ANOVA is used to

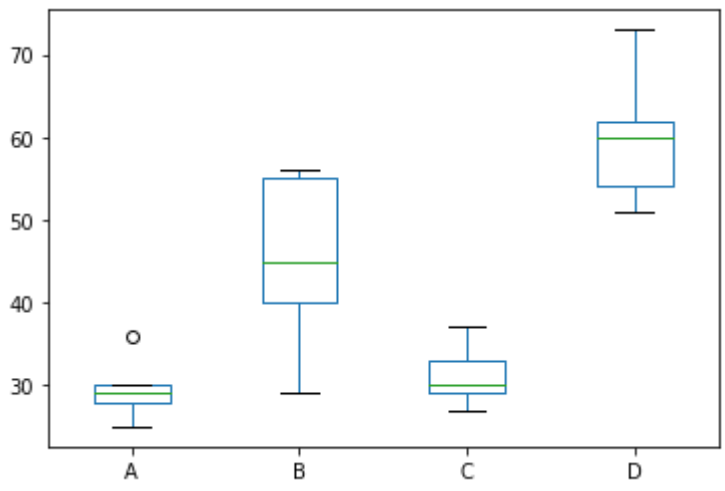
- compare the means of more than 2 groups (t-test can be used to compare 2 groups)
- groups mean differences inferred by analyzing variances.

Main types: One-way (one factor) and two-way (two factors) ANOVA (factor is an independent variable).

In ANOVA, group, factors, and independent variables are similar terms.

```
# load packages
import pandas as pd
import numpy as np
# load data file
d = pd.read_csv("https://reneshbedre.github.io/assets/posts/anova/onewayanova.txt", sep="\t")
# generate a boxplot to see the data distribution by treatments. Using boxplot, we can easily detect the differences
# between different treatments
d.boxplot(column=['A', 'B', 'C', 'D'], grid=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f696cff6278>
```



In ANOVA, we test hypotheses.

- Null hypotheses: Groups means are equal (no variation in means of groups)
- Alternative hypotheses: At least, one group mean is different from other groups.

There are several assumptions:

- Residuals (experimental error) are normally distributed (Shapiro-Wilks Test)
- Homogeneity of variances (variances are equal between treatment groups) (Levene or Bartlett Test)
- Observations are sampled independently from each other

The ANOVA process is as follows:

- Check sample sizes: equal number of observation in each group
- Calculate Mean Square for each group (MS) (SS of group/level-1); level-1 is a degree of freedom (df) for a group
- Calculate Mean Square error (MSE) (SS error/df of residuals)
- Calculate F-value (MS of group/MSE)

```
# load packages
import scipy.stats as stats
# stats f_oneway functions takes the groups as input and returns F and P-value
fvalue, pvalue = stats.f_oneway(d['A'], d['B'], d['C'], d['D'])
print(fvalue, pvalue)
# 17.492810457516338 2.639241146210922e-05

# get ANOVA table as R like output
import statsmodels.api as sm
from statsmodels.formula.api import ols
# reshape the d dataframe suitable for statsmodels package
d_melt = pd.melt(d.reset_index(), id_vars=['index'], value_vars=['A', 'B', 'C', 'D'])
# replace column names
d_melt.columns = ['index', 'treatments', 'value']
# Ordinary Least Squares (OLS) model
model = ols('value ~ C(treatments)', data=d_melt).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

```
17.492810457516338 2.639241146210922e-05
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
  import pandas.util.testing as tm
```

The p-value from ANOVA analysis is significant ($p < 0.05$), and therefore, we conclude that there are significant differences among treatments. If you have unbalanced (unequal sample size for each group) data, you can perform similar steps as described for one-way ANOVA with balanced design (equal sample size for each group).

From ANOVA analysis, we know that treatment differences are statistically significant, but ANOVA does not tell which treatments are significantly different from each other. To know the pairs of significant different treatments, we will perform multiple pairwise comparison (Post-hoc comparison) analysis using Tukey HSD test.

```
# load packages
!pip install pingouin
from pingouin import pairwise_tukey
# perform multiple pairwise comparison (Tukey HSD)
# for unbalanced (unequal sample size) data, pairwise_tukey uses Tukey-Kramer test
m_comp = pairwise_tukey(data=d_melt, dv='value', between='treatments')
print(m_comp)
```

```
Collecting pingouin
  Downloading https://files.pythonhosted.org/packages/fc/8f/8204d5e8365b687a77e44e4fd9dd3b0c6b735b4aa63404a65d32d09dd
     |████████████████████████████████| 225kB 3.5MB/s
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.6/dist-packages (from pingouin) (1.18.5)
Requirement already satisfied: scipy>=1.3 in /usr/local/lib/python3.6/dist-packages (from pingouin) (1.4.1)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.6/dist-packages (from pingouin) (1.0.5)
Requirement already satisfied: matplotlib>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from pingouin) (3.2.2)
Requirement already satisfied: seaborn>=0.9.0 in /usr/local/lib/python3.6/dist-packages (from pingouin) (0.10.1)
Requirement already satisfied: statsmodels>=0.10.0 in /usr/local/lib/python3.6/dist-packages (from pingouin) (0.10.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (from pingouin) (0.22.2.post1)
Collecting pandas_flavor>=0.1.2
  Downloading https://files.pythonhosted.org/packages/9a/57/7fbcff4c0961ed190ac5fcb0bd8194152ee1ee6487edf64fdbae16e2b
Collecting outdated
  Downloading https://files.pythonhosted.org/packages/86/70/2f166266438a30e94140f00c99c0eac1c45807981052a1d4c123660e1
Requirement already satisfied: tabulate in /usr/local/lib/python3.6/dist-packages (from pingouin) (0.8.7)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.24->p
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.24->pingouin) (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.0.2->p
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=3.0.2->pingou
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from statsmodels>=0.10.0->ping
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn->pingouin) (
Requirement already satisfied: xarray in /usr/local/lib/python3.6/dist-packages (from pandas_flavor>=0.1.2->pingouin)
Collecting littleutils
  Downloading https://files.pythonhosted.org/packages/4e/b1/bb4e06f010947d67349f863b6a2ad71577f85590180a935f60543f622
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from outdated->pingouin) (2.23.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.6.1->panda
Requirement already satisfied: setuptools>=41.2 in /usr/local/lib/python3.6/dist-packages (from xarray->pandas_flavor
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (frc
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->outdated->pingc
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->outdated-
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->outdated->
Building wheels for collected packages: pingouin, outdated, littleutils
  Building wheel for pingouin (setup.py) ... done
  Created wheel for pingouin: filename=pingouin-0.3.7-cp36-none-any.whl size=217260 sha256=77b8553df4b4f8cc6aa3a2b2b9
  Stored in directory: /root/.cache/pip/wheels/02/92/32/0ed9ac4a9407227f3f070170a26d05f07d6f2a8a68989a8ac3
  Building wheel for outdated (setup.py) ... done
  Created wheel for outdated: filename=outdated-0.2.0-cp36-none-any.whl size=4961 sha256=7725b01c9a9cc7f743182fe8f4b9
  Stored in directory: /root/.cache/pip/wheels/fd/7c/ef/814f514d31197310872b5abf353feb8fef9d67ee658e1e7e39
  Building wheel for littleutils (setup.py) ... done
  Created wheel for littleutils: filename=littleutils-0.2.2-cp36-none-any.whl size=7051 sha256=7a44aec6828d1f9e03eee6
  Stored in directory: /root/.cache/pip/wheels/53/16/9f/ac67d15c40243754fd73f620e1b9b6dedc20492ecc19a2bae1
Successfully built pingouin outdated littleutils
Installing collected packages: pandas-flavor, littleutils, outdated, pingouin
Successfully installed littleutils-0.2.2 outdated-0.2.0 pandas-flavor-0.2.0 pingouin-0.3.7
   A  B  mean(A)  mean(B)  diff        se         T   p-tukey    hedges
0  A  B     29.6     45.0 -15.4  4.790616 -3.214618  0.010718 -1.836351
1  A  C     29.6     31.2  -1.6  4.790616 -0.333986  0.900000 -0.190790
2  A  D     29.6     60.0 -30.4  4.790616 -6.345739  0.001000 -3.625005
3  B  C     45.0     31.2  13.8  4.790616  2.880632  0.027410  1.645561
4  B  D     45.0     60.0 -15.0  4.790616 -3.131121  0.013679 -1.788654
5  C  D     31.2     60.0 -28.8  4.790616 -6.011753  0.001000 -3.434215
```

The Shapiro-Wilk test can be used to check the normal distribution of residuals

- Null hypothesis: data is drawn from normal distribution.
- Alternate hypothesis: data is not drawn for normal distribution.

```
# load packages
import scipy.stats as stats
w, pvalue = stats.shapiro(model.resid)
print(w, pvalue)
```

```
# 0.9685019850730896 0.7229772806167603
```

```
0.9685019850730896 0.7229772806167603
```

```
# load packages
import scipy.stats as stats
w, pvalue = stats.bartlett(d['A'], d['B'], d['C'], d['D'])
print(w, pvalue)
#5.687843565012841 0.1278253399753447
```

```
5.687843565012841 0.1278253399753447
```

As the p-value (0.12) is non significant, we fail to reject null hypothesis and conclude that treatments have equal variances. The Levene test can be used to check the homogeneity of variances when the data is not drawn from normal distribution.

In this example, there are two factors (independent variables) i.e. genotypes and yield in years. Genotypes and years has five and three levels respectively.

For this experimental design, there are two factors to evaluate, and therefore, two-way ANOVA method is suitable for analysis. Here, using two-way ANOVA, we can simultaneously evaluate how type of genotype and years affects the yields of plants. If you apply one-way ANOVA here, you can evaluate only one factor at a time.

From two-way ANOVA, we can tests three hypotheses

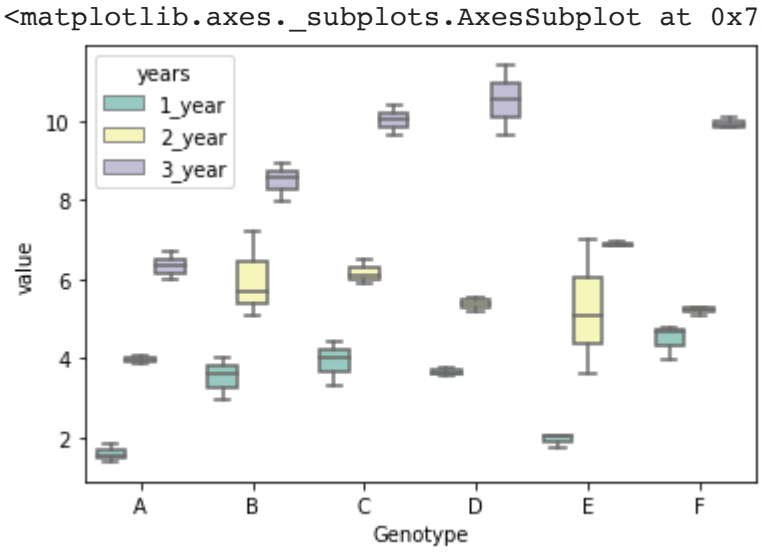
- 1) effect of genotype on yield
- 2) effect of time (years) on yield, and
- 3) effect of genotype and time (years) interactions on yield

```
# load packages
import pandas as pd
import seaborn as sns
# load data file
d = pd.read_csv("https://reneshbedre.github.io/assets/posts/anova/twowayanova.txt", sep="\t")
# reshape the d dataframe suitable for statsmodels package
# you do not need to reshape if your data is already in stacked format. Compare d and d_melt tables for detail
# understanding
d_melt = pd.melt(d, id_vars=['Genotype'], value_vars=['1_year', '2_year', '3_year'])
# replace column names
d_melt.columns = ['Genotype', 'years', 'value']
d_melt.head()
```

	Genotype	years	value
0	A	1_year	1.53
1	A	1_year	1.83
2	A	1_year	1.38
3	B	1_year	3.60
4	B	1_year	2.94

```
# generate a boxplot to see the data distribution by genotypes and years. Using boxplot, we can easily detect the
# differences between different groups
sns.boxplot(x="Genotype", y="value", hue="years", data=d_melt, palette="Set3")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f69554f2f98>
```




```
# load packages
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
# Ordinary Least Squares (OLS) model
# C(Genotype):C(years) represent interaction term
model = ols('value ~ C(Genotype) + C(years) + C(Genotype):C(years)', data=d_melt).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

	sum_sq	df	F	PR(>F)
C(Genotype)	58.551733	5.0	32.748581	1.931655e-12
C(years)	278.925633	2.0	390.014868	4.006243e-25
C(Genotype):C(years)	17.122967	10.0	4.788525	2.230094e-04
Residual	12.873000	36.0	NaN	NaN

The p-value obtained from ANOVA analysis for genotype, years, and interaction are statistically significant ($p < 0.05$).

We conclude that type of genotype significantly affects the yield outcome, time (years) significantly affects the yield outcome, and interaction of both genotype and time (years) significantly affects the yield outcome.

If you have unbalanced (unequal sample size for each group) data, you can perform similar steps as described for two-way ANOVA with the balanced design but set `typ=3`. Type 3 sums of squares (SS) is recommended for an unbalanced design for multifactorial ANOVA.

We know that genotype and time (years) differences are statistically significant, but ANOVA does not tell which genotype and time (years) are significantly different from each other. To know the pairs of significant different genotype and time (years), perform multiple pairwise comparison (post-hoc comparison) analysis using Tukey HSD test.

Similar to one-way ANOVA, you can use Levene and Shapiro-Wilk test to validate the assumptions for homogeneity of variances and normal distribution of residuals.