Statistical Analysis

Section 1: Statistics Concepts

1. Normal distribution

$$\text{PDF} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

$$\underbrace{\text{Log Likelihood}}_{\text{for } n \text{ independent } N(\mu,1)} = \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2 - n\log(\sqrt{2\pi})$$

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

$$\text{SE} = \frac{\sigma}{\sqrt{n}}$$

R functions

- dnorm : Density or height of prob distribution
- pnorm : Cumulative distribution (area) probability
- qnorm : inverse of pnorm give quantile, give prob, returns z score
- rnorm : random number from normal distribution

popultion property

```
u = 10000 # mean
sigma = 1000 # standard deviation

# If you select 10 humans at random, what is the probablity that the average is greater than 10300 ?
N = 10
se = sigma / sqrt(N)
z= (10300 - u) / se
prob= 1 - pnorm(z)
prob
```

```
## [1] 0.1713909
```

```
# What is the probablity that the average number of tastebds they have is between 9500 and 10500
z1 = (9500 - u)/se
z2= (10500 - u)/se
prob2= pnorm(z2) - pnorm(z1)
prob2
```

```
## [1] 0.8861537
```

2. Z Statistics

```
# Z value
z1= qnorm(p = 0.95, mean = 0, lower.tail = T)
z2= qnorm(p = 0.95, mean = 0, lower.tail = F)
z3= qnorm(p = 0.975, mean = 0, lower.tail = T)
sprintf('z1 = %f z2 = %f z3 = %f', z1, z2, z3)
```

```
## [1] "z1 = 1.644854 z2 = -1.644854 z3 = 1.959964"
```

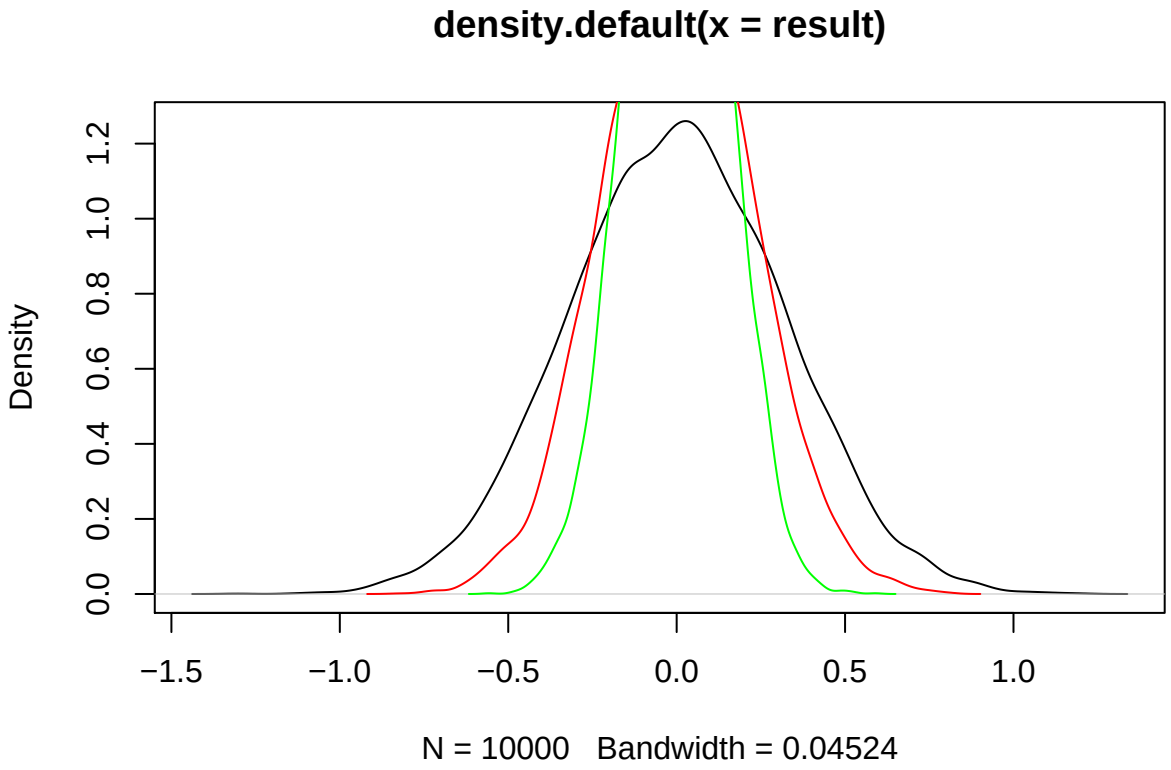
```
sprintf('p1 = %f p1 = %f, p3 = %f', pnorm(z1), pnorm(z2), pnorm(z2))
```

```
## [1] "p1 = 0.950000 p1 = 0.050000, p3 = 0.050000"
```

3. Standard Error: SE

The following code shows that the larger the number of samples draw from a population, the smaller SE of sample mean.

```
# simple line of code
x = 10

my.f = function(N){
  # create a random normal distribution
  z = rnorm(N)
  # Find mean
  zbar = mean(z)
  return(zbar)
}

result = replicate(n = 10000, expr = my.f(10))
result2 = replicate(n = 10000, expr = my.f(20))
result3 = replicate(n = 10000, expr = my.f(50))

plot(density(result))
lines(density(result2) , col='red')
lines(density(result3) , col='green')
```

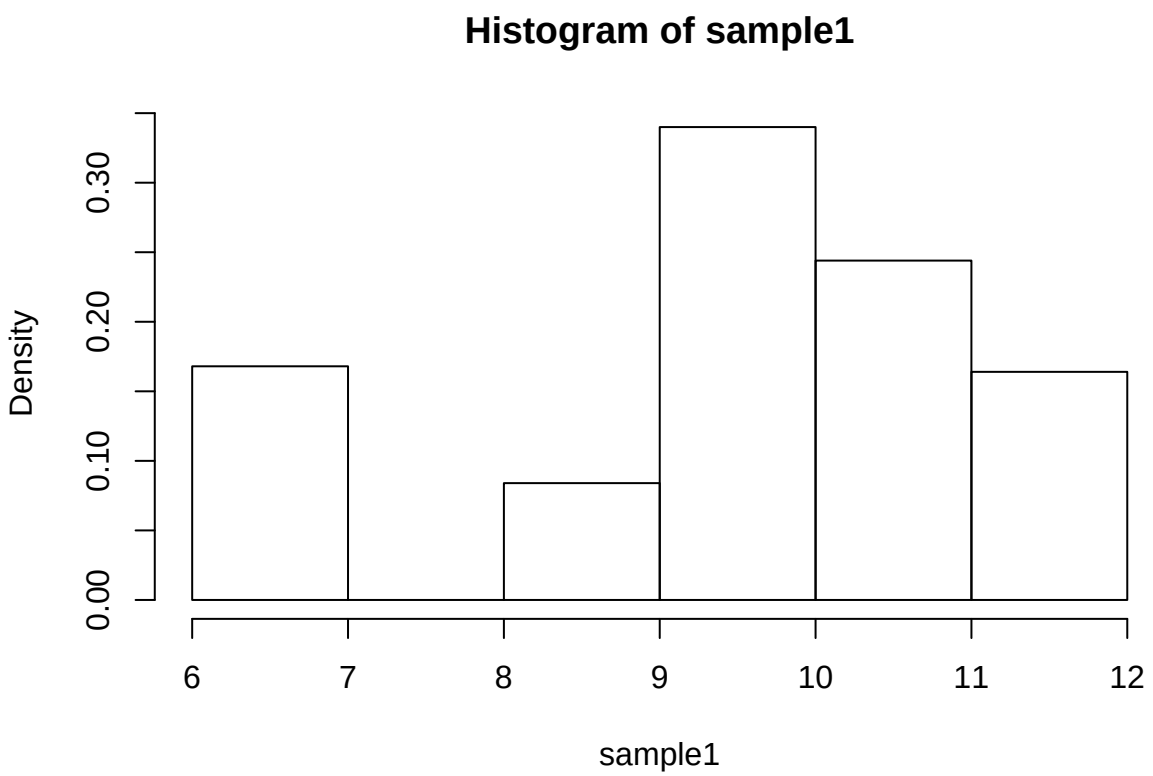


4. Center Limit Theorem

Definition: when independent random variables are added, their properly normalized sum tends toward a normal distribution even if the original variables themselves are not normally distributed.

Sample distribution of variables, we use sample with replacement because population size is small and every draw has the same distribution and can draw many number of times

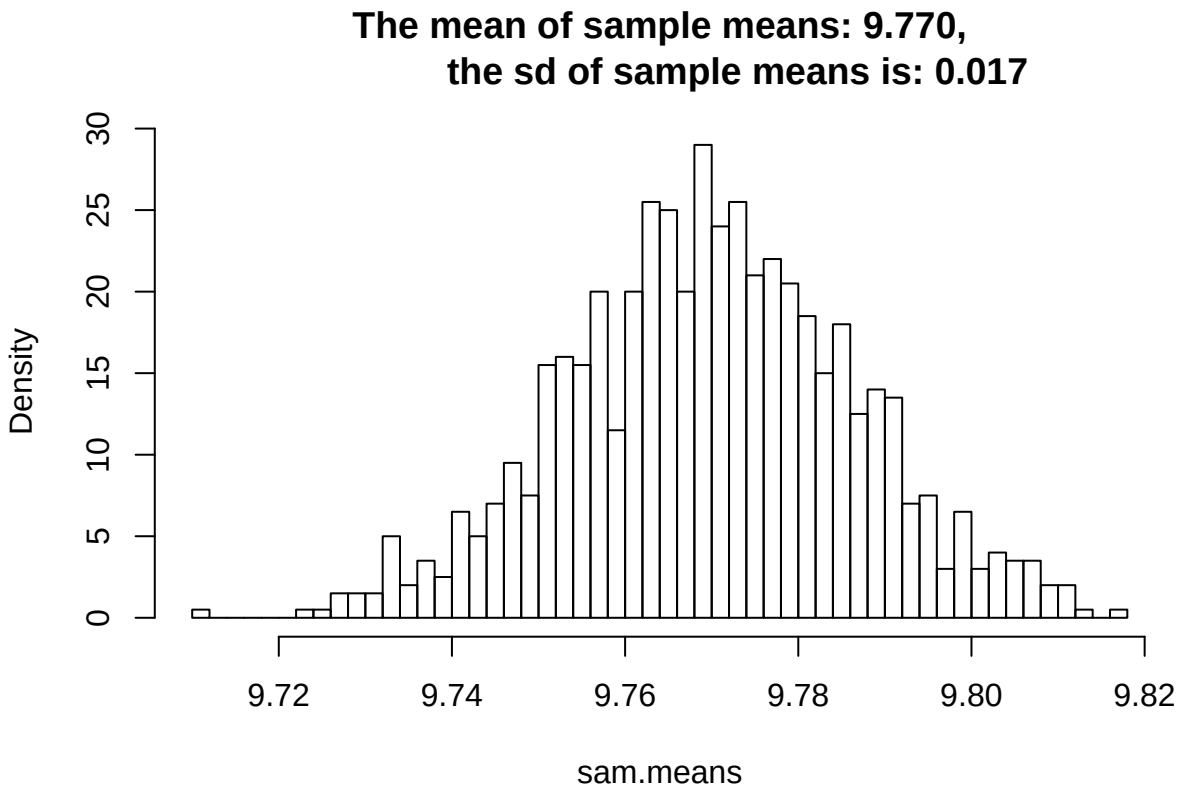
```
# Please note the sample are the shoe sizes
shoe.pop = c(12, 11, 10, 11.5, 11, 11, 8.5,7, 6.5, 9.5, 10, 9.5, 9.5)
N = 1000 # Sample size
sample1 = sample(shoe.pop, size=N, replace=TRUE)
mean.sample <- mean(sample1)
sd.sample <- sd(sample1)
hist(sample1, breaks=5, probability = TRUE)
```



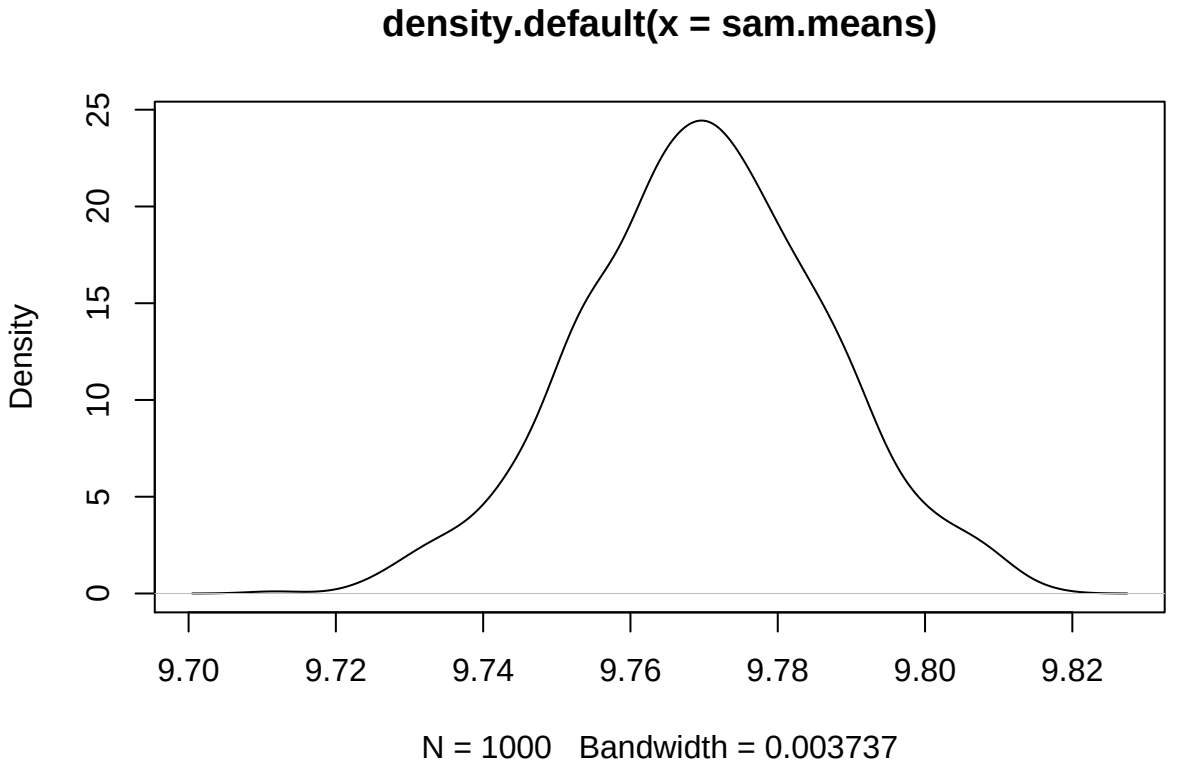
Sample size N determine the SE sigma/sqrt(N), the larger the N, the smaller the SD of the mean, the tight the curve. As N increases, sample mean distribution become close to normal distribution

```
mysamplemean <- function(N) {
    sam1 = sample(shoe.pop, size=N, replace=TRUE)
    sam.mean = mean(sam1)
    return(sam.mean)
}
samplesize <- 10000

sam.means <- replicate(n=1000, mysamplemean(samplesize))
sam.means.mean = mean(sam.means)
sam.means.sd  = sd(sam.means)
prs <- sprintf("The mean of sample means: %2.3f,
               the sd of sample means is: %2.3f", sam.means.mean, sam.means.sd)
```

```r
hist(sam.means, breaks = 50, probability = TRUE, main = prs)
```




```r
plot(density(sam.means))
```



Why central Limint Theorm is Significant? The frequncy distribution of sample means of any underlying distribution with very large sample size is a normal distribution which is a nice assumption. When sample size is large, we use apply property of normal distribution

5. Confidence Interval

Definition: If we were to draw 100 samples from same population, approximately 95 of them would contain the parameter. In other words, We sample from a distribution and calculate the mean, there are 95% probability the mean will fall into the confidence interval

It measure of variability due to sampling error. Different samples drawn from that same population would in general have different values of the sample mean, so there is a distribution of sampled means.

Use normal distribution to approximate the distribution of error about a binomially-distributed observation. The Central Limit Theorem applies poorly to this distribution with a sample size less than 30 or where the proportion is close to 0 or 1.

$$\text{CI} = p \pm 1.96 \times \underbrace{\text{SE for percent}}_{\text{w/ replacement}}$$

```
confint = function(SampleSize) {
    x = runif(SampleSize)
    pop.sd = 0.2886895 # population standard deviation
    x.bar = mean(x) # Sample mean
    se = pop.sd/sqrt(SampleSize) # SE
    upper = x.bar + 1.96*se # upper = mean + z (alpha/2) * SE
    lower = x.bar - 1.96*se # lower = mean - z (alpha/2) * SE
    contained = (lower < 0.5 ) & (0.5 < upper)
    return(contained)
}

res = replicate(1e3 , confint(SampleSize = 1000) )
print(mean(res))
```

```
## [1] 0.95
```

Section 2. Statistical Testing

1. Hypothesis Testing Definition

Definition: Hypothesis statements contain two or more variables that are **measurable** that specify how the variables are related

- H0: null hypothesis
- Ha: alternative hypothesis
- A test is rule of rejecting H0 based on the observed data and risk-level (Reject H0 if . . .)
  - if p_value < 0.05
  - if $|Z| >$ Z_alpha for 2 tails, Z < -Z_alpha, Z > Z_alpha 1 tail
- 2 Actions: Reject H0 or do not reject H0

	H0 True	Ha True
Reject H0	Type 1 Err (FP)	
Not Reject H0		Type 2 err (FN)

Example 1

You take a random sample of 100 Berkeley students to find out if their ground beef consumption is any different than the nation at large. The mean among sample is 2.45 pounds per month. What is the p-value corresponding to the null hypothesis that Berkeley students eat the same amount, on the average compare to the nation at large? what is an appropriate alternative hypothesis?

- H0: u_berkeley = u_nation
- Ha: u_berkeley != u_nation

```
u_berkeley = 2.45 # sample mean
u = 2 # population mean
sd = 2 # population sd
N = 100 # sample size
Z_berkeley = (u_berkeley - u) / (sd/sqrt(N))

alpha = 0.05 # alpha/2 = 0.025
alpha_2_tail = alpha/2
p_value = 2 * (1 - pnorm(Z_berkeley))
p_value
```

```
## [1] 0.02444895
```

```
p_value < alpha_2_tail
```

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

p_value is < 0.025 which is statistically significant, we can reject the H0 that berkeley student's ground beef consumption is the same as average consumption of the nation

Hypothesis Test function

A function that takes a sample data, mean of the null hypothesis, population standard deviation a boolean variable for 1 or 2 tailed test a boolean variable for left or right tail, return P-value for this test, and use 5% critical value α

```
hp_test = function(sample_data, mean_null, sd_p, two_tailed=TRUE, left_tail=NULL) {
    alpha <- 0.05 # Use 5% critical value
    N <- length(sample_data) # Calculate sample size
    sample_mean <- mean(sample_data) # Calculate sample mean from the sample data
    se <- sd_p/sqrt(N) # calculate SE
    z_score <- (sample_mean - mean_null) / se # Calculate Z-score
    # Hypothesis testing
    # two_tailed is true: 2 tailed test, false: 1 tailed test
    # left_tail is true: left, False: right
    if (two_tailed) {
        p_value <- 2 * (1- pnorm(abs(z_score)))
        reject <- p_value < alpha
    } else {
        if (left_tail) {
            p_value <- pnorm(z_score)
            reject <- p_value < alpha
        } else {
            p_value <- 1 - pnorm(z_score)
            reject <- p_value < alpha
        }
    }
```

```
    result <- list(p.value = p_value, reject.null = reject)
    return(result)
}

display_result = function(r) {
    if (r$reject.null) {
        paste("We reject the null hypthesis. The P-value of the test is ", r$p.value)
    } else {
        paste("We do not reject the null hypthesis. The P-value of the test is ", r$p.value)
    }
}
```

2. Test Assumptions

Assumption 1: Normality

- **Shapiro-Wilk**: Test whether a series normally distributed. This is to test assumption data. The **Null** is that the underlying data is normally distributed. We can also use qq-norm plot
- Transform data/Feature transform
  –

```
library(ggplot2)
library(car)
```

```
## Error in library(car): there is no package called 'car'
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
# load the countries dataset, including
# corruption and internet growth variables
load("./data/Countries2.Rdata")
summary(Countries)
```

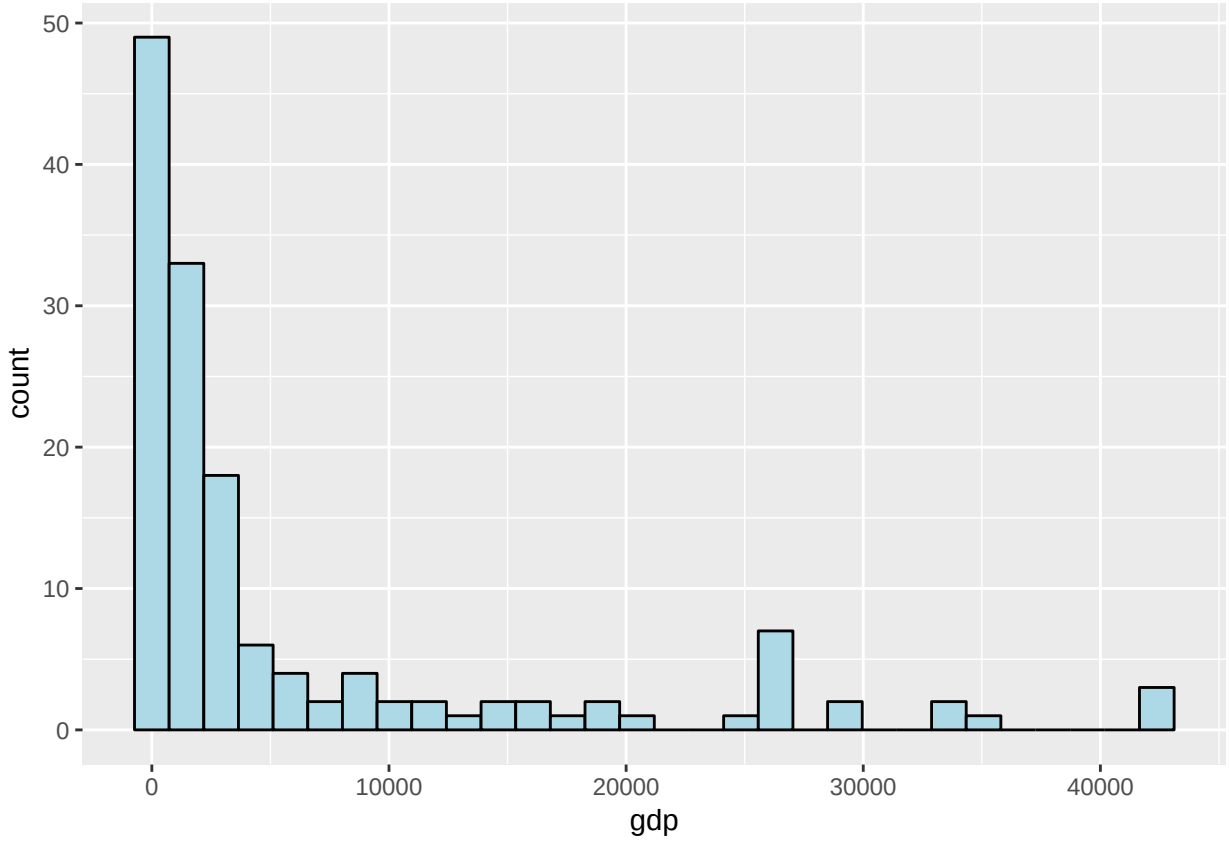
```
##    Country          infant.mortality      gdp        fertility_rate
##  Length:190         Min.   :  3.00   Min.   :   36   Min.   :1.190
##  Class :character   1st Qu.: 13.00   1st Qu.:  435   1st Qu.:1.855
##  Mode  :character   Median : 35.00   Median : 1570   Median :3.070
##                     Mean   : 46.48   Mean   : 6195   Mean   :3.542
##                     3rd Qu.: 72.50   3rd Qu.: 6232   3rd Qu.:5.025
##                     Max.   :154.00   Max.   :42416   Max.   :7.600
##                     NA's   :47       NA's   :45      NA's   :47
##  contraception        region    Country_Code internet_users_2010
##  Min.   : 2.00   Africa  :42   AFG    :  1   Min.   : 0.25
##  1st Qu.:19.75   Americas:26   AGO    :  1   1st Qu.:10.00
##  Median :46.00   Asia    :28   ALB    :  1   Median :27.67
##  Mean   :41.49   Europe  :38   AND    :  1   Mean   :33.61
##  3rd Qu.:61.25   Oceania :14   ARG    :  1   3rd Qu.:53.00
##  Max.   :83.00   NA's    :42   (Other):143   Max.   :95.00
```

```
##  NA's   :78                    NA's   : 42   NA's   :45
##  internet_users_2011 Corruption_Rank       cpi        internet_growth
##  Min.   : 0.98       Min.   :  1.00   Min.   : 8.00   Min.   :-0.12500
##  1st Qu.:11.67       1st Qu.: 44.50   1st Qu.:28.75   1st Qu.: 0.05765
##  Median :32.00       Median : 88.00   Median :37.00   Median : 0.11321
##  Mean   :37.50       Mean   : 87.17   Mean   :43.27   Mean   : 0.21683
##  3rd Qu.:59.50       3rd Qu.:130.75   3rd Qu.:56.25   3rd Qu.: 0.24764
##  Max.   :95.02       Max.   :174.00   Max.   :90.00   Max.   : 2.92000
##  NA's   :50          NA's   :14       NA's   :14      NA's   :51
##    high_cpi         high_gdp
##  Length:190         High:72
##  Class :character   Low :73
##  Mode  :character   NA's:45
##
##
##
##
```

```
# use a histogram to see if the distribution of gdp looks normal
graph1 = ggplot(Countries, aes(gdp))
graph1 + geom_histogram(color='black', fill='light blue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 45 rows containing non-finite values (stat_bin).
```

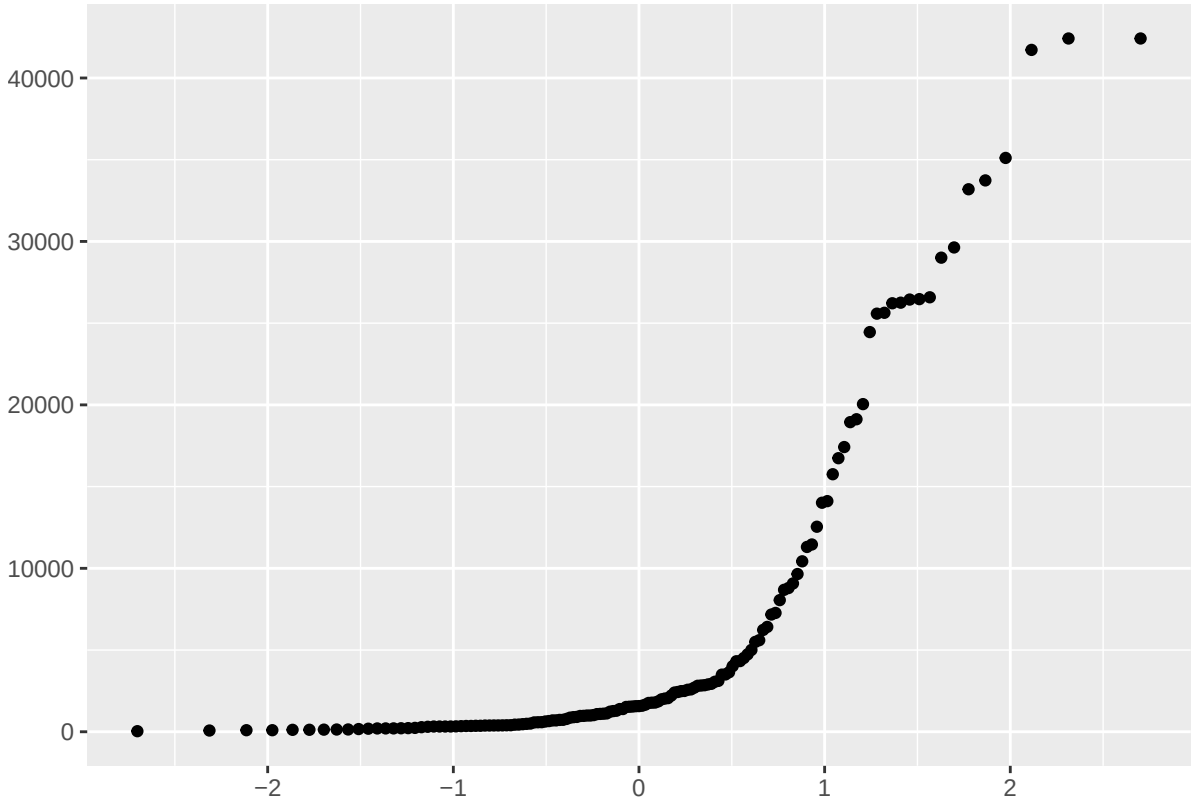



```
# check normality using a qqplot
qqplot = qplot(sample = Countries$gdp, stat="qq")
```

```
## Warning: `stat` is deprecated
```

```
qqplot
```

```
## Warning: Removed 45 rows containing non-finite values (stat_qq).
```




```
# Finally, use a Shapiro-Wilk test to see if normality is a plausible hypothesis
shapiro.test(Countries$gdp)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Countries$gdp
## W = 0.64501, p-value < 2.2e-16
```

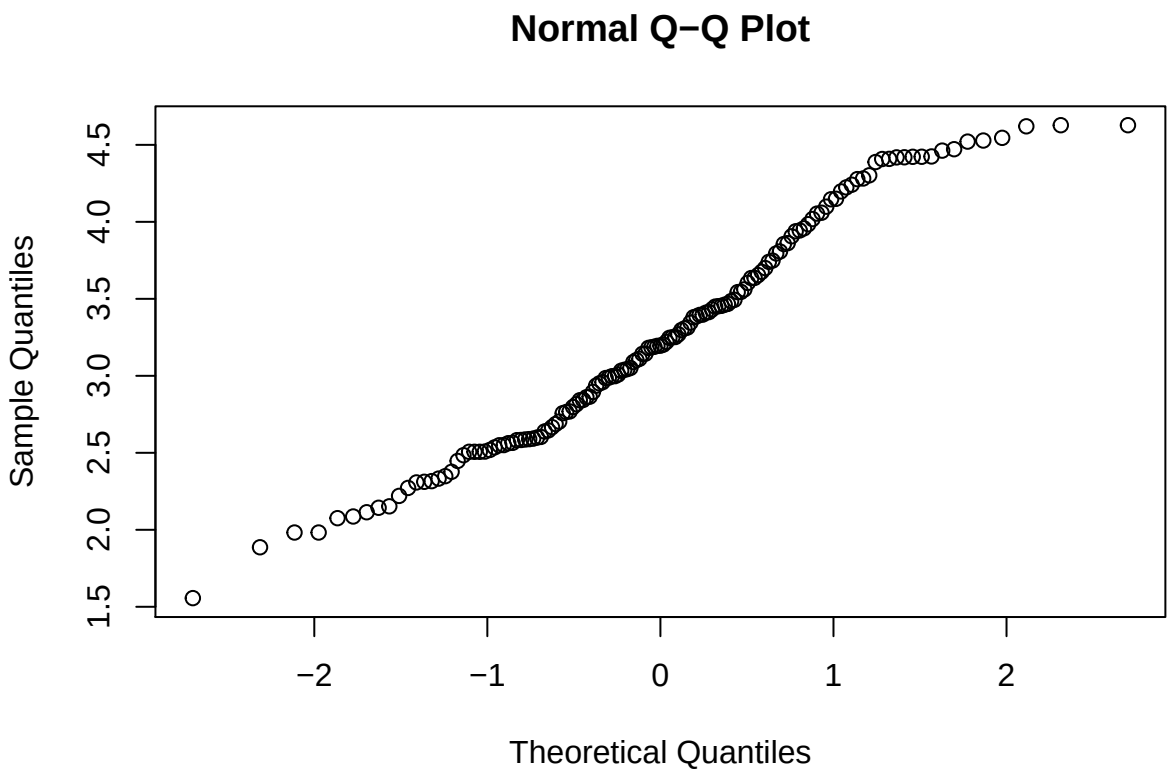
Shapiro-Wilk show p-value < 0.05 which is statistical significant, we can reject the null hypothese that the data is normally distributed

To transform to using log (this is common on econometric data)

```
# Next, let's do the same thing with the log of gdp
# This is a very common transformation in econometrics
Countries$loggdp = log10(Countries$gdp)

# Begin with the Shapiro-Wilk test
shapiro.test(Countries$loggdp)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Countries$loggdp
## W = 0.97303, p-value = 0.005833
```

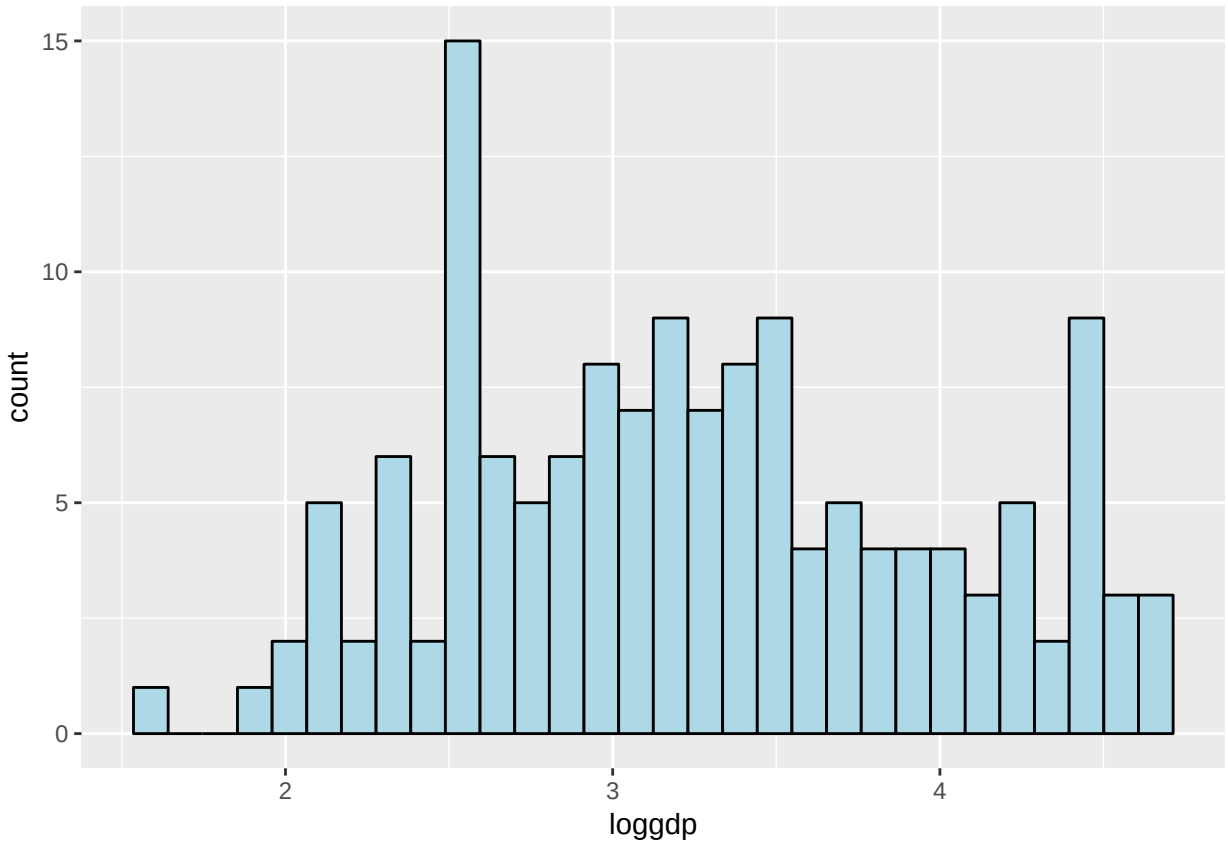
```r
# But look at the shape of the qqplot
qqnorm(Countries$loggdp)
```




```r
# use a histogram to see if the distribution of loggdp looks normal
graph1 = ggplot(Countries, aes(x = loggdp))
graph1 + geom_histogram(color = 'black', fill = 'light blue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 45 rows containing non-finite values (stat_bin).
```



Assumption 1: Homogenity of variance

- **Levene Test**: Test whether two or more series is satisfy Homogeneity of variance assumption. The **null** is the two series's variances is homogeneious.

```
# First, check the means
by(Countries$loggdp, Countries$high_cpi, mean, na.rm = TRUE)
```

```
## Countries$high_cpi: Corrupt
## [1] 2.859729
## ------------------------------------------------------------
## Countries$high_cpi: Trustworthy
## [1] 3.821505
```

```
# check if the variances are the same for both groups
by(Countries$loggdp, Countries$high_cpi, var, na.rm = TRUE)
```

```
## Countries$high_cpi: Corrupt
## [1] 0.2846552
## ------------------------------------------------------------
## Countries$high_cpi: Trustworthy
## [1] 0.3838832
```

```
# use a Levene test to see if equal variances is a plausible hypothesis
leveneTest(Countries$loggdp, Countries$high_cpi)
```

```
## Error in leveneTest(Countries$loggdp, Countries$high_cpi): could not find function "leveneTest"
```

P-value is > 0.05, we cannot reject the hypothese that the variance is homogeneious.

3. Test of association

If numerical => Pearson correlation

If categorical => Chi-square test

Small Example

```
library(foreign)
insurgency = read.dta("./data/lyall2010.dta")
head(insurgency)
```

```
##   ccode yearbg strict lgdpen treat dur  startdate    enddate warid
## 1     2   1832      2     NA     1   4   5/4/1832   2/8/1832    16
## 2     2   1835      2     NA     1  81 28/12/1835  14/8/1842    18
## 3     2   1855      2     NA     1  40   1/5/1855  23/9/1858    37
## 4     2   1855      2     NA     1  34  7/12/1855   8/5/1858    38
## 5     2   1860      2     NA     1  72   1/1/1860 31/12/1865    45
## 6     2   1860      2     NA     1  72   1/1/1860 31/12/1865    46
##                   war yearend incumb              insurg wdl win modern
## 1     Blackhawk's War    1832    USA Sauk and Fox Indians   2   1      0
## 2 Second Seminole War    1842    USA            Seminoles   2   1      0
## 3          Yakima War    1858    USA               Yakima   2   1      0
## 4  Third Seminole War    1858    USA            Seminoles   2   1      0
## 5          Navajo War    1865    USA               Navajo   2   1      0
## 6          Apache War    1865    USA              Apaches   2   1      0
##   gp patron      cinc autoc7    lenerpc ltradegdp elf heli mlevel
## 1  0      0 0.0445450      0 -2.7355239        NA  NA   NA     NA
## 2  0      0 0.0508560      0 -2.4243710        NA  NA   NA     NA
## 3  0      0 0.0802830      0 -0.6670240        NA  NA   NA     NA
## 4  0      0 0.0802830      0 -0.6670240        NA  NA   NA     NA
## 5  0      0 0.1509228      0 -0.5653595        NA  NA   NA     NA
## 6  0      0 0.1509228      0 -0.5653595        NA  NA   NA     NA
##       region weurope eeurop lamerica ssafrica asia nafrme namerica brit fr
## 1 n. america       0      0        0        0    0      0        1    0  0
## 2 n. america       0      0        0        0    0      0        1    0  0
## 3 n. america       0      0        0        0    0      0        1    0  0
## 4 n. america       0      0        0        0    0      0        1    0  0
## 5 n. america       0      0        0        0    0      0        1    0  0
## 6 n. america       0      0        0        0    0      0        1    0  0
##   rus usa china prio cow none fl pitf turkey iraq ger drc spain railway
## 1   0   1     0    0   0    1  0    0      0    0   0   0     0       0
## 2   0   1     0    0   0    1  0    0      0    0   0   0     0       0
## 3   0   1     0    0   0    1  0    0      0    0   0   0     0       0
## 4   0   1     0    0   0    1  0    0      0    0   0   0     0       0
## 5   0   1     0    0   0    1  0    0      0    0   0   0     0       0
## 6   0   1     0    0   0    1  0    0      0    0   0   0     0       0
##   ww2 postwar dec1 dec2 dec3 dec4 dec5 dec6 dec7 dec8 interwar postww2
## 1   0       0   NA   NA   NA   NA   NA   NA   NA   NA       NA      NA
## 2   0       0   NA   NA   NA   NA   NA   NA   NA   NA       NA      NA
## 3   0       0   NA   NA   NA   NA   NA   NA   NA   NA       NA      NA
## 4   0       0   NA   NA   NA   NA   NA   NA   NA   NA       NA      NA
## 5   0       0   NA   NA   NA   NA   NA   NA   NA   NA       NA      NA
## 6   0       0   NA   NA   NA   NA   NA   NA   NA   NA       NA      NA
```

```
##   postcw worldwar2 tie defeat nwstate frhouse1 frhouse2 notes dem_occ
## 1     NA        NA   0      0       0       NA       NA             0
## 2     NA        NA   0      0       0       NA       NA             0
## 3     NA        NA   0      0       0       NA       NA             1
## 4     NA        NA   0      0       0       NA       NA             0
## 5     NA        NA   0      0       0       NA       NA             1
## 6     NA        NA   0      0       0       NA       NA             1
##   mixed_dem mixed_ndem pol5 weakdemo weakauto newdis weakdemo2 weakauto2
## 1         0          0    2        0       NA      1         0         0
## 2         0          0    2        0       NA      1         0         0
## 3         0          0    2        0       NA      1         0         0
## 4         0          0    2        0       NA      1         0         0
## 5         0          0    2        0       NA      1         0         0
## 6         0          0    2        0       NA      1         0         0
##   strongdemo strongauto missing interregnum monarch military single hybrid
## 1          1          0       0           0       0        0      0      0
## 2          1          0       0           0       0        0      0      0
## 3          1          0       0           0       0        0      0      0
## 4          1          0       0           0       0        0      0      0
## 5          1          0       0           0       0        0      0      0
## 6          1          0       0           0       0        0      0      0
##   dem person numlang treat_occ treat_mech democ6 weakstates fiveyearlag
## 1   1      0      NA         0          0      1          0           9
## 2   1      0      NA         0          0      1          0           9
## 3   1      0      NA         0          0      1          0           9
## 4   1      0      NA         0          0      1          0           9
## 5   1      0      NA         0          0      1          0           8
## 6   1      0      NA         0          0      1          0           8
##   flag flag1 occ xconst parcomp nelf duryear mid  cinc100 still endpoint
## 1    0     0   0      7       4   NA       1   0  4.45450     1       NA
## 2    0     0   0      7       4   NA       8   0  5.08560     1       NA
## 3    0     0   0      7       3   NA       4   0  8.02830     1       NA
## 4    0     0   0      7       3   NA       4   0  8.02830     1       NA
## 5    0     0   0      7       3   NA       6   0 15.09228     1       NA
## 6    0     0   0      7       3   NA       6   0 15.09228     1       NA
##   coldwar post1945 ended     ldur docc    lcinc milper support rear
## 1       0        0     1 1.386294    0 1.493915     11       0    0
## 2       0        0     1 4.394449    0 1.626413     13       0    0
## 3       0        0     1 3.688879    0 2.082973     21       0    0
## 4       0        0     1 3.526361    0 2.082973     21       0    0
## 5       0        0     1 4.276666    0 2.714183     29       0    0
## 6       0        0     1 4.276666    0 2.714183     29       0    0
##   ipatron   elev    lelev     ldis      dis mech pol2 winnodraw
## 1       0  99.00 4.595120 6.897700  990.000    0    9         1
## 2       0   1.00 0.000000 7.203044 1343.515    0    9         1
## 3       0 638.25 6.458730 8.191628 3610.596    0    8         1
## 4       0   1.00 0.000000 7.203044 1343.515    0    8         1
## 5       0 715.25 6.572632 8.040888 3105.370    0    8         1
## 6       0 715.25 6.572632 8.040888 3105.370    0    8         1
##   defeatnodraw _st _d       _t _t0    W S s    WoverS xrreg xrcomp xropen
## 1            0   1  0 1.386294   0 0.75 1 1 0.7510685     3      3      4
## 2            0   1  0 4.394449   0 0.75 1 1 0.7510685     3      3      4
## 3            0   1  0 3.688879   0 0.75 1 1 0.7510685     3      3      4
## 4            0   1  0 3.526361   0 0.75 1 1 0.7510685     3      3      4
```

```
## 5            0   1  0 4.276666   0 0.75 1 1 0.7510685     3      3      4
## 6            0   1  0 4.276666   0 0.75 1 1 0.7510685     3      3      4
##   parreg exrec polcomp country comp  part   id abbr
## 1      2     8       9     176 43.5  8.86 3.85  USA
## 2      2     8       9     176 43.5  8.86 3.85  USA
## 3      2     8       7     176 49.1 12.62 6.20  USA
## 4      2     8       7     176 49.1 12.62 6.20  USA
## 5      2     8       7     176 60.1 14.86 8.93  USA
## 6      2     8       7     176 60.1 14.86 8.93  USA
```

```
# variable types
insurgency$dur # Month of war (ratio variable)
```

```
##   [1]   4  81  40  34  72  72  24  48   9  12 129  11  49 241 115 121  36
##  [18]   6  30 641  30 371 135  17  85   2 176 306  36 240  24  23   1   6
##  [35]  70  58  45   8  30  96   3  49   2  31  12  21  48  13  11 120  11
##  [52]   7  24   6   4 134   7   2  39   6  12  10  11  31 261   3  51  86
##  [69]   7   6  18  34  93  42 133  52  65  48 359  64 369   6  52   7  67
##  [86]  71  99   2  12   4  65  12  96  12  44  30   6 126  11  12   7  18
## [103]  18  17  24   5  65  20   1  10  99  24  20  30  31  89  72  80  33
## [120]  17 115  50  47  22  11  65   9 190 156 148  66  25  49  49  47  14
## [137]  12  20   2   1   2  24  22 141  10  48   6  54  44  52   5 120 144
## [154] 342  24  11   1  21  21  20  11  67  33  93  48 105   5  48 162 152
## [171]  53  29  64  11  11  41 276  72  61   3 135  36  76   5 109  37 197
## [188] 310 192 148  95 256 124 178  31 120 109 240 276 209  80   2  14   2
## [205]  30  22  42   4  12   5   2  28  36  44 183  60  12  96   1   1  18
## [222]  26 187   1   4  62   1  87  12   3 145  20  46  67  29 236 212 204
## [239]  96 216  24  12 124  40  34   9  35   5 113  98  45  45 107 252  55
## [256] 120   9  53  84 288 542   2  49  29 209  64 165 157  50 124  54 216
## [273]   3  36 298 349  36 121  84  24 115  60 131  96 110  15
```

```
insurgency$wdl # Categorical / Ordinal
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 1 2 1 1 1 1 1 0 1 1 2 2 2 0 0 2 2
##  [36] 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 1 2 2 2 2 2 2 1 1 0 1
##  [71] 2 0 0 0 2 2 1 0 1 2 2 2 0 2 0 0 2 2 2 2 2 2 0 2 2 1 2 2 2 2 2 0 0 2 2
## [106] 2 2 1 2 0 0 2 2 0 0 0 0 2 2 0 2 2 0 0 2 2 1 0 0 0 2 2 0 2 0 0 0 0 2 2
## [141] 2 2 0 2 0 2 0 0 2 2 0 2 2 2 2 2 1 1 0 0 0 1 1 1 0 2 2 0 0 0 1 0 2 2 2
## [176] 0 1 0 2 2 0 2 1 0 2 1 0 1 1 2 1 0 0 2 2 2 1 1 1 2 0 2 0 2 2 2 0 2 1 0
## [211] 2 0 2 2 1 1 2 1 1 1 2 2 1 2 1 1 2 0 2 2 2 0 0 2 2 2 2 2 2 2 0 0 1 0 0
## [246] 2 2 2 2 0 0 0 1 1 2 1 0 1 1 1 1 2 1 1 2 0 1 0 1 0 2 2 2 2 0 1 2 1 2 2
## [281] 2 2 2 2 0 2
```

```
insurgency$pol2 # Ordinal variable / Interval variable
```

```
##   [1]   9   9   8   8   8   8   8   8  10  10  10  10  10  10  10  10  10
##  [18]  -9  -1  -3  -3   3  -4  -8  -5  10  -5   7   6   7  -3   2  -4  -5
##  [35]  -3   6  -2  -2  -2  -2  -2   3   3   3   3   3   3   3   3   3   3
##  [52]   3   3   3   7   7   7   7   7   7   7   7   7   8   8   8   8   9
##  [69]   8   8  10  10  10  10  10  10  10  10  10  -6  -3  -2  10   6  10
##  [86]  -4  -1  -1  -8  -8  -7  -1   6   7   7   7   7  10   7   7 -10   7
## [103]   7   8   8   9   9   9   9  -9   9  10  10  10  10  10  10  -6  -2
## [120]  -6  -2   1   4   4  -6   6  -7  -9  -9  -9   1   1  -9  -9  -9  -9
## [137]  -9  -9  -9 -10  -4  -4  -4  -6  -9  -3  -5  -6   0   8   5 -10 -10
## [154] -10 -10 -10 -10   4   4   1   0  -7   4  -6  -5  -6   7   6  -9  -7
## [171]  -4   4   0  -3  -9   0   0   3   2  -7  -3  -5  -6  -6  -7  -7   0
```

```
## [188]  -7  -8   4   4   4   4  -9  -8  -7  -7  -7 -10 -10 -10 -10 -10 -10
## [205] -10 -10 -10 -10 -10 -10 -10  -1  -1  -6   7  -5  -7  -9  -9  -9 -10
## [222]  -9   0  -9   9  10  -6   0  -7  -2 -10   0   0  -6  -6  -6  -6  -6
## [239]  -6  -6   0  -5  -5  -5  -5  -8  -8   1   1   1   1   1   9   9   9
## [256]   8   0   8   8  -7   8   8   5   5  -7  -7  -7  -1  -3  -3   3   5
## [273]   0  -1  -7  -7  10  10  -1  -1  -9  -9  -9  -9  -7   1
```

```
insurgency$occ # Binary
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [36] 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1
##  [71] 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 0 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
## [141] 1 1 1 1 1 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [176] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [211] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
## [246] 0 0 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
## [281] 0 0 1 0 1 0
```

```
scatterplot(insurgency$pol2, insurgency$dur)
```

```
## Error in scatterplot(insurgency$pol2, insurgency$dur): could not find function "scatterplot"
```

```
cor.test(insurgency$pol2, insurgency$dur)
```

```
##
##  Pearson's product-moment correlation
##
## data:  insurgency$pol2 and insurgency$dur
## t = -0.94199, df = 284, p-value = 0.347
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.17068852  0.06056555
## sample estimates:
##         cor
## -0.05580994
```

```
cor(insurgency[,c("pol2", "dur")], use = "pairwise.complete.obs")
```

```
##             pol2         dur
## pol2  1.00000000 -0.05580994
## dur  -0.05580994  1.00000000
```

```
table(insurgency$occ, insurgency$wdl)
```

```
##
##      0  1  2
##   0 35 46 93
##   1 43  9 60
```

```
cs = chisq.test(insurgency$occ, insurgency$wdl)
cs
```

```
##
##  Pearson's Chi-squared test
##
## data:  insurgency$occ and insurgency$wdl
## X-squared = 20.345, df = 2, p-value = 3.821e-05
```

```
cs$stdres
```

```
##                insurgency$wdl
## insurgency$occ             0           1           2
##              0 -3.38777319  3.85409426 -0.02038164
##              1  3.38777319 -3.85409426  0.02038164
```

```
cs$expected
```

```
##                insurgency$wdl
## insurgency$occ        0        1        2
##              0 47.45455 33.46154 93.08392
##              1 30.54545 21.53846 59.91608
```

Extensive example

Create data

```
# car gives us nice scatterplots
library(car)
```

```
## Error in library(car): there is no package called 'car'
```

```
# We'll use our Country-by-Country dataset
load("./data/Countries2.Rdata")
summary(Countries)
```

```
##    Country          infant.mortality      gdp         fertility_rate
##  Length:190         Min.   :  3.00   Min.   :   36   Min.   :1.190
##  Class :character   1st Qu.: 13.00   1st Qu.:  435   1st Qu.:1.855
##  Mode  :character   Median : 35.00   Median : 1570   Median :3.070
##                     Mean   : 46.48   Mean   : 6195   Mean   :3.542
##                     3rd Qu.: 72.50   3rd Qu.: 6232   3rd Qu.:5.025
##                     Max.   :154.00   Max.   :42416   Max.   :7.600
##                     NA's   :47       NA's   :45      NA's   :47
##  contraception        region    Country_Code internet_users_2010
##  Min.   : 2.00   Africa  :42   AFG    :  1   Min.   : 0.25
##  1st Qu.:19.75   Americas:26   AGO    :  1   1st Qu.:10.00
##  Median :46.00   Asia    :28   ALB    :  1   Median :27.67
##  Mean   :41.49   Europe  :38   AND    :  1   Mean   :33.61
##  3rd Qu.:61.25   Oceania :14   ARG    :  1   3rd Qu.:53.00
##  Max.   :83.00   NA's    :42   (Other):143   Max.   :95.00
##  NA's   :78                    NA's   : 42   NA's   :45
##  internet_users_2011 Corruption_Rank       cpi        internet_growth
##  Min.   : 0.98       Min.   :  1.00   Min.   : 8.00   Min.   :-0.12500
##  1st Qu.:11.67       1st Qu.: 44.50   1st Qu.:28.75   1st Qu.: 0.05765
##  Median :32.00       Median : 88.00   Median :37.00   Median : 0.11321
##  Mean   :37.50       Mean   : 87.17   Mean   :43.27   Mean   : 0.21683
##  3rd Qu.:59.50       3rd Qu.:130.75   3rd Qu.:56.25   3rd Qu.: 0.24764
##  Max.   :95.02       Max.   :174.00   Max.   :90.00   Max.   : 2.92000
##  NA's   :50          NA's   :14       NA's   :14      NA's   :51
##    high_cpi         high_gdp
##  Length:190         High:72
##  Class :character   Low :73
##  Mode  :character   NA's:45
```

```
##
##
##
##
```

```
# We'll also use Google's dataset of takedown requests -
# that is, orders that come from governments of
# different countries to remove certain content
# from Youtube, search results, and other online products.
# Each row of this dataset corresponds to a specific
# country and a specific online product (you can think
# of the unit of analysis as country x product), and there
# are several variables of interest:
#
# Country - the country making specific takedown requests
# Product - the online product the content is hosted on
#           (Youtube, Blogger, etc)
# Reason - a reason why the content is being targeted
#           (copyright violation, government criticism, etc..)
# Court.Orders - the number of requests from the Country's
#             court system
# Executive..Police..etc. - the number of requests from the
#             executive and other branches of government
# Items.Requested.To.Be.Removed - the number of separate items
#             of content.  However, this variable seems to
#             have a lot of missing values

# Read in the data
Requests = read.csv("./data/Removal_Requests.csv")
head(Requests)
```

```
##   Period.Ending Country CLDR.Territory.Code                  Product
## 1      12/31/09  Brazil                  BR                  Blogger
## 2      12/31/09  Brazil                  BR                    orkut
## 3      12/31/09  Brazil                  BR                    Gmail
## 4      12/31/09  Brazil                  BR Web Search: Autocomplete
## 5      12/31/09  Brazil                  BR                  YouTube
## 6      12/31/09  Brazil                  BR               Web Search
##   Reason Court.Orders Executive..Police..etc.
## 1                 21                      5
## 2                 99                    119
## 3                  4                      0
## 4                  0                      1
## 5                 32                      1
## 6                  9                      0
##   Items.Requested.To.Be.Removed
## 1                            NA
## 2                            NA
## 3                            NA
## 4                            NA
## 5                            NA
## 6                            NA
```

```
# Note that there are multiple rows per country in
# the Requests dataframe.
```

```r
# Create a new variable for total number of requests from
# all branches of government
Requests$total.takedowns = Requests$Court.Orders + Requests$Executive..Police..etc.

# To merge our datasets, we first need to sum together all the
# rows for each country in the Requests dataset, so that
# each country only appears in one row.
# (we'll lose some variables when we do this, such as the product
# the request referred to)
R2 = aggregate(Requests[,c("Court.Orders", "Executive..Police..etc.", "total.takedowns")], list(Country

# Notice that there's one row per country now.
head(R2)
```

```
##      Country Court.Orders Executive..Police..etc. total.takedowns
## 1  Argentina          205                      22             227
## 2  Australia           29                      63              92
## 3    Austria           12                       2              14
## 4 Azerbaijan            1                       1               2
## 5 Bangladesh            0                       3               3
## 6    Belgium            5                      42              47
```

```r
# Perform the merge
Countries = merge(Countries, R2, by="Country", all=T)

head(Countries)
```

```
##       Country infant.mortality  gdp fertility_rate contraception   region
## 1 Afghanistan             154 2848           6.90            NA     Asia
## 2     Albania              32  863           2.60            NA   Europe
## 3     Algeria              44 1531           3.81            52   Africa
## 4     Andorra              NA   NA             NA            NA   Europe
## 5      Angola             124  355           6.69            NA   Africa
## 6   Argentina              22 8055           2.62            NA Americas
##   Country_Code internet_users_2010 internet_users_2011 Corruption_Rank cpi
## 1          AFG                4.0                5.000             174   8
## 2          ALB               45.0               49.000             113  33
## 3          DZA               12.5               14.000             105  34
## 4          AND               81.0               81.000              NA  NA
## 5          AGO               10.0               14.776             157  22
## 6          ARG               40.0               47.704             102  35
##   internet_growth high_cpi high_gdp Court.Orders Executive..Police..etc.
## 1      0.25000000  Corrupt     High           NA                      NA
## 2      0.08888889  Corrupt      Low           NA                      NA
## 3      0.12000000  Corrupt      Low           NA                      NA
## 4      0.00000000     <NA>     <NA>           NA                      NA
## 5      0.47760000  Corrupt      Low           NA                      NA
## 6      0.19260000  Corrupt     High          205                      22
##   total.takedowns
## 1              NA
## 2              NA
## 3              NA
## 4              NA
## 5              NA
```

```
## 6             227
```

Correlation

```
### Correlation: Linear relationships between metric variables

# Let's examine the relationship between corruption
# and takedown requests.

# Use a scatterplot to see how linear the relationship looks
scatterplot(Countries$cpi, Countries$total.takedowns)
```

```
## Error in scatterplot(Countries$cpi, Countries$total.takedowns): could not find function "scatterplot"
```

```
#check the correlation
cor.test(Countries$cpi, Countries$total.takedowns)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Countries$cpi and Countries$total.takedowns
## t = 0.7427, df = 74, p-value = 0.46
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1421955  0.3055476
## sample estimates:
##        cor
## 0.08601762
```

```
# the cor function allows us to construct a correlation matrix
cor(Countries[,c("gdp", "cpi", "total.takedowns")], use = "pairwise.complete.obs")
```

```
##                      gdp        cpi total.takedowns
## gdp             1.00000000 0.77518916      0.02167304
## cpi             0.77518916 1.00000000      0.08601762
## total.takedowns 0.02167304 0.08601762      1.00000000
```

```
# the output is actually a matrix object, so we can
# do things like square each value to get R-squared
cor(Countries[,c("gdp", "cpi", "total.takedowns")], use = "pairwise.complete.obs")**2
```

```
##                         gdp         cpi total.takedowns
## gdp             1.0000000000 0.600918228    0.0004697205
## cpi             0.6009182276 1.000000000    0.0073990311
## total.takedowns 0.0004697205 0.007399031    1.0000000000
```

Chi-square test

```
### Chi-square: Testing for relationships between categorical variables
# Here are three different approaches, depending on structure of dataset

## 1. Two categorical variables

# Look at the frequency table between region and whether a country is corrupt
table(Countries$region, Countries$high_cpi)
```

```
##
```

```
##            Corrupt Trustworthy
##   Africa        35           7
##   Americas      17           7
##   Asia          19           8
##   Europe         9          26
##   Oceania        2           4
```

```
# We store the results of our chi-square test so we can extract more
# values from the output
cs = chisq.test(Countries$region, Countries$high_cpi)
```

```
## Warning in chisq.test(Countries$region, Countries$high_cpi): Chi-squared
## approximation may be incorrect
```

```
# Examine the test result
cs
```

```
##
##  Pearson's Chi-squared test
##
## data:  Countries$region and Countries$high_cpi
## X-squared = 31.08, df = 4, p-value = 2.948e-06
```

```
# Look at the std. residuals to see which regions contribute most to the result
cs$stdres
```

```
##                 Countries$high_cpi
## Countries$region   Corrupt Trustworthy
##         Africa    3.553393   -3.553393
##         Americas  1.069556   -1.069556
##         Asia      1.094985   -1.094985
##         Europe   -5.011243    5.011243
##         Oceania  -1.432885    1.432885
```

```
# Check the expected counts to see if any are less than 5 and
# if we should therefore try Fisher's exact test
cs$expected
```

```
##                 Countries$high_cpi
## Countries$region   Corrupt Trustworthy
##         Africa   25.701493   16.298507
##         Americas 14.686567    9.313433
##         Asia     16.522388   10.477612
##         Europe   21.417910   13.582090
##         Oceania   3.671642    2.328358
```

```
# Use Fisher's exact test in this case:
fisher.test(Countries$region, Countries$high_cpi)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  Countries$region and Countries$high_cpi
## p-value = 1.563e-06
## alternative hypothesis: two.sided
```

```
# For an effect size, we could compute Cramer's V manually
# We may wish to put the code in a function so we can use
# it again whenever we want.
```

```r
cramers_v = function(cs)
{
    cv = sqrt(cs$statistic / (sum(cs$observed) * (min(dim(cs$observed))-1)))
    print.noquote("Cramer's V:")
    return(as.numeric(cv))
}

# run our new function on our chi-square test
cramers_v(cs)
```

```
## [1] Cramer's V:
```

```
## [1] 0.4816019
```

```r
# As a rule of thumb,
# Cramer's V under .2 is weak
# between .2 and .4 is strong
# and above .4 is very strong

## 2. Count data, one variable in columns

# Consider each request to be the unit of analysis, and consider two variables:
# Whether it came from a corrupt or trustworthy country; and whether it came
# through a court order or executive/police action.  We want to know if these
# variables are independent or related.

# We can use aggregate to collapse the rows to just the high_cpi variable
Corrupt_Source = aggregate(Countries[,c("Court.Orders", "Executive..Police..etc.")], list(high_cpi = Co

# Note that we've created a table of counts:
Corrupt_Source
```

```
##      high_cpi Court.Orders Executive..Police..etc.
## 1     Corrupt         2032                    1863
## 2 Trustworthy         2328                    2432
```

```r
# Not required, but we can add row names to make the chi-square output prettier
rownames(Corrupt_Source)=Corrupt_Source$high_cpi

# We want to plug our count table into the chi-square test
# but we first have to remove the first column,
# because it's a factor.
# Otherwise, R will throw an error.
# Notice that we can use a negative index to omit columns
# That is, we can choose columns 2 and 3 with c(2,3)
# or we can get the same thing by skipping column 1 with c(-1)
Corrupt_Source[,c(-1)]
```

```
##             Court.Orders Executive..Police..etc.
## Corrupt             2032                    1863
## Trustworthy         2328                    2432
```

```r
# Plug this into the Chi-square test
cs = chisq.test(Corrupt_Source[,c(-1)])
cs
```

```
##
```

```
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Corrupt_Source[, c(-1)]
## X-squared = 8.9873, df = 1, p-value = 0.002719
```

```
# Look at the standardized residuals to see which direction the effect is in
cs$stdres
```

```
##             Court.Orders Executive..Police..etc.
## Corrupt         3.019497               -3.019497
## Trustworthy    -3.019497                3.019497
```

```
# Check the expected counts to see if any are less than 5 and
# if we should therefore try Fisher's exact test
cs$expected
```

```
##             Court.Orders Executive..Police..etc.
## Corrupt         1962.126                1932.874
## Trustworthy     2397.874                2362.126
```

```
# Since we have a 2x2 matrix, we can measure the effect
# size elegantly as an odds ratio.
# First, get the odds an order came from a Court for
# corrupt countries
corrupt_odds = Corrupt_Source["Corrupt","Court.Orders"] / Corrupt_Source["Corrupt","Executive..Police..

# Do the same for the trustworth countries.
trustworthy_odds = Corrupt_Source["Trustworthy","Court.Orders"] / Corrupt_Source["Trustworthy","Executi

# The odds ratio is just one divided by the other
corrupt_odds / trustworthy_odds
```

```
## [1] 1.13944
```

```
## 3. Count data, both variables in rows

# Let's see if corrupt countries are likely to target different products
# than trustworthy ones.  For this, we can't aggregate our data by Country
# so go back to the original request data, and merge in the high_cpi variable
# also, remove countries that are missing corruption data
Requests2 = merge(Countries[,c("Country", "high_cpi")], Requests, by="Country")
Requests2 = Requests2[ ! is.na(Requests2$high_cpi),]
head(Requests2)
```

```
##     Country high_cpi Period.Ending CLDR.Territory.Code        Product
## 1 Argentina  Corrupt       6/30/11                  AR     Web Search
## 2 Argentina  Corrupt       6/30/11                  AR     Web Search
## 3 Argentina  Corrupt       6/30/11                  AR Google AdWords
## 4 Argentina  Corrupt       6/30/11                  AR     Web Search
## 5 Argentina  Corrupt       6/30/11                  AR        Blogger
## 6 Argentina  Corrupt       6/30/11                  AR        Blogger
##                  Reason Court.Orders Executive..Police..etc.
## 1                 Other            1                       0
## 2 Privacy and Security            8                       0
## 3           Defamation            0                       1
## 4           Defamation            6                       0
## 5           Defamation            4                       0
```

```
## 6 Privacy and Security            1                       0
##   Items.Requested.To.Be.Removed total.takedowns
## 1                             1               1
## 2                            14               8
## 3                             4               1
## 4                            34               6
## 5                             5               4
## 6                             1               1
```

```
# We want separate columns for takedown requests from corrupt countries
# and from trustworthy countries.  Here, we create both columns, and copy
# each value for total.takedowns to the appropriate one.
Corrupt_Product = Requests2[,c("Product","high_cpi")]
Corrupt_Product$Corrupt = ifelse(Requests2$high_cpi == "Corrupt", Requests2$total.takedowns, 0)
Corrupt_Product$Trustworthy = ifelse(Requests2$high_cpi == "Trustworthy", Requests2$total.takedowns, 0)

# Observe that each row only has a positive value in one of the two new columns
head(Corrupt_Product)
```

```
##          Product high_cpi Corrupt Trustworthy
## 1     Web Search  Corrupt       1           0
## 2     Web Search  Corrupt       8           0
## 3 Google AdWords  Corrupt       1           0
## 4     Web Search  Corrupt       6           0
## 5        Blogger  Corrupt       4           0
## 6        Blogger  Corrupt       1           0
```

```
# Next we sum Corrupt and Trustworthy columns for each product.
Corrupt_Product =  aggregate(Corrupt_Product[,c("Corrupt","Trustworthy")], list( Product = Corrupt_Produ

# We are left with a contingency table
Corrupt_Product
```

```
##                                    Product Corrupt Trustworthy
## 1         __\x84\xe2___\xb5\x84\xe2\x84\xdc      26           0
## 2                                  Blogger     600         743
## 3                                    Gmail      22          80
## 4                            Google AdSense       2           1
## 5                            Google AdWords       3         131
## 6                         Google App Engine       1           0
## 7                               Google Apps       0           4
## 8                              Google Books       3           6
## 9                               Google Code       2           0
## 10                              Google Docs       2          11
## 11                             Google Earth       1           0
## 12 Google Earth, Google Maps, and Panoramio       7          20
## 13                            Google Groups       9          59
## 14                            Google Images      24          45
## 15                              Google Knol       1           0
## 16                              Google Maps       1           4
## 17                          Google Notebook       1           0
## 18                            Google Places       2           6
## 19                         Google Play Apps       0           4
## 20                  Google Product Search       0           1
## 21                        Google Profiles       4           1
## 22                          Google Scholar       0           2
```

```
## 23                             Google Sites      12          15
## 24                        Google SMS Channels       1           0
## 25                             Google Videos       7          21
## 26                                   Google+       7          16
## 27                             Google+ Local       4          25
## 28                                     orkut    1247           1
## 29                                 Panoramio       1           3
## 30                           Picasa Web Albums      20          15
## 31                                 Street View       3           4
## 32                                  Web Search     410        1658
## 33                  Web Search: Autocomplete       8          15
## 34                Web Search: Related results       0           3
## 35                                   YouTube    1464        1866
```

```
# We could have also created the table in one step, using the cast command
library(reshape)
```

```
## Error in library(reshape): there is no package called 'reshape'
```

```
Corrupt_Product = cast(Requests2, Product ~ high_cpi , fun = sum, value = c("total.takedowns"))
```

```
## Error in cast(Requests2, Product ~ high_cpi, fun = sum, value = c("total.takedowns")): could not fin
```

```
Corrupt_Product
```

```
##                                     Product Corrupt Trustworthy
## 1         __\x84\xe2___\xb5\x84\xe2\x84\xdc      26           0
## 2                                   Blogger     600         743
## 3                                     Gmail      22          80
## 4                               Google AdSense       2           1
## 5                               Google AdWords       3         131
## 6                            Google App Engine       1           0
## 7                                 Google Apps       0           4
## 8                                Google Books       3           6
## 9                                 Google Code       2           0
## 10                                Google Docs       2          11
## 11                               Google Earth       1           0
## 12 Google Earth, Google Maps, and Panoramio       7          20
## 13                               Google Groups       9          59
## 14                               Google Images      24          45
## 15                                 Google Knol       1           0
## 16                                 Google Maps       1           4
## 17                              Google Notebook       1           0
## 18                                Google Places       2           6
## 19                             Google Play Apps       0           4
## 20                       Google Product Search       0           1
## 21                             Google Profiles       4           1
## 22                              Google Scholar       0           2
## 23                                Google Sites      12          15
## 24                          Google SMS Channels       1           0
## 25                               Google Videos       7          21
## 26                                     Google+       7          16
## 27                               Google+ Local       4          25
## 28                                       orkut    1247           1
## 29                                   Panoramio       1           3
## 30                             Picasa Web Albums      20          15
```

```
## 31                              Street View       3           4
## 32                               Web Search     410        1658
## 33                  Web Search: Autocomplete       8          15
## 34              Web Search: Related results       0           3
## 35                                  YouTube    1464        1866
```

```
# Run a chi-square test as before
cs = chisq.test(Corrupt_Product[,c(-1)])
```

```
## Warning in chisq.test(Corrupt_Product[, c(-1)]): Chi-squared approximation
## may be incorrect
```

```
cs
```

```
##
##  Pearson's Chi-squared test
##
## data:  Corrupt_Product[, c(-1)]
## X-squared = 2292.2, df = 34, p-value < 2.2e-16
```

```
# Check standardized residuals
cs$stdres
```

```
##            Corrupt  Trustworthy
##  [1,]   5.64533503  -5.64533503
##  [2,]  -0.26189811   0.26189811
##  [3,]  -4.78559029   4.78559029
##  [4,]   0.75436229  -0.75436229
##  [5,] -10.02835447  10.02835447
##  [6,]   1.10554096  -1.10554096
##  [7,]  -1.80959184   1.80959184
##  [8,]  -0.70406271   0.70406271
##  [9,]   1.56356135  -1.56356135
## [10,]  -2.14816521   2.14816521
## [11,]   1.10554096  -1.10554096
## [12,]  -1.99562733   1.99562733
## [13,]  -5.28641745   5.28641745
## [14,]  -1.71330740   1.71330740
## [15,]   1.10554096  -1.10554096
## [16,]  -1.12411446   1.12411446
## [17,]   1.10554096  -1.10554096
## [18,]  -1.13775402   1.13775402
## [19,]  -1.80959184   1.80959184
## [20,]  -0.90463908   0.90463908
## [21,]   1.57344856  -1.57344856
## [22,]  -1.27942678   1.27942678
## [23,]  -0.05841843   0.05841843
## [24,]   1.10554096  -1.10554096
## [25,]  -2.13100874   2.13100874
## [26,]  -1.40622461   1.40622461
## [27,]  -3.38398517   3.38398517
## [28,]  42.15374789 -42.15374789
## [29,]  -0.80432757   0.80432757
## [30,]   1.44657633  -1.44657633
## [31,]  -0.11415970   0.11415970
## [32,] -26.38021279  26.38021279
## [33,]  -0.98653930   0.98653930
```

```
## [34,]  -1.56706194   1.56706194
## [35,]  -1.53634841   1.53634841
```

```
# And expected values
cs$expected
```

```
##            Corrupt  Trustworthy
##  [1,]   11.7007510   14.2992490
##  [2,]  604.3887926  738.6112074
##  [3,]   45.9029463   56.0970537
##  [4,]    1.3500867    1.6499133
##  [5,]   60.3038706   73.6961294
##  [6,]    0.4500289    0.5499711
##  [7,]    1.8001155    2.1998845
##  [8,]    4.0502600    4.9497400
##  [9,]    0.9000578    1.0999422
## [10,]    5.8503755    7.1496245
## [11,]    0.4500289    0.5499711
## [12,]   12.1507799   14.8492201
## [13,]   30.6019642   37.3980358
## [14,]   31.0519931   37.9480069
## [15,]    0.4500289    0.5499711
## [16,]    2.2501444    2.7498556
## [17,]    0.4500289    0.5499711
## [18,]    3.6002311    4.3997689
## [19,]    1.8001155    2.1998845
## [20,]    0.4500289    0.5499711
## [21,]    2.2501444    2.7498556
## [22,]    0.9000578    1.0999422
## [23,]   12.1507799   14.8492201
## [24,]    0.4500289    0.5499711
## [25,]   12.6008088   15.3991912
## [26,]   10.3506644   12.6493356
## [27,]   13.0508377   15.9491623
## [28,]  561.6360485  686.3639515
## [29,]    1.8001155    2.1998845
## [30,]   15.7510110   19.2489890
## [31,]    3.1502022    3.8497978
## [32,]  930.6597343 1137.3402657
## [33,]   10.3506644   12.6493356
## [34,]    1.3500867    1.6499133
## [35,] 1498.5961872 1831.4038128
```

```
# The fisher test is probably too computationally intensive to run
#fisher.test(Corrupt_Product[,c(-1)])

# could also use monte-carlo simulation to check significance
chisq.test(Corrupt_Product[,c(-1)], simulate.p.value = T)
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  Corrupt_Product[, c(-1)]
## X-squared = 2292.2, df = NA, p-value = 0.0004998
```

```r
# let's use the function we wrote earlier to check the effect size
cramers_v(cs)
```

```
## [1] Cramer's V:
```

```
## [1] 0.5146303
```

T-test

T-test types

- Independent
- Dependent
- Parametric/non-parametric
- if underline distribution is normally distributed => parametric
- if not non-parametric, can use rank
  - wilcox.test()

dependent (pared) t test

- t.test()
- If dataframe has single column: use t.test(outcome ~ predictor, data, paired = F/T)
- If dataframe has 2 columns: use t.test(score group 1, score group 2, paired = F/T)

R functions for t-distribution

- dt : Density of t-distributon
- pt : Distribution prob of t-distribution
- qt : quantile
- rt : generate random variable

T-tests example 1

```r
# Use the Countries dataset, including takedown variables
load("./data/Countries3.Rdata")
summary(Countries)
```

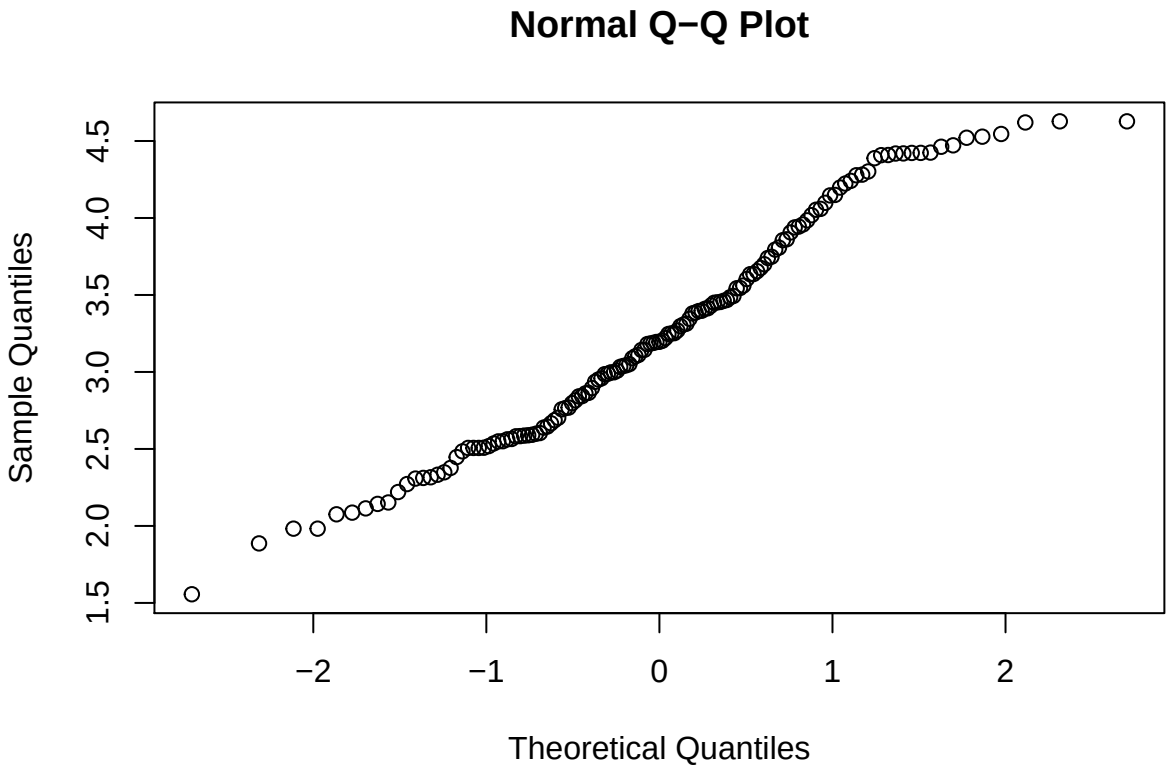
```
##    Country          infant.mortality      gdp        fertility_rate
##  Length:194         Min.   :  3.00   Min.   :   36   Min.   :1.190
##  Class :character   1st Qu.: 13.00   1st Qu.:  435   1st Qu.:1.855
##  Mode  :character   Median : 35.00   Median : 1570   Median :3.070
##                     Mean   : 46.48   Mean   : 6195   Mean   :3.542
##                     3rd Qu.: 72.50   3rd Qu.: 6232   3rd Qu.:5.025
##                     Max.   :154.00   Max.   :42416   Max.   :7.600
##                     NA's   :51       NA's   :49      NA's   :51
##  contraception        region    Country_Code internet_users_2010
##  Min.   : 2.00   Africa  :42   AFG    :  1   Min.   : 0.25
##  1st Qu.:19.75   Americas:26   AGO    :  1   1st Qu.:10.00
##  Median :46.00   Asia    :28   ALB    :  1   Median :27.67
##  Mean   :41.49   Europe  :38   AND    :  1   Mean   :33.61
##  3rd Qu.:61.25   Oceania :14   ARG    :  1   3rd Qu.:53.00
##  Max.   :83.00   NA's    :46   (Other):143   Max.   :95.00
```

```
##  NA's   :82                    NA's   : 46   NA's   :49
##  internet_users_2011 Corruption_Rank       cpi        internet_growth
##  Min.   : 0.98       Min.   :  1.00   Min.   : 8.00   Min.   :-0.12500
##  1st Qu.:11.67       1st Qu.: 44.50   1st Qu.:28.75   1st Qu.: 0.05765
##  Median :32.00       Median : 88.00   Median :37.00   Median : 0.11321
##  Mean   :37.50       Mean   : 87.17   Mean   :43.27   Mean   : 0.21683
##  3rd Qu.:59.50       3rd Qu.:130.75   3rd Qu.:56.25   3rd Qu.: 0.24764
##  Max.   :95.02       Max.   :174.00   Max.   :90.00   Max.   : 2.92000
##  NA's   :54          NA's   :18       NA's   :18      NA's   :55
##    high_cpi         high_gdp   Court.Orders     Executive..Police..etc.
##  Length:194         High:72   Min.   :   0.00   Min.   :  0.00
##  Class :character   Low :73   1st Qu.:   0.00   1st Qu.:  1.00
##  Mode  :character   NA's:49   Median :   1.00   Median :  4.00
##                               Mean   :  53.34   Mean   : 59.57
##                               3rd Qu.:   7.00   3rd Qu.: 11.75
##                               Max.   :1539.00   Max.   :719.00
##                               NA's   :112       NA's   :112
##  total.takedowns
##  Min.   :   1.00
##  1st Qu.:   2.00
##  Median :   5.00
##  Mean   : 112.91
##  3rd Qu.:  23.75
##  Max.   :2258.00
##  NA's   :112
```

```r
# look at log gdp between the corrupt and
# trustworthy Country groups
Countries$loggdp = log10(Countries$gdp)

# The means look different between groups
by(Countries$loggdp, Countries$high_cpi, mean, na.rm = TRUE)
```

```
## Countries$high_cpi: Corrupt
## [1] 2.859729
## ------------------------------------------------------------
## Countries$high_cpi: Trustworthy
## [1] 3.821505
```

```r
# But is this statistically significant?

# From the qqplot, it's not clear if loggdp is normally distributed
qqnorm(Countries$loggdp)
```




```
# The Shapiro test suggests that it's not
shapiro.test(Countries$loggdp)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Countries$loggdp
## W = 0.97303, p-value = 0.005833
```

```
# But we have a large sample size, so we can rely on
# the central limit theorem and use a regular t.test
t.test(Countries$loggdp ~ Countries$high_cpi, Countries)
```

```
##
##  Welch Two Sample t-test
##
## data:  Countries$loggdp by Countries$high_cpi
## t = -9.2317, df = 96.768, p-value = 6.237e-15
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.1685543 -0.7549991
## sample estimates:
##     mean in group Corrupt mean in group Trustworthy
##                  2.859729                  3.821505
```

```
## Computing effect sizes
# We can manually compute Cohen's d, a common measure of effect
# size for the difference between two means.
# Quite simply, Cohen's d is the difference between the means
# divided by their pooled standard error.
# We'll place our code in a function so we can use it again later
```

```r
cohens_d <- function(x, y) {
    # this function takes two vectors as inputs, and compares
    # their means

    # first, compute the pooled standard error
  lx = length(subset(x,!is.na(x)))
  ly = length(subset(y,!is.na(y)))
    # numerator of the pooled variance:
    num = (lx-1)*var(x, na.rm=T) + (ly-1)*var(y, na.rm=T)
    pooled_var = num / (lx + ly - 2) # variance
    pooled_sd = sqrt(pooled_var)

    # finally, compute cohen's d
    cd = abs(mean(x, na.rm=T) - mean(y, na.rm=T)) / pooled_sd
    return(cd)
}

# get the vectors of loggdp for each of our two groups
loggdp_c = Countries$loggdp[Countries$high_cpi=="Corrupt"]
loggdp_t = Countries$loggdp[Countries$high_cpi=="Trustworthy"]

# plug them into our cohen's d function
cohens_d(loggdp_c, loggdp_t)
```

```
## [1] 1.692301
```

```r
# We could also compute the effect size correlation
# this is, quite simply, the correlation between the our metric
# variable and our grouping variable (suitably dummy-coded)
cor.test(Countries$loggdp, as.numeric(factor(Countries$high_cpi)))
```

```
##
##  Pearson's product-moment correlation
##
## data:  Countries$loggdp and as.numeric(factor(Countries$high_cpi))
## t = 9.5463, df = 132, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5265574 0.7295925
## sample estimates:
##       cor
## 0.6390774
```

```r
## 2. Suppose we were just looking at countries in the Americas
Americas = Countries[Countries$region == "Americas",]
summary(Americas)
```

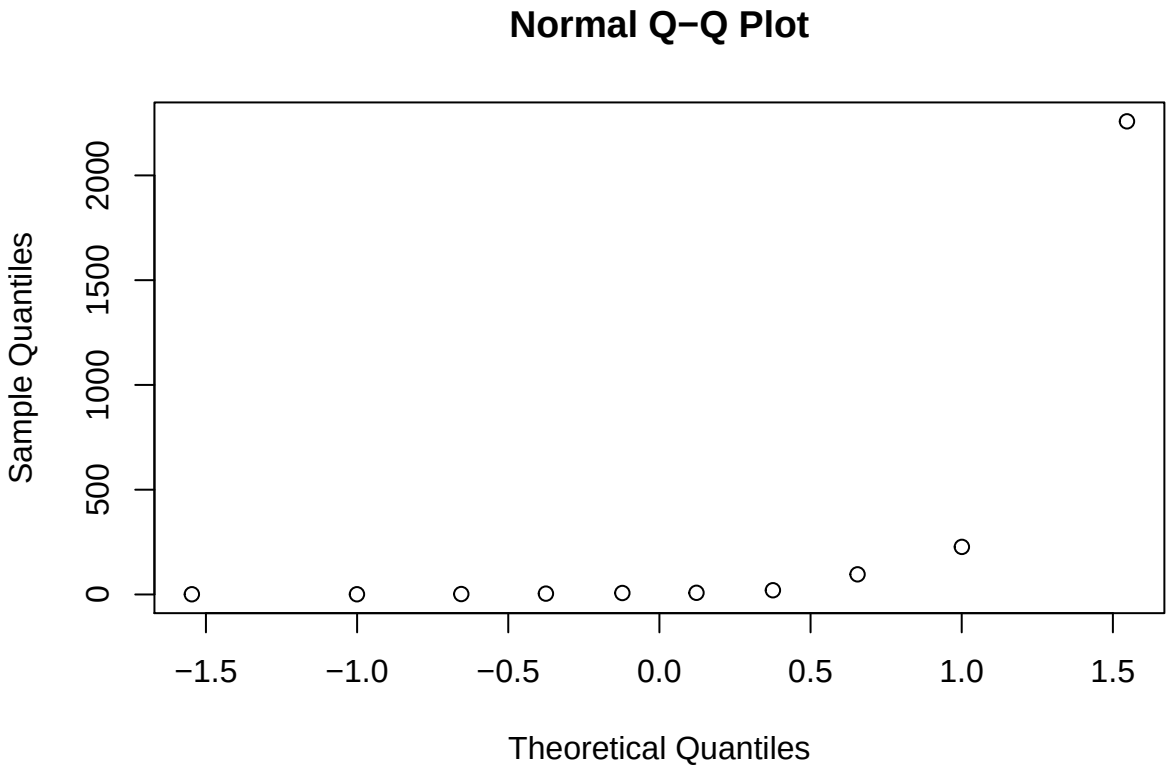
```
##    Country          infant.mortality      gdp        fertility_rate
##  Length:72          Min.   : 6.00    Min.   :  386   Min.   :1.550
##  Class :character   1st Qu.:14.00    1st Qu.: 1435   1st Qu.:2.303
##  Mode  :character   Median :24.00    Median : 2491   Median :2.660
##                     Mean   :30.56    Mean   : 3693   Mean   :2.935
##                     3rd Qu.:42.00    3rd Qu.: 4256   3rd Qu.:3.708
##                     Max.   :82.00    Max.   :18943   Max.   :4.900
##                     NA's   :47       NA's   :46      NA's   :48
```

```
##  contraception        region    Country_Code internet_users_2010
##  Min.   :18.00   Africa  : 0   ARG    : 1    Min.   : 8.37
##  1st Qu.:47.50   Americas:26   BHS    : 1    1st Qu.:20.45
##  Median :54.50   Asia    : 0   BLZ    : 1    Median :32.52
##  Mean   :53.55   Europe  : 0   BOL    : 1    Mean   :32.94
##  3rd Qu.:63.50   Oceania : 0   BRA    : 1    3rd Qu.:40.51
##  Max.   :74.00   NA's    :46   (Other):21    Max.   :80.30
##  NA's   :50                    NA's   :46    NA's   :46
##  internet_users_2011 Corruption_Rank       cpi        internet_growth
##  Min.   :10.60       Min.   :  9.00   Min.   :19.00   Min.   :0.01298
##  1st Qu.:30.70       1st Qu.: 53.75   1st Qu.:31.25   1st Qu.:0.06754
##  Median :36.50       Median : 91.00   Median :36.50   Median :0.10701
##  Mean   :39.45       Mean   : 87.67   Mean   :42.79   Mean   :0.15814
##  3rd Qu.:49.51       3rd Qu.:121.00   3rd Qu.:50.50   3rd Qu.:0.19512
##  Max.   :83.00       Max.   :165.00   Max.   :84.00   Max.   :0.51163
##  NA's   :49          NA's   :48       NA's   :48      NA's   :49
##    high_cpi         high_gdp   Court.Orders    Executive..Police..etc.
##  Length:72          High:18   Min.   :   0.0   Min.   :  0.00
##  Class :character   Low : 8   1st Qu.:   1.0   1st Qu.:  1.25
##  Mode  :character   NA's:46   Median :   2.5   Median :  6.00
##                               Mean   : 179.7   Mean   : 82.70
##                               3rd Qu.:  31.0   3rd Qu.: 19.75
##                               Max.   :1539.0   Max.   :719.00
##                               NA's   :62       NA's   :62
##  total.takedowns      loggdp
##  Min.   :   1.0   Min.   :2.587
##  1st Qu.:   2.5   1st Qu.:3.156
##  Median :   7.5   Median :3.396
##  Mean   : 262.4   Mean   :3.374
##  3rd Qu.:  77.0   3rd Qu.:3.627
##  Max.   :2258.0   Max.   :4.277
##  NA's   :62       NA's   :46
```

```
# We may ask whether the more corrupt countries in this
# group issue more or less takedown requests than the
# more trustworthy ones
by(Americas$total.takedowns, Americas$high_cpi, mean, na.rm = TRUE)
```

```
## Americas$high_cpi: Corrupt
## [1] 315.5
## ------------------------------------------------------------
## Americas$high_cpi: Trustworthy
## [1] 50
```

```
# Notice that total takedowns is not at all normal.
qqnorm(Americas$total.takedowns)
```




```
# Use the Wilcoxon rank-sum test to compare means
wilcox.test(Americas$total.takedowns ~ Americas$high_cpi)
```

```
## Warning in wilcox.test.default(x = c(227L, 1L, 2258L, 7L, 20L, 2L, 8L, 1L:
## cannot compute exact p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Americas$total.takedowns by Americas$high_cpi
## W = 7, p-value = 0.8958
## alternative hypothesis: true location shift is not equal to 0
```

```
# we can compute cohen's d using the function we wrote earlier
takedowns_c = Americas$total.takedowns[Americas$high_cpi == "Corrupt"]
takedowns_t = Americas$total.takedowns[Americas$high_cpi == "Trustworthy"]
cohens_d(takedowns_c, takedowns_t)
```

```
## [1] 0.3596996
```

```
## Let's finally compare the number of takedown requests
# issued by courts, with those issued by executives / police
mean(Countries$Court.Orders, na.rm = T)
```

```
## [1] 53.34146
```

```
mean(Countries$Executive, na.rm = T)
```

```
## [1] 59.57317
```

```
# Because there is just one group of countries, with two
# variables per country, we need a paired-samples test
# (paired = TRUE)
```

```r
#
# In general, we need a paired-sample t-test whenever
# we can pair each observation in one sample with an
# observation in the other sample, and when we expect
# the observations in each pair to vary together to
# some extent.
#
# The pairing could be formed in several ways:
#
# 1. We have two variables for each unit of analysis
# The classic example here is giving a test twice to
# the same group of individuals (pretest-posttest).
# But we could also take two different measurements at
# the same time - such as court ordered takedowns and
# executive-ordered takedowns in our example.
#
# 2. We have a natural pairing between units of analysis
# This could be the case for measurements on twins, or
# spouses.
#
# 3. We create a matched sample by pairing units of
# analysis with similar characteristics


# Because of the large sample size, we can use the parametric
# t-test
t.test(Countries$Court.Orders, Countries$Executive, paired = T)
```

```
##
##  Paired t-test
##
## data:  Countries$Court.Orders and Countries$Executive
## t = -0.36562, df = 81, p-value = 0.7156
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -40.14479  27.68138
## sample estimates:
## mean of the differences
##               -6.231707
```

```r
# effect size
cohens_d(Countries$Court.Orders, Countries$Executive)
```

```
## [1] 0.03441315
```

T-Test Example 2

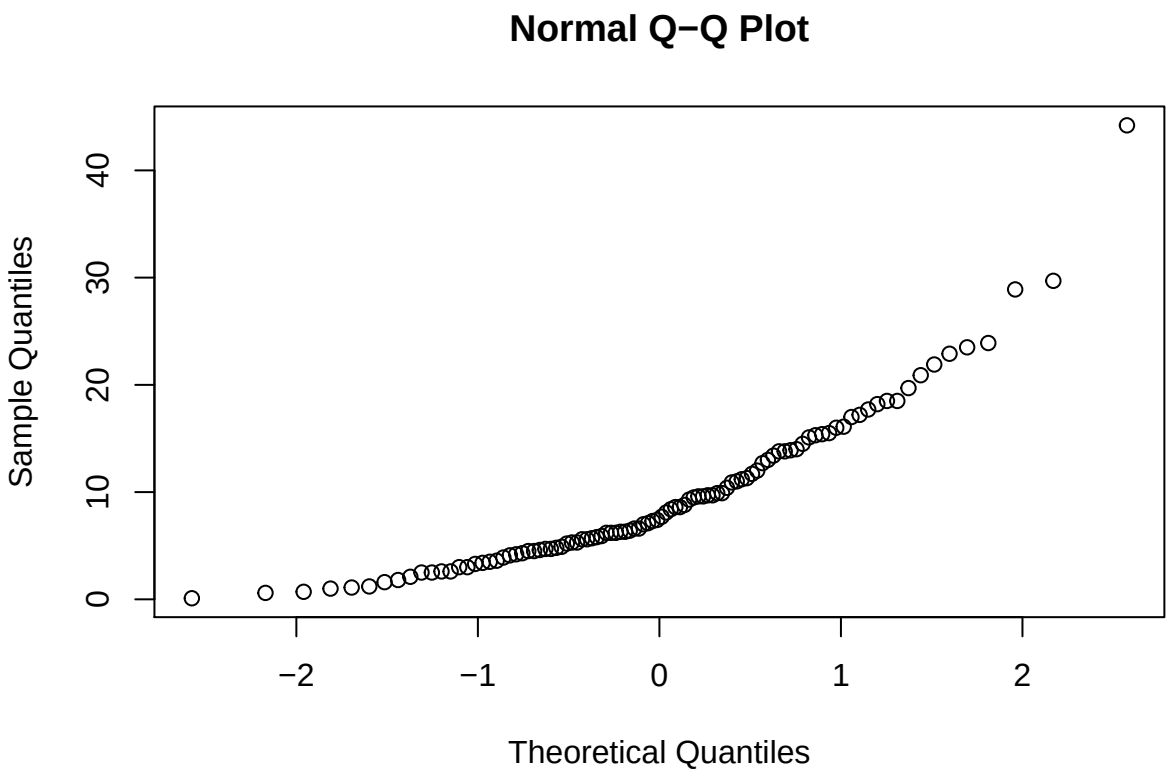
```r
# loading library
library(ggplot2)
library(pastecs)
```

```
## Error in library(pastecs): there is no package called 'pastecs'
```

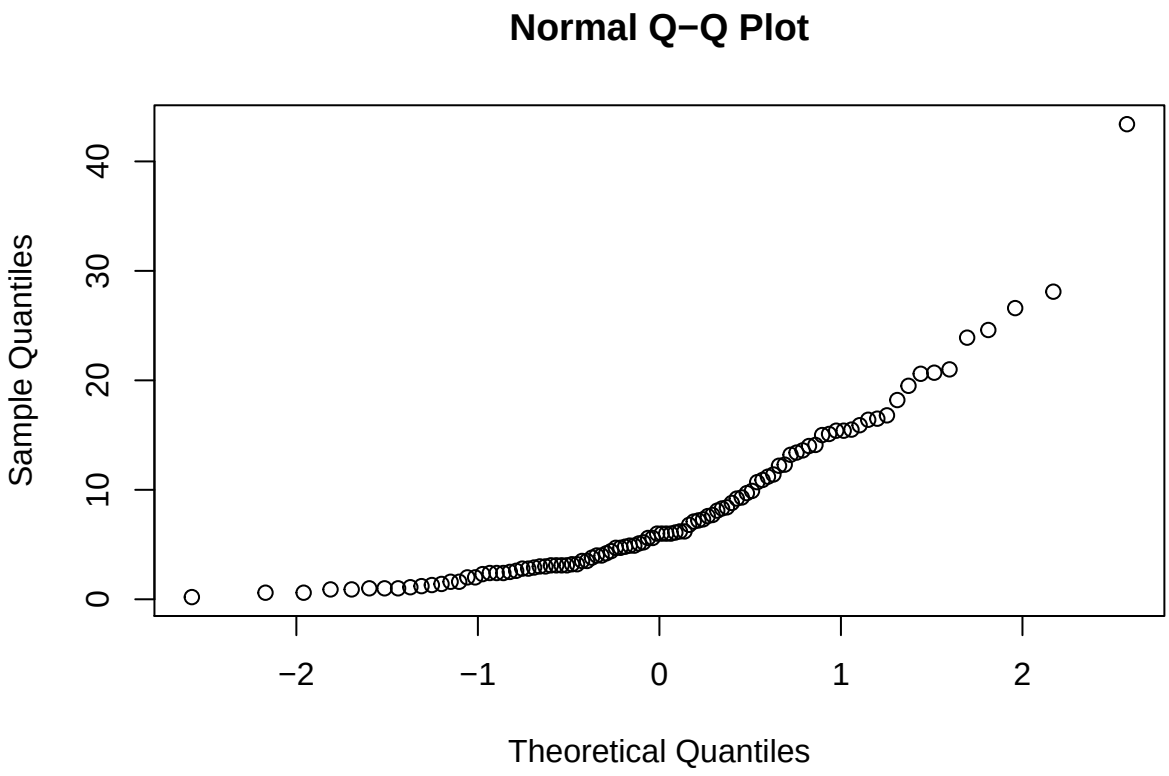
```r
# Reading US senator data
senate_data <- read.csv("data/united_states_senate_2014.csv")

# Rename long column name with shorter names
names(senate_data)[names(senate_data) == "Campaign.Money.Raised..millions.of..."] <- "Raised"
names(senate_data)[names(senate_data) == "Campaign.Money.Spent..millions.of..."] <- "Spent"

# Review the data
summary(senate_data)
```

```
##            Senator.Names    Gender          State            Party
##  Alan "Al" Franken: 1    Female:20   Alabama   : 2   Democrat   :53
##  Amy Klobuchar    : 1    Male  :80   Alaska    : 2   Independent: 2
##  Angus King       : 1                Arizona   : 2   Republican :45
##  Barbara Boxer    : 1                Arkansas  : 2
##  Barbara Mikulski : 1                California: 2
##  Benjamin Cardin  : 1                Colorado  : 2
##  (Other)          :94                (Other)   :88
##             Religion      Raised           Spent
##  Protestant     :49   Min.   : 0.100   Min.   : 0.200
##  Catholic       :27   1st Qu.: 4.575   1st Qu.: 2.975
##  Jewish         :10   Median : 7.550   Median : 6.000
##  Other Christian: 7   Mean   : 9.645   Mean   : 8.227
##  Mormon         : 2   3rd Qu.:13.800   3rd Qu.:12.225
##  Unaffiliated   : 2   Max.   :44.200   Max.   :43.400
##  (Other)        : 3
```

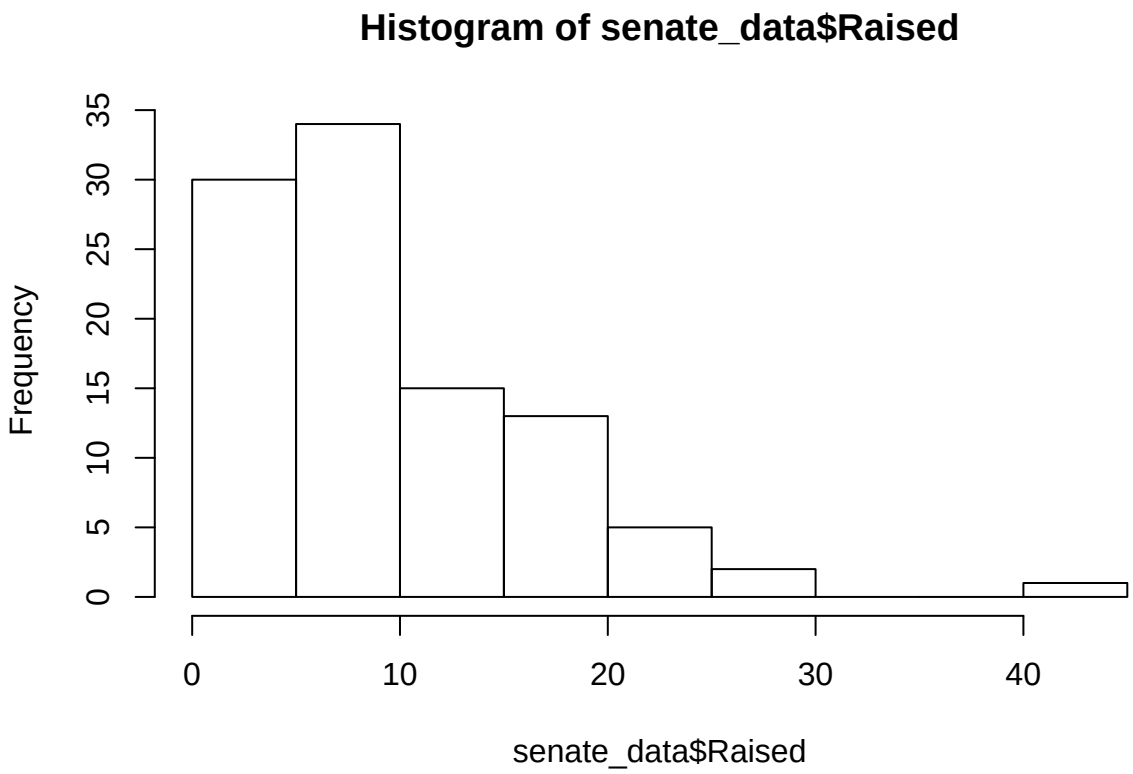
```r
# questions 1
# Is there a difference between the amount of money a senator raises and the amount spent?

# Checking assumption
# Normality
qqnorm(senate_data$Raised)
```



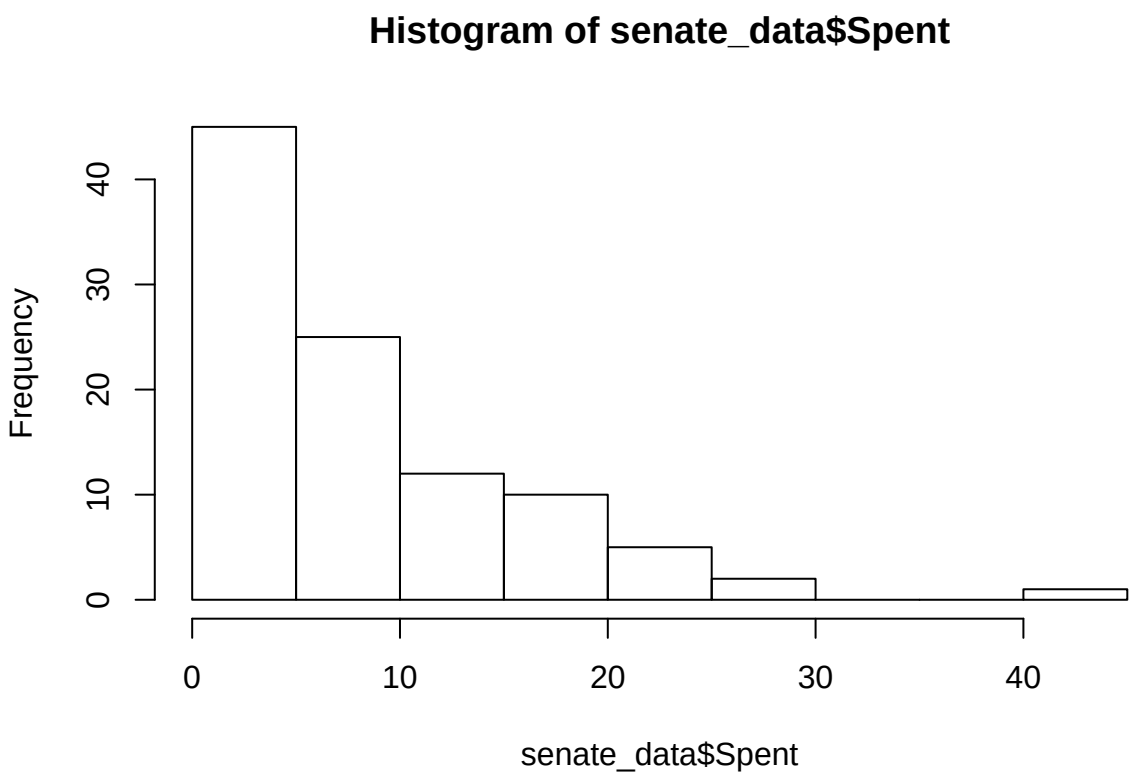

```
qqnorm(senate_data$Spent)
```




```
hist(senate_data$Raised)
```




```r
hist(senate_data$Spent)
```




```r
# Although it is normal but due to large sample size of 100, we can use parametric testing

# dependent (pared) t test
```

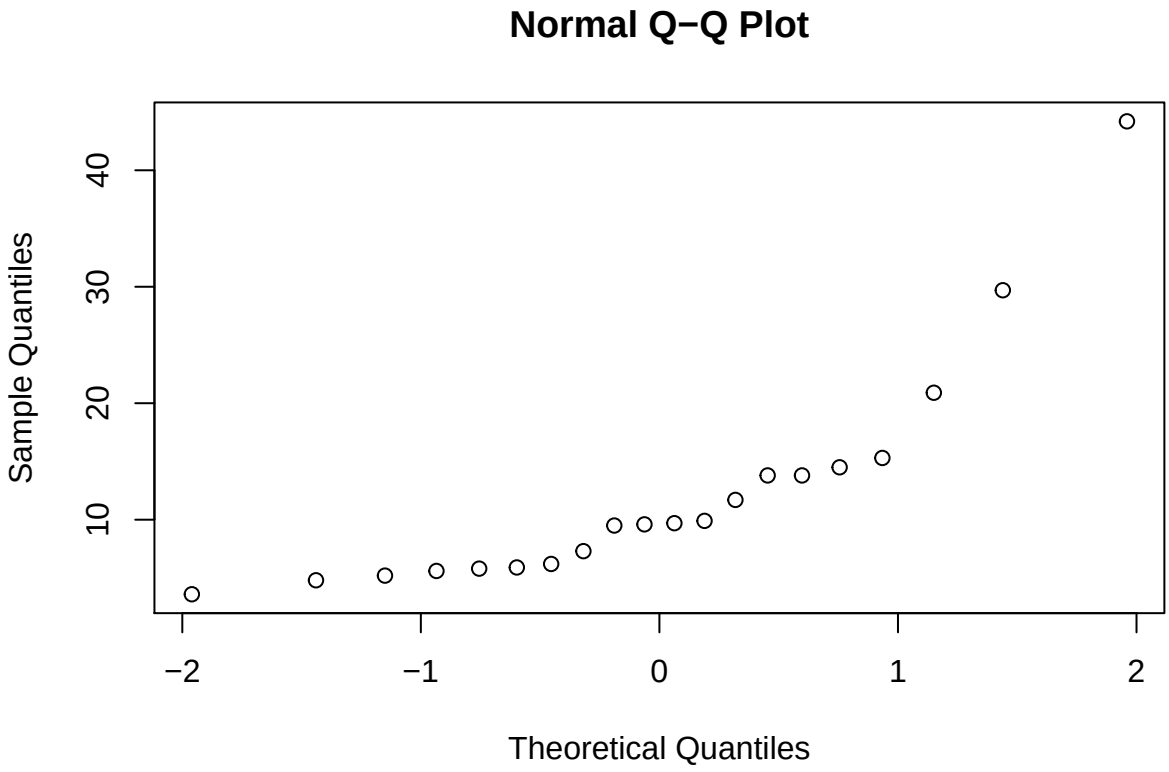
```r
# t.test()
# If dataframe has single column: use t.test(outcome ~ predictor, data, paired = F/T)
# If dataframe has 2 columns: use t.test(score group 1, score group 2, paired = F/T)

# H0: Raised = Spent
# Ha: Raised != Spent

# We have 2 columns, and they are paired or dependent
raised_vs_spent <- t.test(senate_data$Raised, senate_data$Spent, paired=T)
raised_vs_spent
```

```
##
##  Paired t-test
##
## data:  senate_data$Raised and senate_data$Spent
## t = 5.9944, df = 99, p-value = 3.329e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.9486232 1.8873768
## sample estimates:
## mean of the differences
##                   1.418
```

```r
# The results showed that the P-value is < 0.05 which is significant, 95 % CI does not contain
# 0 which means that we can reject the null hypothesis that there is no difference

# calculate the effect size
t <- raised_vs_spent$statistic[[1]]
df <- raised_vs_spent$parameter[[1]]
r <- sqrt(t^2/(t^2 + df))
round(r, 3)
```

```
## [1] 0.516
```

```r
# The effect size (r) is 0.598, it is practical significant

# Question 2
# Do female Democratic senators raise more or less money than female Republican senators?
female_senate = senate_data[senate_data$Gender == "Female",]

# need to check assumption because the sample is small only 20 female senator
qqnorm(female_senate$Raised)
```




```
shapiro.test(female_senate$Raised)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  female_senate$Raised
## W = 0.75051, p-value = 0.0001725
```

```
# Not normal, and due to small sample, we should use non-parametric test
# And it is not paired groups 1) the size of the group are different 2) the are not paired
# wilcox.test()
# if single column, wilcox.test(outcome ~ predictor, data, paired = T/F)
# if two columns, wilcox.test(score group1,  score group2, paired = T/F)

# H0: Female democratic senator raise = female republican
# Ha: Female democratic senator raise != female republican
wilcox_test_out <- wilcox.test(female_senate$Raised ~ female_senate$Party, female_senate)
```

```
## Warning in wilcox.test.default(x = c(15.3, 13.8, 11.7, 9.7, 29.7, 9.9,
## 6.2, : cannot compute exact p-value with ties
```

```
wilcox_test_out
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  female_senate$Raised by female_senate$Party
## W = 58, p-value = 0.01593
## alternative hypothesis: true location shift is not equal to 0
```

```
# P-value is < 0.05 which means it is statistically significant and we can reject the H0
```

```r
# Calculate Effect size
z = qnorm(wilcox_test_out$p.value/2)
n = length(female_senate$Gender)
r = z/sqrt(n)
r
```

```
## [1] -0.5389885
```

```r
# Effect size is -0.538 and it is practically significant as well

# Quesitons 3
# Do protestant Senators spend more or less money than non-protestant senators?
# Create a categorical variable that determine whether a senator is protestant
senate_data$is.protestant = senate_data$Religion == "Protestant"

# look at the column
summary(senate_data$is.protestant)
```

```
##    Mode   FALSE    TRUE
## logical      51      49
```

```r
# H0: protestant senators spend the same is non-protestant
# Ha: Not spent the same
# This is parametric test with reasonable large sample
# This is not a paired test
protestant_test <- t.test(senate_data$Spent ~ senate_data$is.protestant, senate_data)
protestant_test
```

```
##
##  Welch Two Sample t-test
##
## data:  senate_data$Spent by senate_data$is.protestant
## t = 1.2856, df = 97.177, p-value = 0.2016
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.038329  4.857697
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            9.162745            7.253061
```

```r
# p value is > 0.05, we cannot reject the null hypothesis that they spend the same

# Calculate effect size
t <- protestant_test$statistic[[1]]
df <- protestant_test$parameter[[1]]
r <- sqrt(t^2/(t^2 + df))
round(r, 3)
```

```
## [1] 0.129
```

```r
# The effect size is 0.129 which is small

# bootstrap method to compare median
x1 = runif(100)
x2 = runif(100) + .1
median(x1)
```

```
## [1] 0.4979396
```

```
median(x2)
```

```
## [1] 0.60574
```

```
func1 = function() {
    s1 = sample(x=x1, replace=T, size=100)
    s2 = sample(x=x2, replace=T, size=100)
    out = median(s1) - median(s2)
}

bs = replicate(n = 1000, expr = func1())

median(x2)
```

```
## [1] 0.60574
```

ANOVA

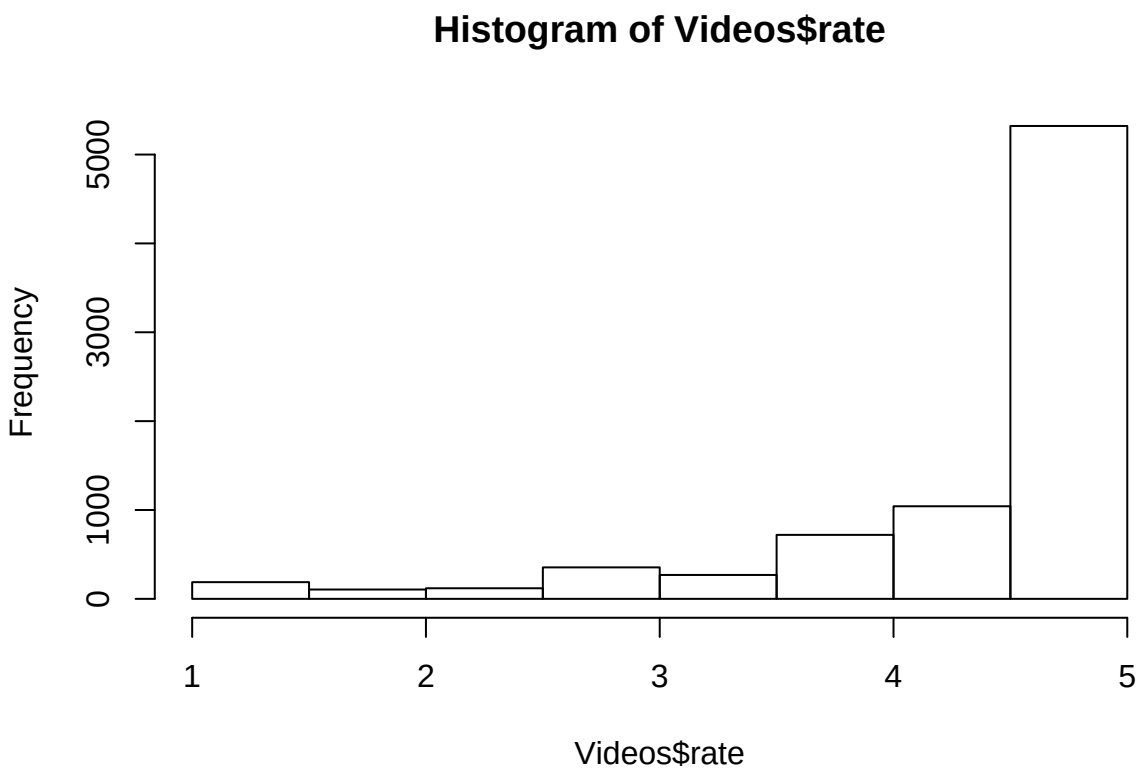
- Test several means of different groups
- Conceptualized as Multiple regression
- Test overall regression model is significant
- F-distribution (experimental manipulation has some effect)
- Evalute overall variation
- compare variability between groups to the variability within groups
- F-Ration: Model Divided by error

```
# A demonstration of ANOVA in R

# Load Youtube video data
load("data/Videos_clean.Rdata")
summary(Videos)
```

```
##         video_id               uploader         age
##  #NAME?     : 129   Pan93bn        :  56   Min.   :  0.0
##  -0Zkx9Sh6DU:   1   nikodora       :  28   1st Qu.: 31.0
##  -0yS9zc_290:   1   WWEOfficialPPVs:  22   Median :142.0
##  -0z5PEZt_Wk:   1   gar6301        :  22   Mean   :209.5
##  -1PT00GVE7k:   1   dermayon       :  20   3rd Qu.:336.0
##  -1RjRtQRoEc:   1   wishinonastar07:  20   Max.   :984.0
##  (Other)    :9484   (Other)        :9450   NA's   :45
##              category        length         views              rate
##  Music           :2676   Min.   :   1   Min.   :      3   Min.   :1.000
##  Entertainment   :2240   1st Qu.:  83   1st Qu.:    348   1st Qu.:4.220
##  People & Blogs  : 811   Median : 193   Median :   1453   Median :4.800
##  Film & Animation: 810   Mean   : 227   Mean   :   9346   Mean   :4.431
##  Comedy          : 621   3rd Qu.: 299   3rd Qu.:   6179   3rd Qu.:5.000
##  (Other)         :2415   Max.   :5289   Max.   :1807640   Max.   :5.000
##  NA's            :  45   NA's   :9      NA's   :9         NA's   :1499
##     ratings           comments
##  Min.   :   0.00   Min.   :   -2.00
##  1st Qu.:   1.00   1st Qu.:    1.00
##  Median :   5.00   Median :    3.00
##  Mean   :  20.66   Mean   :   19.99
##  3rd Qu.:  15.00   3rd Qu.:   13.00
```

```
##  Max.   :3801.00   Max.   :13211.00
##  NA's   :9         NA's   :9
```

```r
# check the rate variable for normality
hist(Videos$rate)
```




```r
# That's not great, but remember that ANOVA is a robust-test
# and the data is on a 1-5 scale, which isn't normally
# a place we'd worry

# Let's look at the means, by each category and overall
by(Videos$rate, Videos$category, mean, na.rm=T)
```

```
## Videos$category: Autos & Vehicles
## [1] 4.262821
## ------------------------------------------------------------
## Videos$category: Comedy
## [1] 4.087505
## ------------------------------------------------------------
## Videos$category: Education
## [1] 4.393839
## ------------------------------------------------------------
## Videos$category: Entertainment
## [1] 4.427854
## ------------------------------------------------------------
## Videos$category: Film & Animation
## [1] 4.52779
## ------------------------------------------------------------
## Videos$category: Gaming
## [1] 3.786579
## ------------------------------------------------------------
```

```
## Videos$category: Howto & Style
## [1] 4.232161
## ------------------------------------------------------------
## Videos$category: Music
## [1] 4.586902
## ------------------------------------------------------------
## Videos$category: News & Politics
## [1] 4.441046
## ------------------------------------------------------------
## Videos$category: Nonprofits & Activism
## [1] 4.438889
## ------------------------------------------------------------
## Videos$category: People & Blogs
## [1] 4.438432
## ------------------------------------------------------------
## Videos$category: Pets & Animals
## [1] 3.849346
## ------------------------------------------------------------
## Videos$category: Science & Technology
## [1] 4.262468
## ------------------------------------------------------------
## Videos$category: Sports
## [1] 4.477944
## ------------------------------------------------------------
## Videos$category: Travel & Events
## [1] 4.478936
```

```
mean(Videos$rate, na.rm=T)
```

```
## [1] 4.431166
```

```
# We can get nicer output with the tapply function
tapply(Videos$rate, Videos$category, mean, na.rm=T)
```

```
##      Autos & Vehicles                Comedy             Education
##              4.262821              4.087505              4.393839
##         Entertainment      Film & Animation                Gaming
##              4.427854              4.527790              3.786579
##         Howto & Style                 Music       News & Politics
##              4.232161              4.586902              4.441046
## Nonprofits & Activism        People & Blogs        Pets & Animals
##              4.438889              4.438432              3.849346
##  Science & Technology                Sports       Travel & Events
##              4.262468              4.477944              4.478936
```

```
# Perform the analysis of variance and check the significance
aovm = aov(rate ~ category, Videos)
summary(aovm)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## category      14    248  17.750   24.93 <2e-16 ***
## Residuals   8068   5744   0.712
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 1535 observations deleted due to missingness
```