R for Statistics and Graphics

Session 3

R for Inferential Statistics (Hypothesis Testing)

I. Categorical Data Analysis

Mehmet Tevfik DORAK, MD PhD

School of Life Sciences, Pharmacy & Chemistry

Kingston University London

Istanbul University, Capa Faculty of Medicine

19 April 2019

Kingston University London

Outline

Useful R Features and Functions

Some Basic Statistical Concepts

Categorical Data Analysis

Kingston University London

Generating a Simulated Vector

You can generate vectors of different kinds

```
seq(1:100)                   # To have sequential numbers from 1 to 100 in a vector
seq(from=1, to=3, by=0.1)    # To have a series of sequential numbers starting from 1 to 3
                               and increasing by 0.1
runif(4)                     # To generate four random numbers within the default range
                               of 0 to 1
runif(10, min=1, max=100)    # To generate 10 random numbers within the range 1 to 100
rnorm(n, mean = 0, sd = 1)   # To generate "n" number of random numbers fitting to
                               normal distribution with default mean=0 and sd=1
rnorm(25, 30, 5)             # To generate a sample of size 25 from a normal
                               distribution with mean 30 and standard deviation 5
a <- rbinom(100, 10, 0.5)    # 100 people tossing a fair coin (third argument=0.5) 10
                               times
                             # mean(a) should be close to 10*0.5 = 5
c(1,5,3,8)                   # to create a vector of these numbers
```

Generating a Simulated Dataset

```
# Generate a dataset/dataframe for a case-control study:
        cc <-
        data.frame(case.id = 1:100,
        age = rnorm(100, mean = 60, sd = 12),
        caco = gl(2, 50, labels = c("Case", "Control")))

        summary(cc)

    case.id                age                 caco
 Min.   :  1.00   Min.   : 29.40   Case      :50
 1st Qu.: 25.75   1st Qu.: 54.31   Control   :50
 Median : 50.50   Median : 61.24
 Mean   : 50.50   Mean   : 61.29
 3rd Qu.: 75.25   3rd Qu.: 66.22
 Max.   :100.00   Max.   :102.47
```

Kingston University London

Generating a Simulated Dataset

```
# Generate a dataset with multiple random variables:

    set.seed(955)
    vvar <- 1:20 + rnorm(20,sd=3)
    wvar <- 1:20 + rnorm(20,sd=5)
    xvar <- 20:1 + rnorm(20,sd=3)
    yvar <- (1:20)/2 + rnorm(20, sd=10)
    zvar <- rnorm(20, sd=6)
    data <- data.frame(vvar, wvar, xvar, yvar, zvar)
    head(data)

#>        vvar       wvar     xvar       yvar      zvar
#> 1 -4.252354  5.1219288 16.02193 -15.156368 -4.086904
#> 2  1.702318 -1.3234340 15.83817 -24.063902  3.468423
#> 3  4.323054 -2.1570874 19.85517   2.306770 -3.044931
#> 4  1.780628  0.7880138 17.65079   2.564663  1.449081
#> 5 11.537348 -1.3075994 10.93386   9.600835  2.761963
#> 6  6.672130  2.0135190 15.24350  -3.465695  5.749642
```

r$_x$c Contingency Table

How to Create a Contingency Table

```
x  <-  matrix(c(22, 46, 66, 58), nrow = 2)     # To create a 2x2 table
```

<u>OR</u>

```
x <- data.frame()     # Assign a name to the contingency table to be created
fix(x)                # Opens the data editor to enter the cell values (rxc)

X                     # Prints the newly created contingency table

     var1 var2
  1   34   42
  2   22   11
```

To run for example, Fisher's test, use the assigned name of the 2x2 table (x):

```
fisher.test(x)
```

To edit the contingency table (or any spreadsheet), use:

```
edit(x)               # Opens the data editor to edit the cell values
```

Factors for Grouping

Factors are grouping variables (like gender, age group or nationality) and can be used to create subsets (strata) of data for subset-specific analysis
http://uc-r.github.io/factors

```
data(iris)
str(iris)        # Shows "Species" as a factor. If it wasn't a factor, but we need
                   it to be a factor for analysis purposes, we would use the
                   following command:

iris$Species <- as.factor(iris$Species)
str(iris)        # Would now show as a factor (but it aleready is)
```

Let's use another built-in dataframe for an exercise

Kingston University London

Factors for Grouping

```
    data(infert)
    str(infert)

'data.frame':   248 obs. of  8 variables:
 $ education     : Factor w/ 3 levels "0-5yrs","6-11yrs",..: 1 1 1 1 2 2 2 2 2 2 ...
 $ age           : num  26 42 39 34 35 36 23 32 21 28 ...
 $ parity        : num  6 1 6 4 3 4 1 2 1 2 ...
 $ induced       : num  1 1 2 2 1 2 0 0 0 0 ...
 $ case          : num  1 1 1 1 1 1 1 1 1 1 ...
 $ spontaneous   : num  2 0 0 0 1 1 0 0 1 0 ...
 $ stratum       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ pooled.stratum: num  3 1 4 2 32 36 6 22 5 19 ...

    infert$case <- as.factor(infert$case)
    str(infert)

'data.frame':   248 obs. of  8 variables:
 $ education     : Factor w/ 3 levels "0-5yrs","6-11yrs",..: 1 1 1 1 2 2 2 2 2 2 ...
 $ age           : num  26 42 39 34 35 36 23 32 21 28 ...
 $ parity        : num  6 1 6 4 3 4 1 2 1 2 ...
 $ induced       : num  1 1 2 2 1 2 0 0 0 0 ...
 $ case          : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
 $ spontaneous   : num  2 0 0 0 1 1 0 0 1 0 ...
 $ stratum       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ pooled.stratum: num  3 1 4 2 32 36 6 22 5 19 ...

The "case" variable is now a factor. You can also check this with:
    class(infert$case)
[1] "factor"
```

Factors for Grouping

```
x <- table(infert$case, infert$spontaneous)
x

    0   1   2
0 113  40  12
1  28  31  24

summary(table(infert$case, infert$spontaneous))
Number of cases in table: 248
Number of factors: 2
Test for independence of all factors:
        Chisq = 32.86, df = 2, p-value = 7.314e-08
```

The variable "case" is numeric:

```
summary(infert)
   education        age            parity          induced            case        spontaneous        stratum      pooled.stratum
 0-5yrs : 12   Min.   :21.00   Min.   :1.000   Min.   :0.0000   Min.   :1.000   Min.   :0.0000   Min.   : 1.00   Min.   : 1.00
 6-11yrs:120   1st Qu.:28.00   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:21.00   1st Qu.:19.00
 12+ yrs:116   Median :31.00   Median :2.000   Median :0.0000   Median :1.000   Median :0.0000   Median :42.00   Median :36.00
               Mean   :31.50   Mean   :2.093   Mean   :0.5726   Mean   :1.335   Mean   :0.5766   Mean   :41.87   Mean   :33.58
               3rd Qu.:35.25   3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:62.25   3rd Qu.:48.25
               Max.   :44.00   Max.   :6.000   Max.   :2.0000   Max.   :2.000   Max.   :2.0000   Max.   :83.00   Max.   :63.00
```

The variable "case" is factor "infert$case <- as.factor(infert$case)":

```
summary(infert)
   education        age            parity          induced        case     spontaneous        stratum      pooled.stratum
 0-5yrs : 12   Min.   :21.00   Min.   :1.000   Min.   :0.0000   1:165   Min.   :0.0000   Min.   : 1.00   Min.   : 1.00
 6-11yrs:120   1st Qu.:28.00   1st Qu.:1.000   1st Qu.:0.0000   2: 83   1st Qu.:0.0000   1st Qu.:21.00   1st Qu.:19.00
 12+ yrs:116   Median :31.00   Median :2.000   Median :0.0000           Median :0.0000   Median :42.00   Median :36.00
               Mean   :31.50   Mean   :2.093   Mean   :0.5726           Mean   :0.5766   Mean   :41.87   Mean   :33.58
               3rd Qu.:35.25   3rd Qu.:3.000   3rd Qu.:1.0000           3rd Qu.:1.0000   3rd Qu.:62.25   3rd Qu.:48.25
               Max.   :44.00   Max.   :6.000   Max.   :2.0000           Max.   :2.0000   Max.   :83.00   Max.   :63.00
```

Kingston University London

apply(), sapply(), lapply(), and tapply()

Guru99 Home Testing SAP Web Must Learn! Big Data

apply(), sapply(), tapply() in R with Examples

This tutorial aims at introducing the apply() function collection. The apply() function is the most basic of all collection. We will also learn sapply(), lapply() and tapply(). The apply collection can be viewed as a substitute to the loop

The apply() collection is bundled with **r essential** package if you install R with Anaconda. The apply() function can be feed with many functions to perform redundant application on a collection of object (data frame, list, vector, etc.). The purpose of apply() is primarily to avoid explicit uses of loop constructs. They can be used for an input list, matrix or array and apply a function. Any function can be passed into apply().

apply()

RDocumentation

Apply Functions Over Array Margins

Returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Keywords array, iteration

Usage

```
apply(X, MARGIN, FUN, ...)
```

Arguments

X an array, including a matrix.

MARGIN a vector giving the subscripts which the function will be applied over. E.g., for a matrix `1` indicates rows, `2` indicates columns, `c(1, 2)` indicates rows and columns. Where `X` has named dimnames, it can be a character vector selecting dimension names.

FUN the function to be applied: see 'Details'. In the case of functions like `+`, `%*%`, etc., the function name must be backquoted or quoted.

```
data(iris)
apply(iris[-5], 2, mean)        # "2" as the second argument denotes columns

Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
```

Kingston University London

tapply()

RDocumentation   Search for packages, functions, etc   R Enterprise Training   R package   Leaderboard

tapply

From base v3.5.1
by R-core R-core@R-project.org

18th
Percentile

Apply A Function Over A Ragged Array

Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

Keywords iteration, category

Usage

```
tapply(X, INDEX, FUN = NULL, …, default = NA, simplify = TRUE)
```

Arguments

X an R object for which a `split` method exists. Typically vector-like, allowing subsetting with `[` .

INDEX a `list` of one or more `factor` s, each of same length as `X` . The elements are coerced to factors by `as.factor` .

FUN the function to be applied, or `NULL` . In the case of functions like `+` , `%*%` , etc., the function name must be backquoted or quoted. If `FUN` is `NULL` , tapply returns a vector which can be used to subscript the multi-way array `tapply` normally produces.

The `tapply()` function is useful when we need to break up a vector into groups defined by some classifying factor, compute a function on the subsets, and return the results in a convenient form.

```
data(iris)
tapply(iris$Petal.Length, iris$Species, mean)
    setosa versicolor  virginica
     1.462      4.260      5.552
```

Kingston University London

aggregate()

Aggregate Multiple Columns At Once

The formula method of the `aggregate()` function allows running the same function on multiple columns at once (like the `apply()` function). The formula is the first argument of the function: " . " represents all variables other than the grouping factor (like Species in iris). The grouping factor is defined on the right hand side of (~). Second argument is the dataframe name, and the third is the function to be applied.

```
data(iris)
aggregate(.~Species, iris, var)      # var = variance

     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa    0.1242490  0.14368980   0.03015918  0.01110612
2 versicolor    0.2664327  0.09846939   0.22081633  0.03910612
3  virginica    0.4043429  0.10400408   0.30458776  0.07543265
```

Kingston University London

with()

The **with( )** function applys an expression to a dataset. It is similar to DATA= in SAS.

```
# with(data, expression)
# example applying a t-test to a data frame mydata
with(mydata, t.test(y ~ group))
```

```
data(infert)
with(infert, t.test(infert$parity ~ infert$case))

# parity is a count variable and case is the case-
control indicator (grouping factor)
```

```
> data(infert)
> with(infert, t.test(infert$parity ~ infert$case))

        Welch Two Sample t-test

data:  infert$parity by infert$case
t = -0.13841, df = 160.23, p-value = 0.8901
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3600963  0.3129258
sample estimates:
mean in group 0 mean in group 1
       2.084848        2.108434
```

Kingston University London

with

Evaluate An Expression In A Data Environment

by()

by

Apply A Function To A Data Frame Split By Factors

A data frame is split by row into data frames subsetted by the values of one or more factors, and function `FUN` is applied to each subset in turn.

```
data(iris)
by(iris$Sepal.Length, iris$Species, summary)    # summary statistics of
                                  Sepal.Length is requested for each Species

iris$Species: setosa
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   4.800   5.000   5.006   5.200   5.800
------------------------------------------------------------------------------
iris$Species: versicolor
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.900   5.600   5.900   5.936   6.300   7.000
------------------------------------------------------------------------------
iris$Species: virginica
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.900   6.225   6.500   6.588   6.900   7.900
```

R Function of the Day

Kingston University London

Variables: Equivalent Terms

Input
Independent
Explanatory
Exposure
Predictor
Regressor

Output
Dependent
Response
Outcome
Predicted
Regressand

Co-variate
(Potential) Confounder

Effect Size

The strength/magnitude/degree of a correlation/association is assessed by the effect size.
The *P* value is a measure of statistical significance (or the strength of statistical significance). The effect size is the measure (and not the *P* value) to use to compare two results.

RR / OR / HR / ARR / NNT
Correlation coefficient
Cohen's d value *
Regression coefficient

* https://www.rdocumentation.org/packages/DescTools/versions/0.99.19/topics/CohenD

Kingston University London

Statistical Tests

No adjustment	With adjustment
Chi-squared or Exact test	(Logistic) regression
t-test	Multivariable regression
Log-rank test (survival test)	Cox regression
Correlation	Multivariable regression
Linear regression	Multivariable regression
ANOVA - Linear regression	

Statistical Tests for Categorical Data (rxc Table) Analysis

Test	Context
Chi-squared or exact test	Observed counts in groups
McNemar's test	Observed counts in groups, but matched data
Goodness-of-fit test	Observed counts and expected proportions
Mantel-Haenszel test	Multiple rxc tables are analyzed simultaneously (like a meta-analysis)
Trend test (Cochran-Mantel-Haenszel test)	Observed counts in at least three groups
Z-test	An observed proportion and an hypothesized proportion; or two observed proportions

Kingston University London

Inferential Statistics with R

Count Data Analysis

Define your *r* x *c* table as a matrix; assign a name to it (e.g., mymatrix), and enter your matrix name within the brackets:

```
chisq.test(mymatrix)
fisher.test(mymatrix)
mcnemar.test(mymatrix)     # For matched observations
```

Define your array as multiple matrices; assign a name to it (e.g., myarray), and enter your array name within the brackets:

```
cmh.test(myarray)          # The array is multiple contingency tables
```

Define the first column of your table (exposed) in a vector (x) and the totals for each category in a different vector (n) for the trend test:

```
prop.trend.test(x,n)       # Like smokers (x) out of total subjects (n)
```

Define a vector of your observed counts (x), and a vector of expected proportions (p) for the goodness-of-fit test:

```
chisq.test(x, p=p)         # Note that x is counts, and p is proportions
```

Script: `s3.R`

Kingston University London

Inferential Statistics with R

Count Data Analysis

R for Categorical Data Analysis

Steele H. Valenzuela

Using R for Biomedical Statistics

Biomedical statistics

This booklet tells you how to use the R software to carry out some simple analyses that are common in biomedical statistics. In particular, the focus is on cohort and case-control studies that aim to test whether particular factors are associated with disease, randomised trials, and meta-analysis.

This booklet assumes that the reader has some basic knowledge of biomedical statistics, and the principal focus of the booklet is not to explain biomedical statistics analyses, but rather to explain how to carry out these analyses using R.

A Little Book of R For Biomedical Statistics

Release 0.2

Kingston University London

2.4 Testing for an Association Between Disease and Exposure, in a Cohort or Case-Control Study

In a case-control or cohort study, it is interesting to do a statistical test for association between having the disease and being exposed to some treatment or environment (for example, smoking or taking a certain drug).

In R, you can test for an association using the Chi-squared test, or Fisher's exact test. For example, using our data from the example above:

```
> print(mymatrix)
          Disease Control
Exposure1      30      24
Exposure2      76     241
Unexposed      82     509
> chisq.test(mymatrix)
     Pearson's Chi-squared test

 data:  mymatrix
 X-squared = 60.5762, df = 2, p-value = 7.015e-14

> fisher.test(mymatrix)
    Fisher's Exact Test for Count Data

 data:  mymatrix
 p-value = 5.263e-12
 alternative hypothesis: two.sided
```

Here the P-value for the Chi-squared test is about 7e-14, and the P-value for Fisher's exact test is about 5e-12. Both are very tiny (<0.05), indicating a significant association between exposure and disease (using a cutoff of P<0.05 for statistical significance).

2.5 Calculating the (Mantel-Haenszel) Odds Ratio when there is a Stratifying Variable

You may have data from a cohort study or case-control study that is stratified, for example, the data may be separated (stratified) by the sex of the people studied. For example, we may have two different tables giving information on the relationship between exposure (eg. to a certain drug or smoking cigarettes) and having a particular disease. One of the tables may given information for women, and the other give information for men.

Requires the R package "`lawstat`"

```
> myarray <- array(c(mymatrix1,mymatrix2),dim=c(2,2,2))
> cmh.test(myarray)
    Cochran-Mantel-Haenszel Chi-square Test

  data:  myarray
  CMH statistic = 40.512, df = 1.000, p-value = 0.000,
  MH Estimate = 23.001,
  Pooled Odd Ratio = 25.550,
  Odd Ratio of level 1 = 16.480,
  Odd Ratio of level 2 = 28.667
```

dim = c(2,2,2) : two 2x2 tables

This tells you that the odds ratio for the first stratum (women) is 16.480, the odds ratio for the second stratum (men) is 28.667, and the aggregate odds ratio that we would get if we pooled the data for men and women is 25.550. The Mantel-Haenszel odds ratio is estimated to be 23.001.

Inferential Statistics with R

Count Data Analysis

Dose Response (Trend) Test

In a dose-response analysis, it is usual to have information on the incidence of a disease in people who were exposed to different doses of some factor (for example, number of cigarettes smoked per day, dose of a certain drug taken, etc.). For example, your data may look like this:

	Disease	Control
Dose=2	35	82
Dose=9.5	250	293
Dose=19.5	196	190
Dose=37	136	71
Dose=50	32	13

We can enter our data into R as follows (note that you need to type "nrow=5" to tell R that there are 5 rows of data):

```
> mymatrix <- matrix(c(35,82,250,293,196,190,136,71,32,13),nrow=5,byrow=TRUE)
> colnames(mymatrix) <- c("Disease","Control")
> rownames(mymatrix) <- c("2","9.5","19.5","37","50")
> print(mymatrix)
      Disease Control
 2         35      82
 9.5      250     293
 19.5     196     190
 37       136      71
 50        32      13
```

Kingston University London

More suited for epidemiological analysis

A Little Book of R For Biomedical Statistics

Release 0.2

Inferential Statistics with R

Count Data Analysis

Dose Response (Trend) Test

prop.trend.test

From stats v3.5.3
by R-core R-core@R-project.org

99.99th
Percentile

Test For Trend In Proportions

Performs chi-squared test for trend in proportions, i.e., a test asymptotically optimal for local alternatives where the log odds vary in proportion with `score`. By default, `score` is chosen as the group numbers.

```
> prop.trend.test(x = c(47, 31, 9, 5), n = c(49, 40, 18, 10))

        Chi-squared Test for Trend in Proportions

data:  c(47, 31, 9, 5) out of c(49, 40, 18, 10) ,
 using scores: 1 2 3 4
X-squared = 21.031, df = 1, p-value = 4.519e-06
```

Inferential Statistics with R

Count Data Analysis

Dose Response (Trend) Test

```
. ptrendi 47 2 1 \ 31 9 2 \ 9 9 3 \ 5 5 4

          r    nr    _prop      x
   1.    47     2    0.959   1.00
   2.    31     9    0.775   2.00
   3.     9     9    0.500   3.00
   4.     5     5    0.500   4.00

Trend analysis for proportions
------------------------------

Regression of p = r/(r+nr) on x:
Slope = -.18261, std. error =  .03982, Z =   4.586

Overall chi2(3) =          22.407,  pr>chi2 = 0.0001
Chi2(1) for trend =        21.031,  pr>chi2 = 0.0000
Chi2(2) for departure =     1.376,  pr>chi2 = 0.5026
```

Stata output

R output

```
> prop.trend.test(x = c(47, 31, 9, 5), n = c(49, 40, 18, 10))

        Chi-squared Test for Trend in Proportions

data:  c(47, 31, 9, 5) out of c(49, 40, 18, 10) ,
 using scores: 1 2 3 4
X-squared = 21.031, df = 1, p-value = 4.519e-06
```

Kingston University London

R for Categorical Data Analysis

Steele H. Valenzuela

Inferential Statistics with R

Count Data Analysis

Dose Response (Trend) Test

EXERCISE:
Define x and n for the following command to work correctly:

```
prop.trend.test(x, n)
```

Kingston University London

Inferential Statistics with R

Count Data Analysis

Dose Response (Trend) Test

EXERCISE:
Define x and n for the following command to work correctly:

```
prop.trend.test(x, n)
```

```
x = c(20, 30, 40)
n = c(60, 60, 60)
prop.trend.test(x,n)

    Chi-squared Test for Trend in Proportions

data:  x out of n ,
       using scores: 1 2 3
X-squared = 13.333, df = 1, p-value = 0.0002607
```

```
. ptrendi 20 40 1 \ 30 30 2 \ 40 20 3

          r    nr    _prop      x
  1.     20    40    0.333   1.00
  2.     30    30    0.500   2.00
  3.     40    20    0.667   3.00

Trend analysis for proportions
------------------------------

Regression of p = r/(r+nr) on x:
Slope =  .16667, std. error =  .04564, Z =   3.651

Overall chi2(2) =          13.333,  pr>chi2 = 0.0013
chi2(1) for trend =        13.333,  pr>chi2 = 0.0003
chi2(1) for departure =     0.000,  pr>chi2 = 0.9976
```

Inferential Statistics with R

Count Data Analysis

Dose Response (Trend) Test

Generate a matrix:

```
matrix1 <- matrix(c(35, 82, 250, 293, 196, 190, 136, 71, 32,
13), nrow=5, byrow=TRUE)
colnames(matrix1) <- c("cases", "controls")
rownames(matrix1) <- c("0", "1", "2", "3", "4+")
matrix1        # to inspect the matrix with row and column names included

   cases controls
0     35       82
1    250      293
2    196      190
3    136       71
4+    32       13
```

 # Run DescTools:MHChisqTest()

```
MHChisqTest(matrix1)

        Mantel-Haenszel Chi-Square
data:  matrix1
X-squared = 47.158, df = 1, p-value = 6.55e-12
```

Inferential Statistics with R

Count Data Analysis

Goodness-of-fit Test

```
#Conducting a one-way chi-square (replace stars with appropriate observed frequencies e.g
26,31,26,27 obtained from step 1 and expected ratio as proportions e.g.,1/4,1/4,1/4,1/4)
chisq.test(c(**,**,**,**),p=c((*/*),(*/*),(*/*),(*/*)))
```

3. Identifying the key elements of the output

Following the instructions above will produce the following output in the **R Console** window: the key elements are annotated in blue.

```
> #Importing data from tab delimited file (replace stars with an appropriate object name e.g.,peas)
> peas<-read.table(file.choose(),header=TRUE)
> attach(peas)
> names(peas)
[1] "form"     "colour"   "category"
>
> #Calculating observed frequencies (replace stars with approriate text eg., category, category)
> tapply(category,category,length)
   Round.Green   Round.Yellow  Wrinkled.Green Wrinkled.Yellow
           26            31              26              27
>
> #Conducting a one-way chi-square (replace stars with appropriate observed frequencies and expected
ratio as proportions e.g 26,31,26,27 and 1/4,1/4,1/4,1/4)
> chisq.test(c(26,31,26,27),p=c((1/4),(1/4),(1/4),(1/4)))

        Chi-squared test for given probabilities

data:  c(26, 31, 26, 27)
X-squared = 0.6182, df = 3, p-value = 0.8923
```

NOTE: Use these frequencies generated by step 1 in the code in step 2

Statistic **Degrees of Freedom** **P Value**

For paired or matched count data, ordinary Chi-squared test is not appropriate, and McNemar's test should be used.

Define the matrix for your 2x2 table, and run the test as shown.

```
> mcnemar.test(mymatrix)
    McNemar's Chi-squared test with continuity correction

data:  mymatrix
McNemar's chi-squared = 26.4143, df = 1, p-value = 2.755e-07
```

Inferential Statistics with R

Count Data Analysis

Z-test

Not much different from Chi-squared test; it is used to compare a proportion with an hypothesized proportion (one-sample test), or to test the equality of two proportions (two-sample test).

prop.test

Exact And Approximate Tests For Proportions

```
prop.test(x, n, p = NULL, alternative = c("two.sided", "less",
  "greater"), conf.level = 0.95, data = NULL, success = NULL, ...)
```

```
    prop.test(76, 100, 0.50)    # One-sample Z-test

1-sample proportions test with continuity correction
data:  76 out of 100, null probability 0.5
X-squared = 26.01, df = 1, p-value = 3.397e-07
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
   0.6623089 0.8373345
sample estimates:
   p
0.76
```

Inferential Statistics with R

Count Data Analysis

Z-test

```
prop.test(c(76, 50), c(100, 100))   # If continuity correction
          is not needed, correct = FALSE argument can be added

    2-sample test for equality of proportions with continuity
    correction

data:  c(76, 50) out of c(100, 100)
X-squared = 13.406, df = 1, p-value = 0.0002508
alternative hypothesis: two.sided
95 percent confidence interval:
 0.1211184 0.3988816
sample estimates:
prop 1 prop 2
  0.76   0.50
1-sample proportions test with continuity correction
data:  76 out of 100, null probability 0.5
X-squared = 26.01, df = 1, p-value = 3.397e-07
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
   0.6623089 0.8373345
sample estimates:
   p
0.76
```

Kingston University London

Inferential Statistics with R

Count Data Analysis

Now, run Script: `s3.R`

Kingston University London

Inferential Statistics with R

2x2 Table Analysis

Information that can be obtained from a 2x2 table

Association statistics
(Chi-squared and Fisher's exact test)

Effect size
(Odds ratio, relative risk, absolute risk reduction, number needed to treat)

Measures of diagnostic accuracy
(Sensitivity, specificity, positive/negative predictive tests, ROC/AUC)

Association measures
(Phi coefficient, Cramer's V, Yule's Q and others)

Now, run Script: `contingency.R`

Kingston University London

Next

R for Inferential Statistics

II. t-test, ANOVA, regression

Kingston University London