    sd = sdTemp)

plot(At.data, cumul.data, type="l")
```



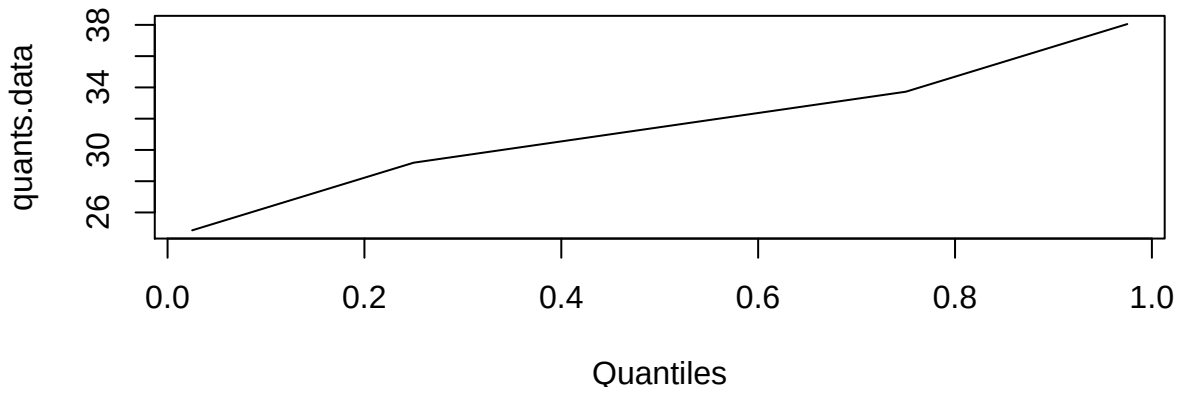
Calculate the 2.5th, 25, 50th, 75th, and 97.5th quantiles from the distribution using `qnorm()`

R code:

```r
# create vector of probabilities as input
Quantiles <- c(0.025, 0.25, 0.5, 0.75, 0.975)
# calculate quantiles
quants.data <- qnorm(Quantiles,
                     mean = MeanTemp, sd = sdTemp)
# plot
plot(Quantiles, quants.data, type="l")
```

ANSWER

```r
# create vector of probabilities as input
Quantiles <- c(0.025, 0.25, 0.5, 0.75, 0.975)
# calculate quantiles
quants.data <- qnorm(Quantiles,
                     mean = MeanTemp, sd = sdTemp)
# plot
plot(Quantiles, quants.data, type="l")
```



What is the difference between `pnorm()` and `qnorm()`?

I had a go at explaining it myself now I want to see the ANSWER

`pnorm()` takes quantiles as an input and calculates the cumulative density function at those quantiles. It shows the probability of having an observed value less than or equal to each quantile.

`qnorm()` is the inverse of `pnorm()`. It takes a vector of probabilities as an input and gives quantiles as an output. i.e. it finds the value of `x` (our variable of interest) at a given percentile of the distribution.

There is quite a nice summary with examples here: https://www.datascienceblog.net/post/basic-statistics/distributions/

Once you have had a go, you can watch the summary video to go through this section.

Click here or watch below

The Normal Distribution: the log likelihood

If we are modelling our data, we have observations y_i, then we assume they were drawn from a normal distribution with mean μ and variance σ^2. The likeihood for this data is the probability density function (now as a function of the parameters): the log-likelihood for a single datapoint is

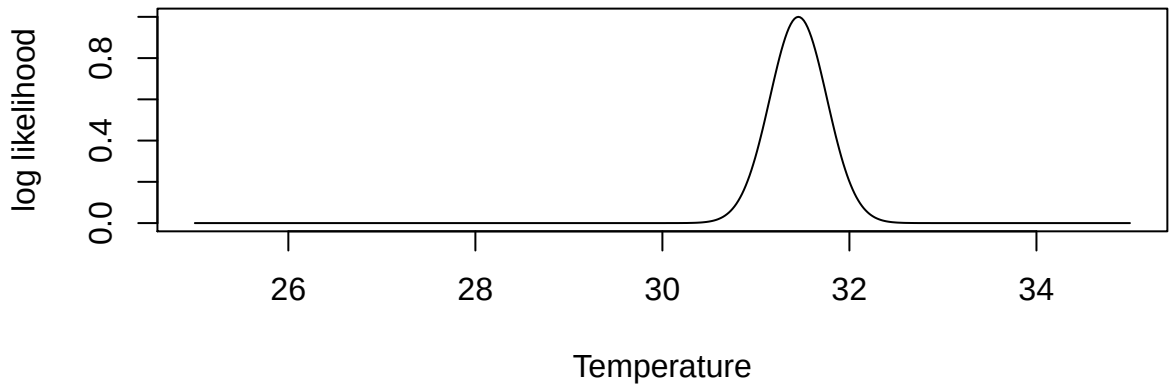
$$l(\mu, \sigma^2|y_1) = -\frac{1}{2}log(2\pi\sigma^2) - \frac{(y_1 - \mu)^2}{2\sigma^2}$$

In practice, we can calculate this with `dnorm(..., log=TRUE)`. If we have n data points, the likelihood for them all is

$$\begin{aligned} l(\mu, \sigma^2|y_1, y_2, ..., y_n) &= \sum_{i=1}^{n} \left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(y_1 - \mu)^2}{2\sigma^2} \right) \\ &= -\frac{1}{2}n\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_1 - \mu)^2 \end{aligned}$$

And we can calculate this using `sum(dnorm(..., log=TRUE))`.

```
Means <- seq(25, 35, length=500)
# sapply applies a function (here CalcNormlogLh) across a set of numbers
# (here Means)
loglhoods <- sapply(Means, CalcNormlogLh, sigma=sdTemp,
                    data=GDay$Temperature)
plot(Means, exp(loglhoods-max(loglhoods)), type="l",
     xlab="Temperature", ylab = "log likelihood")
```



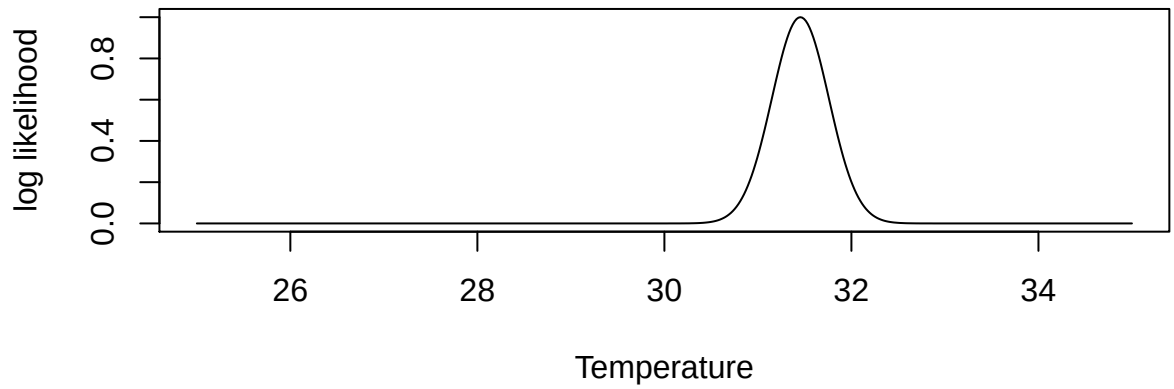
This is the likelihood for μ given a guess for σ^2. But we want to estimate these parameters, so we need to maximise this likelihood with repsect to them.

Finding the Estimate: the mean

In practice, $\hat{\mu}$ is more important, because we will be modelling μ as a function of different effects. First we can calculate the likelihood for different values of μ, using the sample estimate of the standard deviation of the data as the standard deviation of the distribution.

This code uses `CalcNormlogLh()` to calculate the likelihood. We create a vector of possible values of μ, calculate the likelihood for each of these, and then plot the likeihood against their values.

```
Means <- seq(25, 35, length=500)
# Calcualte the normal likelihood for differnet means
#  sapply() loops ove rhte vlaues of Means, and applies the CalcNormlogLh() to each one
loglhoods <- sapply(Means, CalcNormlogLh, sigma=sdTemp,
                    data=GDay$Temperature)
plot(Means, exp(loglhoods-max(loglhoods)), type="l",
     xlab="Temperature", ylab = "log likelihood")
```



You may notice that `loglhoods` is the log-likeihood, but we plot the likelihood, so we have to back-transform it, using `exp(loglhoods-max(loglhoods))`. This is the same as `likelihood/(maximum likelihood)`, so we are just scaling the y-axis. if we don't do this, we can get numerical problems: the largest value of the log-likelihood is -320.66, which give a likelihood of $5.5090705 \times 10^{-140}$.

Given this, we can estimate the maximum likelihood estimate with

```
max(loglhoods)
```

```
## [1] -320.6555
```

```
Means[loglhoods==max(loglhoods)]
```

```
## [1] 31.45291
```

In the second line `loglhoods==max(loglhoods)` tests whether each value of `loglhoods` equals the maximum value.

Deriving the m.l.e for μ

But we can do better than this: we can find the maximum with a bit of maths. We want to maximise the likelihood in the same way we did for p for the binomial distribution by doing this:

- calculate the slope
- set the slope to 0
  - we should really check that it is the maximum, not minimum
- find the parameter where the slope is 0: that is the m.l.e.

Our log-likelihood is

$$l(\mu, \sigma^2 | y_1, y_2, ..., y_n) = -\frac{1}{2} n log(2\pi\sigma^2) - \sum_{i=1}^{n} \frac{(y_1 - \mu)^2}{2\sigma^2}$$

We can ignore the $-\frac{1}{2} n log(2\pi\sigma^2)$, because it does not involve μ. And we can re-arrange the sum:

$$\begin{aligned} l(\mu, \sigma^2 | y_1, y_2, ..., y_n) &= -\sum_{i=1}^{n} \frac{(y_1 - \mu)^2}{2\sigma^2} \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i^2 - 2y_i\mu + \mu^2) \\ &= -\frac{1}{2\sigma^2} \left(\sum_{i=1}^{n} y_i^2 - 2\mu \sum_{i=1}^{n} y_i + n\mu^2 \right) \end{aligned}$$

If we differentiate with respect to μ and set that slope to zero we get

$$0 = \frac{1}{2\sigma^2} \left(2 \sum_{i=1}^{n} y_i - 2n\mu \right)$$

Which we can re-arrange so the MLE is

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n}$$

The sample mean! And it doesn't depend on σ^2.

The MLE for σ^2

This is usually less important. We are generally not interested in the standard deviation, but it is a parameter of the distribution, so it has to be estimated. What we do is differentiate with respect to σ^2, set to zero, re-arrange, and get

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2$$

For details, you can do it yourself or follow the proof at the StatLect site.

The estimate $\hat{\mu}$ is just the sample mean, and $\hat{\sigma}^2$ is the sample variance.

These two statistics can be used to summarise the whole distibution. Note that the denominator for the estimate of $\hat{\sigma}^2$ is n, not $n - 1$. But nobody actually uses this: it assumes the MLE for $\hat{\mu}$ is the true value, but in reality we do not know the true value of μ, so it turns out that using $(n - 1)$ is better because it takes into account the uncertainty

Confidence Intervals

This next video is a bit long. It tries to put the maximum likelihood estimation in the context of modelling and explain how all of the different things we have covered so far link together. Hopefully it is helpful.

Click here or watch below

We can simulate data and calculate the likelihood for different values of the mean (we will fix the standard deviation for now).

We can look at the distribution of $\hat{\mu}$, and (for example) estimate confidence intervals:

You will need to download and source the script

```
# Simulate data sets with mean 1 (mu in function below), standard deviation 2 (sigma in function below)
# For each one calculate the MLE for the mean
Sims <- SimNormMeans(mu=1, sigma=2, N=10, NReps = 10000)
hist(Sims)
quantile(Sims, c(0.025, 0.975))
```

Use this to simulate the Punxsutawney temperature data to estimate the mean temperature and its 95% confidence interval. To do this you should:

- estimate the MLEs for μ and σ^2
- Use these and `SimNormMeans()` to simulate replicate data (with the same number of data points as in the original data), and estimate the means of each simulated data set
- from the distribution of estimated means, calculate the 95% confidence interval (using the quantiles of the distribution)

Use the code above as a basis for this but use R hints if you get stuck.

R help if you need it

First, find MLE for μ and σ^2 for the Punxsutawney temperature data.

It says above that you find these by taking the mean and variance of `x`, respectively. In this case, `x` is temperature data from Punxsutawney.

You can do this in R using the functions `mean()` and `var()`.

Second, use the `SimNormMeans()` function to simulate repeat data collection of the temperature, pick a high number of repeat samples e.g. 1000-10000. (Too high will take too long). For this part, you need one extra bit of information: argument `N`, which is the number of data points. This needs to match the number in the original data.

Remember: although σ^2 is the parameter we wanted the MLE for, `SimNormMeans()` takes the input σ. So take the square root of σ^2 using `sqrt()` to get σ.

You can find this using `length()` e.g.

```
N = length(YOURDATA$Temperature)
```

`SimNormMeans()` will calculate the mean for you.

Third, use the quantile function like the example above to get a 95% confidence interval.

ANSWER

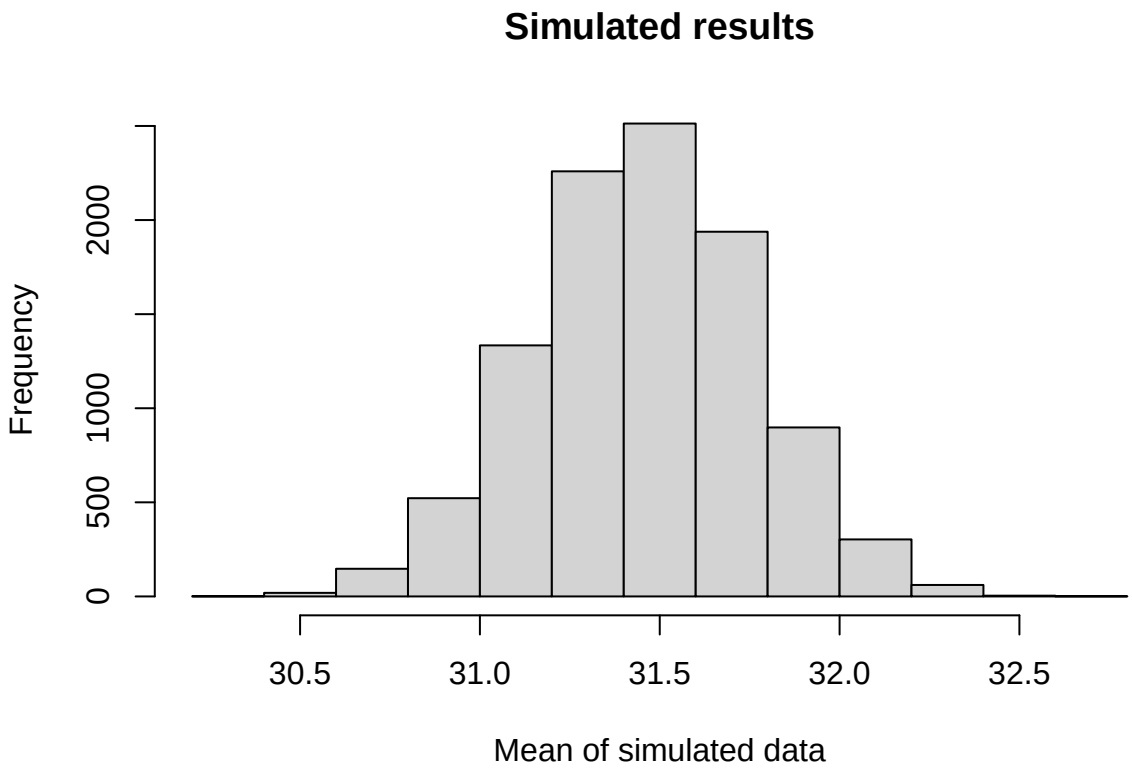
```
# Find MLE estimates of mean and variance
mu <- mean(GDay$Temperature)
sigma2 <- var(GDay$Temperature)
sigma <- sqrt(sigma2)

# Find number of data points
N <- length(GDay$Temperature)

MySims <- SimNormMeans(mu=mu, sigma=sigma, N=N,
                       NReps = 10000)

# Plot the results of the simulation (yours will be a bit different)
hist(MySims,
     xlab = "Mean of simulated data",
```

```
     ylab = "Frequency",
     main = "Simulated results")
```




```
# Get confidence intervals
quantile(MySims, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 30.85480 32.04945
```

Your answers will not be exactly the same (simulations are random)! But should be close.

The distribution of $\hat{\mu}$

We could use the simulations whenever we want to calculate the confidence intervals, but instead we can rely on some beer-inspired mathematics. It turns out that the distribution of $\hat{\mu}$ is a t-distribution.

So, we can calculate the density, quantiles etc. in the same way we did for the normal distribution:

```
dt(x=0.5, df=5)
```

```
## [1] 0.3279185
```

```
pt(q=1.5, df=5)
```

```
## [1] 0.9030482
```

```
qt(p=0.7, df=5)
```

```
## [1] 0.5594296
```

```
rt(n=2, df=5)
```

```
## [1] -0.4002121 -1.1594407
```

The t-distribution has a parameter, called `df`, which is there 'degrees of freedom', and is roughly the amount of information in the distribution.

"But what about the mean and variance?" you're thinking. Well, what we have above is the standard t-distribution. Recall that for the normal distribution we can calculate this:

$$z = \frac{x - \mu}{\sigma}$$

and z follows a normal distirbution with mean 0 and variance 1. If we do the same transformation and x follows a t distribution, z follows a standard t distribution.

So, to calculate the confidence interval for our mean, we can use the same approach as we use for the normal distribution. For the normal distribution, the interval is the mean plus/minus 1.96 times the standard error. We use 1.96 because that is the 97.5% quantile of the standard normal distribution (and -1.96 is the 2.5% quantile). So, we can just replace ± 1.96 with the relevant quantiles. We call the qˆth quantile $t_\nu(q)$, where ν is the degrees of freedom. For the mean of the normal distribution this is $n - 1$, where n is the sample size:

$(\hat{\mu} + t_{n-1}(0.025)\frac{\hat{\sigma}}{\sqrt{n}}, \hat{\mu} + t_{n-1}(0.975)\frac{\hat{\sigma}}{\sqrt{n}})$

```
# Use this function to calculate the confidence interval for the mean.
#  This is a trivial example, doing it for the numbers 1 to 5
CalcNormalMeanCI(1:5)
```

If we have enough data, this distribution look like a normal distribution

Calculate the 95% confidence interval for the mean of the Punxsutawney temperature data.

ANSWER

Use the code from above and plug in the temperature data.

```
# Use this function to calculate the confidence interval for the mean.
CalcNormalMeanCI(GDay$Temperature)
```

```
## [1] 30.85348 32.05488
```

These answers should be the same because there are no simulations, it is observed data.

How the amount of data affects confidence intervals

Those of you who are comfortable with mathematics might be able to work out from the equations how the confidence intervals are affected by the sample size (hint: look at the the standard error). For the rest of you...

Simulate data with the same mean & standard deviation as the Groundhog Day data, but with different sample sizes: 10, 100 and 100 data points. For each of these calculate the confidence intervals. How do they change as the sample size increases?

To do this you should use `rnorm()` for the simulation. With `n` = number of observations, `mean` = mu, `sd` = sigma.

Then use `CalcNormMeanCI()`.

ANSWER

```
MySims10 <- rnorm(n = 10,
                  mean=mean(GDay$Temperature),
                  sd=sd(GDay$Temperature))

MySims100 <- rnorm(n = 100,
                   mean=mean(GDay$Temperature),
```

```
                  sd=sd(GDay$Temperature))

MySims1000 <- rnorm(n = 1000,
                  mean=mean(GDay$Temperature),
                  sd=sd(GDay$Temperature))

# Get confidence intervals
CalcNormalMeanCI(MySims10)
```

```
## [1] 29.35147 32.80877
```

```
CalcNormalMeanCI(MySims100)
```

```
## [1] 30.45799 31.78492
```

```
CalcNormalMeanCI(MySims1000)
```

```
## [1] 31.53963 31.95947
```

Your results will not be the same because each simulation is random, but you should see the confidence interval gets narrower as we get more observations (higher sample size).

(Optional) If you want to some more advanced coding, calculate the standard errors for different sample sizes, and plot the standard error against sample size.

ANSWER

To do this, you need to create your own function to calculate the standard error. Then apply it to your simulated datasets above and then plot the result.

The standard error =

$$\frac{\hat{\sigma}}{\sqrt{n}}$$

```
# Function
SECalc <- function(x){
  # calculate sigma (standard deviation)
  sigma <- sd(x)
  # find number of data points and take squareroot
  n <- sqrt(length(x))

  # calculate SE

  SE <- sigma/n

  return(SE)
}

# try it out on simulated data

SECalc(MySims10)
```

```
## [1] 0.8054954
```

```
SECalc(MySims100)
```

```
## [1] 0.3360558
```

```
SECalc(MySims1000)
```

```
## [1] 0.1070273
```

Again, your answers won't be the same as mine but you should see a pattern of the SE getting smaller as the sample size increases.

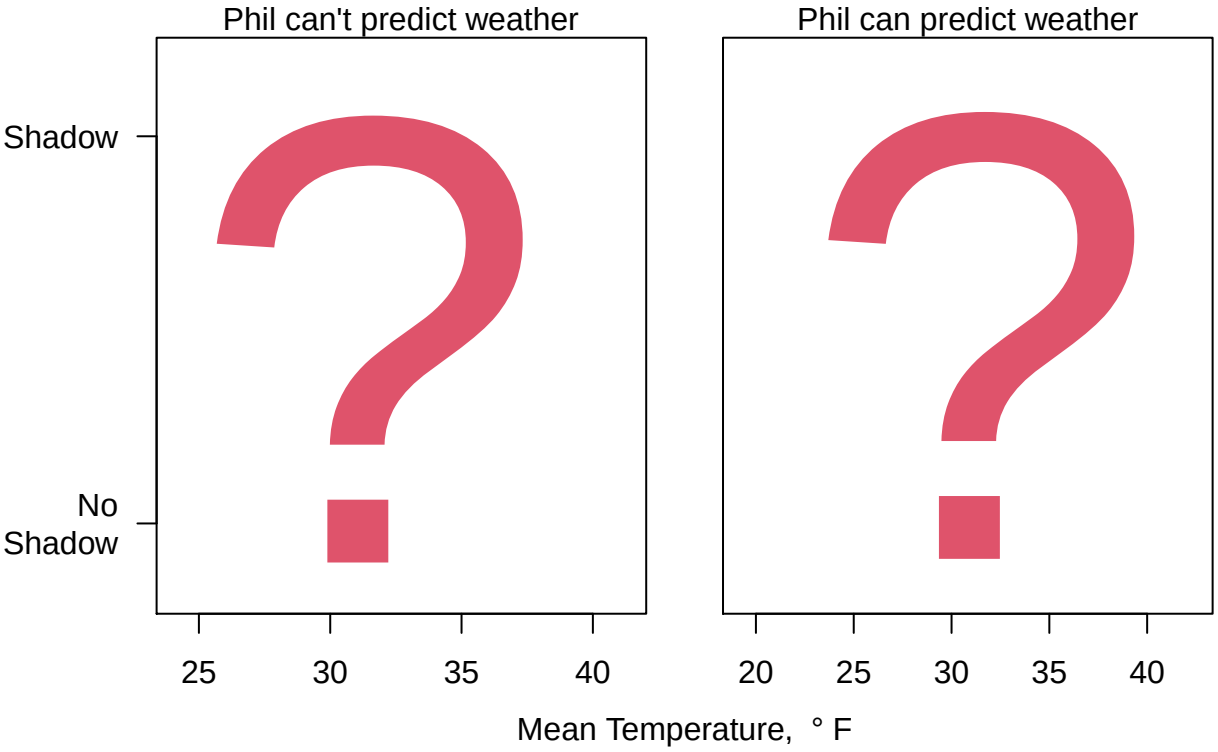
Predicting the End of Winter

So far we have been learning about statistical inference and statisical programming. Now we can start to use this in modelling. We will start with a simple model (a t-test), but put it in the context of maximum likelihood.

The legend is that if Punxsutawney Phil sees his shadow, there will be 6 more weeks of winter. So if he is good at predicting winter, we should see lower average temperatures in the 2 months after the prediction.

The Modelling

What would we expect if Punxsutawney Phil can predict winter, and if he cannot? Draw some pictures - you don't need to be too precise.



ANSWER

If Phil cannot predict winter, you would expect to see no difference in the temperature when Phil can see his shadow or not.

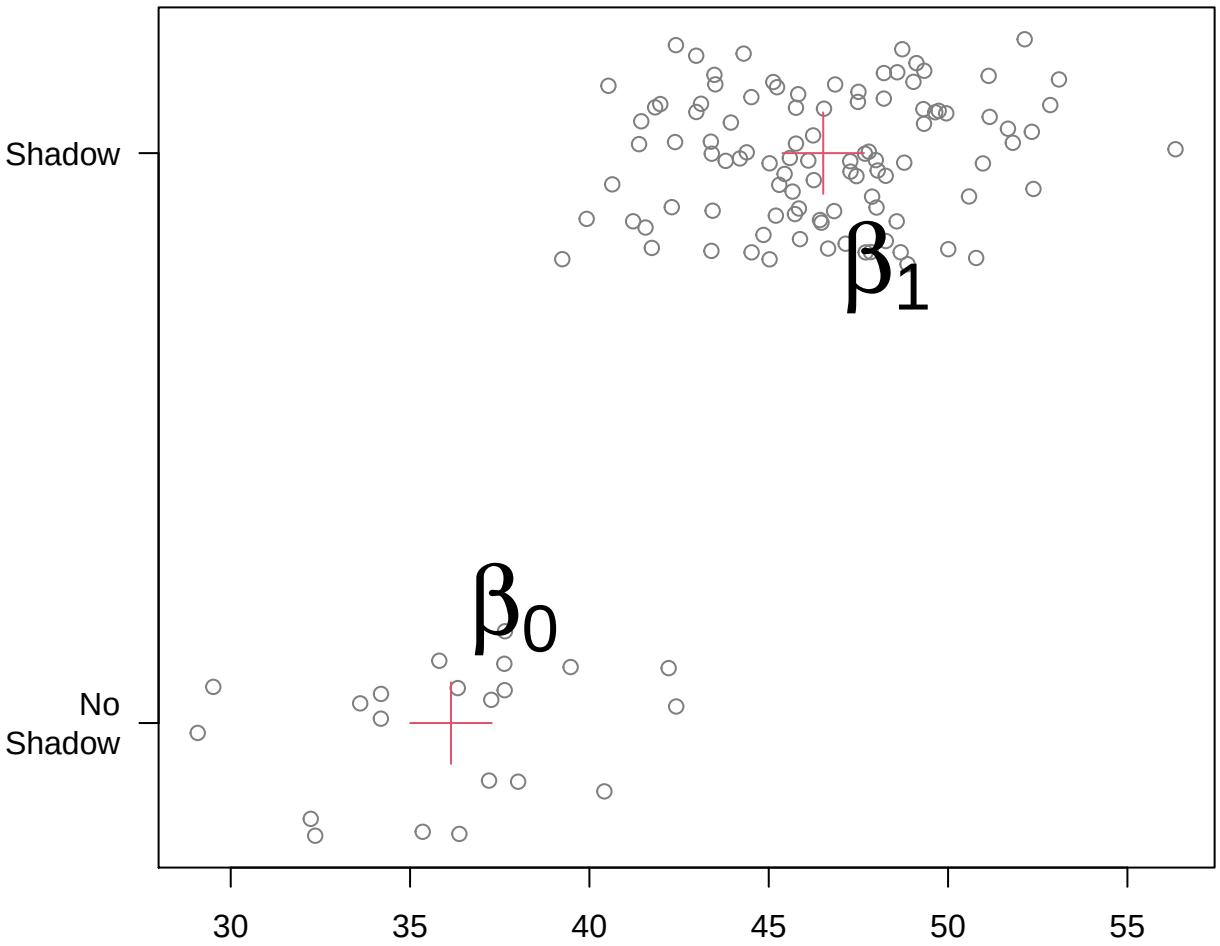
If Phil can predict winter, you would expect temperatures to be colder when he can see his shadow and warmer when he cannot. I.e you expect to see a **difference**.

We have to build a model where there are different mean temparatures when he does see his shadow, and when he does not.

We can do this in a few ways, but here we will assume $y_i \sim N(\mu_i, \sigma^2)$ (i.e. the data follow a normal distribution with the same variance). We also will define a covariate X_i: $X_i = 1$ if Phil saw his shadow (and hence predicted winter),$X_i = 0$ if he did not.

The core of the modelling is μ_i. We are asking whether it depends on X_i. We can write the model like this:

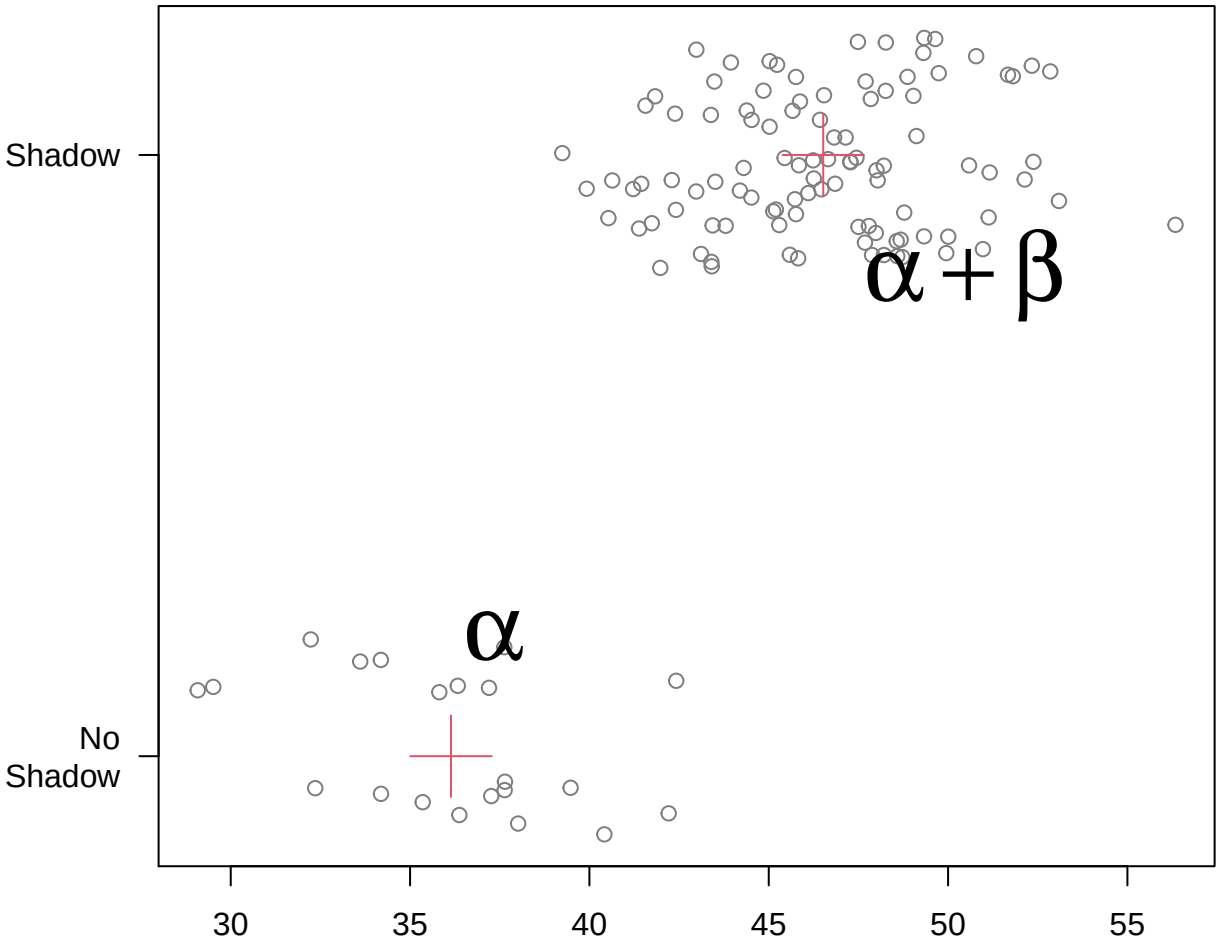
$$\mu_i = \begin{cases} \beta_0 & \text{if } X_i = 0 \\ \beta_1 & \text{if } X_i = 1 \end{cases}$$



If the legend is correct, then μ_i is smaller when $X_i = 1$. Thus, we are interested in the difference between β_0 and β_1. So it makes more sense to focus on that:

$$\mu_i = \begin{cases} \alpha & \text{if } X_i = 0 \\ \alpha + \beta & \text{if } X_i = 1 \end{cases}$$

The difference is β, and this is what we are interested in, in particular whether it is 0.



We can also write the model as $\mu_i = \alpha + \beta X_i$. The difference is β (because X_i can only be 0 or 1). Although this seems a bit indirect, it is the easiest to extend to more complex models, as we will see later in teh course.

Calculating the Likelihood

The log-likelihood is not too difficult to write down

$$\log L(\mu_i, \sigma | x_1, ..., x_n) = C - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu_i)^2$$

where mu_i is described above. We now have 3 parameters (two for the means, and the standard deviation). We can look at the likelihood for the difference, using the MLEs for the other parameters.

```
# Use this function to simulate the MLE for the difference between 2 groups.
#  XX is an indicator (0 or 1) for which group the data is in
#  YY is the data (e.g. temperature)
# nSims should be set to be much larger than 5
SimttestLhood(XX=GDay$Shadow, YY=GDay$Temperature, nSims=5)
```

Run more than 5 simulations and look at the likelihood for the difference. What is the mean and the 95% confidence interval?

ANSWER

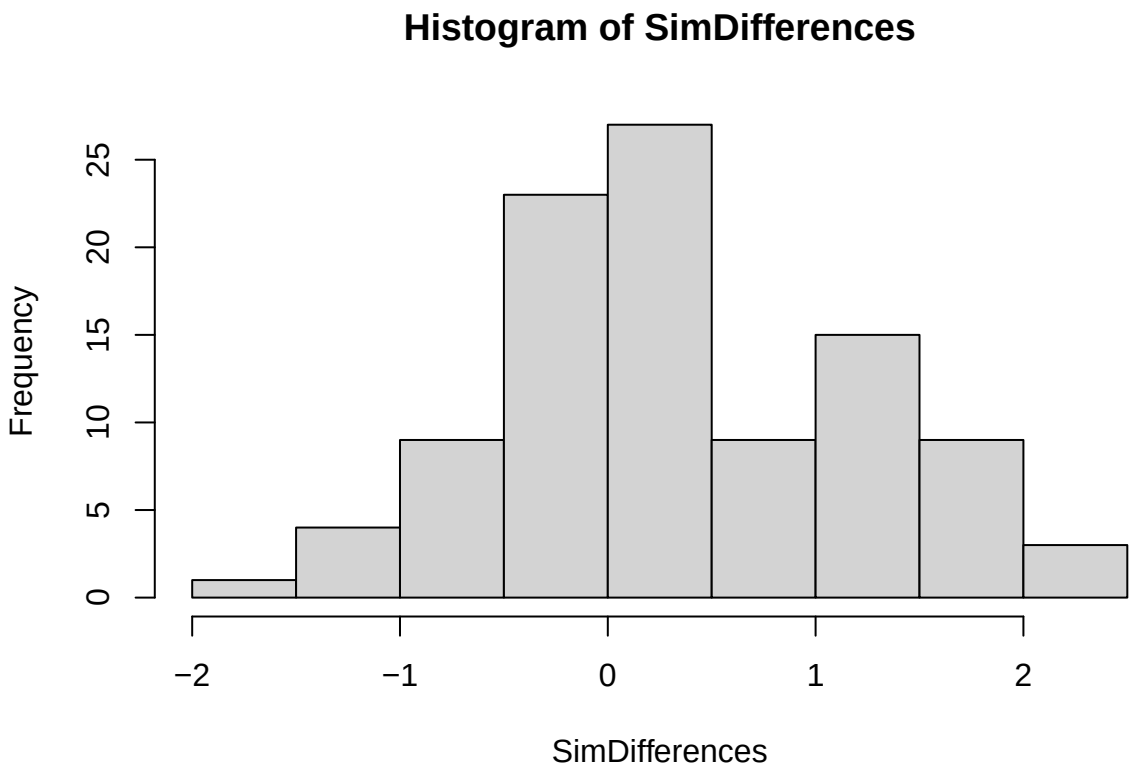
```
# Use this function to simulate mean differences between 2 groups
```

```
# This will give 100 simulated datasets

SimDifferences <- SimttestLhood(XX=GDay$Shadow,
                                YY=GDay$Temperature,
                                nSims=100)

# Look at the simulated mean differences

hist(SimDifferences)
```




```
# Calculate confidence intervals using quantile function

quantile(SimDifferences, c(0.025, 0.975))
```

```
##      2.5%     97.5%
## -1.335984  2.053809
```

t-tests In Practice

In practice, we don't need to simulate the likeihood as a student has already done the maths for us. It turns out that the difference also follows a t-distribution. And R has functions to do the calculations for us:

```
t.test(GDay$Temperature[GDay$Shadow], GDay$Temperature[!GDay$Shadow], var.equal = TRUE)
```

There is much more about this in Chapter 3 of *A New Stats with R*.

- what is the estimate of the difference in temperature between when Phil sees his shadow and when he does not?
- what is the confidence interval?
- how sure can you be about the direction of any difference? And how large could a difference be?
- do you think Punxsutawney Phil can predict whether there will be another 6 weeks of winter?

ANSWER

This takes a few steps. First, run the t-test, it will print a result.

Read the result carefully to find the information you need.

```
# Run t-test

t.test(GDay$Temperature[GDay$Shadow], GDay$Temperature[!GDay$Shadow], var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  GDay$Temperature[GDay$Shadow] and GDay$Temperature[!GDay$Shadow]
## t = 0.46769, df = 120, p-value = 0.6409
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.224467  1.981842
## sample estimates:
## mean of x mean of y
##  31.51937  31.14068
```

```
# Find the difference

31.51937 - 31.14068
```

```
## [1] 0.37869
```

The output does not directly show the difference but you can calculate it from the means of x and y which are shown at the bottom of the output. Here it is a difference of 0.37869 degrees Fahrenheit.

To work out the direction, you need to know which group is which. In this case, Shadow = TRUE is the higher mean. So, when Phil sees his shadow, the temperature is 0.37869 degrees warmer than when he doesn't.

The 95% confidence interval is -1.22 to 1.98, which spans 0. Therefore, you cannot be sure of any direction of difference. It could be as low as over -1 and up to +2 (for the most plausible range).

As a result, I do not think Phil can predict the end of winter (sorry Phil).

Finish with Video 4

Click here or watch below