HCI Statistics in R

Getting Started

STEP 1. Get R and install it - http://r-project.org/

STEP 2. Install some packages

`Packages & Data` - `Package Installer`

- psych
- pwr
- car

Some Basics

Faking data

rnorm generates a set of data that follows the normal distribution with the given mean and standard deviation

```
rnorm(n, mean = 0, sd = 1)
```

For the sake of our R experiments, imagine that these are number of seconds to complete a task. I'm centering everything on 0 to make some concepts clear, but obviously, it can never take -2 seconds to complete a task.

Descriptive statistics

Create some fake data

```
mydata = rnorm(100, 0, 1)
```

Get the min, max, median, mean, 1st quart and 3rd quartiles

```
summary(a)
```

Get some additional values (range, skew, kurtosis, standard error, standard deviation)

```
library(psych) # this "loads" the psych library of functions
describe(mydata)

# Note the se = standard error of the mean (standard estimate of the mean) --
you can check this yourself with the formula: [sd value]/ sqrt(n)
```

Basic Charts

R comes "preloaded" with some datasets

```
women
summary(women)
describe(women)
women$height # gets you just that piece of data
attach(women) # makes the "women" database first class
height # now this prints the "height" piece of the women database
```

Let's do some basic plots of the women data set.

```
plot(weight, height)
abline(lm(height~weight))
title("Regression of Height on Weight")
```

A histogram (frequency plot) lets us see what the distribution looks like.

```
mydata = rnorm(1000, 0, 1)
hist(mydata)
hist(mydata, breaks=10)
```

A boxplot is a useful diagram, too. A boxplot shows the median, and interquartile values of the dataset, and the whiskers represent the min/max values. Actually, in practice, the whiskers can mean a lot of different things.

```
boxplot(mydata)
```

For more interesting datasets, we can look at this by "chunks of data" at a time.

```
mtcars
# Boxplot of MPG by Car Cylinders
boxplot(mpg~cyl,data=mtcars, main="Car Milage Data", xlab="Number of Cylinders",
ylab="Miles Per Gallon")
```

t-test and How alpha Works

Set up for basic t-test - here, each variable (a and b) contain a list of 10 numbers -- each corresponding to a different sample of 10.

```
a = rnorm(10, 0, 1)
b = rnorm(10, 0, 1)

# Unpaired t-test
t.test(a, b)
```

For the above example, try varying the # of samples, and run it multiple times. Note that here, the null hypothesis (that a == b) is actually true. Nevertheless, it is possible for the p-value to be less than 0.05 or 0.1 (which might be what you set your alpha to). The above should show you that sometimes, the t-test (and inferential statistics) will essentially "get it wrong" sometimes. In these cases, where the p-value is smaller than alpha, we would reject the null hypothesis. This is a Type I error!

Now, continue trying the above, except up the sample size a bit.

```
a = rnorm(1000, 0, 1)
b = rnorm(1000, 0, 1)
t.test(a, b)
```

Now let's explore how the alpha works. Imagine that your alpha is 0.05. Note that here, the null hypothesis is false (a is != b). Over several runs of these three lines, you'll see that sometimes it gets it right, sometimes it gets it wrong (i.e. based on your alpha). The alpha is your guard against Type I errors.

```
a = rnorm(10, 0, 1)
b = rnorm(10, 0.5, 1)
t.test(a, b)
```

For small sample sizes, we don't have a lot of ability to detect differences (particularly if

differences are quite small). Upping the sample sizes makes it "easier" to detect these differences.

```
a = rnorm(10000, 0, 1)
b = rnorm(10000, 0.5, 1)
t.test(a, b)
```

Loading Data

Let's create some data. Imagine this is a test of mouse vs. stylus pointing. The "perf" number is the amount of time it takes to complete the task. Each row in this data represents a different trial by a different person.

```
interface = c("mouse", "mouse", "mouse", "mouse", "mouse", "stylus", "stylus",
"stylus", "stylus", "stylus")
# for simplicity, let's just make the perf all come from the same distribution
(regardless of the condition)
perf = rnorm(10, 10, 1)
# now let's stick it into a table
data = data.frame(interface, perf)
data
write.table(data, "interfacedata.txt", sep='\t')
```

To check (and set) which directory you're saving and loading your data from:

```
getwd()
setwd('/Users/tonyt/Dropbox/temp')
```

Now open up Finder/Explorer, and load up the text file so you can see what it looks like.

Notice this is a "tab-delimited" file. R can also handle comma-separated value files (.csv) that excel generates by default.

Now let's load up the data again

```
mydata = read.table("interfacedata.txt", header=TRUE, sep='\t')
mydata
```

Let's do up a data set where the samples are paired. In this experiment, each person used each interface at least once. For the sake of making this interesting, let's pretend that the mouse is

faster than the stylus.

```
mousedata = rnorm(10, 10, 1)
stylusdata = rnorm(10, 12, 1)
data = data.frame(mousedata, stylusdata)
write.table(data, "pairedinterfacedata.txt", sep='\t')
```

Parametric Inferential Tests

t-test: unpaired two-sample t-test

Based on the "interface.txt" you created in the last section. Since we created this example from the same "sample generation function" (i.e. rnorm(10, 10, 1)), H0 is actually true. Let's see if the p-value checks out.

```
# t-test - unpaired two-sample t-test
data = read.table("interfacedata.txt", header=TRUE, sep='\t')
attach(data)
t.test(perf~interface)
detach(data)
```

t-test: paired sample t-test

With paired samples, the basic idea is: for each pair, subtract one value from the other. For the resulting set, now check if there is a difference from 0.

```
# t-test - paired sample t-test
data = read.table("pairedinterfacedata.txt", header=TRUE, sep='\t')
attach(data)
t.test(mousedata, stylusdata, paired=TRUE)
detach(data)
```

Aside: Levene's Test

Parametric tests make two assumptions: (1) the sample data is normal; (2) that there is homogeneity of the variances (i.e. same distribution of variance). A typical test you need to run on your data before you can use a parametric test (e.g. t-test, ANOVA) is to assess for homogeneity of variance. If this doesn't check out, you need to use a non-parametric test.

```
# Levene's test for homogeneity of variance
# Here, you're rooting to NOT reject the H0. You want H0 to be true.
data = read.table("interfacedata.txt", header=TRUE, sep='\t')
library(car)
leveneTest(perf ~ interface, data=data)
```

One-way ANOVA

Let's pretend to be doing a typing speed comparison: Regular keyboard, iPhone prediction, and Android prediction. In advance, let's set it up so that we know that Android prediction is faster than iPhone prediction, and that both are faster than a regular keyboard. In this experiment, a person only does one condition.

```
kbperf = rnorm(10, 10, 1)
ipperf = rnorm(10, 9.5, 1)
anperf = rnorm(10, 9, 1)

# next, let's label each piece of performance data
# Teacher: explain rep("kb", 10)
keyboarddata = data.frame(kbperf, rep("kb", 10))
iphonedata = data.frame(ipperf, rep("ip", 10))
androiddata = data.frame(anperf, rep("ad", 10))

# change the column names so they can be merged
colnames(keyboarddata) = c("perf", "type")
colnames(iphonedata) = c("perf", "type")
colnames(androiddata) = c("perf", "type")

# merge it all
data = rbind(keyboarddata, iphonedata, androiddata)
```

Most of the above was a lot of mumbo-jumbo to set up some data. Let's see how to use the ANOVA. In this ANOVA, what we're going to check to see if one of these samples is different from the others.

```
# one-way ANOVA
# null hypothesis: kbperf = ipperf = anperf

# First, let's check homogeneity of variance
leveneTest(perf ~ type, data=data)

# If that checks out, let's do our one-way ANOVA
aov = aov(perf ~ type, data=data)
summary(aov)
```

Just for kicks, mess around with the initial data set to see what happens. Some ways you can mess around with it: recreate the experiment multiple times over. Try different sizes of n, or different levels of difference between the conditions (or again, perhaps no differences). Note the frequency that the tool gives us the wrong "guess" (aka inference).

Tukey's Honest-Significant Difference (pairwise comparisons)

Now if there is a difference, where are the actual differences? If we conducted separate t-tests for each comparison, we would be increasing the odds that we would make a Type I error. i.e. kb vs. ip = 0.05, ip vs. an = 0.05, kb vs. an = 0.05 Each of these tests incurs a possibility of having a Type I error.

There exist methods to keep this possibility at a set level. Tukey's post-hoc analysis (which compares all possible things with one another at once) is an easy choice. This also makes a bunch of assumptions (e.g. equal number of samples per condition), but is an easy fun one to use.

```
TukeyHSD(aov)
```

One-way Repeated Measures ANOVA

So, let's imagine doing the exact same experiment as before, except this time, a person will do each condition. This is called a "repeated measures" experiment. We basically do the same thing, except we interpret the "error term" differently.

```
# one-way repeated-measure ANOVA
kbperf = rnorm(10, 10, 1)
ipperf = rnorm(10, 9.5, 1)
anperf = rnorm(10, 9, 1)
# Teacher: explain seq(1,10,1)
keyboarddata = data.frame(kbperf, rep("kb", 10), seq(1,10,1))
iphonedata = data.frame(ipperf, rep("ip", 10), seq(1,10,1))
androiddata = data.frame(anperf, rep("ad", 10), seq(1,10,1))
colnames(keyboarddata) = c("perf", "type", "participant")
colnames(iphonedata) = c("perf", "type", "participant")
colnames(androiddata) = c("perf", "type", "participant")
data = rbind(keyboarddata, iphonedata, androiddata)

aov = aov(perf ~ factor(type) + Error(factor(participant)/factor(type)), data)
summary(aov)
```

At this point, if you found a difference, you would use Tukey's HSD to figure out where the differences were.

```
TukeyHSD(aov)
```

Just for fun, you can compare the data as if it were not a repeated-measure ANOVA.

```
aov = aov(perf ~ type, data=data)
summary(aov)
```

Notice when you do this:

```
SS(residuals from one-way ANOVA) == SS(residuals from RM ANOVA-participant) +
SS(residuals from RM ANOVA)
```

This is essentially saying: part of the error is due to the participants. What is the impact of that error? Also notice that the SS(type) == SS(type)

Two-way ANOVA

With a two-way ANOVA, you're exploring the effect of two separate variables (each with different levels). The model that we are thinking about here are thus: SS(A) -- deviation due to the A factor, SS(B) -- deviation due to the B factor, SS(AB) -- deviation due to the interaction between A and B factors, and SSE -- the "error".

```
SS(total) = SS(A) + SS(B) + SS(AB) + SSE
```

The interaction chunk refers to several possible interactions. See, for example the following two references: 1 or 2

Koji Yatani has a very nice example on his site that I'm going to replicate here.

The experiment here imagines that you have designed two cool new interaction techniques. You want to see how these interaction techniques does in relation to the existing old interaction technique (factor A), and also across interaction devices -- a pen vs. touch (factor B). This is called a 2x2 factorial design: in factor A (interaction technique), we have three levels; in factor B (device), we have two levels (pen, touch). In both cases, these factors are "between-subjects" factors.

Prepare the data

```
Device <- factor(c(rep("Pen",24), rep("Touch",24)))
Technique <- factor(rep(c(rep("A",8), rep("B",8), rep("C",8)), 2))
Time <- c(1.2,1.4,1.8,2.0,1.1,1.5,1.5,1.7,
    + 2.1,2.5,2.2,2.2,2.9,2.3,2.3,2.6,
    + 3.5,3.4,3.3,3.2,2.9,2.8,3.8,3.4,
    + 2.4,1.8,2.5,2.1,2.2,1.9,1.7,2.3,
    + 2.8,3.1,3.2,4.0,2.9,3.6,3.2,3.7,
    + 4.5,4.8,4.7,4.1,4.1,4.2,4.6,4.9)
```

Do a Levene's test for homogeneity of variance

```
library(car)
levene.test(Time ~ Device * Technique)
```

Do the two-way ANOVA

```
options(contrasts=c("contr.sum", "contr.poly"))
Anova(lm(Time ~ Device * Technique), type="III")
```

Two-way Repeated Measures

Here is a within-subjects design.

```
Device <- factor(c(rep("Pen",24), rep("Touch",24)))
Technique <- factor(rep(c(rep("A",8), rep("B",8), rep("C",8)), 2))
Time <- c(1.2,1.4,1.8,2.0,1.1,1.5,1.5,1.7,
    + 2.1,2.5,2.2,2.2,2.9,2.3,2.3,2.6,
    + 3.5,3.4,3.3,3.2,2.9,2.8,3.8,3.4,
    + 2.4,1.8,2.5,2.1,2.2,1.9,1.7,2.3,
    + 2.8,3.1,3.2,4.0,2.9,3.6,3.2,3.7,
    + 4.5,4.8,4.7,4.1,4.1,4.2,4.6,4.9)
data <- data.frame(Device, Technique, Time)
data2 <- with(data, cbind(Time[Device=="Pen"&Technique=="A"],
    + Time[Device=="Pen"&Technique=="B"],
    + Time[Device=="Pen"&Technique=="C"],
    + Time[Device=="Touch"&Technique=="A"],
    + Time[Device=="Touch"&Technique=="B"],
    + Time[Device=="Touch"&Technique=="C"]))
data2

D <- factor(c("P","P","P","T","T","T"))
T <- factor(c("A","B","C","A","B","C"))
factor <- data.frame(D,T)
factor

options(contrasts=c("contr.sum", "contr.poly"))
model <- lm(data2 ~ 1)
library(car)
aov <- Anova(model, idata=factor, idesign=~D*T, type="III")
summary(aov, multivariate=FALSE)
```

Mixed-design ANOVA

Sometimes, you will have a factor that you've designed to be between-subjects, and another factor that is within-subjects. For example, let's suppose now you set up your experiment that device is the between-subjects variable, and that interaction technique is the within-subjects variable. Here, you'd have "pen" people, and "touch" people. Pen people do all the interaction techniques, but only with the pen. Touch people also do all the interaction techniques, but again, only with the touch. This is called a mixed-factor design.

```
Participant <- factor(cbind(rep(c("1", "2", "3", "4", "5", "6", "7", "8"), 3),
    + rep(c("9", "10", "11", "12", "13", "14", "15", "16"), 3)))
Device <- factor(c(rep("Pen",24), rep("Touch",24)))
Technique <- factor(rep(c(rep("A",8), rep("B",8), rep("C",8)), 2))
Time <- c(1.2,1.4,1.8,2.0,1.1,1.5,1.5,1.7,
    + 2.1,2.5,2.2,2.2,2.9,2.3,2.3,2.6,
    + 3.5,3.4,3.3,3.2,2.9,2.8,3.8,3.4,
    + 2.4,1.8,2.5,2.1,2.2,1.9,1.7,2.3,
    + 2.8,3.1,3.2,4.0,2.9,3.6,3.2,3.7,
    + 4.5,4.8,4.7,4.1,4.1,4.2,4.6,4.9)
data <- data.frame(Device, Technique, Participant, Time)
data

library(ez)
options(contrasts=c("contr.sum", "contr.poly"))
ezANOVA(data=data, dv=.(Time), wid=.(Participant), within=.(Technique),
between=.(Device), type=3)
```

Unbalanced Designs

It is possible to run ANOVAs when you have an unbalanced number of participants in each cell, but I'm not going to get into it here. Generally, it is just easier to avoid this, and get a balanced number of participants. ;-)

Effect Size

To this point, all we have done is worried about whether there is a significant difference. But, there is the issue of whether the difference is substantive. Imagine the following test, where a and b represent the number of days one is sick with the flu, and b represents the data from the "new drug."

```
> a = rnorm(1000, 10, 1)
> b = rnorm(1000, 9.9, 1)
> t.test(a,b)
    Welch Two Sample t-test
t = -2.2385, df = 1997.908, p-value = 0.0253
```

Even though there is a significant difference here, it is not really that substantive. 9.9 days compared to 10? Come on now.

There is a way of quantifying this difference, and it is called measuring the **effect size**. This is a normalized measure. So, for instance, while 0.1 days is not a lot in comparison to 10 days, it might be a lot if the typical variance/standard deviation of each patient is (say) 0.001 days. So,

the effect size takes this into account.

Each statistical test has a different way of measuring effect size (depends on the nature of the data, and the test that you are using). If you find there is a statistical difference, you are normally obligated to report the effect size, too.

See the links below for resources on which effect size metric you should use (it will be based on the test that you use).

Cohen's d for a paired t-test

For illustrative purposes, let's look at Cohen's d, which you would normally use for a paired t-test. In this case, Cohen's d is calculated as follows:

```
d = |M|/SD

where M is the mean of the differences, and
      SD is the standard deviation of the differences
```

If you think about what's going on here, you'll get a bigger value for d if the differences are huge, and/or the standard deviations between the differences are small.

There is a rule of thumb about the size of the effect: 0.2 - 0.3 is small, around 0.5 is medium, and 0.8 or larger is "large." But, this depends on context.

- A nice visual explanation for Cohen's D

Intuitive explanation for eta squared

A common measure for effect size when using ANOVAs is eta squared. It looks something like this for a one-way ANOVA:

```
eta squared = SSbetween / SStotal
```

The way to think about this is that there is variance in your data, and this is captured by the SSs. Remember that SStotal = SSbetween + SSwithin, where SSwithin is all the random variation due to measurement error, or individual differences. What the equation above tells you is that eta squared is essentially telling us the following: what is the proportion of the total variance that the "between" variable is responsible for? So, eta squared is basically a proportion measure. If it were 0, then there is basically no effect--everything in SStotal is due to the SSwithin. If it were 1, then there is no variation among the measurements.

Another one you will hear about is partial eta squared. This is basically used in factorial ANOVA

designs, and is asking the proportion that is due to each factor. For example, imagine an AxB design. You'd calculate the following partial eta squareds:

```
partial eta squared(a) = SS(a) / SStotal

partial eta squared(b) = SS(b) / SStotal

partial eta squared(axb) = SS(axb) / SStotal // interaction effect

partial eta squared(error) = SS(error) / SStotal // error gets its own
```

If you add up all these partial eta squareds, you should end up with 1.

- Explanation of the difference between eta squared and partial eta squared

Understanding Correlations

Correlations are not used much in HCI research, as far as I can tell. Nevertheless, it is useful to understand how Pearson's r Correlation measure works. r ranges from -1 to 1. |r| tells you the strength of the correlation, with 1 being a perfect correlation. The positive/negative of the value tells you which direction the relationship goes in.

- A nice visual explanation for correlations
- Interpreting r values

Non-Parametric Tests

The parametric tests are used when the data is ratio data, and the samples fulfill a number of assumptions (normality, and homogeneity of variances). When you have ordinal data, or your data doesn't fulfill those assumptions, then we need to turn to non-parametric tests.

Here are two different scenarios where you are comparing keyboard vs. android text input.

Within-Subjects Design: Imagine each participant gets to try both conditions. At the end, you ask your participants to fill out a questionnaire, asking the question: "On a scale of 1 (very uncomfortable) - 7 (very comfortable), how would you rate [X]?" Imagine that you ask the question twice -- once for each keyboard and Android. This data is considered "paired." Now you end up with a data set that looks like this:

Participant Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Keyboard	1	3	2	4	3	2	1	1	3	2
Android Prediction	3	5	6	4	2	4	7	6	3	5

Between-Subjects Design: Imagine participants are assigned a group--each person only does the keyboard **or** the android condition. Now, you ask them the same question about the comfort level. This data is considered "unpaired." Your data looks like this:

Participant Number	Keyboard	Android Prediction
1	2	
2	4	
3	3	
4	1	
5	2	
6	3	
7	3	
8	2	
9	3	
10		1
11		3
12		5
13		4
14		2
15		4
16		3
17		5
18		5
19		3
20		2

In the first scenario, you would use a **Wilcoxon Signed-Rank test** -- this is like a paired t-test.

```
GroupA <- c(1,3,2,4,3,2,1,1,3,2)
GroupB <- c(3,5,6,4,2,4,7,6,3,5)
library(coin)
wilcoxsign_test(GroupA ~ GroupB, distribution="exact")
```

In the second scenario, you would use a **Mann-Whitney test** (sometimes known as **Mann-Whitney-Wilcoxon test**). This is like an unpaired t-test. In the R package, this is called just a regular wilcox test. I don't know why.

```
GroupA = c(2,4,3,1,2,3,3,2,3,1)
GroupB = c(3,5,4,2,4,3,5,5,3,2)
wilcox.test(GroupA, GroupB)
```

- Koji Yatani provides a nice explanation for how these work here: Mann-Whitney Test Wilcoxon Signed Test.

Reporting Statistics

- **t-test**: With a Welch's t test, we found a significant effect for techniques ($t(15) = 2.20$, $p < 0.05$, Cohen's $d=0.98$) with Technique 2 outperforming Technique 1.
- **one-way ANOVA**: Bartlett's test did not show a violation of homogeneity of variances ($X^2(2) = 1.03$, $p = 0.60$). With one-way ANOVA, we found a significant effect of Group on Value ($F(2,21)=12.01$, $p<0.01$, partial $eta^2=0.54$).
- **two-way ANOVA**: With two-way ANOVA, we found significant main effects of Device ($F(1,42)=94.5$, $p<0.01$, partial $eta^2=0.69$ and Technique ($F(2,42)=165$, $p<0.01$, partial $eta^2=0.89$) on the performance time. We also found a significant interaction of Device and Technique ($F(2,42)=3.63$, $p<0.05$, partial $eta^2=0.15$).
- **Mann-Whitney-Wilcoxon U Test**: The medians of Group A and Group B were 2.5 and 3.5, respectively. We ran a Mann-Whitney's U test to evaluate the difference in the responses of our 5-Likert scale question. We found a significant effect of Group (The mean ranks of Group A and Group B were 7.8 and 13.2, respectively; $U = 23$, $Z = -2.11$, $p < 0.05$, $r = 0.47$).
- **Wilcoxon Signed Rank Test**: The medians of Group A and Group B were 2.0 and 4.5, respectively. An Wilcoxon Signed-rank test shows that there is a significant effect of Group ($W = 1$, $Z = -2.39$, $p < 0.05$, $r = 0.53$).

Some Resources

- Koji Yatani's Statistics for HCI - Many of the examples here are drawn from his site.
- http://www.statmethods.net/
- http://personality-project.org/r/
- http://www.r-tutor.com/elementary-statistics
- http://www.cookbook-r.com/

Some Datasets

- http://personality-project.org/r/datasets/

Issues

- Can't see the x-axis labels on my charts

`Preferences` -> `Quartz` -> `"Override R Quartz width/height parameters"` (source)