Tutorial 3: Comparisons and Inference

Jan Vogler (jan.vogler@duke.edu)

September 11, 2015

Today's Agenda

1. Covariance
2. Correlation
3. Cross-tabs
4. Central Limit Theorem
5. t-tests and p-values

Question: sometimes R gives you output like this:

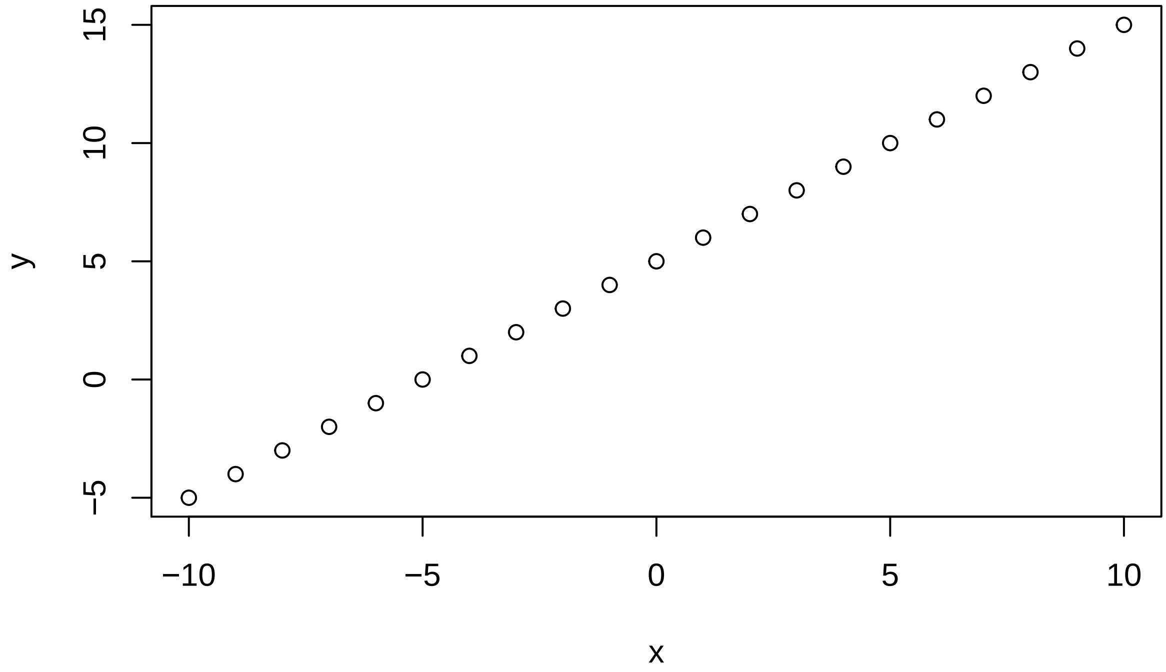
2.43e-05

What does this mean?

Topic 1: Covariance

Let us create two variables that are clearly linearly dependent on each other.

```
x = seq(-10, 10)
y = (x + 5)
plot(x, y)
```



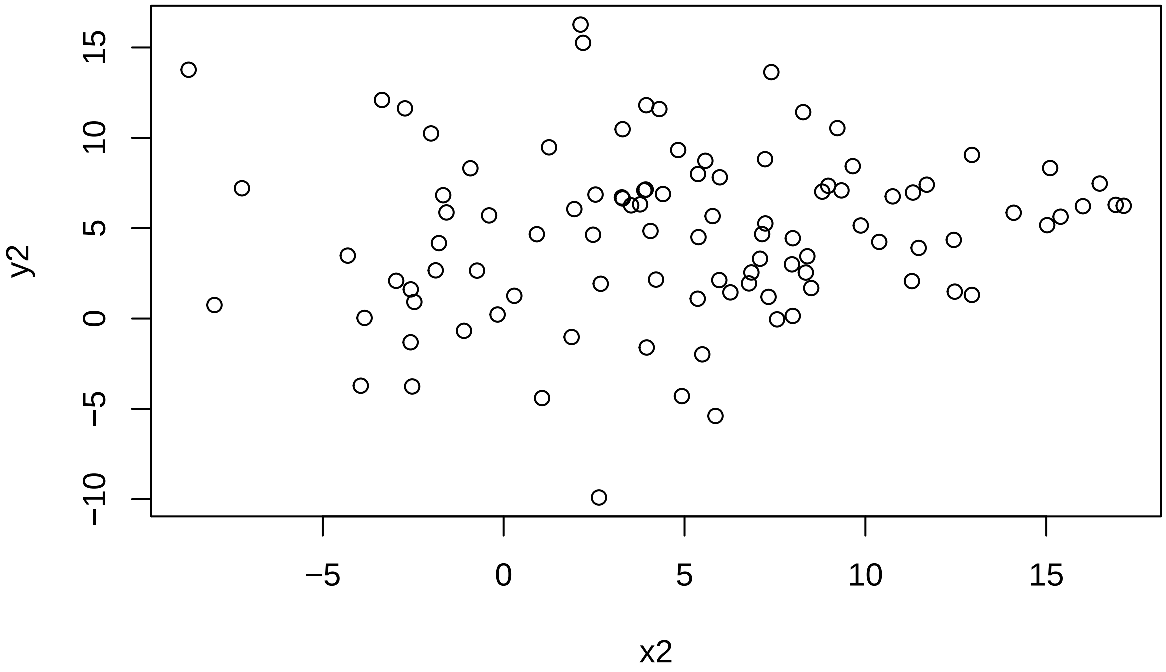
The covariance of these two variables is positive - as x increases so does y. A positive covariance indicates that as x is above its mean value, y is above its mean value (on average).

```
cov(x, y)
```

```
## [1] 38.5
```

Let us create two variables that have no relationship to each other:

```
x2 = rnorm(100, mean = 5, sd = 5)
y2 = rnorm(100, mean = 5, sd = 5)  # Both variables are just random drows from the normal distribution
plot(x2, y2)
```



The covariance should be close to zero - due to the randomness of the data it is most likely not exactly zero though.

```
cov(x2, y2)
```

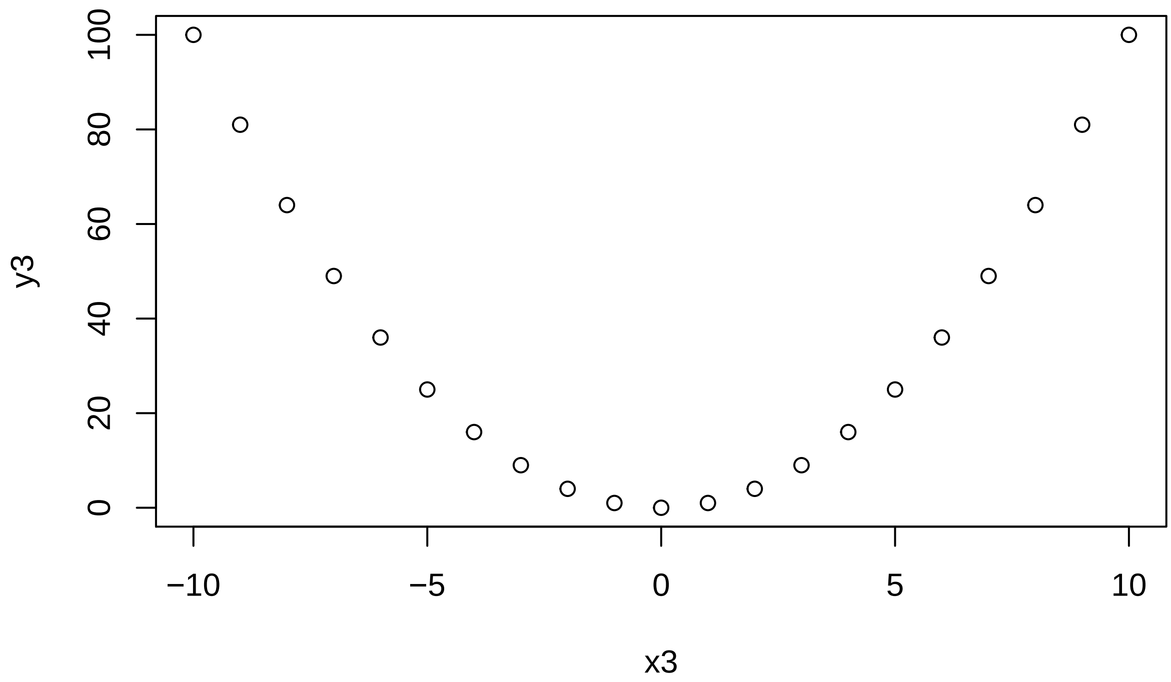
```
## [1] 2.71493
```

Note that this means even variables that are completely random and not related to each other may produce a non-zero covariance. However, the E(Cov(x2,y2)) = 0, so the distribution of the covariance is centered on the value 0. If our variables have large variances though, we might get a large value of the covariance. The problem is that covariance is not to-scale.

Independence implies that the expected value of the covariance is zero.

Does a covariance of zero imply independence?

```
x3 = seq(-10, 10)
y3 = x3^2
plot(x3, y3)
```



As we can clearly see from the plot, there is a curvilinear relationship of the two variables - they are not independent. When one variable is a function of another variable, even a quadratic function, they are not independent.

```
cov(x3, y3)
```

```
## [1] 0
```

The formula tells us that the covariance is zero. Why?

Covariance captures linear relationships.

When x3 is below its mean, the values of y3 vary in the exact same way as when x3 is above its mean.

The formula can't capture the curvilinear relationship because it looks at the variation of y3 relative to x3's deviation from its mean. y3 varies in the exact same way when x3 moves above and below its mean. Thus, there is no linear relationship that could be captured.

Topic 2: Correlation

As you've learned in the lecture, the problem with covariance is that it is not to scale. It doesn't really tell us that much about how much variables vary with each other because it doesn't account for their individual variation magnitudes. However, correlation standardizes covariance by the standard deviation of the two variables. The result is that the measure of correlation is bound between -1 and 1.

```r
cor(x, y)  ### Why does this produce '1'. What is the meaning of this value?
```

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

How about x2 and y2 that are completely random?

```r
cor(x2, y2)  # The correlation is extremely close to zero, indicating that there is no systematic linea
```

```
## [1] 0.1017667
```

Does correlation capture non-linear relationships equally well?

Correlation is a mathematical concept. We cannot find the correlation between a numeric and a character vector.

```r
y4=rep(c("a","b","c"),7)
y4
```

```
##  [1] "a" "b" "c" "a" "b" "c" "a" "b" "c" "a" "b" "c" "a" "b" "c" "a" "b"
## [18] "c" "a" "b" "c"
```

```r
is.numeric(y4) # Checks whether y4 is numeric and returns the argument FALSE.
```

```
## [1] FALSE
```

```r
cor(x,y4) # Gives us the error message "y must be numeric".
```

```
## Error in cor(x, y4): 'y' must be numeric
```

Interestingly, however, R allows us to find the correlation between a numeric and a logical vector, although a logical vector is not numeric.

```r
y5 = rep(c(T, F, F), 7)
y5
```

```
##  [1]  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [12] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
```

```r
is.numeric(y5)  # Checks whether y5 is numeric and returns the argument FALSE.
```

```
## [1] FALSE
```

```r
class(y5)  # Returns the class of the vector.
```

```
## [1] "logical"
```

```r
cor(x, y5)  # Returns a value.
```

```
## [1] -0.1167748
```

How do we have to think about this?

Assume that T=1 and F=0.

```r
y6 = rep(c(1, 0, 0), 7)
cor(x, y6)  # Returns the same value as above, meaning that R views T=1 and F=0
```

```
## [1] -0.1167748
```

Topic 3: Cross-tabs

R has several built-in datasets, let's have a look at them.

```r
library(datasets)
data(occupationalStatus)
occupationalStatus
```

```
##       destination
## origin   1   2   3   4   5   6   7   8
##      1  50  19  26   8   7  11   6   2
##      2  16  40  34  18  11  20   8   3
##      3  12  35  65  66  35  88  23  21
##      4  11  20  58 110  40 183  64  32
##      5   2   8  12  23  25  46  28  12
##      6  12  28 102 162  90 554 230 177
##      7   0   6  19  40  21 158 143  71
##      8   0   3  14  32  15 126  91 106
```

According to the documentation this is "Cross-classification of a sample of British males according to each subject's occupational status and his father's occupational status."

The source is a journal article from 1979: "Goodman, L. A. (1979) Simple Models for the Analysis of Association in Cross-Classifications having Ordered Categories."

Let us assume that 1 is a low occupational status and 8 is a high occupational status (it might be the opposite). Is there a relationship between the status of the father and the son?

Before using the command below, use the following command: install.packages("gmodels")

```r
library(gmodels)
```

```
## Warning: package 'gmodels' was built under R version 3.2.2
```

```r
CrossTable(occupationalStatus)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  3498
##
##
##              | destination
##       origin |         1 |         2 |         3 |         4 |         5 |         6 |         7 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---
##            1 |        50 |        19 |        26 |         8 |         7 |        11 |         6 |
##              |   561.961 |    29.430 |    15.717 |     4.708 |     0.444 |    24.504 |    11.515 |
##              |     0.388 |     0.147 |     0.202 |     0.062 |     0.054 |     0.085 |     0.047 |
##              |     0.485 |     0.119 |     0.079 |     0.017 |     0.029 |     0.009 |     0.010 |
##              |     0.014 |     0.005 |     0.007 |     0.002 |     0.002 |     0.003 |     0.002 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---
##            2 |        16 |        40 |        34 |        18 |        11 |        20 |         8 |
##              |    30.377 |   161.485 |    27.842 |     0.144 |     0.028 |    18.723 |    11.946 |
##              |     0.107 |     0.267 |     0.227 |     0.120 |     0.073 |     0.133 |     0.053 |
##              |     0.155 |     0.252 |     0.103 |     0.039 |     0.045 |     0.017 |     0.013 |
##              |     0.005 |     0.011 |     0.010 |     0.005 |     0.003 |     0.006 |     0.002 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---
##            3 |        12 |        35 |        65 |        66 |        35 |        88 |        23 |
##              |     0.334 |    23.798 |    32.359 |     9.492 |     4.969 |     7.176 |    21.531 |
##              |     0.035 |     0.101 |     0.188 |     0.191 |     0.101 |     0.255 |     0.067 |
##              |     0.117 |     0.220 |     0.197 |     0.144 |     0.143 |     0.074 |     0.039 |
##              |     0.003 |     0.010 |     0.019 |     0.019 |     0.010 |     0.025 |     0.007 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---
##            4 |        11 |        20 |        58 |       110 |        40 |       183 |        64 |
##              |     1.186 |     0.534 |     1.707 |    25.988 |     0.414 |     0.309 |     6.458 |
##              |     0.021 |     0.039 |     0.112 |     0.212 |     0.077 |     0.353 |     0.124 |
##              |     0.107 |     0.126 |     0.176 |     0.240 |     0.164 |     0.154 |     0.108 |
##              |     0.003 |     0.006 |     0.017 |     0.031 |     0.011 |     0.052 |     0.018 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---
##            5 |         2 |         8 |        12 |        23 |        25 |        46 |        28 |
##              |     1.464 |     0.117 |     0.502 |     0.313 |    18.318 |     0.898 |     0.091 |
##              |     0.013 |     0.051 |     0.077 |     0.147 |     0.160 |     0.295 |     0.179 |
##              |     0.019 |     0.050 |     0.036 |     0.050 |     0.102 |     0.039 |     0.047 |
##              |     0.001 |     0.002 |     0.003 |     0.007 |     0.007 |     0.013 |     0.008 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---
##            6 |        12 |        28 |       102 |       162 |        90 |       554 |       230 |
##              |    19.508 |    18.320 |     5.219 |     1.404 |     0.216 |    19.474 |     0.000 |
##              |     0.009 |     0.021 |     0.075 |     0.120 |     0.066 |     0.409 |     0.170 |
##              |     0.117 |     0.176 |     0.309 |     0.353 |     0.369 |     0.467 |     0.388 |
##              |     0.003 |     0.008 |     0.029 |     0.046 |     0.026 |     0.158 |     0.066 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---
##            7 |         0 |         6 |        19 |        40 |        21 |       158 |       143 |
##              |    13.486 |    10.547 |    13.563 |     6.721 |     3.751 |     0.047 |    55.016 |
##              |     0.000 |     0.013 |     0.041 |     0.087 |     0.046 |     0.345 |     0.312 |
##              |     0.000 |     0.038 |     0.058 |     0.087 |     0.086 |     0.133 |     0.241 |
##              |     0.000 |     0.002 |     0.005 |     0.011 |     0.006 |     0.045 |     0.041 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--
##            8 |         0 |         3 |        14 |        32 |        15 |       126 |        91 |
##              |    11.395 |    12.103 |    13.878 |     6.946 |     5.330 |     0.207 |     9.829 |
##              |     0.000 |     0.008 |     0.036 |     0.083 |     0.039 |     0.326 |     0.235 |
##              |     0.000 |     0.019 |     0.042 |     0.070 |     0.061 |     0.106 |     0.153 |
##              |     0.000 |     0.001 |     0.004 |     0.009 |     0.004 |     0.036 |     0.026 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--
## Column Total |       103 |       159 |       330 |       459 |       244 |      1186 |       593 |
##              |     0.029 |     0.045 |     0.094 |     0.131 |     0.070 |     0.339 |     0.170 |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--
##
##
```

How can we interpret this table?

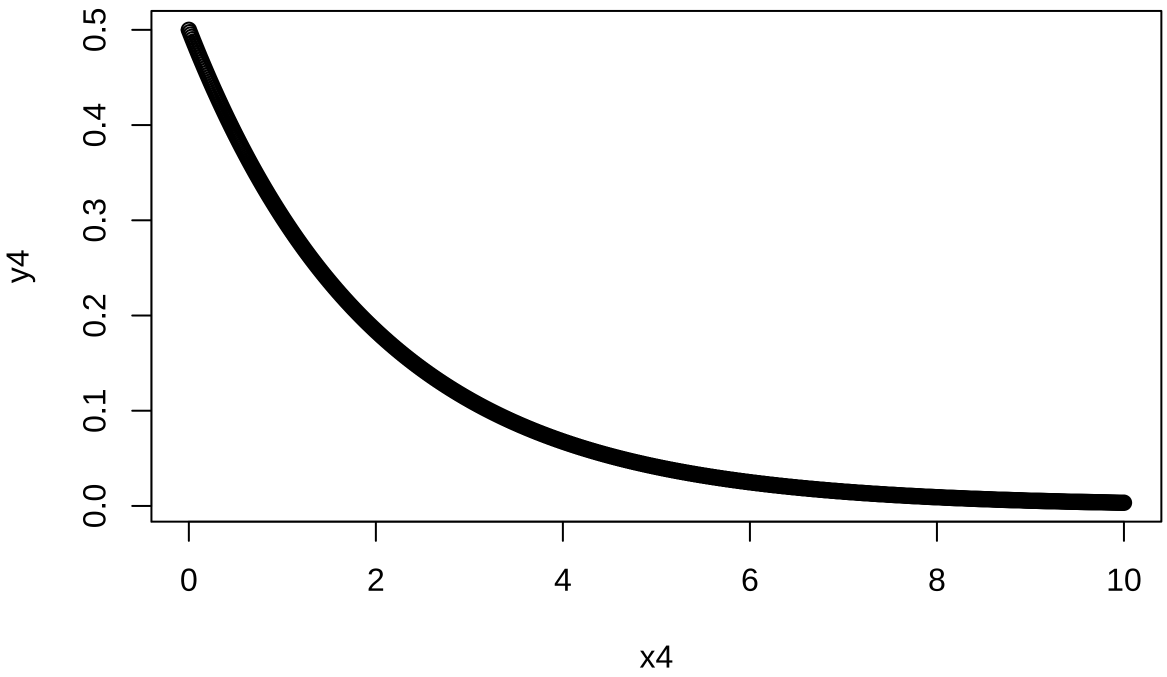
Topic 4: Central Limit Theorem

The Central Limit Theorem states that if we have infinitely many draws of the same size from a specific distribution, the mean of this distribution will be approximately normally distributed.

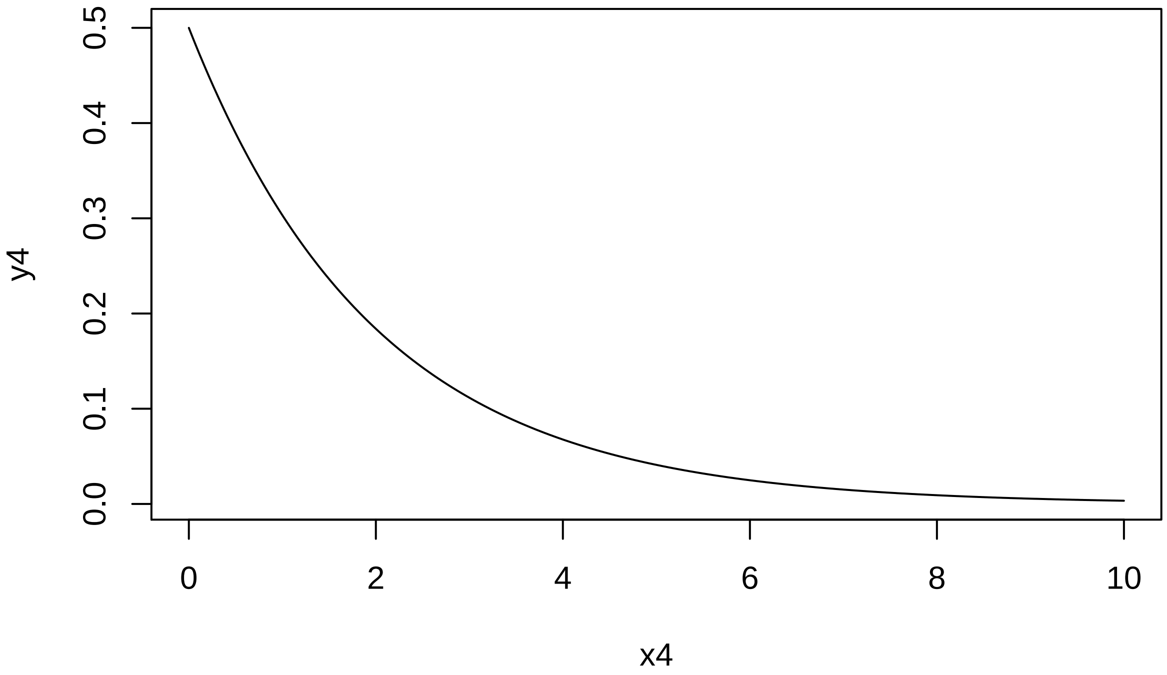
Let us illustrate this with a simple example of the exponential distribution. The exponential distribution doesn't look like a normal distribution.

How does an exponential distribution look like?

```
x4 = seq(0, 10, by = 0.01)
y4 = dexp(x4, rate = 0.5)  # Returns the density
plot(x4, y4)  # This doesn't look nice
```



```r
plot(x4, y4, type = "l")  # Use type='l' for a line plot
```



We can also use the ggplot2 package to make it look even nicer.

Use the command the following command before you run this code: install.packages("ggplot2")

```
library(ggplot2)
```

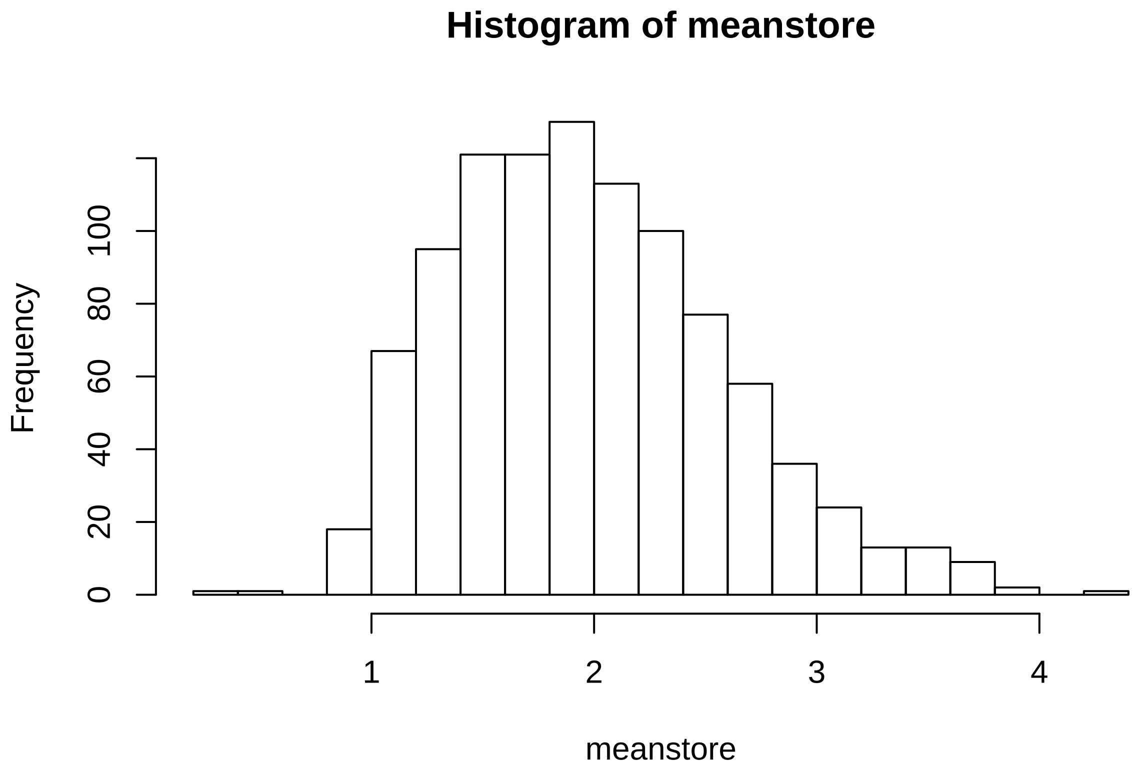
```
## Warning: package 'ggplot2' was built under R version 3.2.2
```

```
plot1 = qplot(x4, y4)  # Now that looks even nicer
```

Recall: The Central Limit Theorem states that if we have multiple samples of the same size, their mean will be approximately normally distributed.

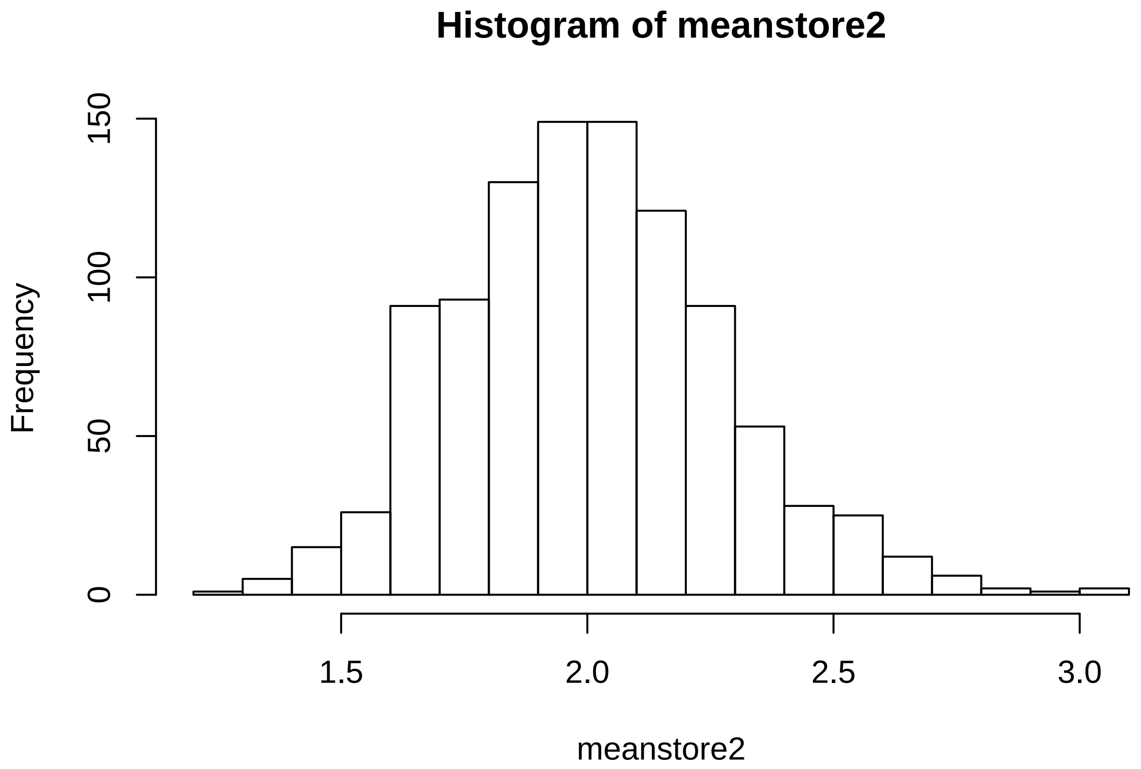
So, what happens if we draw 1000 times 10 samples from this distribution, how will their mean be distributed?

```
meanstore = rep(0, 1000)
for (i in 1:1000) {
    expdraw = rexp(10, rate = 0.5)
    meanstore[i] = mean(expdraw)
}
hist(meanstore, breaks = 20)  # It is approximately normally distributed, as predicted by the CLT.
```



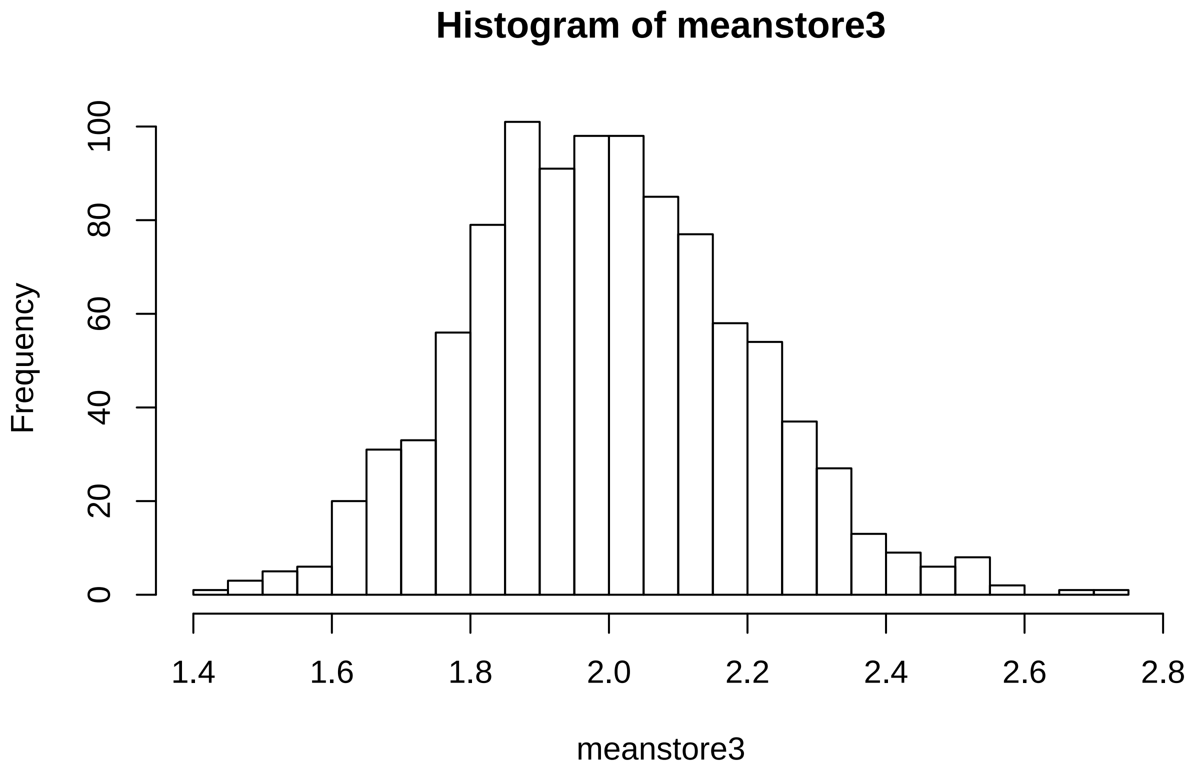
Let us expand this example and increase the number of draws to 50:

```
meanstore2 = rep(0, 1000)
for (i in 1:1000) {
    expdraw = rexp(50, rate = 0.5)
    meanstore2[i] = mean(expdraw)
}
hist(meanstore2, breaks = 20)  # It is approximately normally distributed, as predicted by the CLT.
```



And a final example with 100 draws:

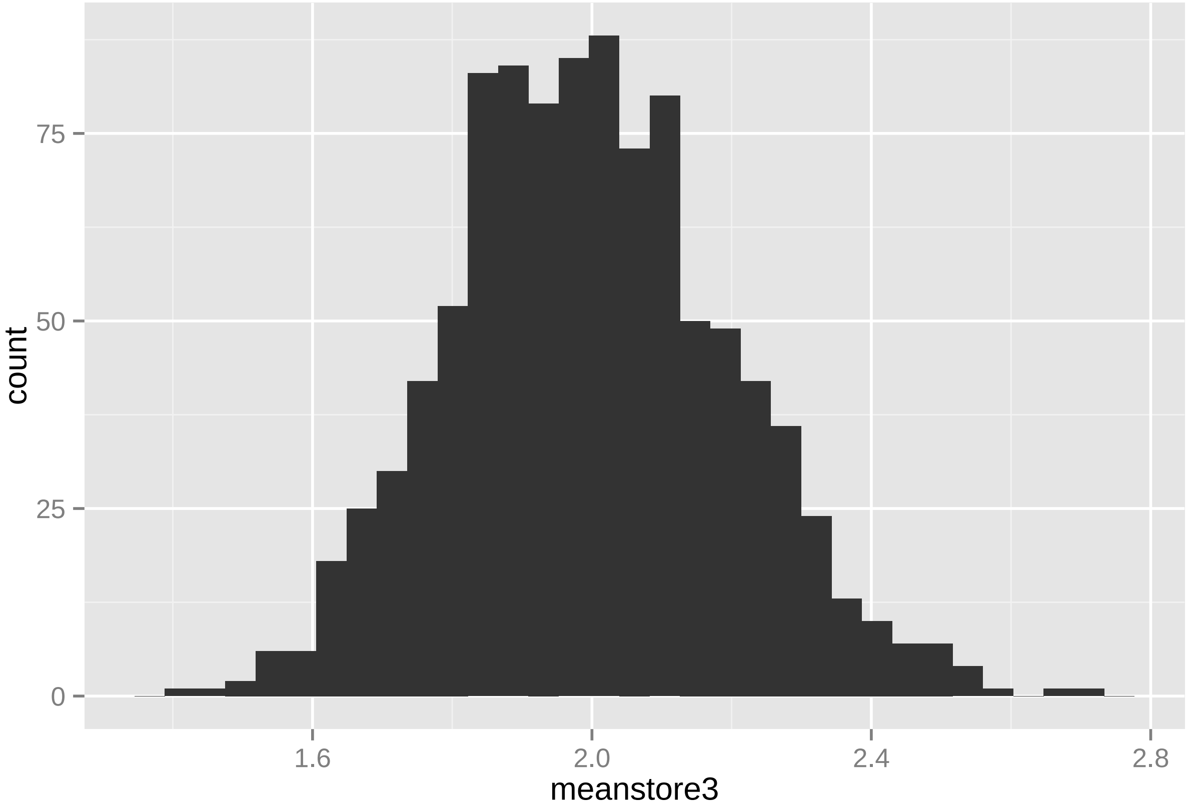
```
meanstore3 = rep(0, 1000)
for (i in 1:1000) {
    expdraw = rexp(100, rate = 0.5)
    meanstore3[i] = mean(expdraw)
}
hist(meanstore3, breaks = 20)  # It is approximately normally distributed, as predicted by the CLT.
```



Let’s use ggplot again to make a nicer histogram:

```
library(ggplot2)
qplot(meanstore3, geom = "histogram")
```

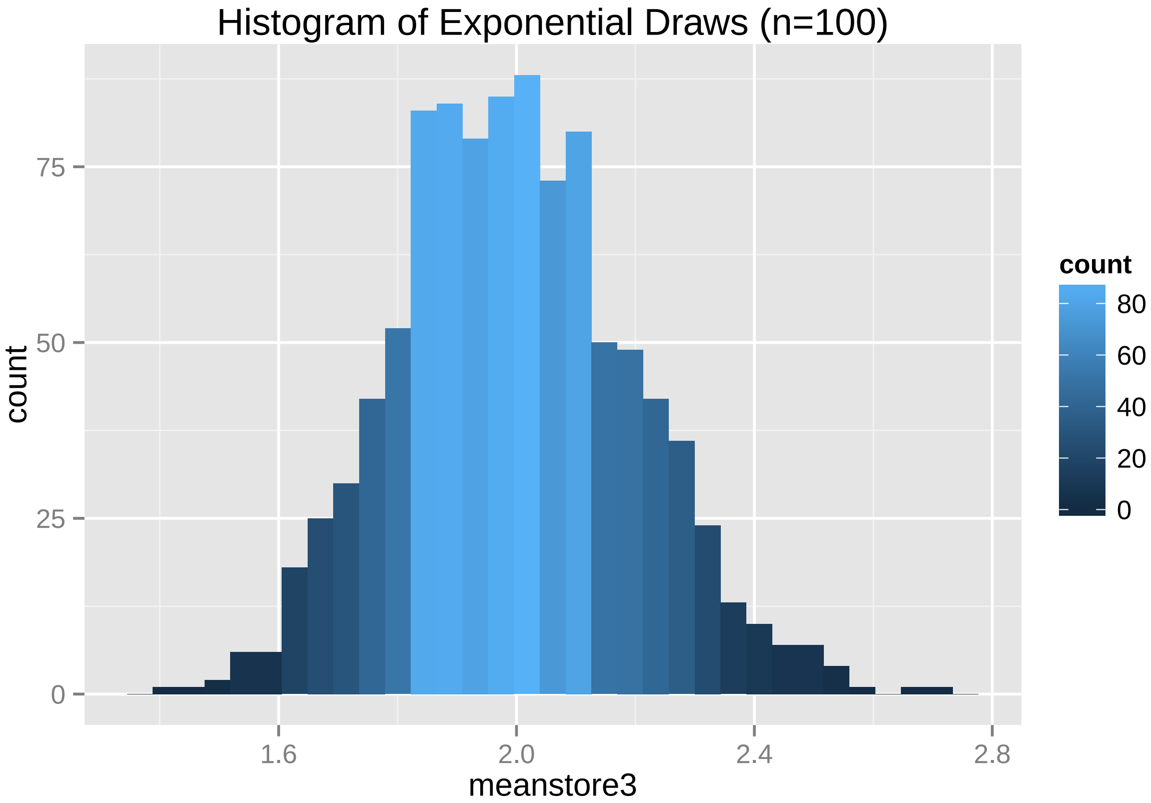
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



This is not perfect though, let's play a little with the function:

```
library(ggplot2)
plot2 = qplot(meanstore3, geom = "histogram", main = "Histogram of Exponential Draws (n=100)",
    bindwidth = 30)  # Now that looks even nicer
plot2 + geom_histogram(aes(fill = ..count..))  # Looks great
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



Topic 5: t-tests and p-values

t-tests allow us to either compare the mean of two populations or to compare the mean of one population against a theoretical mena value.

Let us create two sets of numbers that come from normal distributions with different means.

```
vec1 = rnorm(30, mean = 2, sd = 1)
vec2 = rnorm(30, mean = 3, sd = 1)
```

The t-test allows us to find out the likelihood that these two come from the same distribution:

```
t.test(vec1, vec2)
```

```
##
##  Welch Two Sample t-test
##
## data:  vec1 and vec2
## t = -2.7752, df = 57.31, p-value = 0.007437
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.1422212 -0.1848032
## sample estimates:
## mean of x mean of y
##  2.101640  2.765152
```

What does this t-value mean? What does this p-value mean?

We can also compare a single sample against a mean that we define to be our H0.

```
t.test(vec1, mu = 2)
```

```
##
##  One Sample t-test
##
## data:  vec1
## t = 0.63718, df = 29, p-value = 0.529
## alternative hypothesis: true mean is not equal to 2
## 95 percent confidence interval:
##  1.775395 2.427884
## sample estimates:
## mean of x
##   2.10164
```

```
t.test(vec1, mu = 2.5)
```

```
##
##  One Sample t-test
##
## data:  vec1
## t = -2.4973, df = 29, p-value = 0.01844
## alternative hypothesis: true mean is not equal to 2.5
## 95 percent confidence interval:
##  1.775395 2.427884
## sample estimates:
## mean of x
##   2.10164
```

```
t.test(vec1, mu = 3)
```

```
##
##  One Sample t-test
##
## data:  vec1
## t = -5.6318, df = 29, p-value = 4.392e-06
## alternative hypothesis: true mean is not equal to 3
## 95 percent confidence interval:
##  1.775395 2.427884
## sample estimates:
## mean of x
##   2.10164
```

```
t.test(vec1, mu = 3.5)
```

```
##
##  One Sample t-test
##
## data:  vec1
## t = -8.7663, df = 29, p-value = 1.197e-09
## alternative hypothesis: true mean is not equal to 3.5
## 95 percent confidence interval:
##  1.775395 2.427884
## sample estimates:
## mean of x
##   2.10164
```

What do these t-values mean? What do these p-values mean?

Typically, a p-value is related to a type-1 error rate (alpha) that we define in advance. Type-1 erors refer to the incorrect rejection of a true null hypothesis. Often we want alpha to be smaller than 0.05.

Typically, our null hypothesis (H0) is that (1) the vectors have the same mean or (2) the mean of the observations is the same as our theoretical mean.

The p-value is the probability that we get data with evidence that is such strong AGAINST H0 if H0 was true. Think about what this means. If we define a threshold of p to be $p < 0.05$, then we have a type-1 error rate of alpha = 0.05.

This produces a very small p-value:

```r
t.test(vec1, mu = 3)
```

```
##
##  One Sample t-test
##
## data:  vec1
## t = -5.6318, df = 29, p-value = 4.392e-06
## alternative hypothesis: true mean is not equal to 3
## 95 percent confidence interval:
##  1.775395 2.427884
## sample estimates:
## mean of x
##   2.10164
```

Note that there are four important levels of statistical significane:

$p <= 0.001$, corresponds to a type-1 error rate (alpha) of 0.001

$p <= 0.01$, corresponds to a tye-1 error rate (alpha) of 0.01

$p <= 0.05$, corresponds to a type-1 error rate (alpha) of 0.05

$p <= 0.1$, corresponds to a type-1 error rate (alpha) of 0.1

A p-value of $p < 0.001$ implies that a test is significant at all common levels of statistical significance.

How would we compute the confidence interval of a test like the following?

```r
t.test(vec1, vec2)
```

```
##
##  Welch Two Sample t-test
##
## data:  vec1 and vec2
## t = -2.7752, df = 57.31, p-value = 0.007437
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.1422212 -0.1848032
## sample estimates:
## mean of x mean of y
##  2.101640  2.765152
```

Let us first get the standard error:

```
standerr = sqrt(var(vec1)/length(vec1) + var(vec2)/length(vec2))
standerr
```

```
## [1] 0.2390878
```

To construct the 95-percent confidence interval get the difference in means and subtract and add the standard error:

```
(1.754153 - 3.180158) - (2 * standerr)  # lower bound
```

```
## [1] -1.904181
```

```
(1.754153 - 3.180158) + (2 * standerr)  # upper bound
```

```
## [1] -0.9478294
```