R Examples from BIOST 511 - Set 7 and 8

Taught by Lurdes Inoue, Ph.D.

Brian D. Williamson

University of Washington
Department of Biostatistics

November 14, 2014

Disclaimer

All of the following slides and functions are to be used when given sample descriptive statistics

For raw data, refer to the documentation for `ttest()` and other functions given on http://www.emersonstatistics.com/R

Test of Proportions

Use the `prop.test()` function from the `stats` package (automatically loaded into R along with the base package)

```
prop.test(110, 200, p=.3, alternative="two.sided")

1-sample proportions test with continuity correction

data:  110 out of 200, null probability 0.3
X-squared = 58.3393, df = 1, p-value = 2.206e-14
alternative hypothesis: true p is not equal to 0.3
95 percent confidence interval:
 0.4782703 0.6197775
sample estimates:
   p
0.55
```

– If we want a different test, we change the `alternative` argument

t-test using Sample Descriptive Statistics

Use the `ttesti()` function from the `uwIntroStats` package

```
ttesti(24, 175, 35, 43, null.hyp=230)

 One-sample t-test
  Obs Mean Std. Error SD 95 %CI
x 24  175  7.1443     35 160.221 189.779

 t-statistic = -7.698396 , df = 23

 Ho: mean =  230

 Ha: mean != 230 , Pr(|T| > |t|) = 8.24e-08
```

– Again, if we want a different test, we change the `alternative` argument

Binomial Test

Use the `binom.test()` function from the `stats` package

```
binom.test(110, 200, p=.3, alternative="two.sided")

Exact binomial test

data:  110 and 200
number of successes = 110, number of trials = 200, p-value = 3.125e-13
alternative hypothesis: true probability of success is not equal to 0.3
95 percent confidence interval:
 0.4782493 0.6202464
sample estimates:
probability of success
                  0.55
```

– Again, if we want a different test, we change the `alternative` argument

Group 1 and Group 2 One-Sample Tests

```
## One sample t-test for Group 1
ttesti(obs=10, mean=-1.6, sd=1.5, null.hyp=0)

 One-sample t-test
  Obs Mean Std. Error SD  95 %CI
x 10  -1.6 0.4743     1.5 -2.673 -0.527

 t-statistic = -3.373096 , df = 9

 Ho: mean =  0

 Ha: mean != 0 , Pr(|T| > |t|) = 0.0082165
## For Group 2
ttesti(obs=30, mean=-.7, sd=2.1, null.hyp=0)

 One-sample t-test
  Obs Mean Std. Error SD  95 %CI
x 30  -0.7 0.3834     2.1 -1.484 0.084

 t-statistic = -1.825742 , df = 29

 Ho: mean =  0

 Ha: mean != 0 , Pr(|T| > |t|) = 0.078203
```

Difference of Means with Sample Descriptive Statistics

Unequal Variances

Use the `ttesti()` function in the `uwIntroStats` package
For the Group 1 Group 2 example,

```
## Two sample for the difference of means, with variance unequal
ttesti(10, -1.6, 1.5, 30, -.7, 2.1)

 Two-sample t-test with unequal variances
     Obs Mean Std. Error SD  95 %CI
x    10  -1.6 0.4743     1.5 -2.673 -0.527
y    30  -0.7 0.3834     2.1 -1.484 0.084
diff 40  -0.9 0.6099     NA  -2.135 0.335

 t-statistic = -1.475608 , Satterthwaite's df = 21.72386

 Ho: mean(x) - mean(y) = diff =  0

 Ha: diff != 0 , Pr(|T| > |t|) = 0.1544
```

Proportions

Use the binom.test() function in the stats package

```
## Exact binomial test
binom.test(7, 26, .5)

Exact binomial test

data:  7 and 26
number of successes = 7, number of trials = 26, p-value = 0.02896
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.1157322 0.4778748
sample estimates:
probability of success
             0.2692308
```

Chi-squared Test

Use the `chisq.test()` function in the `stats` package

If given raw data, use the `tabulate()` function in the `uwIntroStats` package

The `correct=FALSE` argument tells R to NOT use Yates' continuity correction

For this small data example, you will get a warning from R, because the expected cell counts are too small for a reliable Chi-squared test

```
chisq.test(matrix(c(2,23,5,30), nrow=2, byrow=T), correct=FALSE)

Pearson's Chi-squared test

data:  matrix(c(2, 23, 5, 30), nrow = 2, byrow = T)
X-squared = 0.5591, df = 1, p-value = 0.4546
```

Fisher's Exact Test

Use the `fisher.test()` function in the `stats` package
If given raw data, use the `tabulate()` function in the `uwIntroStats` package with the correct arguments

```
fisher.test(matrix(c(2,23,5,30), nrow=2, byrow=T))

Fisher's Exact Test for Count Data

data:  matrix(c(2, 23, 5, 30), nrow = 2, byrow = T)
p-value = 0.6882
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.04625243 3.58478157
sample estimates:
odds ratio
  0.527113
```

Chi-squared Goodness of Fit Test

Use the `chisq.test()` function in the `stats` package

Simply enter in a vector rather than a matrix and a goodness of fit test will be performed

Homogeneity Test

Use the `chisq.test()` function in the `stats` package

If given raw data, use the `tabulate()` function in the `uwIntroStats` package with the correct arguments

```
## Chi-squared example
chisq.test(matrix(c(7, 55, 489, 475,
                    293, 38, 61, 129, 570, 431, 154, 12),
                  nrow=2, byrow=T))

Pearson's Chi-squared test

data:  matrix(c(7, 55, 489, 475, 293, 38, 61, 129, 570, 431, 154, 12), nrow = 2, byrow = T)
X-squared = 137.7193, df = 5, p-value < 2.2e-16
```

Independence Test

```
chisq.test(matrix(c(52, 79, 342, 226, 62, 153, 417, 262, 53, 213, 629, 375,
                    54, 231, 571, 244, 18, 46, 139, 74, 25, 139, 330, 116),
                  nrow=4, byrow=T))
Pearson's Chi-squared test

data:  matrix(c(52, 79, 342, 226, 62, 153, 417, 262, 53,
213, 629, 375, 54, 231, 571, 244, 18, 46, 139, 74, 25, 139, 330, 116), nrow = 4, byrow = T)
X-squared = 2041.811, df = 15, p-value < 2.2e-16
```

Sign Test

No direct analog for this test in R
Since it ignores a lot of information, use the binomial test or McNemar's test

Wilcoxon Signed Rank Test

Use the `wilcoxon` function from the `uwIntroStats` package
First you must have data vectors, so create them if necessary

```
cf <- c(1153, 1132, 1165, 1460, 1162, 1493, 1358, 1453, 1185, 1824, 1793, 1930, 2075)
healthy <- c(996, 1080, 1182, 1452, 1634, 1619, 1140, 1123, 1113, 1463, 1632, 1614, 1836)
wilcoxon(cf, healthy, paired=TRUE)

 Wilcoxon signed rank test
        obs sum ranks expected
positive  10        71     45.5
negative   3        20     45.5
zero       0         0      0.0
all       13        91     91.0

unadjusted variance   204.75
adjustment for ties     0.00
adjustment for zeroes   0.00
adjusted variance     204.75
                    H0 Ha
Hypothesized Median 0  two.sided
  Test Statistic p-value
V 71             0.080322
Z 1.7821         0.037368
```

Wilcoxon Rank Sum Test

Use the `wilcoxon` function from the `uwIntroStats` package

First you must have data vectors, so create them if necessary

```
ebv <- c(2.9, 12.1, 2.6, 2.5, 2.8, 15.8, 3.2, 1.8, 7.8,
         2.9, 3.2, 8.0, 1.5, 6.3, 1.2, 3.5, 4.5, 1.3,
         1.0, 1.0, 1.3, 1.9, 1.3, 2.1, 2.1, 1.0)
sero <- c(rep(1,16), rep(0,10))
data <- cbind(ebv, sero)
wilcoxon(data[sero==1], data[sero==0])
```

```
 Wilcoxon rank sum test
         obs rank sum expected
Y         20      369      689
X         32     1009      689
combined  52     1378     1378

unadjusted variance 2826.666667
adjustment for ties    2.968326
adjusted variance   2823.698341
                    H0 Ha
Hypothesized Median 0  two.sided
  Test Statistic p-value
W 481           0.0018285
Z 3.1168        0.00091424
Warning message:
In wilcoxon.do(x = y, y = x, alternative = alternative, mu = mu,  :
  cannot compute exact p-value with ties
```

The warning message means that there are tied values, so the p-values are approximate

Systolic Blood Pressure Example

Create the data set by saving it as a text file, then load it into R and attach it

```
systolic <- read.table("P:/TA/systolic.txt", header=TRUE, quote="\"", stringsAsFactors=FALSE)
```

Next regress `systolic` (the y-variable) vs age using the `regress()` function, with first argument `"mean"`

```
regress("mean", systolic$systolic, model=uModel(systolic$age))

Call:
regress(fnctl = "mean", y = systolic$systolic, model = uModel(systolic$age))

Residuals:
    Min      1Q  Median      3Q     Max
-7.1395 -2.3314 -0.2163  2.1872  5.8605

Coefficients:
               Estimate  Naive SE  Robust SE    95%L      95%H          F stat    df Pr(>F)
Intercept       67.68     3.191     2.440        62.45     72.91          769.25 1  < 0.00005
systolic$age    6.153     0.9283    0.6511       4.757     7.550           89.32 1  < 0.00005

Residual standard error: 3.403 on 14 degrees of freedom
Multiple R-squared:  0.7584, Adjusted R-squared:  0.7411
F-statistic: 43.94 on 1 and 14 DF,  p-value: 1.135e-05
```

The `uModel()` function gives the correct names and allows multiple variables to be listed in the `model` argument

Prediction of the mean

Again use the `regress()` function
We will need items from the list it returns, so get the predicted $\hat{y}$'s with `$linearPredictor` and the standard deviation with `$sigma`:

```
yhat <- sysRegress$linearPredictor
## get standard error about line
se <- sysRegress$sigma/sqrt(14)
## calculate confidence interval around the line using the quantile of the t distribution
lower <- yhat - qt(.975, 14)*se
upper <- yhat + qt(.975, 14)*se
cbind(round(lower, 3), round(yhat, 3), round(upper,3))
        [,1]   [,2]    [,3]
 [1,] 84.189 86.140  88.090
 [2,] 90.342 92.293  94.244
 [3,] 84.189 86.140  88.090
 [4,] 78.035 79.986  81.937
 [5,] 90.342 92.293  94.244
 [6,] 96.496 98.447 100.397
 [7,] 78.035 79.986  81.937
 [8,] 84.189 86.140  88.090
 [9,] 96.496 98.447 100.397
[10,] 90.342 92.293  94.244
[11,] 78.035 79.986  81.937
[12,] 84.189 86.140  88.090
[13,] 84.189 86.140  88.090
[14,] 90.342 92.293  94.244
[15,] 84.189 86.140  88.090
[16,] 84.189 86.140  88.090
```

Prediction of a new value

Again use the `regress()` function

```
yhat <- sysRegress$linearPredictor
## get standard error about line
se <- sysRegress$sigma*sqrt(1+1/16+(systolic$age-mean(systolic$age))^2/(sd(systolic$age)^2))
## calculate confidence interval around the line using the quantile of the t distribution
lower <- yhat - qt(.975, 14)*se
upper <- yhat + qt(.975, 14)*se
cbind(round(lower, 3), round(yhat, 3), round(upper,3))
        [,1]   [,2]    [,3]
 [1,] 78.240 86.140  94.039
 [2,] 83.090 92.293 101.497
 [3,] 78.240 86.140  94.039
 [4,] 67.375 79.986  92.597
 [5,] 83.090 92.293 101.497
 [6,] 83.416 98.447 113.478
 [7,] 67.375 79.986  92.597
 [8,] 78.240 86.140  94.039
 [9,] 83.416 98.447 113.478
[10,] 83.090 92.293 101.497
[11,] 67.375 79.986  92.597
[12,] 78.240 86.140  94.039
[13,] 78.240 86.140  94.039
[14,] 83.090 92.293 101.497
[15,] 78.240 86.140  94.039
[16,] 78.240 86.140  94.039
```