StatNotes

http://www.ling.upenn.edu/~joseff/rstudy/week3.html#Normality

The Normal Distribution

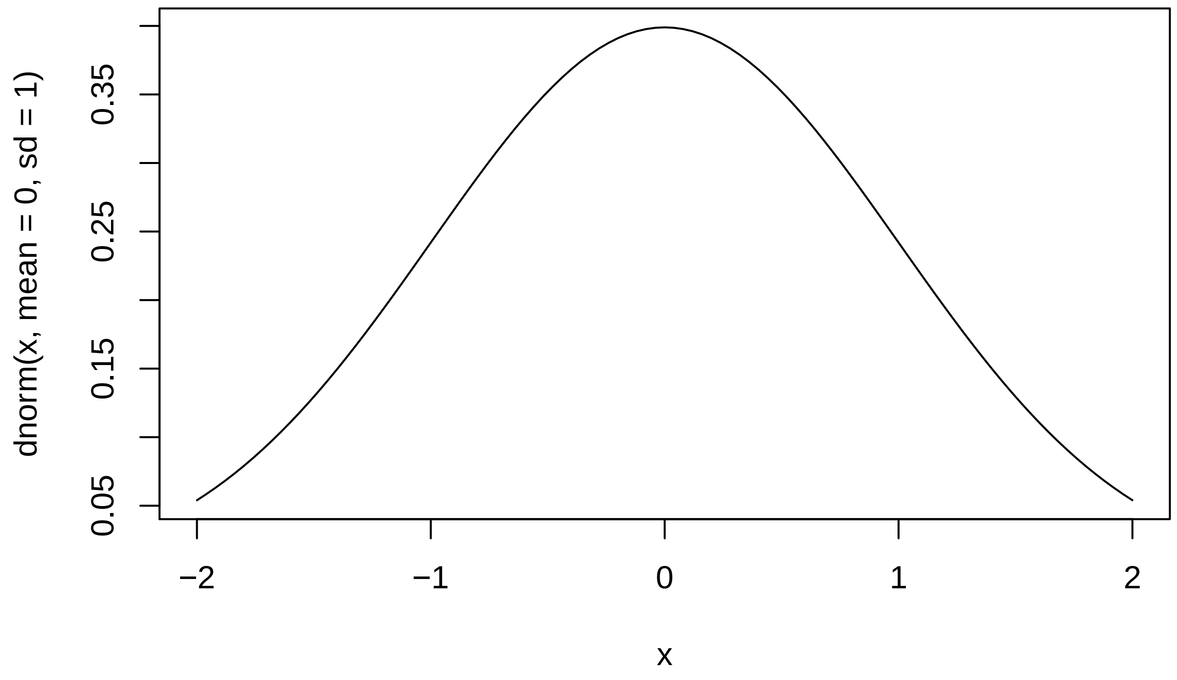
It is generally assumed that most population distributions are normal, or at least could be transformed to be normal. What is nice about a normal distibution is that it can be defined strictly by a mean (&mu) and standard deviation (&sigma).

R has a number of built in functions that give you access to a few statistical distributions (see here for a comprehensive list). Here are the functions for accessing the normal distribution:

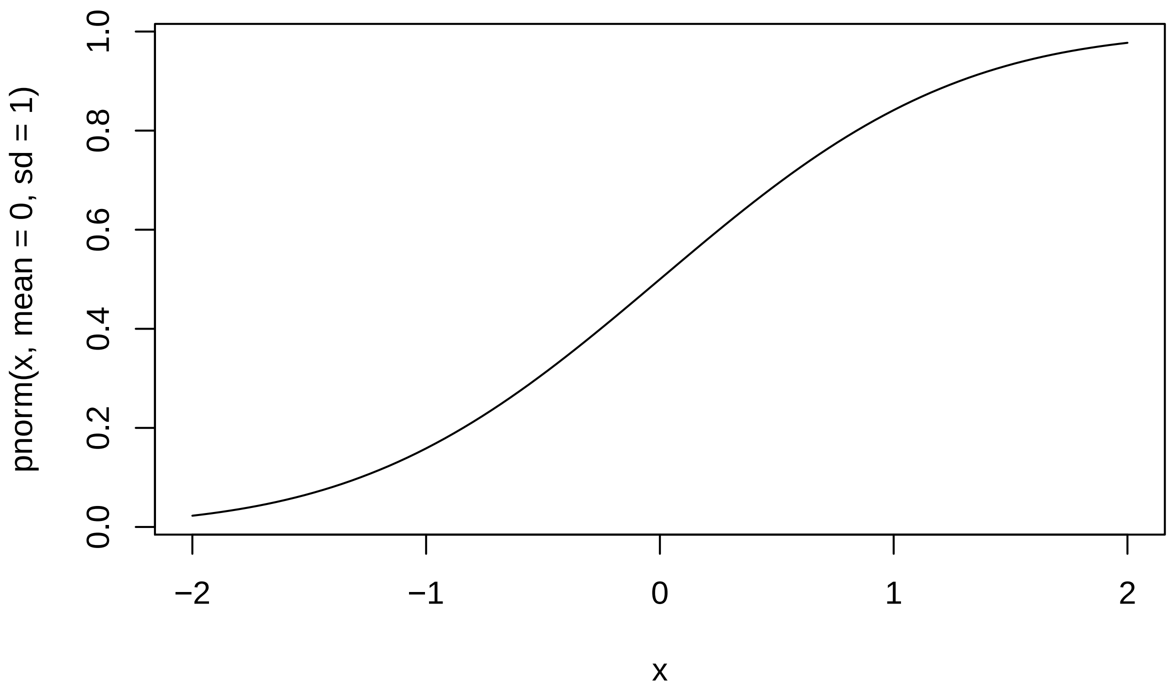
- dnorm(x, mean, sd) Gives the density of a normal distribution with a given mean and sd at x
- pnorm(q, mean, sd) Gives the probabilty of getting a value &le q from the cumulative probability function (the percentile)
- qnorm(p, mean, sd) Gives the value that is the pth percentile from the cumulative probability function
- rnorm(n, mean, sd) Gives n random numbers from a normal distribution

To plot any of these distributions, we can use the curve() function, which takes an algebraic function as its first argument, then plots the function from some value to some other value.

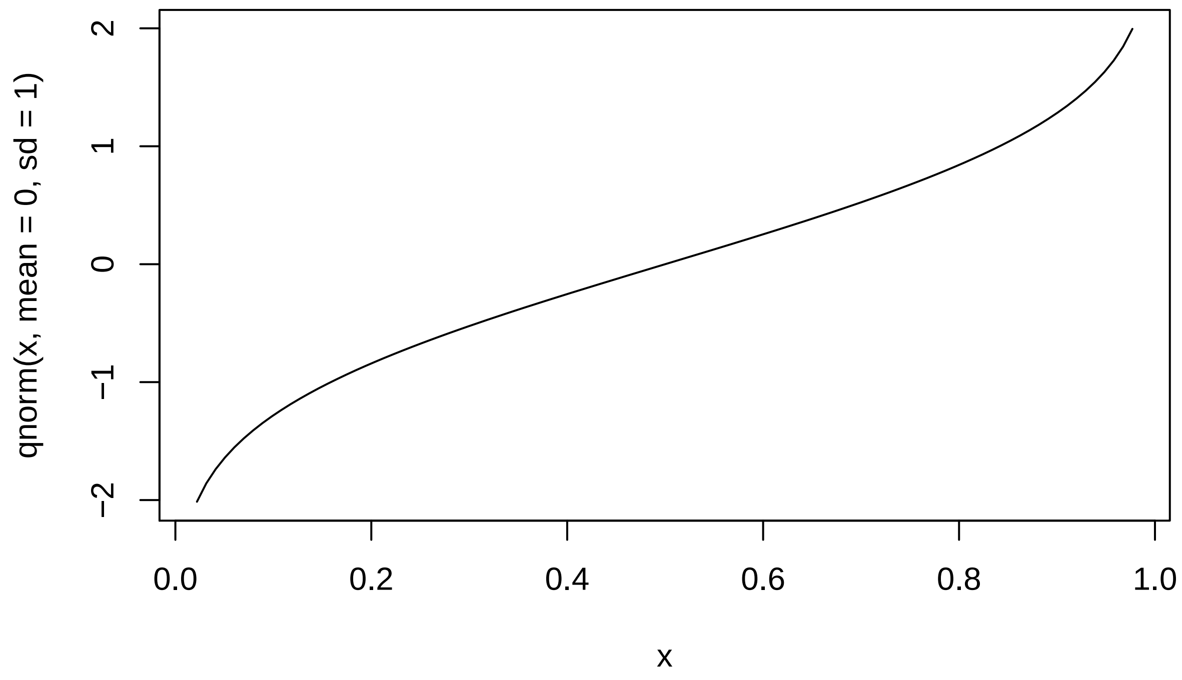
```
curve(dnorm(x, mean = 0, sd = 1), from = -2, to = 2)
```



```
##plots the shape of the density function
   curve(pnorm(x, mean = 0, sd = 1), from = -2, to = 2)
```



```
##plots the cumulative probability function
curve(qnorm(x, mean = 0, sd = 1), from = 0.022, to = 0.977)
```



```
##plots the percentiles against their values
```

An Example

Following the discussion in Johnson 2008, lets look at a ficticious rating task. 36 subject are given a sentence, and are asked to rate its grammaticality on a scale of 1-10. First, let's simulate the data. I do this a little differently than Johnson 2008. I don't know if the difference is significant, but I think my way is more principled.

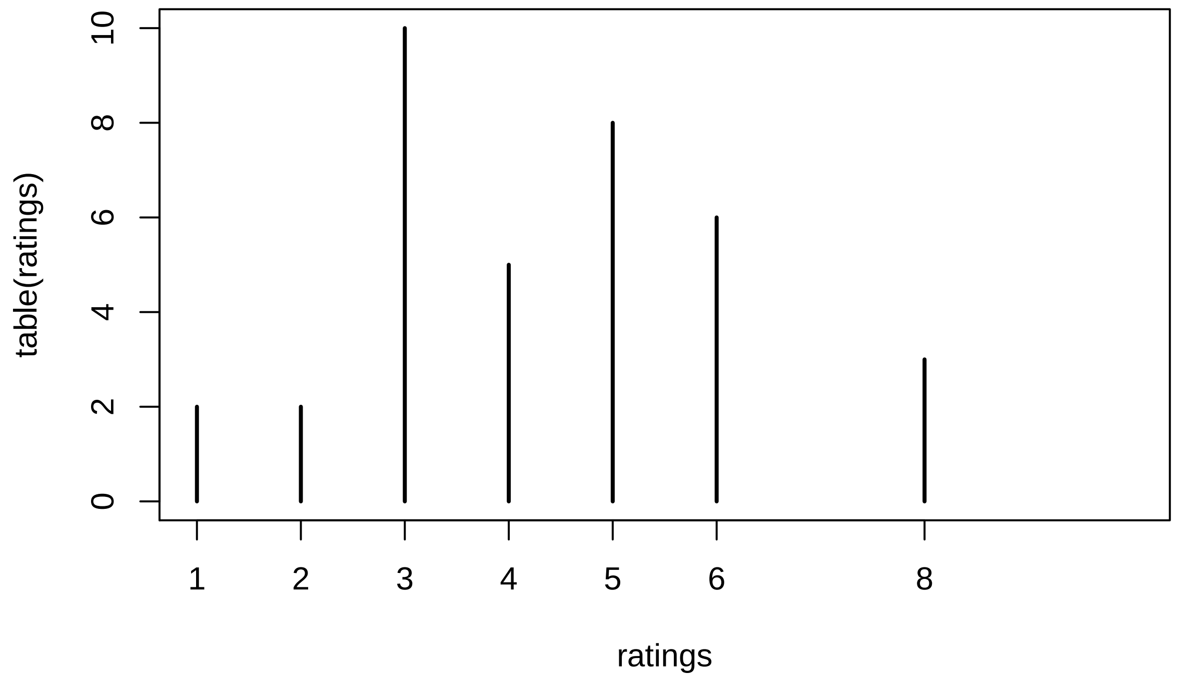
```
samp.prob<-dnorm(1:10,mean = 4.5,sd = 2)
## returns the probability of getting a particular rating assuming
## &mu = 4.5 and &sigma = 2
ratings <- sample(1:10, 36, replace = T, prob = samp.prob)
# ^^ this is the step to repeat to get new ratings
```

Now that we've simulated the data, lets look at the frequencies of each response using table() and plot a histogram of it. (The default behavior for plotting a table is as a histogram.)

```
table(ratings)
```

```
## ratings
##  1  2  3  4  5  6  8
##  2  2 10  5  8  6  3
```

```r
plot(table(ratings),xlim = c(1,10))
```



Here, we can obviously see how the data has clustered. It looks the sentence got ratings of 5 the most, followed by 4. With this task, what is the probability of getting a particular response? We can approximate this by dividing the frequency of each response by the number of tokens.

```r
table(ratings) / 36
```

```
## ratings
##          1          2          3          4          5          6
## 0.05555556 0.05555556 0.27777778 0.13888889 0.22222222 0.16666667
##          8
## 0.08333333
```

Based on the data I generated for the plot above, the probability of the sentence recieving a rating of 5 &asymp 28%, and the probabilty of getting a 4 rating &asymp 22%. The probability of getting a 4 or 5 &asymp 50%. This is a little higher than you would expect from the population we sampled.

```r
sum(samp.prob[4:5])
```

```
## [1] 0.3866681
```

```r
## should be 0.3866681
```

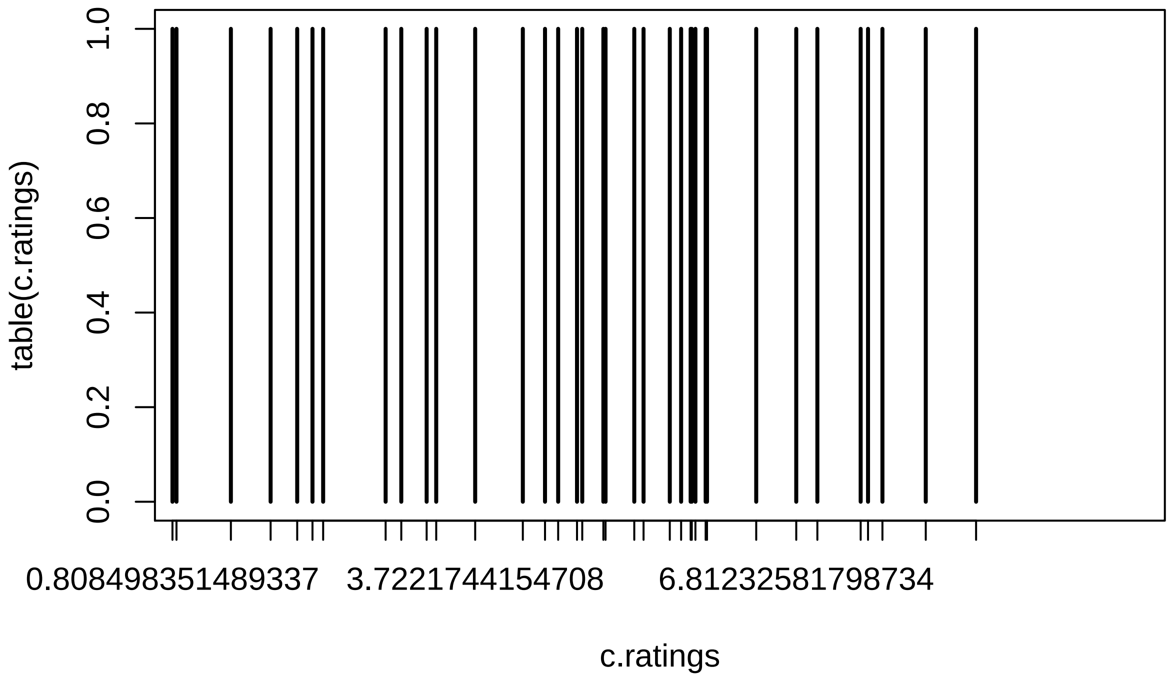
Continuous measurements

What would happen if we hadn't restricted subjects to giving us an integer rating, but had instructed them to report grammaticality by moving a slider left to right. We would then be collecting data with continuous values. Let's simulate and examine the data:

```
c.ratings <- rnorm(36,mean = 4.5,sd = 2)
## rnorm() draws random samples from a normal distribution.
table(c.ratings)
```

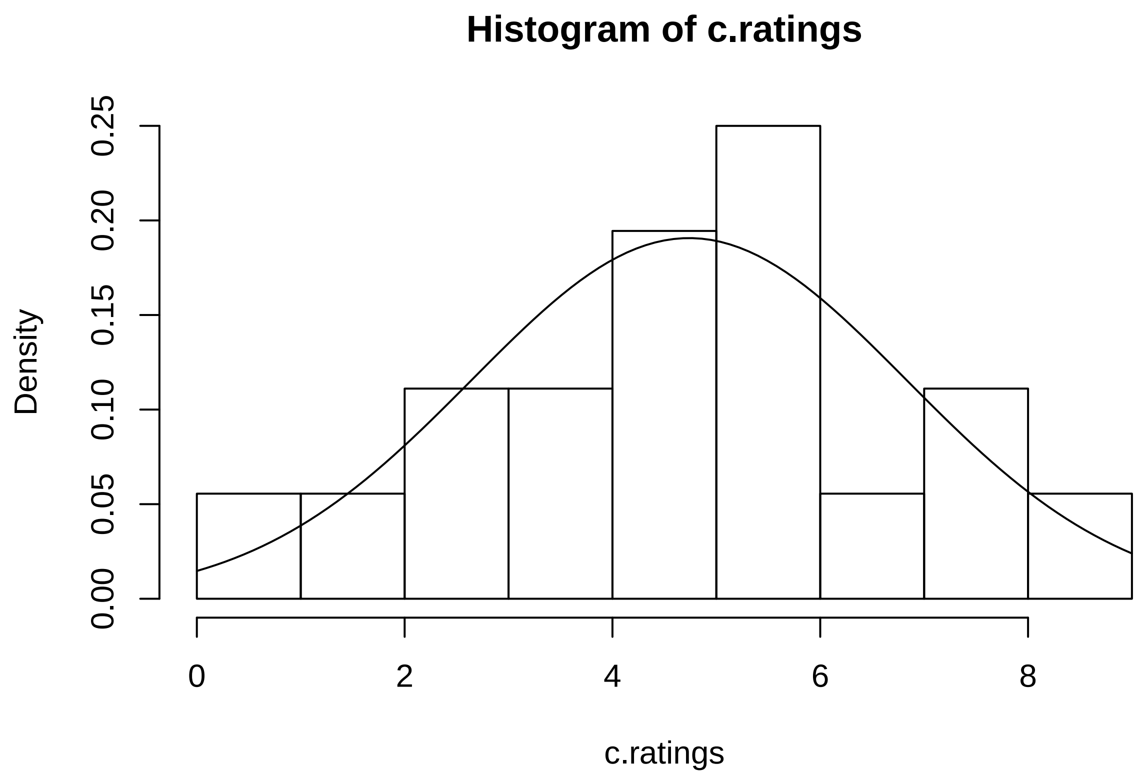
```
## c.ratings
## 0.808498351489337 0.847356635853598  1.37088356835411  1.75341490718609
##                 1                 1                 1                 1
##  2.00933124263362   2.1564007594199  2.25896035164065  2.86021813711715
##                 1                 1                 1                 1
##  3.01095302880586  3.25516255156686  3.34725600110673   3.7221744154708
##                 1                 1                 1                 1
##  4.18089926314672  4.39437017027817  4.52086995615395  4.70242267932271
##                 1                 1                 1                 1
##  4.75272837431397  4.95576737361886  4.97685870515977  5.25346387483643
##                 1                 1                 1                 1
##  5.34272983700947  5.59510202533459  5.70425140101536  5.79596587946362
##                 1                 1                 1                 1
##  5.80694562527598  5.84197321516925  5.94041108845611  5.95344161840095
##                 1                 1                 1                 1
##  6.42693275302467  6.81232581798734  7.01552929530336  7.43231454104879
##                 1                 1                 1                 1
##   7.5032577471963  7.64175348136197  8.05875006568045  8.54327187179069
##                 1                 1                 1                 1
```

```
plot(table(c.ratings),xlim = c(1,10))
```



So, you can see how useless it is like this. The probability of getting the same value on a continuous scale is infinitely low. If we did get the same value twice, it would actually be evidence of the limited precision of our measurement tools. We could try binning the continuous values, then plotting a histogram of frequencies within each bin. hist() does this automatically. We'll give the argument prob the value True, so that the y-axis values will be the probability of an observation appearing in that bin. However, if we are willing to assume normality, we can do better than this by examining the normal density curve. Here, we'll overlay it on the histogram we just plotted.

```r
hist(c.ratings,prob = T)
curve(dnorm(x,mean = mean(c.ratings),sd = sd(c.ratings)),add = T)
```



What is nice about the normal distribution density function is that you can calculate the probability of a number appearing within a given range by finding the area under the curve.

Normality of Sample Means

If you were to take a random sample from a normally distributed population 1000 times, you would probably have 1000 different sample means. However, those sample means themselves are normally distributed, and the larger the sample, the lower the variance.

Not surprisingly, if the variance of the population were lower, the sample variance is also lower. The variance of the possible mean values is called the Standard Error of the Mean, and can be calculated as the standard deviation of the sample divided by the square of the size of the sample. The 95% confidence interval of a mean &asymp the sample mean ± (SEM*1.96). Let's look at our ratings example.

```
r.mean<-mean(c.ratings)
##the sample mean
r.sem<-sd(c.ratings)/sqrt(length(c.ratings))
c(r.mean-(r.sem*1.96),r.mean+(r.sem*1.96))
```

```
## [1] 4.054085 5.421079
```

```
##95% of means of samples of the same size
##drawn from the same population will be between
##these two values
```

What We've Got

As Johnson 2008 points out, this reasoning about distributions buys us the following: * We can make probability statements about normal distributions * We can calculate sample means and variances * Sample means will fall into a normal distribution * You can estimate the variance of means from the sample mean and variance * Therefor we can make probability statements about population means.

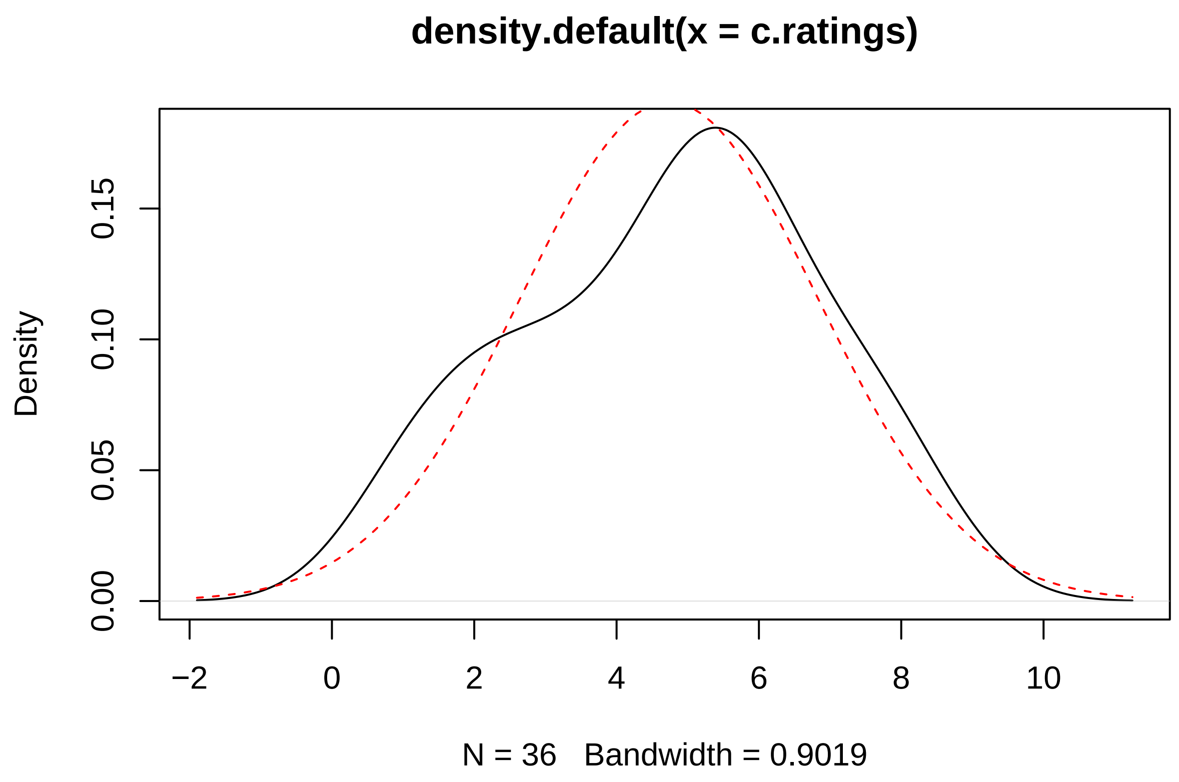
This now gives us a boot strap into testing hypotheses about populations from our samples.

Normality Testing

Since the normal distribution is so important to many assumptions in statistical test, it's good to know some ways to gauge the normality of a sample. If your data is not normally distributed, then the results of most statistical tests and regressions can be meaningless, so you really really need to inspect the data this way. Here are two graphical techniques, and one statistical test.

First, you can plot the empirical density of the data, and compare that to the density function of a normal distribution with the same mean and variance.

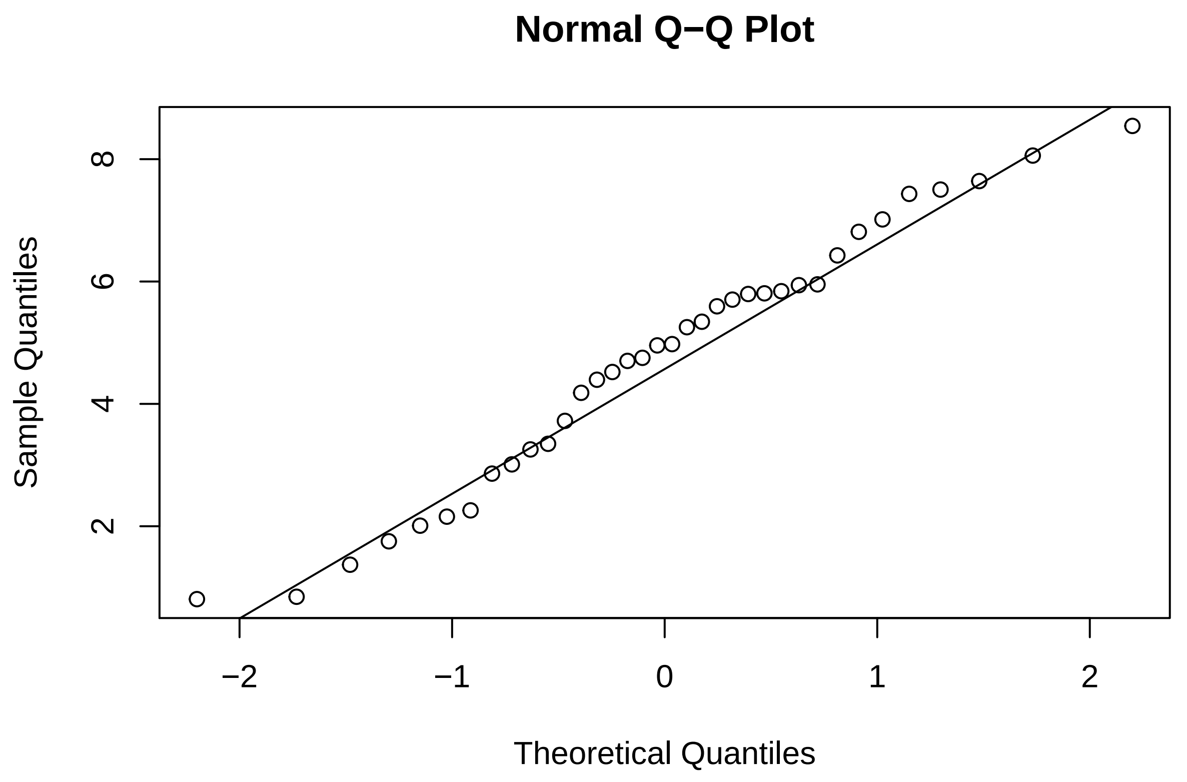
```r
plot(density(c.ratings))
curve(dnorm(x,mean = mean(c.ratings),sd = sd(c.ratings)),add = T,lty =2, col = "red")
```



Just eyeballing it, the sample looks pretty normal, and it should, since it was randomly generated from a normal distribution.

Another graphical technique is the qqplot.

```r
qqnorm(c.ratings)
qqline(c.ratings)
```



What this plot does is compare which quantile a point empirically is in the data (the y axis) to what quantile that point would be if it had been drawn from a normal disribution with the mean and sd of the sample (the x axis). I find these hard to interpret sometimes. What is important is that if the points are closely distributed to the line, then the distribution is fairly normal.

The statistical test for normality is the Shapiro-Wilk normality test. It returns a significant p-value if the distribution is significantly non-normal.

```
shapiro.test(c.ratings)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  c.ratings
## W = 0.9708, p-value = 0.4478
```

The p-value is > 0.05, so we cannot reject the hypothesis that the distribution is normal.

Tests for the mean

Since the properties of a normal distribution are rather well known, we can test hypotheses about means of distributions. The questions we can ask are: Could this sample have been drawn from a population with mean equal to, greater or less than x? Could these two samples have been drawn from the same populations? Are the populatio means of these two samples greater or less than eachother?

t-tests

The t-test is a test for the job. **Note** t-tests can only be performed meaningfully on data that is normally distributed.

In practice, we compute a t-statisticwhich measures the number of standard deviations that ^ ß1 is away from 0. If there really is no relationship between X and Y , then we expect that (3.14) will have a t-distribution with n-2 degrees of freedom. The tdistribution has a bell shape and for values of n greater than approximately 30 it is quite similar to the normal distribution. Consequently, it is a simple matter to compute the probability of observing any value equal to |t| or larger, assuming ß1 = 0. We call this probability the p-value. Roughly p-value speaking, we interpret the p-value as follows: a small p-value indicates that it is unlikely to observe such a substantial association between the predictor and the response due to chance, in the absence of any real association between the predictor and the response. Hence, if we see a small p-value, then we can infer that there is an association between the predictor and the response. We reject the null hypothesis—that is, we declare a relationship to exist between and Y —if the p-value is small enough. Typical p-value cutoffs for rejecting the null hypothesis are 5 or 1%.

One Sample t-test Let's look at some data from Daalgard 2008. This data is of the daily energy intake of some women.

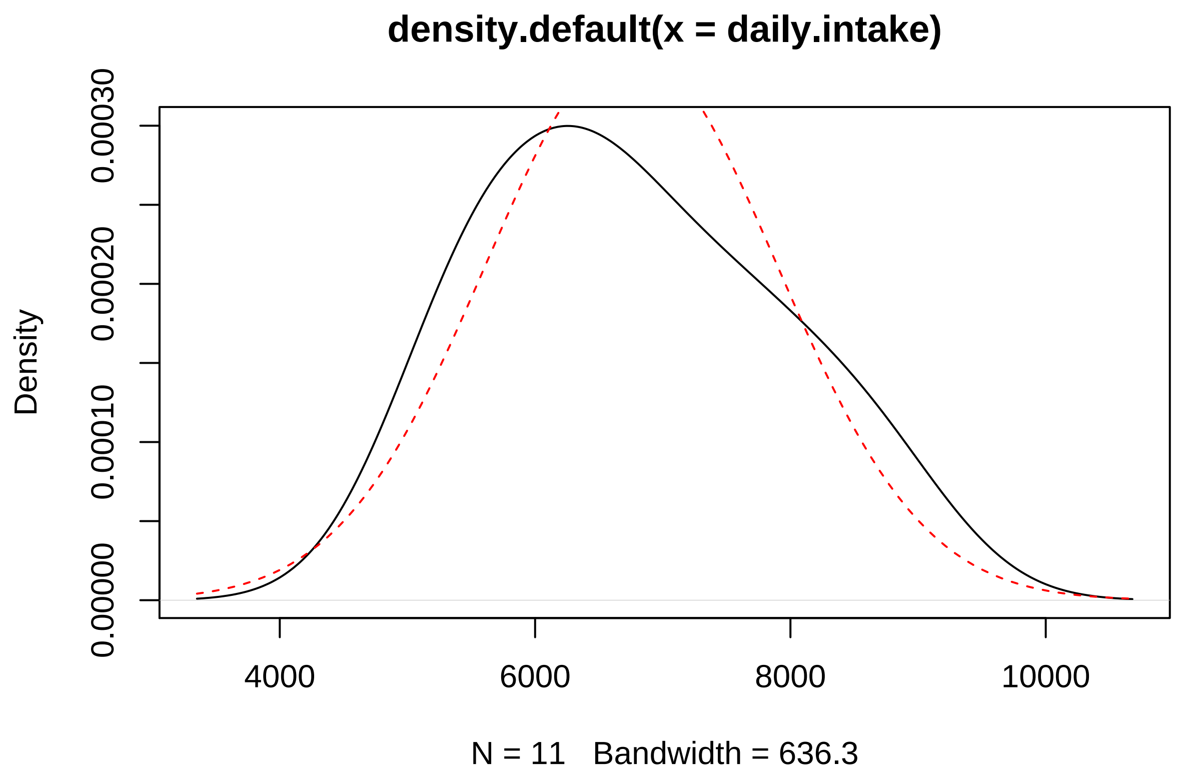
```r
daily.intake<-c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770)
mean(daily.intake)
```

```
## [1] 6753.636
```

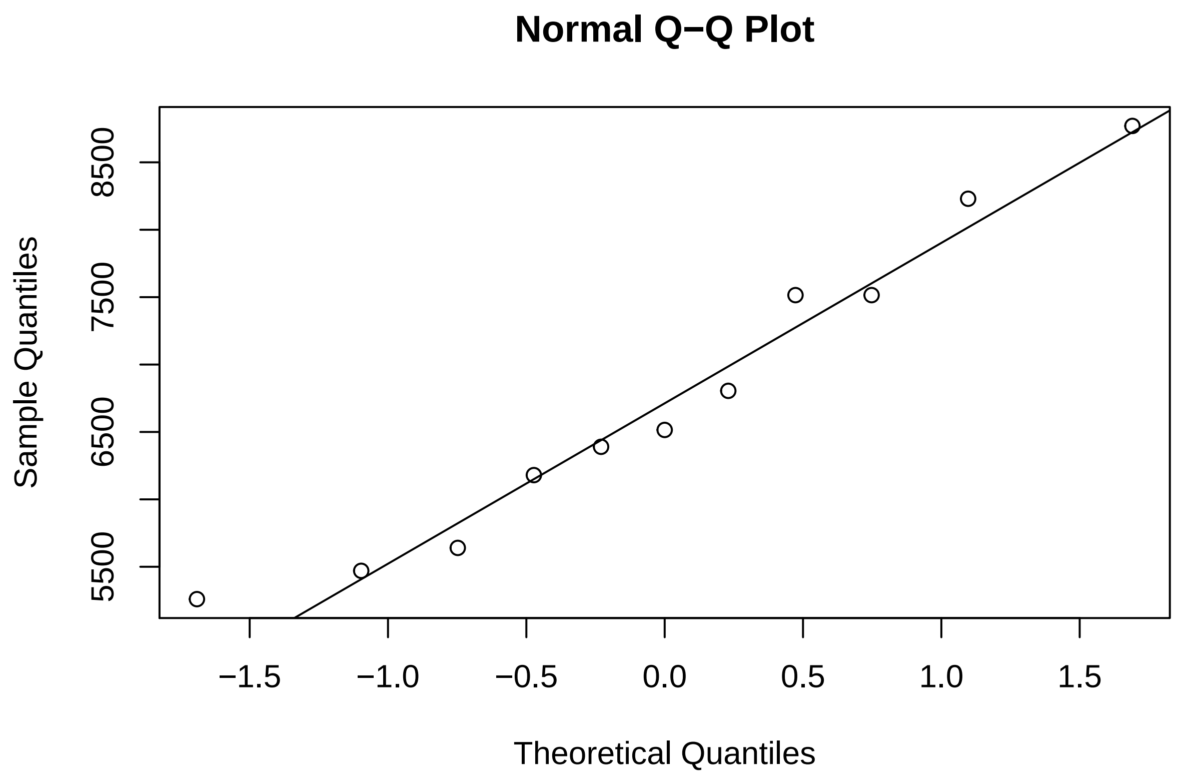
```r
sd(daily.intake)
```

```
## [1] 1142.123
```

```r
plot(density(daily.intake))
curve(dnorm(x,mean = mean(daily.intake),sd = sd(daily.intake)),add = T, col = "red",lty = 2)
```



```r
qqnorm(daily.intake)
qqline(daily.intake)
```



```r
shapiro.test(daily.intake)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  daily.intake
## W = 0.95237, p-value = 0.6743
```

Everything looks good and normal. As Daalgard suggests, we may want to know if this sample could have come from a population of women with a mean intake of 7725 kJ.

```r
kJ<-t.test(daily.intake,mu=7725)
kJ
```

```
##
##  One Sample t-test
##
## data:  daily.intake
## t = -2.8208, df = 10, p-value = 0.01814
## alternative hypothesis: true mean is not equal to 7725
## 95 percent confidence interval:
##  5986.348 7520.925
## sample estimates:
## mean of x
##  6753.636
```

The output of t.test() is the first specialized R object we've come across so far. The default print out is rather useful, but we can also access specific values from the test. To see what all is accessible, use names()

```
names(kJ)
```

```
## [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
## [6] "null.value"  "alternative" "method"      "data.name"
```

```
kJ$alternative
```

```
## [1] "two.sided"
```

```
kJ$p.value
```

```
## [1] 0.01813724
```

This was a two-sided t-test, which means we were testing the hypothesis that &mu = 7725. The p-value is sufficiently low, so we can reject this hypothesis at the p = 0.018 level, and adopt the alternative, that &mu &ne 7725. The default printout actually presents this in a much more readable way, but it' good to know how to access values from an object.

If you wanted to do a one sided t-test, to examine the altenative hypotheses that &mu < or > x, you can pass a value of "greater" or "less" to the alternative argument of t.test()

```
t.test(daily.intake, mu = 7725, alternative = "greater")
```

```
##
##  One Sample t-test
##
## data:  daily.intake
## t = -2.8208, df = 10, p-value = 0.9909
## alternative hypothesis: true mean is greater than 7725
## 95 percent confidence interval:
##  6129.492      Inf
## sample estimates:
## mean of x
##  6753.636
```

```
t.test(daily.intake, mu = 7725, alternative = "less")
```

```
##
##  One Sample t-test
##
## data:  daily.intake
## t = -2.8208, df = 10, p-value = 0.009069
## alternative hypothesis: true mean is less than 7725
## 95 percent confidence interval:
##     -Inf 7377.781
## sample estimates:
## mean of x
##  6753.636
```

Two Sample t-tests

If you have two samples, and want to see if they could have come from the same population, you want to test if they could have the same &mu. Daalgard has data on energy consumption of people with different satures.

```
library(ISwR)
summary(energy)
```

```
##      expend        stature
##  Min.   : 6.130   lean :13
##  1st Qu.: 7.660   obese: 9
##  Median : 8.595
##  Mean   : 8.979
##  3rd Qu.: 9.900
##  Max.   :12.790
```

There are two ways you could peform the t-test. Either pass two vectors to t-test, or give it a formula.

```
obese<-subset(energy,stature == "obese")$expend
lean<- subset(energy,stature == "lean")$expend
t.test(lean,obese)
```

```
##
##  Welch Two Sample t-test
##
## data:  lean and obese
## t = -3.8555, df = 15.919, p-value = 0.001411
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.459167 -1.004081
## sample estimates:
## mean of x mean of y
##  8.066154 10.297778
```

```
##or
t.test(energy$expend ~ energy$stature)
```

```
##
##  Welch Two Sample t-test
##
## data:  energy$expend by energy$stature
## t = -3.8555, df = 15.919, p-value = 0.001411
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.459167 -1.004081
## sample estimates:
##  mean in group lean mean in group obese
##            8.066154           10.297778
```

The p-value is very low, so we can reject the hypothesis that the population of obese people and the population of lean people have the same &mu when it comes to energy consumption. How big is the difference? The 95% confidence interval give the range we can be 95% confident that &mulean-&muobese lies within.

Again, we could do a one-sided t-test by defining alternative hypotheses

```r
t.test(energy$expend ~ energy$stature, alt = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  energy$expend by energy$stature
## t = -3.8555, df = 15.919, p-value = 0.9993
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -3.242484       Inf
## sample estimates:
##  mean in group lean mean in group obese
##            8.066154           10.297778
```

```r
t.test(energy$expend ~ energy$stature, alt = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  energy$expend by energy$stature
## t = -3.8555, df = 15.919, p-value = 0.0007053
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -1.220763
## sample estimates:
##  mean in group lean mean in group obese
##            8.066154           10.297778
```

If you're willing to commit to the variances of these two populations being the same, then the two-sample t-test can be done with a bit more power. To compare variances, use var.test()

```r
var.test(energy$expend ~ energy$stature)
```

```
##
##  F test to compare two variances
##
## data:  energy$expend by energy$stature
## F = 0.78445, num df = 12, denom df = 8, p-value = 0.6797
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.1867876 2.7547991
## sample estimates:
## ratio of variances
##           0.784446
```

If this returned a significant p-value, then we could not assume that the variances were equal. However, the p-value is not, so let's do the t-test again assuming equal variances.

```r
 t.test(energy$expend ~ energy$stature, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  energy$expend by energy$stature
## t = -3.9456, df = 20, p-value = 0.000799
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.411451 -1.051796
## sample estimates:
##  mean in group lean mean in group obese
##            8.066154           10.297778
```

Looks like we've got a smaller p-value, and a narrower 95% confidence interval, so success.

Paired t-test

Daalgard also has data on energy intake from women pre- and postmenstrual women in intake. However, what is important about this data is that it was not collected from different groups of women, but from the same women at different times. It's important to use a paired t-test for this kind of data. Any time a single individual provides 2 points of measurement, sort of pre- and post-treatment, a paired t-test should be used. (If you have more than two data points from an individual, you should do a repeated measures ANOVA, but I don't know how that works yet.)

```r
t.test(intake$pre, intake$post,paired = T)
```

```
##
##  Paired t-test
##
## data:  intake$pre and intake$post
## t = 11.941, df = 10, p-value = 3.059e-07
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1074.072 1566.838
## sample estimates:
## mean of the differences
##                1320.455
```

Non-parametric tests

If for some reason you can't assume normality of a population (because you got a significant result on a Shapiro-Wilk test for instance), you could try using a non-parametric test. These tests make no assumptions about the distribution of the data, except that it is symmetrically distributed around its µ.

One Sample Wilcoxon test

```r
wilcox.test(daily.intake, mu = 7725)
```

```
## Warning in wilcox.test.default(daily.intake, mu = 7725): cannot compute
## exact p-value with ties
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  daily.intake
## V = 8, p-value = 0.0293
## alternative hypothesis: true location is not equal to 7725
```

Two Sample Wilcoxon test

```r
wilcox.test(energy$expend ~ energy$stature)
```

```
## Warning in wilcox.test.default(x = c(7.53, 7.48, 8.08, 8.09, 10.15, 8.4, :
## cannot compute exact p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  energy$expend by energy$stature
## W = 12, p-value = 0.002122
## alternative hypothesis: true location shift is not equal to 0
```

Matched Pairs Wilcoxon test

```r
wilcox.test(intake$pre, intake$post, paired = T)
```

```
## Warning in wilcox.test.default(intake$pre, intake$post, paired = T): cannot
## compute exact p-value with ties
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  intake$pre and intake$post
## V = 66, p-value = 0.00384
## alternative hypothesis: true location shift is not equal to 0
```