One-way Analysis Of Variance (ANOVA)

One-way ANOVA is a statistical method to test the claim that the means of **two or more** samples x_i with sample size n_i ($i = 1 \ldots k$) are equal. The test makes the following assumptions:

- the shape of the population from which the samples are drawn should be normal
- all samples have equal variance
- all samples should be drawn independently

Again there are two hypothesis:

- The null hypothesis H_0 states that the means $\overline{x}_i$ with $i = 1 \ldots k$ are equal.
- The alternative hypothesis H_1 states that the means $\overline{x}_i$ with $i = 1 \ldots k$ are not equal.

As with the two sample t-test, the ANOVA test can also be formulated as a test to determine whether two or more independent samples with equal variance were drawn from populations having the same normal distribution.

One-way ANOVA is a generalization of the two sample t-test. To illustrate this we assume to have access to two gene expression level populations with a difference in mean. We draw one sample with sample size $n = 9$ from each population and perform a two sample t-test and a one-way ANOVA test to compute a p-value for the null hypothesis:

```
mu_geneA = 0.26
sigma_geneA = 0.01
mu_geneB = 0.28
sigma_geneB = 0.01
N = 100000
population_geneA = np.random.normal(size=N,loc=mu_geneA,scale=sigma_geneA)
population_geneB = np.random.normal(size=N,loc=mu_geneB,scale=sigma_geneB)
n = 9
geneA = np.random.choice(population_geneA,size=n)
geneB = np.random.choice(population_geneB,size=n)
print "two sample t-test:"
print stats.ttest_ind(geneA,geneB)
print "one-way ANOVA:"
print stats.f_oneway(geneA,geneB)
```

```
two sample t-test:
Ttest_indResult(statistic=-3.9482390041166515, pvalue=0.0011507408616140947)
one-way ANOVA:
F_onewayResult(statistic=15.588591233628064, pvalue=0.0011507408616140908)
```

We can see that the p-values computed by both tests are exactly the same. It is the test statistic that is different. ANOVA computes an **F-statistic** that is simply a ratio of two variances. So, where the two sample t-test T-statistic looks at the difference in mean between two samples, the one-way ANOVA F-statistics looks at the **ratio of two variances**.

Conidere a typical ANOVA example in which we want to test the effect of a certain treatment on the expression level of a specific gene. There are three types of treatment: "T1", "T2" and "T3". We have four samples. The first sample contains gene expression levels of mice that did not undergo any treatment. The next three samples contain gene expression levels of mice treated with treatment "T1", "T2" and "T3" respectively. We assume each sample has sample size $n_1 = n_2 = n_3 = n_4 = 12$.

We want to test if the means of the four samples are equal (i.e. there is no effect of any of the treatments). The following code creates these samples such that the means of the populations from which the samples are drawn is indeed equal:

```
mu_control = 0.26
mu_T1 = 0.26
mu_T2 = 0.26
mu_T3 = 0.26
sigma = 0.05 #equal variance is assumed
n = 12
sample_control = np.random.normal(size=n,loc=mu_control,scale=sigma)
sample_T1 = np.random.normal(size=n,loc=mu_T1,scale=sigma)
sample_T2 = np.random.normal(size=n,loc=mu_T2,scale=sigma)
sample_T3 = np.random.normal(size=n,loc=mu_T3,scale=sigma)
```

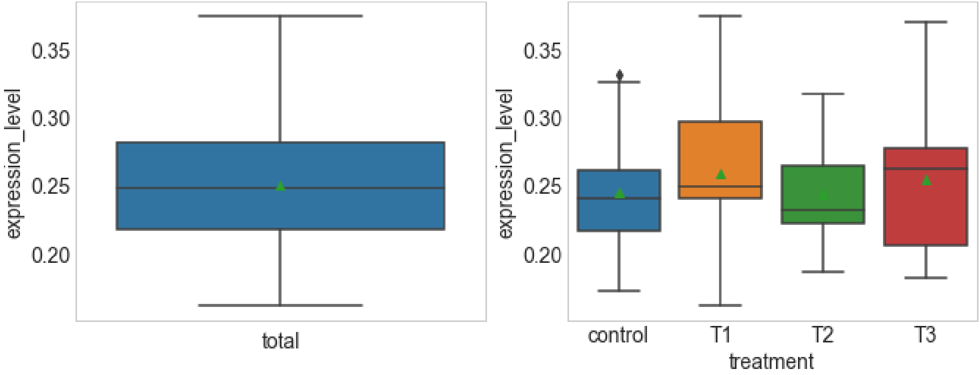
In ANOVA we don't talk about samples (eventhough they are samples) but instead we refer to the difference between the samples as a **factor**. For our dataset we call the factor "treatment" ((concidering no treatment (control) as a special type of treatment). This factor has four **levels** or **groups**: "control", "T1", "T2" and "T3".

To make this more clear we create a dataset with two columns: "treatment" (our factor) that tells the type of treatment (the level) of an observation and "expression_level" that contains the measured gene expression level for an observation (the random variable):

```
df = pd.DataFrame()
df["treatment"] = ["control"]*n + ["T1"]*n + ["T2"]*n + ["T3"]*n
df["expression_level"] = np.concatenate((sample_control,sample_T1,sample_T2,sample_T3))
```

To gain insight into the dataset we plot a boxplot for the total gene expression level and then we create four boxplots for the gene expression level in each of the four levels of the factor "treatment":

```
plt.figure(figsize=(11,5))
plt.subplot(1,2,1)
ax = sns.boxplot(y="expression_level",data=df,showmeans=True)
ax.set_xlabel("total")
plt.subplot(1,2,2)
sns.boxplot(x="treatment",y="expression_level",data=df,showmeans=True)
plt.show()
```



The mean for each distribution is indicated by the green triangle. We see that these sample means differ. But again, is this difference significant? Is it bigger than expected due to the uncertainty in the sampling? We know that it is not.

Let's compute the F-statistic. The first variance in the statistic is the variation between the $k = 4$ sample means $\overline{x}_i$. This variation is also known as the "**between group variation**" or as the "**explained variance**" as it is the variance in the gene expression that is caused (or can be explained) by the treatment. The variance is computed as what is known as the **treatment sum of squares**:

$$SST = \sum_{i=1}^{k} n_i(\overline{x}_i - \overline{x}_{total})^2,$$

where $\overline{x}_i$ is the mean of the observations in level i and $\overline{x}_{total}$ is the mean of all observations in the dataset.

The code below computes the SST for our four samples:

```
def compute_SST(control,T1,T2,T3):
    mean_total = np.mean(np.concatenate((control,T1,T2,T3)))
    mean_control = np.mean(control)
    mean_T1 = np.mean(T1)
    mean_T2 = np.mean(T2)
    mean_T3 = np.mean(T3)
    SST = len(control)*(mean_control-mean_total)**2
    SST += len(T1)*(mean_T1-mean_total)**2
    SST += len(T2)*(mean_T2-mean_total)**2
    SST += len(T3)*(mean_T3-mean_total)**2
    return SST
SST = compute_SST(sample_control,sample_T1,sample_T2,sample_T3)
print "SST = %f" % SST
```

```
SST = 0.001968
```

The second variance in the F-statistic is the variation within the samples, also known as the "**within group variation**" or "**unexplained variance**". This is computed as the **sum of squares of the error**:

$$SSE = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \overline{x}_i)^2,$$

where x_{ij} id the jth observation in sample i.

The following code computes the SSE:

```
def compute_SSE(control,T1,T2,T3):
    mean_control = np.mean(control)
    mean_T1 = np.mean(T1)
    mean_T2 = np.mean(T2)
    mean_T3 = np.mean(T3)
    SSE = np.sum([(y-mean_control)**2 for y in control])
    SSE += np.sum([(y-mean_T1)**2 for y in T1])
    SSE += np.sum([(y-mean_T2)**2 for y in T2])
    SSE += np.sum([(y-mean_T3)**2 for y in T3])
    return SSE

SSE = compute_SSE(sample_control,sample_T1,sample_T2,sample_T3)
print "SSE = %f" % SSE
```

```
SSE = 0.107293
```

What is important to realize is that we have just partioned the total variance of the expression levels into two parts: one part SST caused by the treatment and one part SSE caused by an unknown source.

We can check this. The variance of the expression level in the dataset (not taking treatment into account) is computed by the **total sum of squares**:

$$TSS = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \overline{x}_{total})^2.$$

The following code computes the TTS:

```
def compute_TSS(control,T1,T2,T3):
    mean_total = np.mean(np.concatenate((control,T1,T2,T3)))
    TSS = np.sum([(y-mean_total)**2 for y in control])
    TSS += np.sum([(y-mean_total)**2 for y in T1])
    TSS += np.sum([(y-mean_total)**2 for y in T2])
    TSS += np.sum([(y-mean_total)**2 for y in T3])
    return TSS

TSS = compute_TSS(sample_control,sample_T1,sample_T2,sample_T3)
print "TSS = %f" % TSS
```

```
TSS = 0.109261
```

We now have that TSS = SST + SSE:

```
print SST + SSE
```

```
0.10926068778284104
```

The F-statistic computed by ANOVA is the ratio of SST and SSE (with some variance normalization as we have seen before):

$$\frac{SST/(k-1)}{SSE/(n_t-k)},$$

where n_t is total number of observations in the dataset. The F-statistic increases as the SST increases and/or as the SSE decreases. To state this differently: the F-statistic increases as the explained variance gets larger than the unexplained variance. Note that a larger explained variance means a larger effect of the factor on the total variance in the dataset, i.e. a larger difference between the means of the factor levels.

If the null hypothesis is true then the F-statistic follows an F-distribution with $df_1 = k - 1$ and $df_2 = n_t - k$ degrees of freedom. For our dataset the F-statistic is:

```
k = 4
N = k*n
F = (SST/(k-1))/(SSE/(N-k))
print F
```

```
0.2690322819972136
```

The Python function to perform one-way ANOVA shows that this is indeed the correct value and also computes the p-value for the F-statistic:

```
print stats.f_oneway(sample_control,sample_T1,sample_T2,sample_T3)
```

```
F_onewayResult(statistic=0.26903228199721374, pvalue=0.847367420790879)
```

There is another Python function for ANOVA that provides more details on the SST and the SSE:

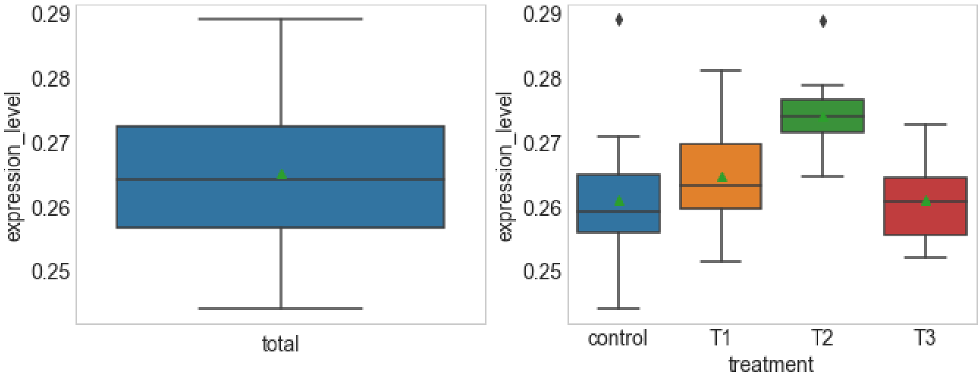
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('expression_level ~ treatment',
                data=df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
             sum_sq    df         F    PR(>F)
treatment  0.001968   3.0  0.269032  0.847367
Residual   0.107293  44.0       NaN       NaN
```

The column "sum_sq" shows the sum of squares SST in row "treatment" and SSE in row "Residual" (the unexplained variance is sometimes also refered to as the **residuals**). The column "df" shows the degrees of freedom for each variance and the column "F" shows the F-statistic. Finally the last column shows the p-value for the factor "treatment". We can see that the values for SST and SSE are the same as what we computed above.

Consider the example above with treatment T2 showing an increased gene expression level:

```
mu_control = 0.26
mu_T1 = 0.26
mu_T2 = 0.27
mu_T3 = 0.26
sigma = 0.01 #equal variance is assumed
n = 12
sample_control = np.random.normal(size=n,loc=mu_control,scale=sigma)
sample_T1 = np.random.normal(size=n,loc=mu_T1,scale=sigma)
sample_T2 = np.random.normal(size=n,loc=mu_T2,scale=sigma)
sample_T3 = np.random.normal(size=n,loc=mu_T3,scale=sigma)
df = pd.DataFrame()
df["treatment"] = ["control"]*n + ["T1"]*n + ["T2"]*n + ["T3"]*n
df["expression_level"] = np.concatenate((sample_control,sample_T1,sample_T2,sample_T3))
plt.figure(figsize=(11,4))
plt.subplot(1,2,1)
ax = sns.boxplot(y="expression_level",data=df,showmeans=True)
ax.set_xlabel("total")
plt.subplot(1,2,2)
sns.boxplot(x="treatment",y="expression_level",data=df,showmeans=True)
plt.show()
```



If we now perform ANOVA we see that the p-value for treatment is very small and we reject the null hypothesis if $\alpha = 0.01$:

```
mod = ols('expression_level ~ treatment',
                data=df).fit()

aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
             sum_sq    df         F    PR(>F)
treatment  0.001384   3.0  5.945948  0.001702
Residual   0.003413  44.0       NaN       NaN
```

The ANOVA says that the means of the four samples are not equal. But we do not know which mean(s) differ(s). Other statistical tests such as the least significant difference test can gain more insight into which level(s) differ(s). This test is only used when the null hypothesis is rejected as a result of your hypothesis test results.

Also, we could perform several pair-wise two sample t-tests. Then why perform one-way ANOVA if we can just do all pair-wise two-sample comparisons? The answer lies again in the power of the test. An ANOVA analysis is more likely to correctly reject the null hypothesis than any of the more sepecific pair-wise t-test, i.e. ANOVA has higher power.

two-way ANOVA

In our gene expression dataset we called the treatment a factor. A factor has two or more levels. In our dataset there are $l_t = 4$ levels: "control", "T1", "T2" and "T3". In two-way ANOVA there are **two factors**, each with two or more levels.

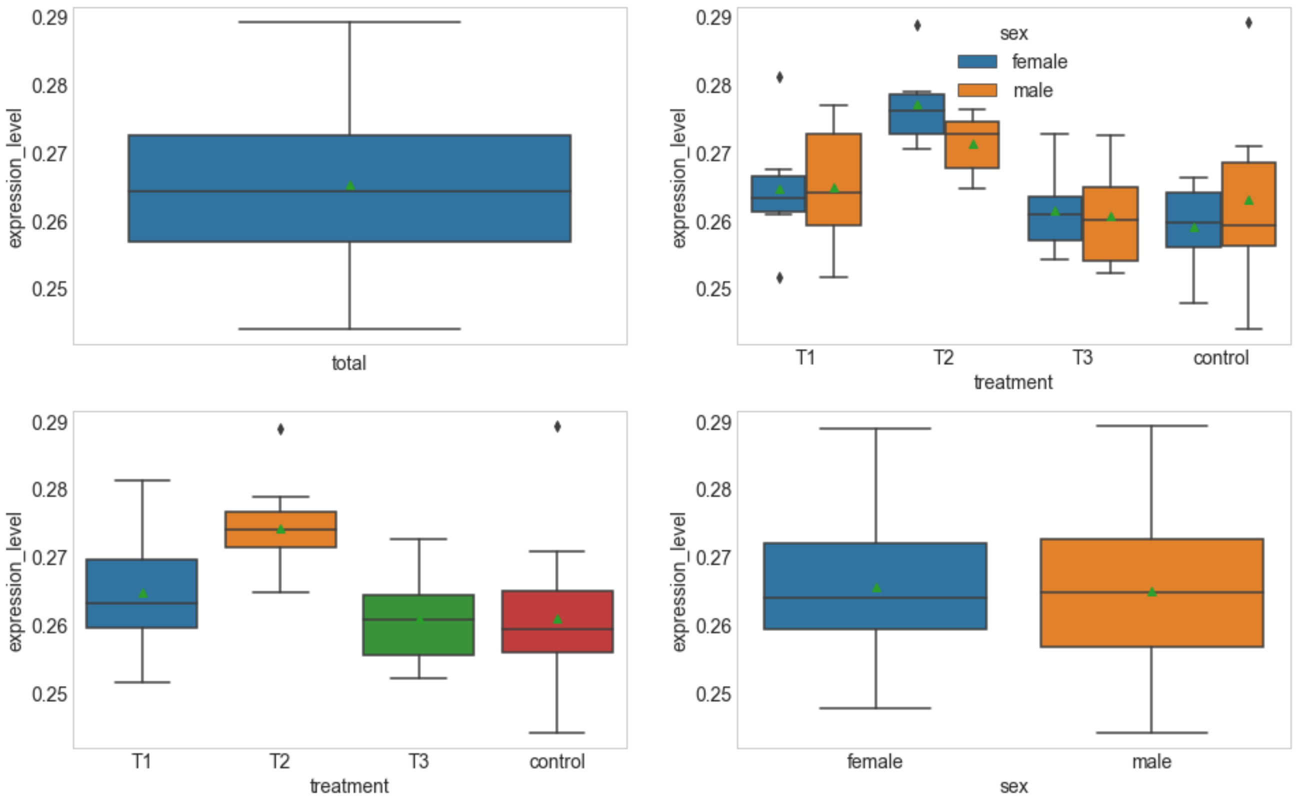
Let's add another factor to our dataset. Suppose we take the sex of a model mouse into account. We add (randomly) a second factor "sex" to the dataset with $l_s = 2$ levels: "female" and "male":

```
df["sex"] = ["female" if x%2==0 else "male" for x in range(k*n)]
df = df[["treatment","sex","expression_level"]].sort_values(["treatment","sex"])
```

For each of the $l_t \times l_s = 8$ level combinations the dataset contains $r = 6$ observations.

We can plot this dataset as boxplots for each factor level and for each combination of levels between the factors (remember that treatment "T2" has an increased expression level in our dataset):

```
plt.figure(figsize=(16,10))
plt.subplot(2,2,1)
ax = sns.boxplot(y="expression_level",data=df,showmeans=True)
ax.set_xlabel("total")
plt.subplot(2,2,2)
ax = sns.boxplot(x="treatment",y="expression_level",hue="sex",data=df,showmeans=True)
plt.subplot(2,2,3)
ax = sns.boxplot(x="treatment",y="expression_level",data=df,showmeans=True)
plt.subplot(2,2,4)
ax = sns.boxplot(x="sex",y="expression_level",data=df,showmeans=True)
plt.show()
```



We can perform a one-way ANOVA for the factor "treatment" again (so not taking the factor "sex" into account):

```
mod = ols('expression_level ~ treatment',
                data=df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
              sum_sq    df         F    PR(>F)
treatment  0.001384   3.0  5.945948  0.001702
Residual   0.003413  44.0       NaN       NaN
```

At $\alpha = 0.01$ the test correctly rejects the null hypothesis.

Also, we can perform a one-way ANOVA for the factor "sex" (so not taking the factor "treatment" into account):

```
mod = ols('expression_level ~ sex',
                data=df).fit()
```

```
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
            sum_sq    df         F   PR(>F)
sex       0.000003   1.0  0.033092  0.85645
Residual  0.004793  46.0       NaN      NaN
```

At $\alpha = 0.01$ the test correctly accepts the null hypothesis as we assigned the levels for factor "sex" randomly.

In two-way ANOVA both factors are taken into account when computing the F-statistic. Similar to one-way ANOVA the total variance TSS of the gene expresion level is partitioned into the variance $SS_{treatment}$ caused by the factor "treatment", the variance SS_{sex} caused by the factor "sex" and the variance SSE caused by an unknown source:

$$TSS = SS_{treatment} + SS_{sex} + SSE.$$

So, without going into too much mathematics, what does this mean?

First we perform a two-way ANOVA on our dataset. We can do this in Python as follows:

```
mod = ols('expression_level ~ treatment + sex',
                data=df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
             sum_sq    df         F    PR(>F)
treatment  0.001384   3.0  5.816689  0.001986
sex        0.000003   1.0  0.043488  0.835794
Residual   0.003410  43.0       NaN       NaN
```

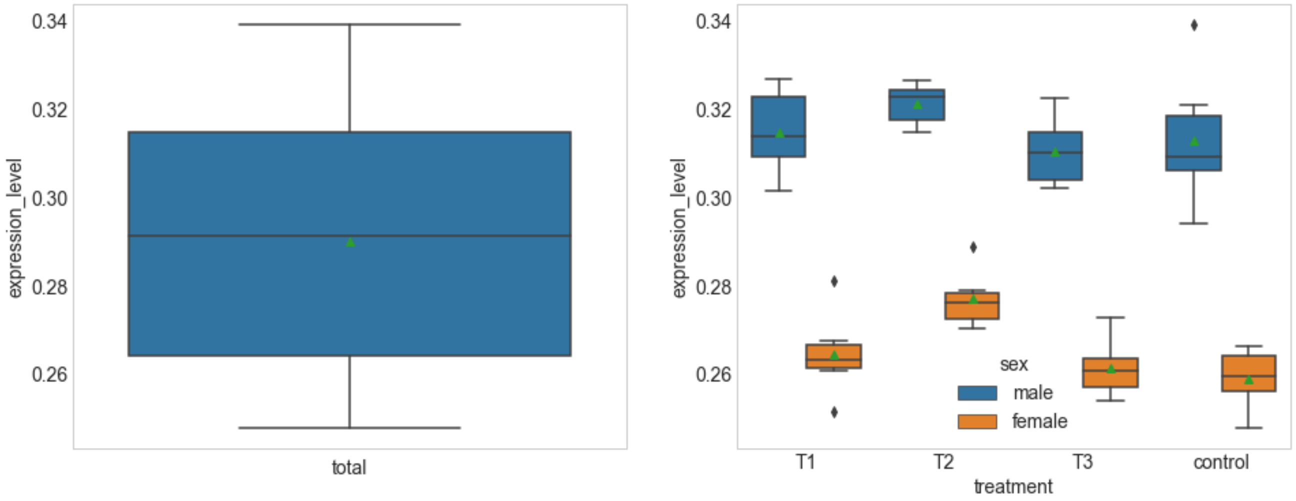
Notice how a row is added for the factor "sex". We have two p-values: one for factor "treatment" and one for factor "sex". At $\alpha = 0.01$ only factor "treatment" shows a difference in mean for one or more of the levels.

Now we make a drastic change in our dataset. We increase the gene expression level for male mice consistently with 0.05:

```
df_male = df[df.sex=="male"]
df_female = df[df.sex=="female"]
df_male.loc[:,"expression_level"] = df_male.loc[:,"expression_level"] + 0.05
df2 = pd.concat([df_male,df_female])
```

If we plot our dataset now we get:

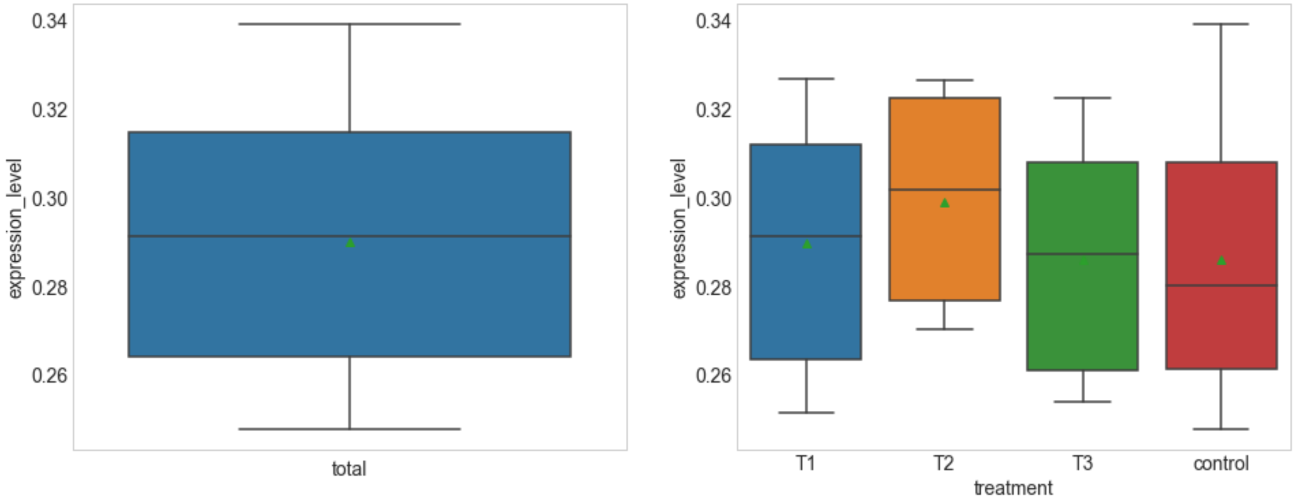
```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
ax = sns.boxplot(y="expression_level",hue="treatment",data=df2,showmeans=True)
ax.set_xlabel("total")
plt.subplot(1,2,2)
sns.boxplot(x="treatment",y="expression_level",hue="sex",data=df2,showmeans=True)
plt.show()
```



The first thing we notice is that the total variance (the variance for all the observations (not taking the two factors into account)) is much larger than the variance of the eight factor level combinations individually. This large variance is caused by the fact that male mice have much higher geneA expression levels than female mice.

When we perform one-way ANOVA for the factor "treatment" then the factor "sex" (which we don't take into account when doing a one-way ANOVA on "treatment") increases the variance in the four levels of factor "treatment". We can plot this as follows:

```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
ax = sns.boxplot(y="expression_level",hue="treatment",data=df2,showmeans=True)
ax.set_xlabel("total")
plt.subplot(1,2,2)
sns.boxplot(x="treatment",y="expression_level",data=df2,showmeans=True)
plt.show()
```



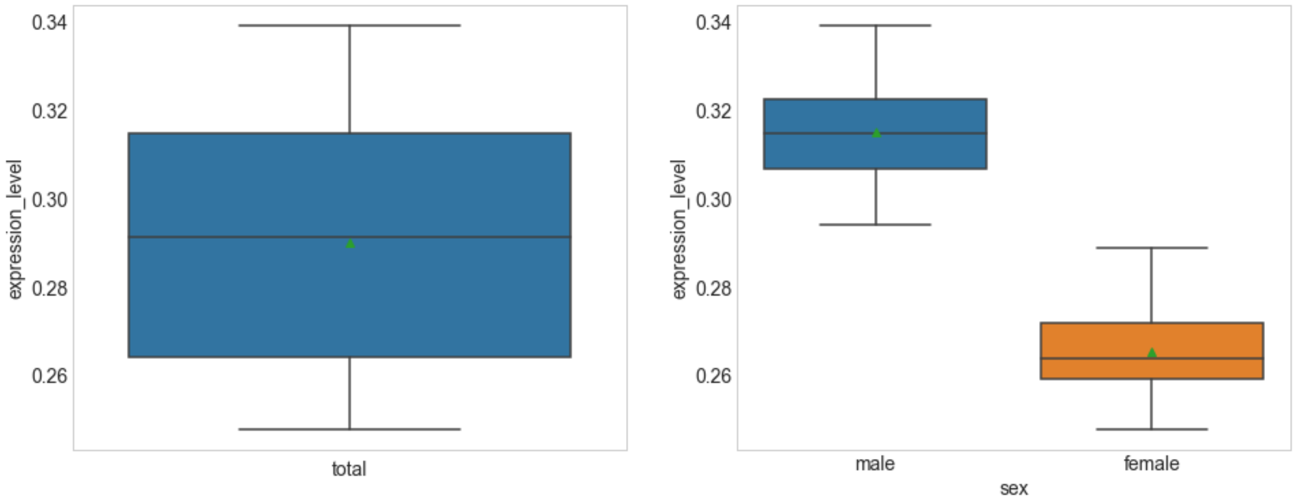
When we perform a one-way ANOVA we get:

```
mod = ols('expression_level ~ treatment',
                data=df2).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
             sum_sq    df         F    PR(>F)
treatment  0.001384   3.0  0.619269  0.606255
Residual   0.032770  44.0       NaN       NaN
```

So, even though we know that level "T2" in factor "treatment" does have an effect on the gene expression (increased expression level), it is not detected by the hypothesis test. This is because of the large variance caused by the factor "sex". We can see this clearly in the following boxplots showing expression levels for the factor "sex":

```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
ax = sns.boxplot(y="expression_level",hue="treatment",data=df2,showmeans=True)
ax.set_xlabel("total")
plt.subplot(1,2,2)
sns.boxplot(x="sex",y="expression_level",data=df2,showmeans=True)
plt.show()
```



We can see how most of the total variance can be explained by the factor "sex". The one-way ANOVA confirms this:

```
print
mod = ols('expression_level ~ sex',
                data=df2).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
            sum_sq    df           F        PR(>F)
sex       0.029360   1.0  281.771448  3.053987e-21
Residual  0.004793  46.0         NaN           NaN
```

We do obtain a very small p-value.

Let's compute the two-way ANOVA p-values:

```
mod = ols('expression_level ~ treatment + sex',
                data=df2).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
             sum_sq    df           F        PR(>F)
treatment  0.001384   3.0    5.816689  1.986050e-03
sex        0.029360   1.0  370.284849  9.454352e-23
Residual   0.003410  43.0         NaN           NaN
```

Now we can see that, as expected, both p-values are below $\alpha = 0.01$. This is because in two-way ANOVA there are two assignable sources of variation ("treatment" and "sex" in our example) and **this reduces error variation** thereby giving the test more power. The main difference with one-way ANOVA is the computation of the sum of squares of the error SSE which is computed in two-way ANOVA as

$$SSE = \sum_{k=1}^{r}\sum_{j=1}^{l_s}\sum_{i=1}^{l_t}(x_{ijk} - \overline{x}_{ij})^2,$$

where x_{ijk} is the kth observation of a total of r observations in level i for factor "treatment" and level j for factor "sex".

Let's again compare the one-way ANOVA on the factor "treatment" with the two-way ANOVA on "treatment" and "factor":

```
print "one-way ANOVA on treatment"
mod = ols('expression_level ~ treatment',
                data=df2).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
print
print "two-way ANOVA on treatment and sex"
mod = ols('expression_level ~ treatment + sex',
                data=df2).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
one-way ANOVA on treatment
             sum_sq    df         F    PR(>F)
treatment  0.001384   3.0  0.619269  0.606255
Residual   0.032770  44.0       NaN       NaN

two-way ANOVA on treatment and sex
             sum_sq    df           F        PR(>F)
treatment  0.001384   3.0    5.816689  1.986050e-03
sex        0.029360   1.0  370.284849  9.454352e-23
Residual   0.003410  43.0         NaN           NaN
```

We can see that $SS_{treatment}$ is the same in both tests. However SSE (the residuals) is much smaller for the two-way ANOVA F-statistic. This is because in the computation of SSE the variation caused by both factors is taken into account. Since SSE is in the denominator of the F-statistic, a smaller SSE means a larger F-statistic which in turn implies a smaller p-value.

As in two-way ANOVA there are two factors we have to take into account that there could be an **interaction** between these factors. If this is the case then we should be very careful with the conclusions we make about the individual factors. In two-way ANOVA with interaction the total variance is partitoned as

$$TSS = SS_{treatment} + SS_{sex} + SS_{treatment:sex} + SSE,$$

where $SS_{treatment:sex}$ is the variance caused by the (potential) interaction of the two factors "treatment" and "sex".

Let's check for an interaction effect in our dataset. We can do this in Python as follows:

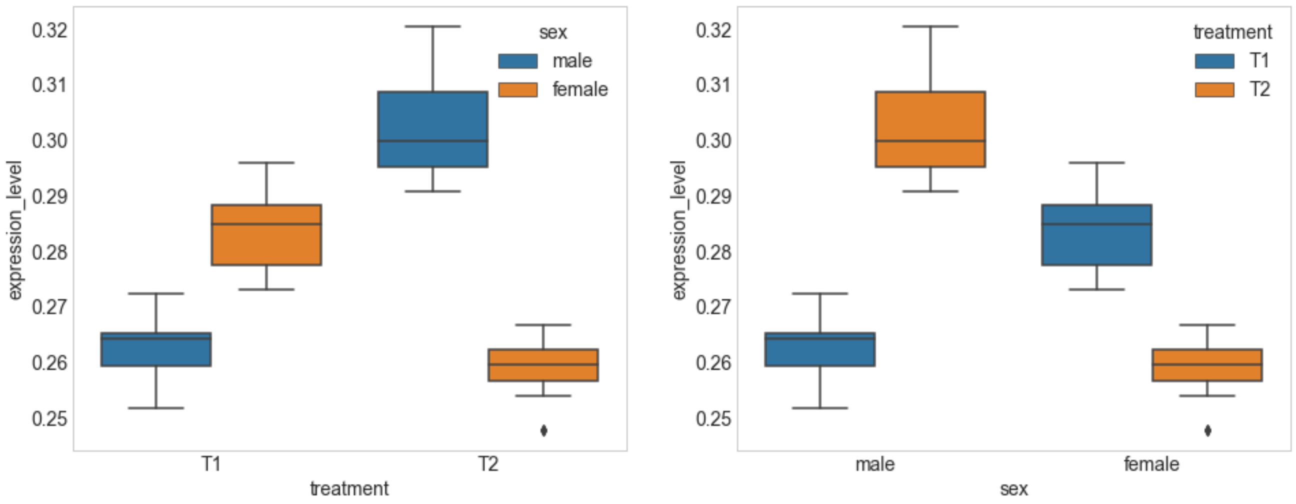
```
print "two-way ANOVA with interaction term:"
mod = ols('expression_level ~ treatment * sex',
                data=df2).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
two-way ANOVA with interaction term:
                 sum_sq    df           F        PR(>F)
treatment      0.001384   3.0    5.660833  2.495141e-03
sex            0.029360   1.0  360.363196  1.294282e-21
treatment:sex  0.000151   3.0    0.615943  6.087133e-01
Residual       0.003259  40.0         NaN           NaN
```

At $\alpha = 0.01$ we conclude that there is no significant interation between the two factors in our dataset.

Now consider the following example where there is a clear interaction. The factor "treatment" has just $l_t = 2$ levels: "T1" and "T2" and we have again a factor "sex" with also $l_s = 2$ levels: "female" and "male". For level "T1" the mean female mouse gene expression level is higher than for males. However, for level "T2" the mean female mouse gene expression level is lower than for males. Also, there is a difference between the means of levels "T1" and "T2" as well as there is a difference between the means of levels "male" and "female":

```
mu_T1_male = 0.26
mu_T1_female = 0.28
mu_T2_male = 0.30
mu_T2_female = 0.26
sigma = 0.008 #equal variance is assumed
r = 12
group_T1_male = np.random.normal(size=r,loc=mu_T1_male,scale=sigma)
group_T1_female = np.random.normal(size=r,loc=mu_T1_female,scale=sigma)
group_T2_male = np.random.normal(size=r,loc=mu_T2_male,scale=sigma)
group_T2_female = np.random.normal(size=r,loc=mu_T2_female,scale=sigma)
df3 = pd.DataFrame()
df3["treatment"] = ["T1"]*2*r + ["T2"]*2*r
df3["sex"] = ["male"]*r + ["female"]*r + ["male"]*r + ["female"]*r
df3["expression_level"] = np.concatenate((group_T1_male,group_T1_female,group_T2_male,group_T2_female))
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.boxplot(x="treatment",y="expression_level",hue="sex",data=df3)
plt.subplot(1,2,2)
sns.boxplot(x="sex",y="expression_level",hue="treatment",data=df3)
plt.show()
```



First we compute the one-way ANOVA for each factor:

```
print "one-way ANOVA for treatment:"
mod = ols('expression_level ~ treatment',
                data=df3).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
print
print "one-way ANOVA for sex:"
mod = ols('expression_level ~ sex',
                data=df3).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
one-way ANOVA for treatment:
             sum_sq    df         F    PR(>F)
treatment  0.000681   1.0  1.830809  0.182645
Residual   0.017122  46.0       NaN       NaN

one-way ANOVA for sex:
            sum_sq    df         F    PR(>F)
sex       0.001525   1.0  4.309095  0.043526
Residual  0.016278  46.0       NaN       NaN
```

At $\alpha = 0.01$ both null hypothesis are accepted. Now we perform a two-way ANOVA without taking the interaction into account:

```
print "two-way ANOVA without interaction term:"
mod = ols('expression_level ~ treatment + sex',
                data=df3).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
two-way ANOVA without interaction term:
             sum_sq    df         F    PR(>F)
treatment  0.000681   1.0  1.966113  0.167722
sex        0.001525   1.0  4.399597  0.041596
Residual   0.015597  45.0       NaN       NaN
```

Again both null hypothesis are accepted. This is incorrect as we know that both "treatment" and "sex" have a significant effect on the gene expression level. The problem is that both factors even each other out as we will see in the next plots. The two-way ANOVA that takes an interaction between the factors into account gives:

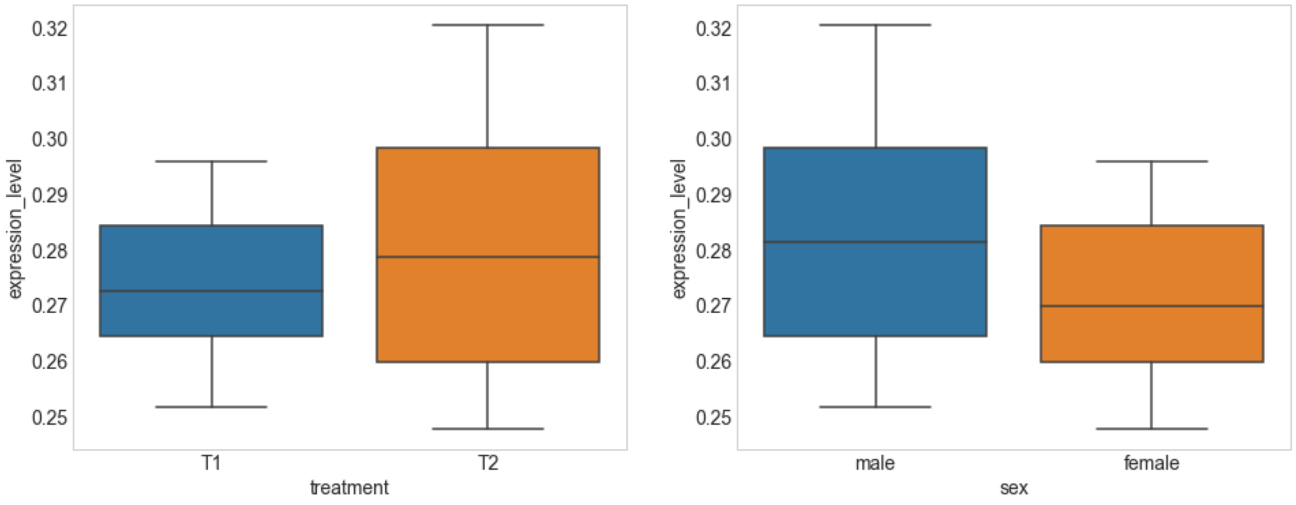
```
print "two-way ANOVA with interaction term:"
mod = ols('expression_level ~ treatment * sex',
                data=df3).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print aov_table
```

```
two-way ANOVA with interaction term:
                 sum_sq    df           F        PR(>F)
treatment      0.000681   1.0   11.335033  1.587863e-03
sex            0.001525   1.0   25.364546  8.551073e-06
treatment:sex  0.012952   1.0  215.433911  1.457354e-18
Residual       0.002645  44.0         NaN           NaN
```

At $\alpha = 0.01$ we see that this time both null hypothesis are correctly rejected. We also see a third p-value (for the interaction "treatment:sex" that shows that there is indeed a significant interaction between the factors.

To better illustrate the effect of the interaction we can plot the distribution of expression levels between the levels for each factor:

```
#because of interaction no difference is observed
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.boxplot(x="treatment",y="expression_level",data=df3)
plt.subplot(1,2,2)
sns.boxplot(x="sex",y="expression_level",data=df3)
plt.show()
```



Notice how because of the interaction we observe no clear difference between the means of the levels in a factor.

Power analysis

We have seen that the power of a hypothesis test is defined as the probability of correctly rejecting the null hypothesis in case the alternative hypothesis is true. In power analysis the power is defined as $1-\beta$ where β is the probability of accepting the null hypothesis even though the alternative hypothesis is true.

When preparing a statistical analysis it is good practice to estimate the sample size required to find the effect of interest if it exists. The "effect of interest" is known as the **effect size**.

Suppose we want to test if there is a difference between the mean expression level of two genes (geneA and geneB). We want to design a statistical experiment that is able to detect (if it exists) a minimum 0.02 in expression difference between the expression level of both genes. This "minimum 0.02 difference" is the **minimum** effect size we want to be able to detect (again, if it exists).

Let's illustrate this. We know that the power of a two sample t-test (and all other hypothesis tests) depends on the variance of the (normal) population from which the sample(s) are drawn. We fix the sample size n, the effect size (noted as variable `diff` in the code below) and the significance level α. Next we vary the variance (or standard deviation σ) of the population from which the samples are drawn and compute the power of a two sample t-test using 1000 samples drawn from each population:

```
num_samples = 1000
alpha = 0.01
mu_geneA = 0.26
#different population variances
n = 9
diff = 0.02
mu_geneB = mu_geneA + diff
for sigma in [0.01,0.015,0.02,0.025,0.03,0.04,0.05]:
    power = 0
    for i in range(num_samples):
        geneA = np.random.normal(size=n,loc=mu_geneA,scale=sigma)
        geneB = np.random.normal(size=n,loc=mu_geneB,scale=sigma)
        pvalue = stats.ttest_ind(geneA,geneB).pvalue
        if pvalue < alpha:
            power += 1
    print "Power for sample size %i, mean difference %f and population sigma %f: %f" %
            (n,diff,sigma,power/(float(num_samples)))
```

```
Power for sample size 9, mean difference 0.020000 and population sigma 0.010000: 0.883000
Power for sample size 9, mean difference 0.020000 and population sigma 0.015000: 0.500000
Power for sample size 9, mean difference 0.020000 and population sigma 0.020000: 0.225000
Power for sample size 9, mean difference 0.020000 and population sigma 0.025000: 0.149000
Power for sample size 9, mean difference 0.020000 and population sigma 0.030000: 0.091000
Power for sample size 9, mean difference 0.020000 and population sigma 0.040000: 0.060000
Power for sample size 9, mean difference 0.020000 and population sigma 0.050000: 0.028000
```

We see that the power of the two sample t-test deceases as the variance of the population increases. We also know that the power of a two sample t-test (and all other hypothesis tests) depends on the sample size n. We fix the population variance, the effect size and the significance level α. Next we vary the samples size n and compute the power of a two sample t-test using 1000 samples drawn for different values for n:

```
#different sample sizes
sigma = 0.01
diff = 0.02
mu_geneB = mu_geneA + diff
for n in range(2,15):
    power = 0
    for i in range(num_samples):
        geneA = np.random.normal(size=n,loc=mu_geneA,scale=sigma)
        geneB = np.random.normal(size=n,loc=mu_geneB,scale=sigma)
        pvalue = stats.ttest_ind(geneA,geneB).pvalue
        if pvalue < alpha:
            power += 1
    print "Power for sample size %i, mean difference %f and population sigma %f: %f" %
            (n,diff,sigma,power/(float(num_samples)))
```

```
Power for sample size 2, mean difference 0.020000 and population sigma 0.010000: 0.050000
Power for sample size 3, mean difference 0.020000 and population sigma 0.010000: 0.168000
Power for sample size 4, mean difference 0.020000 and population sigma 0.010000: 0.330000
Power for sample size 5, mean difference 0.020000 and population sigma 0.010000: 0.480000
Power for sample size 6, mean difference 0.020000 and population sigma 0.010000: 0.630000
Power for sample size 7, mean difference 0.020000 and population sigma 0.010000: 0.744000
Power for sample size 8, mean difference 0.020000 and population sigma 0.010000: 0.836000
Power for sample size 9, mean difference 0.020000 and population sigma 0.010000: 0.889000
Power for sample size 10, mean difference 0.020000 and population sigma 0.010000: 0.923000
Power for sample size 11, mean difference 0.020000 and population sigma 0.010000: 0.949000
Power for sample size 12, mean difference 0.020000 and population sigma 0.010000: 0.967000
Power for sample size 13, mean difference 0.020000 and population sigma 0.010000: 0.978000
Power for sample size 14, mean difference 0.020000 and population sigma 0.010000: 0.991000
```

We see that the power of the two sample t-test increases as the sample size increases. Finally the power of a two sample t-test (and all other hypothesis tests) also depends on the minimum effect size we hope to detect if it exists. We fix the population variance, the sample size and the significance level α. Next we vary the effect size (variable "diff" in the code below) and compute the power of a two sample t-test using 1000 samples drawn for different effect sizes:

```
#different effect sizes
sigma = 0.01
n= 9
for diff in [0.001,0.005,0.01,0.02,0.03,0.04]:
    mu_geneB = mu_geneA + diff
    power = 0
    for i in range(num_samples):
        geneA = np.random.normal(size=n,loc=mu_geneA,scale=sigma)
        geneB = np.random.normal(size=n,loc=mu_geneB,scale=sigma)
        pvalue = stats.ttest_ind(geneA,geneB).pvalue
        if pvalue < alpha:
            power += 1
    print "Power for sample size %i, mean difference %f and population sigma %f: %f" %
            (n,diff,sigma,power/(float(num_samples)))
```

```
Power for sample size 9, mean difference 0.001000 and population sigma 0.010000: 0.009000
Power for sample size 9, mean difference 0.005000 and population sigma 0.010000: 0.054000
Power for sample size 9, mean difference 0.010000 and population sigma 0.010000: 0.253000
Power for sample size 9, mean difference 0.020000 and population sigma 0.010000: 0.880000
Power for sample size 9, mean difference 0.030000 and population sigma 0.010000: 1.000000
Power for sample size 9, mean difference 0.040000 and population sigma 0.010000: 1.000000
```

We see that the power of the two sample t-test increases as the effect size increases, which makes sense.

So the power of a hypothesis test depends on these four numbers:

- the effect size
- the population variance
- the sample size
- the significance level

This means that when we fix the expected power of a test, the minimum effect size, the population variance and the significance level we can compute the minimum sample size n required to achieve this expected power.

In Python we can compute the minimum sample size required for a certain minimum statistical power as follows:

```
import statsmodels.stats.power as smp
sigma = 0.01
diff = 0.02
for power in [0.99,0.95,0.9,0.8]:
    print "Minimum sample size required for power = %f: %f" %
        (power,smp.TTestIndPower().solve_power(diff/sigma, power=power, alpha=alpha, alternative='two-sided'))
```

```
Minimum sample size required for power = 0.990000: 13.805334
Minimum sample size required for power = 0.950000: 10.710971
Minimum sample size required for power = 0.900000: 9.251530
Minimum sample size required for power = 0.800000: 7.660540
```

In this code the population variance is taken into account by dividing the effect size (variable `diff`) by the standard deviation.

According to the power analysis, if we expect a minimum power of 95% then we need to draw at least $n = 10.710971 = 11$ observations in each sample. We can verify this with the code we used above for varying the sample size and computing the power:

```
#different sample sizes
mu_geneB = mu_geneA + diff
n = 11
power = 0
for i in range(num_samples):
    geneA = np.random.normal(size=n,loc=mu_geneA,scale=sigma)
    geneB = np.random.normal(size=n,loc=mu_geneB,scale=sigma)
    pvalue = stats.ttest_ind(geneA,geneB).pvalue
    if pvalue < alpha:
        power += 1
print "Power for sample size %i, mean difference %f and population sigma %f: %f" %
    (n,diff,sigma,power/(float(num_samples)))
```

```
Power for sample size 11, mean difference 0.020000 and population sigma 0.010000: 0.959000
```

We can see that our estimations are indeed correct.

One problem with a power analysis is that the variance of the population needs to be known, while in most cases this information is not available. Typical practice is to find an estimate for the variance in the scientific literature or perform a "pilot" experiment with small sample size to estimate the population variance from this small sample.

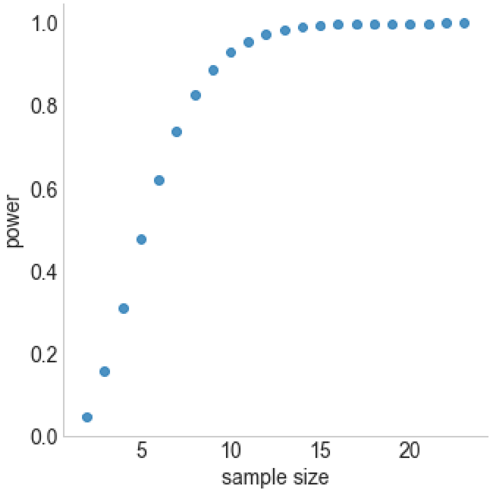
In the design of a statistical study it is important to consider the trade-off between the power of the hypothesis test and the sample size of the study. We can see this by computing the power of our two sample t-test for different sample sizes:

```
df = []
for n in range(2,24):
    power = smp.TTestIndPower().solve_power(diff/sigma, nobs1=n, alpha=alpha, alternative='two-sided')
    print "Power for sample size %i: %f" % (n,power)
    df.append([n,power])
df = pd.DataFrame(df,columns=["sample size","power"])
```

```
Power for sample size 2: 0.048628
Power for sample size 3: 0.156856
Power for sample size 4: 0.313033
Power for sample size 5: 0.476935
Power for sample size 6: 0.622611
Power for sample size 7: 0.739367
Power for sample size 8: 0.826419
Power for sample size 9: 0.887894
Power for sample size 10: 0.929485
Power for sample size 11: 0.956656
Power for sample size 12: 0.973892
Power for sample size 13: 0.984556
Power for sample size 14: 0.991011
Power for sample size 15: 0.994844
Power for sample size 16: 0.997082
Power for sample size 17: 0.998368
Power for sample size 18: 0.999098
Power for sample size 19: 0.999506
Power for sample size 20: 0.999732
Power for sample size 21: 0.999856
Power for sample size 22: 0.999923
Power for sample size 23: 0.999959
```

If we plot the power against the sample size we get:

```
sns.lmplot(x="sample size",y="power",data=df,fit_reg=False)
plt.show()
```



This plot shows how for small sample sizes adding one more sample significantly increases the power of our test. From the moment we have ten observations in each level we see that the added power decreases exponentially.