- **Introduction to RStudio (continuation)**
  - Making your data R-friendly
  - The aggregate() function
  - Building a dataframe
- **Probability density functions**
  - Using R to find critical values and probabilities
- **Hypothesis testing (t-tests)**
- **Power determination (t-tests)**

Introduction to RStudio (continuation)

Making your data R-friendly

If you've looked at the homework yet (Question 1), you'll see a data table that looks like this:

Sample 1	38	20	22	50	46	25	45	40	39	43
Sample 2	37	24	27	25	38	58	50	32	57	38
Sample 3	42	46	45	40	50	43	45	29	38	51
Sample 4	39	45	31	42	21	60	48	26	51	38
Sample 5	49	36	33	50	42	42	46	33	44	25
Sample 6	39	40	50	30	47	72	60	34	47	34

While this may look fine to you, R would prefer to have the data in a different format for importing and subsequent analysis. Dataframes in R are organized in columns, and each column should have a heading. Each number in the above table has one classifier associated with it: Sample. And the numbers are Measurements associated with that classifier. The data can be re-formatted thusly:

Sample	Meas
1	38
1	20
1	22
...	...
6	34
6	47
6	34

To accomplish this re-formatting, I recommend using Excel. Simply copy the data from the homework, paste (transpose) into Excel, and then stack into a column (**always use copy-paste, NEVER type numbers!**). Once you're done, save the worksheet as a comma delimited file (.csv); and now you have R-ready data.

For more detailed instructions, refer to the supplementary document "Importing Data into RStudio" found on the course website.

The aggregate() function

To find the mean of each of the 6 samples above, you *could* use the subset() function we talked about last time. For example, for Sample 1:

```
mean ( subset ( ques1.dat, Sample == 1 )$Meas )
```

You could then run six lines of script like this, one for each sample. You could then do the same for standard deviation, CV, etc. Functional, but ugly.

Another way to tackle this is with the aggregate() function.

Example 1 *[Lab2ex1.R]*

The aggregate() function allows you to apply R functions to systematic subsets of data, provided these subsets are defined by levels of a *factor*. Let's convert the imported data into a dataframe and then ask R to tell us about the current structure of that dataframe:

```
ques1_dat<-as.data.frame(ques1_dat)
str(ques1_dat, give.attr=F)
```

The output:

```
'data.frame':  60 obs. of  2 variables:
 $ Sample: int  1 1 1 1 1 1 1 1 1 1 ...
 $ Meas  : int  38 20 22 50 46 25 45 40 39 43 ...
```

What this tells us is that R thinks the Measurements are integers (which they R), but it also thinks the Sample ID's are integers -- but they aren't. The Sample ID's are, in fact, levels of the classification variable "Sample"; that is, "Sample" is a *factor*. We need to tell R this:

```
ques1_dat$Sample<-as.factor(ques1_dat$Sample)
```

Now if we ask R about the structure of the dataframe, this is the output we get:

```
'data.frame':  60 obs. of  2 variables:
 $ Sample: Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Meas  : int  38 20 22 50 46 25 45 40 39 43 ...
```

Perfect, now we can proceed to use the aggregate() function to find the means of all six samples, using a single line of script:

```
means <- aggregate(ques1_dat$Meas, list(ques1_dat$Sample), mean)
means
```

```
  Group.1    x
1       1 36.8
2       2 38.6
3       3 42.9
4       4 40.1
5       5 40.0
6       6 45.3
```

There is nothing special about the mean() function. This same approach can be used to find the standard deviations, CV's, etc. Now, once you have calculated all these things, how can you efficiently grab all the results? Here's one way:

```
ns <- aggregate(ques1_dat$Meas, list(ques1_dat$Sample), length)
ids <- unique(ques1_dat$Sample)

summary = data.frame(ids,means$x,ns$x)
summary
```

```
  ids means.x ns.x
1   1    36.8   10
2   2    38.6   10
3   3    42.9   10
4   4    40.1   10
5   5    40.0   10
6   6    45.3   10
```

If you really want to get fancy, you can also rename the headers in this newly constructed summary dataframe, as follows:

```
names(summary)[1:3] <- c("Sample","Mean","n")
summary
```

```
  Sample Mean  n
1      1 36.8 10
2      2 38.6 10
3      3 42.9 10
4      4 40.1 10
5      5 40.0 10
6      6 45.3 10
```

Probability density functions

R is a powerful statistical computing software package that makes tables of critical values (Z, t, F, chi-squared, etc.) unnecessary. With some simple code, you can easily access *exact* critical values and their associated probabilities, as illustrated in the following script. Play around! These are the functions that will make your life easy in Question 2 of the first homework.

Example 2 *[Lab2ex2.R]*

```
#BIOL933
#Lab 2
#Example 2

# This script illustrates the use of R to calculate densities (critical
# values) and their corresponding probabilities for common distributions

#Normal distribution
Nvalue<-qnorm(0.975)
Nvalue
pNvalue<-pnorm(1.959964)
pNvalue

#t distribution
Tvalue<-qt(0.975,10)
Tvalue
pTvalue<-pt(2.228139,10)
pTvalue

#chi^2 distribution
Chivalue<-qchisq(0.975,10)
Chivalue
pChivalue<-pchisq(20.48318,10)
pChivalue

#F distribution
Fvalue<-qf(0.975,10,4)
Fvalue
pFvalue<-pf(8.843881,10,4)
pFvalue

#visualizing distributions
par(mfrow=c(2,2))
plot(function(x) dnorm(x), -6, 6, ylim = c(0, 0.5), main = "Z - Density")
plot(function(x) dt(x, df = 10), -6, 6, ylim = c(0, 0.5), main = "t - Density")
plot(function(x) dchisq(x, df = 10), 0, 50, ylim = c(0, 0.1), main = "chi^2 - Density")
plot(function(x) df(x, df1 = 10, df2 = 4), 0, 15, ylim = c(0, 0.7), main = "F - Density")

#generating random numbers from a normal distribution
rNvalues<-rnorm(10,5,2)
rNvalues
```

t-tests

To demonstrate hypothesis testing and t-test power determination in R, let's revisit the barley dataset from Lecture 1.

Example 3 *[Lab2ex3.R]*

For starters, go ahead and import the Lab2ex3.csv dataset and name the dataframe "extract_dat" as we did last week. If you go into the R help page for the t.test() function, you'll run into this:

```
t.test(  x,
         y = NULL,
         alternative = c("two.sided", "less", "greater"),
         mu = 0,
         paired = FALSE,
         var.equal = FALSE,
         conf.level = 0.95, ...)
```

This is a template meant to illustrate the usage of this particular function. To conduct a simple one-sample, two-tailed t-test on the barley extract data:

```
t.test(extract_dat$Extract, mu=78)
```

And the output:

```
        One Sample t-test

data:  extract.dat$Extract
t = -6.2727, df = 13, p-value = 2.866e-05
alternative hypothesis: true mean is not equal to 78
95 percent confidence interval:
 75.23436 76.65135
sample estimates:
mean of x
 75.94286
```

And what was the power of this particular test?

```
power.t.test(  n = length(extract.dat$Extract),
               delta = 78-mean(extract.dat$Extract),
               sd = sd(extract.dat$Extract),
               sig.level = 0.05,
               type = "one.sample",
               alternative = "two.sided")
```

And the results:

```
           One-sample t test power calculation

                    n = 14
                delta = 2.057143
                   sd = 1.227076
            sig.level = 0.05
                power = 0.9999288
          alternative = two.sided
```

And variations on the theme

A t-test for two independent samples without an assumption of equal variances (default of t.test function).

```
height <- c(2,2,2,4)
height2 <- c(2,2,3,5)
t.test(height, height2)

      Welch Two Sample t-test

data:  height and height2
t = -0.5774, df = 5.4, p-value = 0.5869
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.677493  1.677493
sample estimates:
mean of x mean of y
      2.5       3.0
```

A t-test for two independent samples, assuming equal variances

```
t.test(height, height2, var.equal = T)
pooled.var <- (1/2)*(var(height)+var(height2))
power.t.test(  n = length(height),
               delta = mean(height)-mean(height2),
               sd = sqrt(pooled.var),
               sig.level = 0.05,
               type = "two.sample",
               alternative = "two.sided")
```

```
      Two Sample t-test

data:  height and height2
t = -0.5774, df = 6, p-value = 0.5847
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.619088  1.619088
sample estimates:
mean of x mean of y
      2.5       3.0

     Two-sample t test power calculation

              n = 4
          delta = 0.5
             sd = 1.224745
      sig.level = 0.05
          power = 0.07102622
    alternative = two.sided

NOTE: n is number in *each* group
```

A t-test for paired samples

```
t.test(height, height2, paired=T)
diff<-height-height2
power.t.test(  n = length(height),
               delta = mean(height)-mean(height2),
               sd = sd(diff),
               sig.level = 0.05,
               type = "paired",
               alternative = "two.sided")

      Paired t-test

data:  height and height2
t = -1.7321, df = 3, p-value = 0.1817
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.4186931  0.4186931
sample estimates:
mean of the differences
                   -0.5

     Paired t test power calculation

              n = 4
          delta = 0.5
             sd = 0.5773503
      sig.level = 0.05
          power = 0.2310195
    alternative = two.sided

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within
pairs
```