Module 11

Statistics

Andrew Jaffe
Instructor

Statistics

Now we are going to cover how to perform a variety of basic statistical tests in R.

- Correlation
- T-tests
- Proportion tests
- Chi-squared
- Fisher's Exact Test
- Linear Regression

Note: We will be glossing over the statistical theory and "formulas" for these tests. There are plenty of resources online for learning more about these tests, as well as dedicated Biostatistics series at the School of Public Health

Correlation

cor() performs correlation in R

```
cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))
```

```
> load("charmcirc.rda")
> cor(dat2$orangeAverage, dat2$purpleAverage)
```

```
[1] NA
```

```
> cor(dat2$orangeAverage, dat2$purpleAverage, use = "complete.obs")
```

```
[1] 0.9208
```

Correlation

You can also get the correlation between matrix columns

```
> signif(cor(dat2[, grep("Average", names(dat2))], use = "complete.obs"), 3)
```

```
              orangeAverage purpleAverage greenAverage bannerAverage
orangeAverage         1.000         0.889        0.837         0.441
purpleAverage         0.889         1.000        0.843         0.441
greenAverage          0.837         0.843        1.000         0.411
bannerAverage         0.441         0.441        0.411         1.000
```

Or between columns of two matrices, column by column.

```
> signif(cor(dat2[, 3:4], dat2[, 5:6], use = "complete.obs"), 3)
```

```
              greenAverage bannerAverage
orangeAverage        0.837         0.441
purpleAverage        0.843         0.441
```

Correlation

You can also use cor.test() to test for whether correlation is significant (ie non-zero). Note that linear regression is probably your better bet.

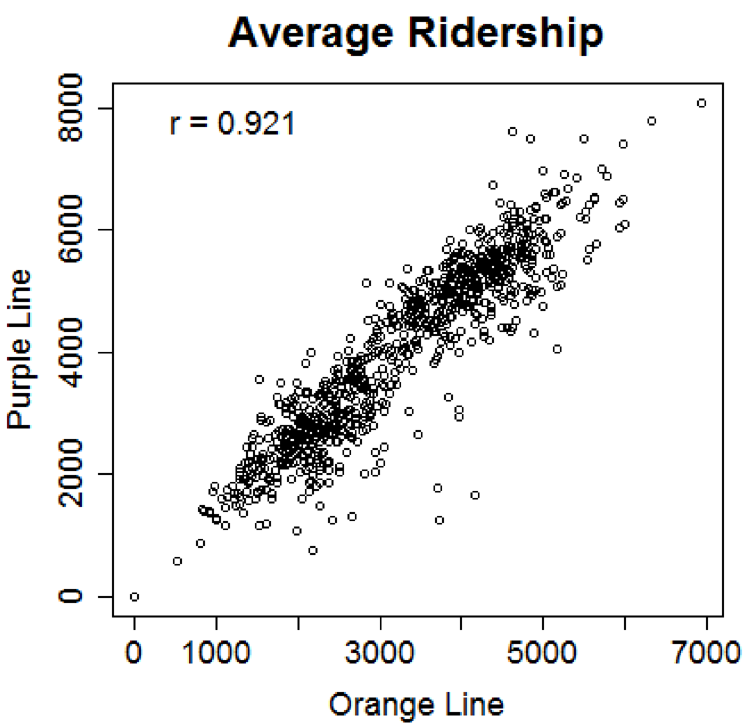
```
> ct = cor.test(dat2$orangeAverage, dat2$purpleAverage, use = "complete.obs")
> ct
```

```

    Pearson's product-moment correlation

data:  dat2$orangeAverage and dat2$purpleAverage
t = 69.65, df = 871, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9100 0.9303
sample estimates:
   cor
0.9208
```

Correlation

Note that you can add the correlation to a plot, via the legend() functinon.

```
> plot(dat2$orangeAverage, dat2$purpleAverage, xlab = "Orange Line", ylab = "Purple Line",
+     main = "Average Ridership", cex.axis = 1.5, cex.lab = 1.5, cex.main = 2)
> legend("topleft", paste("r =", signif(ct$estimate, 3)), bty = "n", cex = 1.5)
```



Correlation

For many of these testing result objects, you can extract specific slots/results as numbers, as the 'ct' object is just a list.

```
> # str(ct)
> names(ct)
```

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

```
> ct$statistic
```

```
    t
69.65
```

```
> ct$p.value
```

```
[1] 0
```

T-tests

The T-test is performed using the t.test() function, which essentially tests for the difference in means of a variable between two groups.

```
> tt = t.test(dat2$orangeAverage, dat2$purpleAverage)
> tt
```

```

    Welch Two Sample t-test

data:  dat2$orangeAverage and dat2$purpleAverage
t = -16.22, df = 1745, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1141.5  -895.2
sample estimates:
mean of x mean of y
     2994      4013
```

```
> names(tt)
```

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

T-tests

You can also use the 'formula' notation.

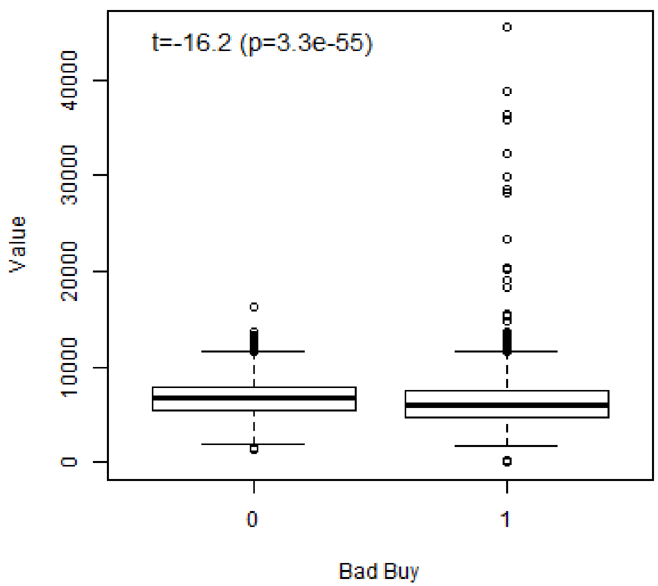
```
> cars = read.csv("http://biostat.jhsph.edu/~ajaffe/files/kaggleCarAuction.csv",
+     as.is = T)
> tt2 = t.test(VehBCost ~ IsBadBuy, data = cars)
> tt2$estimate
```

```
mean in group 0 mean in group 1
           6797            6259
```

T-tests

You can add the t-statistic and p-value to a boxplot.

```
> boxplot(VehBCost ~ IsBadBuy, data = cars, xlab = "Bad Buy", ylab = "Value")
> leg = paste("t=", signif(tt$statistic, 3), " (p=", signif(tt$p.value, 3), ")",
+     sep = "")
> legend("topleft", leg, cex = 1.2, bty = "n")
```



Proportion tests

prop.test() can be used for testing the null that the proportions (probabilities of success) in several groups are the same, or that they equal certain given values.

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

```
> prop.test(x = 15, n = 32)
```

```

    1-sample proportions test with continuity correction

data:  15 out of 32, null probability 0.5
X-squared = 0.0312, df = 1, p-value = 0.8597
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.2951 0.6497
sample estimates:
     p
0.4688
```

Chi-squared tests

chisq.test() performs chi-squared contingency table tests and goodness-of-fit tests.

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

```
> tab = table(cars$IsBadBuy, cars$IsOnlineSale)
> tab
```

```

        0     1
  0 62375  1632
  1  8763   213
```

Chi-squared tests

```
> cq = chisq.test(tab)
> cq
```

```

	Pearson's Chi-squared test with Yates' continuity correction

data:  tab
X-squared = 0.9274, df = 1, p-value = 0.3356
```

```
> names(cq)
```

```
[1] "statistic" "parameter" "p.value"   "method"    "data.name" "observed"
[7] "expected"  "residuals" "stdres"
```

```
> cq$p.value
```

```
[1] 0.3356
```

Chi-squared tests

Note that does the same test as prop.test, for a 2x2 table.

```
> chisq.test(tab)
```

```

    Pearson's Chi-squared test with Yates' continuity correction

data:  tab
X-squared = 0.9274, df = 1, p-value = 0.3356
```

```
> prop.test(tab)
```

```

    2-sample test for equality of proportions with continuity
    correction

data:  tab
X-squared = 0.9274, df = 1, p-value = 0.3356
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.005208  0.001674
sample estimates:
prop 1 prop 2
0.9745 0.9763
```

Linear Regression

Now we will briefly cover linear regression. I will use a little notation here so some of the commands are easier to put in the proper context.

y_i = \alpha + \beta * x_i + \epsilon_i

where:

- y_i is the outcome for person i
- \alpha is the intercept
- \beta is the slope
- x_i is the predictor for person i
- \epsilon_i is the residual variation for person i

Linear Regression

The 'R' version of the regression model is:

y ~ x

where:

- y is your outcome
- x is/are your predictor(s)

Linear Regression

```
> fit = lm(VehOdo ~ VehicleAge, data = cars)
> fit
```

```

Call:
lm(formula = VehOdo ~ VehicleAge, data = cars)

Coefficients:
 (Intercept)   VehicleAge
       60127         2723
```

'(Intercept)' is \alpha

'VehicleAge' is \beta

Linear Regression

```
> summary(fit)
```

```
Call:
lm(formula = VehOdo ~ VehicleAge, data = cars)

Residuals:
   Min     1Q Median     3Q    Max 
-71097  -9500   1383  10323  41037 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  60127.2      134.8   446.0   <2e-16 ***
VehicleAge    2722.9       29.9    91.2   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13800 on 72981 degrees of freedom
Multiple R-squared:  0.102, Adjusted R-squared:  0.102 
F-statistic: 8.31e+03 on 1 and 72981 DF,  p-value: <2e-16
```

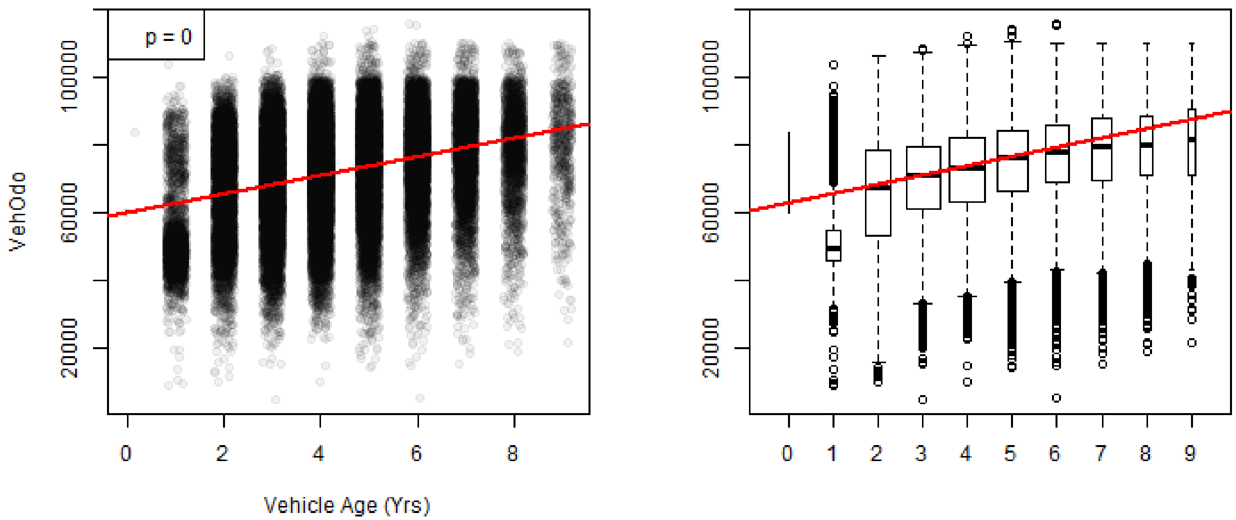
Linear Regression

```
> summary(fit)$coef
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    60127     134.80  446.04        0
VehicleAge      2723      29.86   91.18        0
```

Linear Regression

```
> library(scales)
> par(mfrow = c(1, 2))
> plot(VehOdo ~ jitter(VehicleAge, amount = 0.2), data = cars, pch = 19, col = alpha("black",
+     0.05), xlab = "Vehicle Age (Yrs)")
> abline(fit, col = "red", lwd = 2)
> legend("topleft", paste("p =", summary(fit)$coef[2, 4]))
> boxplot(VehOdo ~ VehicleAge, data = cars, varwidth = TRUE)
> abline(fit, col = "red", lwd = 2)
```



Linear Regression

Note that you can have more than 1 predictor in regression models.

The interpretation for each slope is change in the predictor corresponding to a one-unit change in the outcome, holding all other predictors constant.

```
> fit2 = lm(VehOdo ~ VehicleAge + WarrantyCost, data = cars)
> summary(fit2)
```

```

Call:
lm(formula = VehOdo ~ VehicleAge + WarrantyCost, data = cars)

Residuals:
   Min     1Q Median     3Q    Max 
-67895  -8673    940   9305  45765 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.24e+04   1.46e+02   359.1   <2e-16 ***
VehicleAge   1.94e+03   2.89e+01    67.4   <2e-16 ***
WarrantyCost 8.58e+00   8.25e-02   104.0   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12900 on 72980 degrees of freedom
Multiple R-squared:  0.218, Adjusted R-squared:  0.218 
F-statistic: 1.02e+04 on 2 and 72980 DF,  p-value: <2e-16
```

Linear Regression

Factors get special treatment in regression models - lowest level of the factor is the comparison group, and all other factors are relative to its values.

```
> fit3 = lm(VehOdo ~ factor(TopThreeAmericanName), data = cars)
> summary(fit3)
```

```

Call:
lm(formula = VehOdo ~ factor(TopThreeAmericanName), data = cars)

Residuals:
   Min     1Q Median     3Q    Max 
-71947  -9634   1532  10472  45936 

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)                          68249         93  733.98  < 2e-16 ***
factor(TopThreeAmericanName)FORD      8524        158   53.83  < 2e-16 ***
factor(TopThreeAmericanName)GM        4952        129   38.39  < 2e-16 ***
factor(TopThreeAmericanName)NULL     -2005       6362   -0.32  0.75267    
factor(TopThreeAmericanName)OTHER      585        160    3.66  0.00026 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14200 on 72978 degrees of freedom
Multiple R-squared:  0.0482,	Adjusted R-squared:  0.0482 
F-statistic:  924 on 4 and 72978 DF,  p-value: <2e-16
```

Probability Distributions

These are included in base R

- Normal
- Binomial
- Beta
- Exponential
- Gamma
- Hypergeometric
- etc

Probability Distributions

Each has 4 options:

- `r` for random number generation [e.g. `rnorm()`]
- `d` for density [e.g. `dnorm()`]
- `p` for probability [e.g. `pnorm()`]
- `q` for quantile [e.g. `qnorm()`]

```
> rnorm(5)
```

```
[1] -1.0539  2.2844 -0.5777  1.6222  1.0054
```

Sampling

The `sample()` function is pretty useful for permutations

```
> sample(1:10, 5, replace = FALSE)
```

```
[1]  6  7  3  5 10
```