Bentley University MA255 in Python

Content extracted from the How to Data Website

This PDF generated on 05 November 2022

Contents

MA255 is an undergraduate statistics course at Bentley University on the Design of Experiments. The description from the course catalog can be found here.

The course covers various experimental designs including factorial and fractional factorial designs, interaction among factors, and applications in management (including cost savings and policy making) as well as in marketing.

The sequence of topics below is not necessarily the final version; this topic page is under construction.

Summarizing data and exploratory analysis

Experiments with one treatment factor

Analyzing data from a larger design

Content last modified on 07 December 2021.

How to summarize a column

Description

When provided with a dataset in which you want to focus on one column, how would you compute descriptive statistics for that column?

Related task:

- How to compute summary statistics
- How to summarize and compare data by groups

Solution in Python

The solution below uses an example dataset about the teeth of 10 guinea pigs at three Vitamin C dosage levels (in mg) with two delivery methods (orange juice vs. ascorbic acid). (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('ToothGrowth')
```

Let us consider qualitative and quantitative variables separately.

Consider the qualitative column "supp" in the dataset (which type of supplement the animal received). To count the distribution of each categorical value, use `value_counts()`:

```
df['supp'].value_counts()
# Or use df['supp'].value_counts(normalize = True) for proportions instead.
```

```
VC    30
OJ    30
Name: supp, dtype: int64
```

The output says that there are 30 observations under each of the two levels, Orange Juice and Ascorbic Acid.

If you wish to jointly summarize two categorical columns, provide both to `value_counts()`:

```
df[['supp','dose']].value_counts()
```

```
supp  dose
OJ    0.5     10
      1.0     10
      2.0     10
VC    0.5     10
      1.0     10
      2.0     10
dtype: int64
```

This informs us that there are 10 observations for each of the combinations.

Now consider the quantitative column `len` in the dataset (the length of the animal's tooth). We can compute summary statistics for it just as we can for a whole dataframe (as we cover in how to compute summary statistics).

```
df['len'].describe() # Summary statistics
```

```
count    60.000000
mean     18.813333
std       7.649315
min       4.200000
25%      13.075000
50%      19.250000
75%      25.275000
max      33.900000
Name: len, dtype: float64
```

The individual functions for mean, standard deviation, etc. covered under "how to compute summary statistics" apply to individual columns as well. For example, we can compute quantiles:

```
df['len'].quantile([0.25,0.5,0.75])   # These chosen values give quartiles.
```

```
0.25    13.075
0.50    19.250
0.75    25.275
Name: len, dtype: float64
```

Content last modified on 24 October 2021.

See a problem? Tell us or edit the source.

How to compute summary statistics

Description

The phrase "summary statistics" usually refers to a common set of simple computations that can be done about any dataset, including mean, median, variance, and some of the others shown below.

Related tasks:

- How to summarize a column
- How to summarize and compare data by groups

Solution in Python

We first load a famous dataset, Fisher's irises, just to have some example data to use in the code that follows. (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data( 'iris' )
```

How big is the dataset? The output shows number of rows then number of columns.

```
df.shape
```

```
(150, 5)
```

What are the columns and their data types? Are any values missing?

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Sepal.Length  150 non-null    float64
 1   Sepal.Width   150 non-null    float64
 2   Petal.Length  150 non-null    float64
 3   Petal.Width   150 non-null    float64
 4   Species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

What do the first few rows look like?

```
df.head()  # Default is 5, but you can do df.head(20) or any number.
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

The easiest way to get summary statistics for a pandas DataFrame is with the `describe` function.

```
df.describe()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

The individual statistics are the row headings, and the numeric columns from the original dataset are listed across the top.

We can also compute these statistics (and others) one at a time for any given set of data points. Here, we let `xs` be one column from the above DataFrame, but you could use any NumPy array or pandas DataFrame instead.

```
xs = df['Sepal.Length']

import numpy as np

np.mean( xs )           # mean, or average, or center of mass
np.median( xs )         # 50th percentile
np.percentile( xs, 25 ) # compute any percentile, such as the 25th
np.var( xs )            # variance
np.std( xs )            # standard deviation, the square root of the variance
np.sort( xs )           # data in increasing order
np.sum( xs )            # sum, or total
```

Content last modified on 26 July 2021.

See a problem? Tell us or edit the source.

How to summarize and compare data by groups

Description

When given a set of data that has different treatment conditions and an outcome variable, we need to perform some exploratory data analysis. How would you quantitatively compare the treatment conditions with regards to the outcome variable?

Related tasks:

- How to compute summary statistics

Solution in Python

The solution below uses an example dataset about the teeth of 10 guinea pigs at three Vitamin C dosage levels (in mg) with two delivery methods (orange juice vs. ascorbic acid). (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('ToothGrowth')
```

To obtain the descriptive statistics of the quantitative column (`len` for length of teeth) based on the treatment levels (`supp`), we can combine the `groupby` and `describe` functions.

```
df.groupby('supp')['len'].describe()
```

supp	count	mean	std	min	25%	50%	75%	max
OJ	30.0	20.663333	6.605561	8.2	15.525	22.7	25.725	30.9
VC	30.0	16.963333	8.266029	4.2	11.200	16.5	23.100	33.9

To choose which statistics you want to see, you could use the `agg` function and list the statistics you want.

```
df.groupby('supp')['len'].agg(['min','median','mean','max','std','count'])
```

supp	min	median	mean	max	std	count
OJ	8.2	22.7	20.663333	30.9	6.605561	30
VC	4.2	16.5	16.963333	33.9	8.266029	30

If your focus is on just one statistic, you can often use its name in place of `agg`, as shown below, using the `quantile` function.

```
df.groupby('supp')['len'].quantile([0.25,0.5,0.75]) # Quartiles - default is median, i.e. 0.5
```

```
supp
OJ    0.25    15.525
      0.50    22.700
      0.75    25.725
VC    0.25    11.200
      0.50    16.500
      0.75    23.100
Name: len, dtype: float64
```

In this example, we grouped by just one category (`supp`), but the `groupby` function accepts a list of columns if you need to create subcategories, etc.

Content last modified on 24 October 2021.

See a problem? Tell us or edit the source.

How to create bivariate plots to compare groups

Description

Suppose we have a dataset with different treatment conditions and an outcome variable, and we want to perform exploratory data analysis. How would we visually compare the treatment conditions with regards to the outcome variable?

Related tasks:

- How to create basic plots (on website)
- How to add details to a plot (on website)
- How to create a histogram (on website)
- How to create a box (and whisker) plot (on website)
- How to change axes, ticks, and scale in a plot (on website)
- How to plot interaction effects of treatments

Solution in Python

The solution below uses an example dataset about the teeth of 10 guinea pigs at three Vitamin C dosage levels (in mg) with two delivery methods (orange juice vs. ascorbic acid). (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('ToothGrowth')
```

If you wish to understand the distribution of a numeric variable (here "len") compared across different values of a categorical variable (here "supp"), you can construct a bivariate histogram. We use Seaborn and Matplotlib to do so.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.displot(df, x="len", col="supp", stat="density")
plt.show()
```

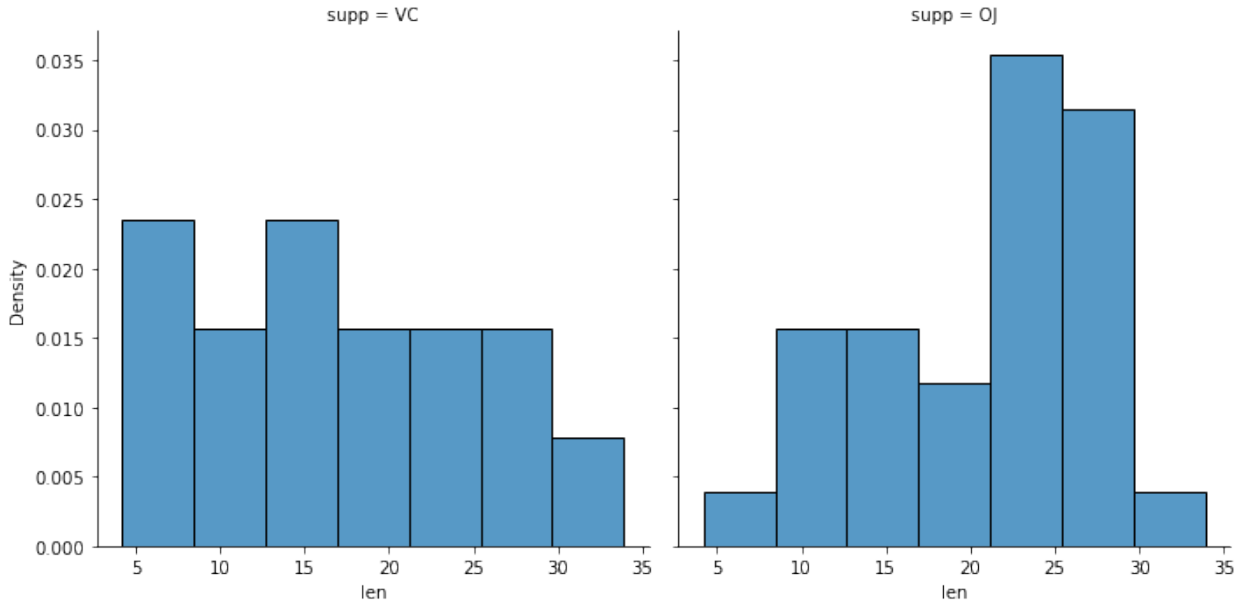

Figure 1: png

To visualize the same information summarized using quartiles only, you can construct a bivariate box plot.

```
sns.boxplot(x="supp", y="len", data = df, order = ['OJ','VC'])
plt.show()
```

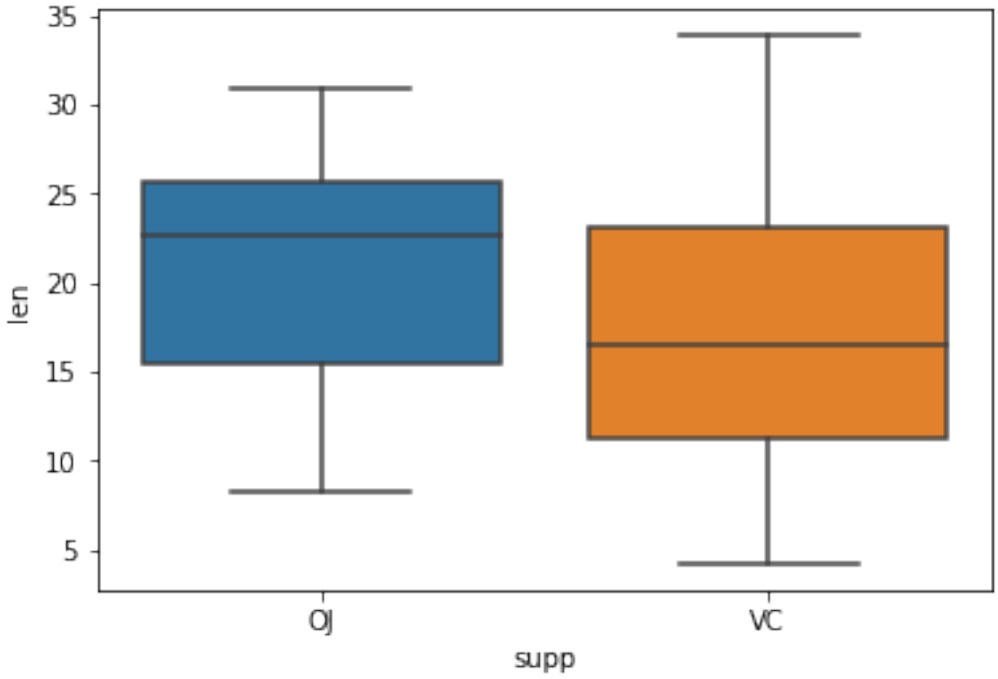

Figure 2: png

Even more simply, we may wish to plot just the means and 95% confidence intervals around the mean for the quantitative variable, for each of the values of the categorical variable. We do so with a point plot.

```
sns.pointplot(x = 'supp', y = 'len', data = df,
              ci = 95,         # Which confidence interval?  Here 95%.
              capsize = 0.1)  # Size of "cap" drawn on each confidence interval.
plt.show()
```

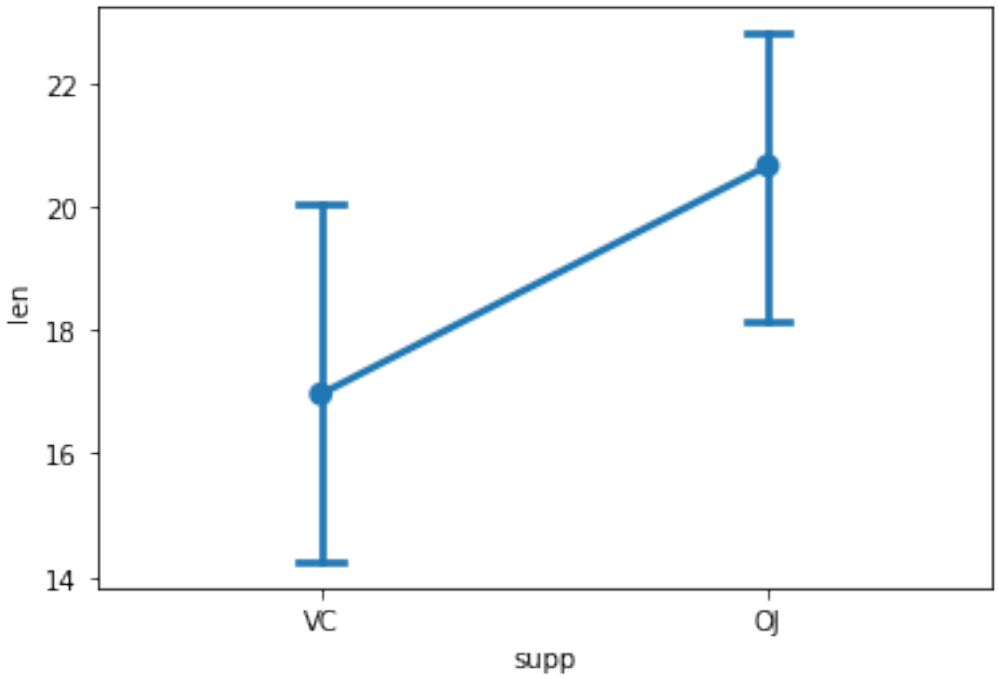

Figure 3: png

Content last modified on 26 July 2021.

See a problem? Tell us or edit the source.

How to check the assumptions of a linear model

Description

If you plan to use a linear model to describe some data, it's important to check if it satisfies the assumptions for linear regression. How can we do that?

Solution in Python

When performing a linear regression, the following assumptions should be checked.

1. We have two or more columns of numerical data of the same length.

The solution below uses an example dataset about car design and fuel consumption from a 1974 Motor Trend magazine. (See how to quickly load some sample data (on website).) We can see that our columns all have the same length.

```
from rdatasets import data
df = data('mtcars')
df = df[['mpg','cyl','wt']]  # Select the 3 variables we're interested in
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   mpg     32 non-null     float64
 1   cyl     32 non-null     int64
 2   wt      32 non-null     float64
dtypes: float64(2), int64(1)
memory usage: 896.0 bytes
```

2. Scatter plots we've made suggest a linear relationship.

Scatterplots are covererd in how to create basic plots (on website), but after making the model, we can also examine the residuals.

So let's make the model. Our predictors will be the number of cylinders and the weight of the car and the response will be miles per gallon. (See also how to fit a linear model to two columns of data (on website).)

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()

predictors = df[['cyl','wt']]
response = df['mpg']
model.fit( X=predictors, y=response )

predictions = model.predict(predictors)
```

We test for linearity with residual plots. We show just one residual plot here; you should make one for each predictor. Seaborn has a function for just this purpose. (See also how to compute the residuals of a linear model (on website).)

```
import seaborn as sns
import matplotlib.pyplot as plt
# The "lowess" parameter adds a smooth line through the data:
sns.residplot(x = df['wt'], y = response, data=df, lowess=True)
plt.xlabel("Weight")
plt.title('Miles per gallon')
plt.show()
```

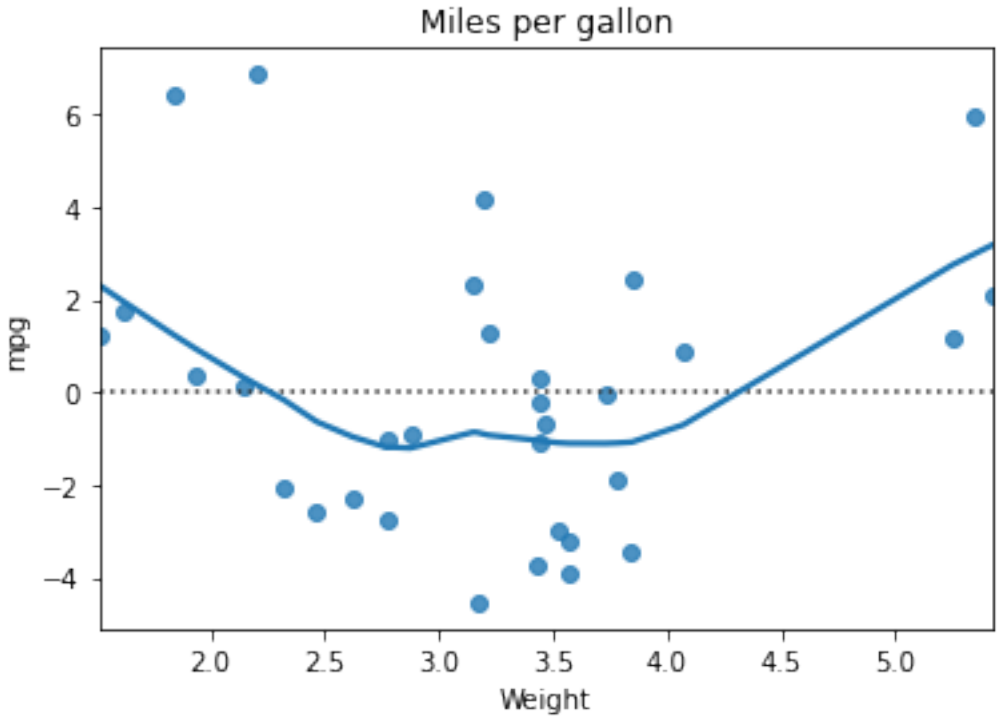

Figure 4: png

3. After making the model, the residuals seem normally distributed.

We can check this by constructing a QQ-plot, which compares the distribution of the residuals to a normal distribution. Here we use SciPy, but there are other methods; see how to create a QQ-plot (on website).

```
from scipy import stats
residuals = response - predictions  # Compute the residuals
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("Normal Q-Q Plot")
plt.show()
```

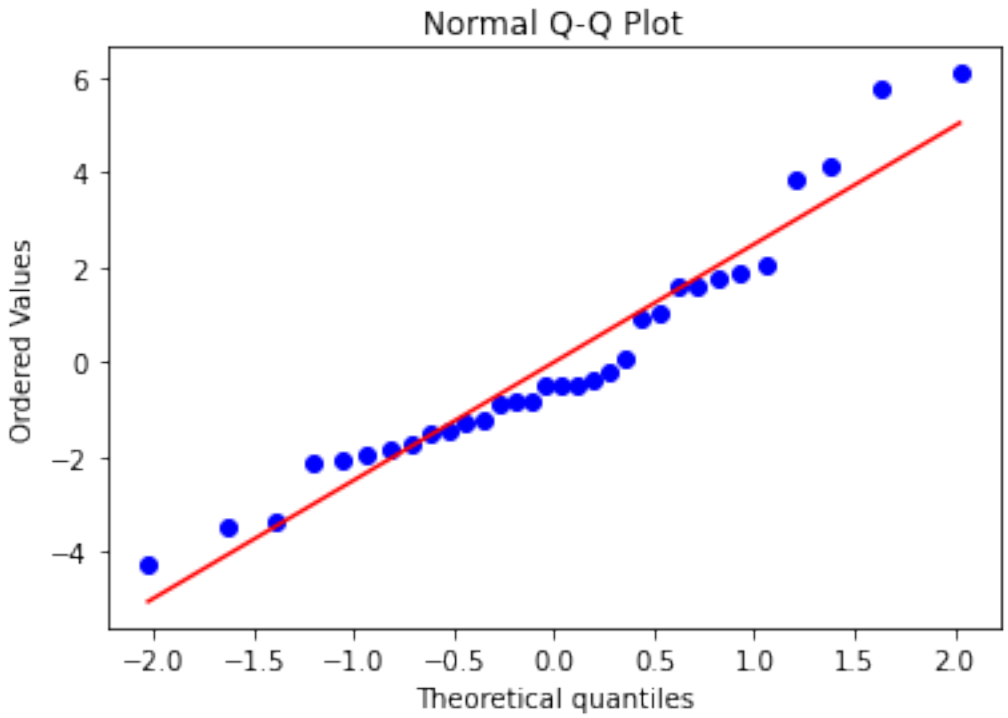

Figure 5: png

4. After making the model, the residuals seem homoscedastic.

This assumption is sometimes called "equal variance," and can be checked by the `regplot` function in Seaborn. We must first standardize the residuals, which we can do with NumPy. We want to see a plot with no clear pattern; a cone shape to the data would indicate heteroscedasticity, the opposite of homoscedasticity.

```
import numpy as np
standardized_residuals = np.sqrt(np.abs(residuals))
sns.regplot(x = predictions, y = standardized_residuals, scatter=True, lowess=True)
plt.ylabel("Standarized residuals")
plt.xlabel("Fitted value")
plt.title("Scale-Location")
plt.show()
```

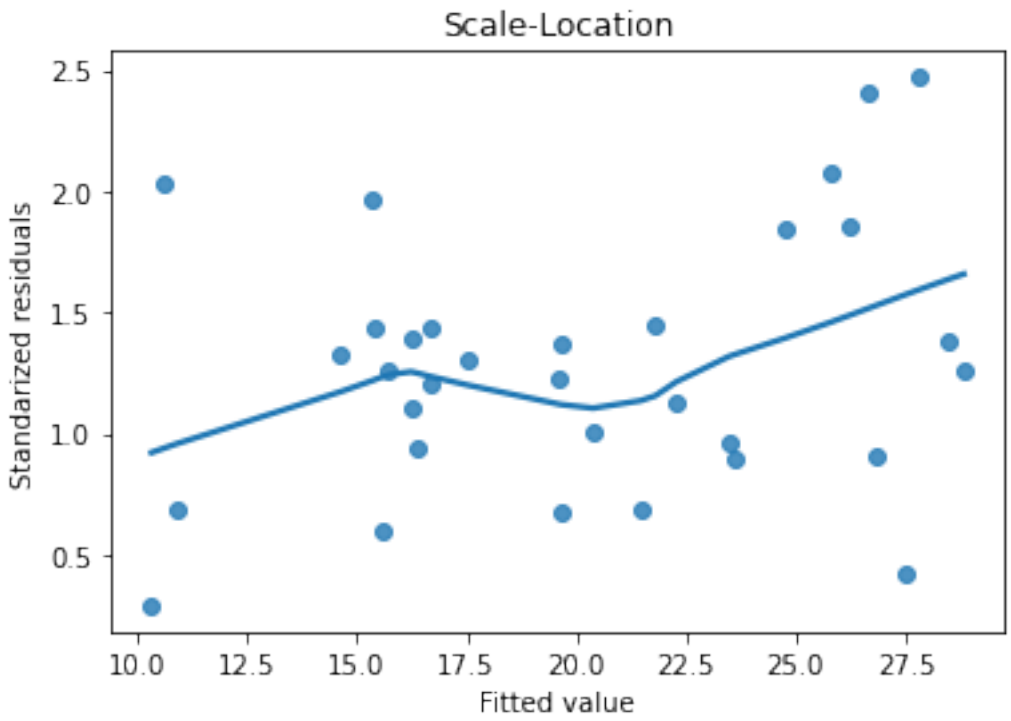

Figure 6: png

Content last modified on 14 September 2021.

See a problem? Tell us or edit the source.

How to compute the power of a test comparing two population means

Description

When creating a factorial design, it is important that it has adequate power to detect significant main effects and interaction effects of interest. How can we calculate the power of a two-sample t test that we aim to perform in such a situation?

Related tasks:

- How to choose the sample size in a study with two population means (on website)

Solution in Python

From `statsmodels`, we use the `solve_power` function in the `TTestIndPower` class. That function embodies a relationship among five variables; you provide any four of them and it will compute the fifth to be consistent with the first four, regarding the two-sample t-test you plan to perform. Let's get started by importing the package and create a `TTestIndPower` object.

```
from statsmodels.stats.power import TTestIndPower
analysis = TTestIndPower()
```

For this example, let's say that:

- You plan to create a balanced 4×2 factorial experiment with 32 subjects.
- You expect the effect size for the main effect of factor A to be medium (0.25 according to Cohen's 1988 text).
- You want to know the expected power for the test of a main effect of factor A.
- Your significance level is $\alpha = 0.05$.

We proceed as follows.

```
obs = 32       # number of subjects (or observations)
effect = 0.25  # effect size
alpha = 0.05   # significance level
ratio = 1      # ratio of the number of observations in one sample to the other

# We leave power unspecified, so that solve_power will compute it for us:
analysis.solve_power( effect_size=effect, power=None, nobs1=obs, ratio=ratio, alpha=alpha )
```

```
0.1662985260871502
```

The power is 0.1663, which means that the probability of rejecting the null hypothesis when in fact it is false OR the probability of avoiding a Type II error is 0.1663.

Content last modified on 30 November 2021.

See a problem? Tell us or edit the source.

How to perform an analysis of covariance (ANCOVA)

Description

Recall that covariates are variables that may be related to the outcome but are unaffected by treatment assignment. In a randomized experiment with one or more observed covariates, an analysis of covariance (ANCOVA) addresses this question: How would the mean outcome in each treatment group change if all groups were equal with respect to the covariate? The goal is to remove any variability in the outcome associated with the covariate from the unexplained variability used to determine statistical significance.

Related tasks:

- How to do a one-way analysis of variance (ANOVA) (on website)
- How to compare two nested linear models
- How to conduct a mixed designs ANOVA
- How to conduct a repeated measures ANOVA

Solution in Python

The solution below uses an example dataset about car design and fuel consumption from a 1974 Motor Trend magazine. (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('mtcars')
```

Let's use ANCOVA to check the effect of the engine type (0 = V-shaped, 1 = straight, in the variable `vs`) on the miles per gallon when considering the weight of the car as a covariate. We will use the `ancova` function from the `pingouin` package to conduct the test.

```
from pingouin import ancova
ancova(data=df, dv='mpg', covar='wt', between='vs')
```

	Source	SS	DF	F	p-unc	np2
0	vs	54.228061	1	7.017656	1.292580e-02	0.194839
1	wt	405.425409	1	52.466123	5.632548e-08	0.644024
2	Residual	224.093877	29	NaN	NaN	NaN

```
/opt/conda/lib/python3.9/site-packages/outdated/utils.py:14: OutdatedPackageWarning: The package pingouin is out
Set the environment variable OUTDATED_IGNORE=1 to disable these warnings.
  return warn(
```

The p-value for each variable is in the `p-unc` column.

The p-value for the `wt` variable tests the null hypothesis, "The quantities `wt` and `mpg` are not related." Since it is below 0.05, we reject the null hypothesis, and conclude that `wt` is significant in predicting `mpg`.

The p-value for the `vs` variable tests the null hypothesis, "The quantities `vs` and `mpg` are not related if we hold `wt` constant." Since it is below 0.05, we reject the null hypothesis, and conclude that `vs` is significant in predicting `mpg` even among cars with equal weight (`wt`).

Note: Unfortunately, a two-factor ANCOVA is not possible in pingouin. However, a model with more than one covariate is possible, as you can provide a list as the `covar` parameter when calling `ancova`.

Content last modified on 07 December 2021.

See a problem? Tell us or edit the source.

How to perform pairwise comparisons

Description

When analyzing data from a completely randomized single-factor design, suppose that you have performed an ANOVA and noticed that there's a significant difference between at least one pair of treatment levels. How can pairwise comparisons help us explore which pairs of treatment levels are different?

Related tasks:

- How to do a one-way analysis of variance (ANOVA) (on website)
- How to perform post-hoc analysis with Tukey's HSD test

Solution in Python

The solution below uses an example dataset that details the counts of insects in an agricultural experiment with six types of insecticides, labeled A through F. (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('InsectSprays')
df
```

	count	spray
0	10	A
1	7	A
2	20	A
3	14	A
4	14	A
...	...	...
67	10	F
68	26	F
69	26	F
70	24	F
71	13	F

72 rows × 2 columns

Before we perform any post hoc analysis, we need to see if the count of insects depends on the type of insecticide given by conducting a one way ANOVA. (See also how to do a one-way analysis of variance (ANOVA) (on website).)

```
from statsmodels.formula.api import ols
model = ols('count ~ spray', data = df).fit()
import statsmodels.api as sm
sm.stats.anova_lm(model, typ=1)
```

	df	sum_sq	mean_sq	F	PR(>F)
spray	5.0	2668.833333	533.766667	34.702282	3.182584e-17
Residual	66.0	1015.166667	15.381313	NaN	NaN

At the 5% significance level, we see that the count differs according to the type of insecticide used. We assume that the model assumptions are met, but do not verify that here.

If we would like to compare the pairs without any corrections, we can use the 'pairwise t test' in the `scikit_posthocs` package.

```
import scikit_posthocs as sp
sp.posthoc_ttest(df, val_col='count', group_col='spray', p_adjust=None, pool_sd=True )
```

	A	B	C	D	E	F
A	1.000000e+00	6.044761e-01	7.266893e-11	9.816910e-08	2.753922e-09	1.805998e-01
B	6.044761e-01	1.000000e+00	8.509776e-12	1.212803e-08	3.257986e-10	4.079858e-01
C	7.266893e-11	8.509776e-12	1.000000e+00	8.141205e-02	3.794750e-01	2.794343e-13
D	9.816910e-08	1.212803e-08	8.141205e-02	1.000000e+00	3.794750e-01	4.035610e-10
E	2.753922e-09	3.257986e-10	3.794750e-01	3.794750e-01	1.000000e+00	1.054387e-11
F	1.805998e-01	4.079858e-01	2.794343e-13	4.035610e-10	1.054387e-11	1.000000e+00

Techniques to adjust the above table for multiple comparisons include the Bonferroni correction, Fisher's Least Significant Difference (LSD) method, Dunnett's procedure, and Scheffe's method. These can be used in place of 'None' for the `p.adjust` argument; see details here.

You can also determine the magnitude of these differences; see how to perform post-hoc analysis with Tukey's HSD test.

Content last modified on 16 September 2021.

See a problem? Tell us or edit the source.

How to perform post-hoc analysis with Tukey's HSD test

Description

If we run a one-way ANOVA test and find that there is a significant difference between population means, we might want to know which means are actually different from each other. One way to do so is with Tukey's Honestly Significant Differences (HSD) method. It creates confidence intervals for each pair of samples, while controlling for Type I error rate across all pairs. Thus the resulting intervals are a little wider than those produced using Fisher's LSD method. How do we make these confidence intervals, with an appropriate visualization?

Solution in Python

We load here the same data that appears in the solution for how to perform pairwise comparisons. That solution used ANOVA to determine which pairs of groups have significant differences in their means; follow its link for more details.

```
from rdatasets import data
df = data('InsectSprays')
df
```

	count	spray
0	10	A
1	7	A
2	20	A
3	14	A
4	14	A
...	...	...
67	10	F
68	26	F
69	26	F
70	24	F
71	13	F

72 rows × 2 columns

We now want to perform an unplanned comparison test on the data to determine the magnitudes of the differences between pairs of groups. We do this by applying Tukey's HSD approach to perform pairwise comparisons and generate confidence intervals that maintain a specified experiment-wide error rate. Before that, the `pairwise_tukeyhsd` module needs to be imported from the `statsmodels` package.

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
tukey = pairwise_tukeyhsd(endog=df['count'], groups=df['spray'], alpha=0.05)
print(tukey)
```

```
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower    upper  reject
-----------------------------------------------------
     A      B   0.8333    0.9  -3.8659  5.5326  False
     A      C -12.4167  0.001 -17.1159 -7.7174   True
     A      D  -9.5833  0.001 -14.2826 -4.8841   True
     A      E    -11.0  0.001 -15.6992 -6.3008   True
     A      F   2.1667  0.728  -2.5326  6.8659  False
     B      C   -13.25  0.001 -17.9492 -8.5508   True
     B      D -10.4167  0.001 -15.1159 -5.7174   True
     B      E -11.8333  0.001 -16.5326 -7.1341   True
     B      F   1.3333    0.9  -3.3659  6.0326  False
     C      D   2.8333 0.4921  -1.8659  7.5326  False
     C      E   1.4167    0.9  -3.2826  6.1159  False
     C      F  14.5833  0.001   9.8841 19.2826   True
     D      E  -1.4167    0.9  -6.1159  3.2826  False
     D      F    11.75  0.001   7.0508 16.4492   True
     E      F  13.1667  0.001   8.4674 17.8659   True
-----------------------------------------------------
```

Because the above table contains a lot of information, it's often helpful to visualize these intervals. Python's `statsmodels` package does not have a built-in way to do so, but we can create our own as follows.

```
import matplotlib.pyplot as plt
rows = tukey.summary().data[1:]
plt.hlines( range(len(rows)), [row[4] for row in rows], [row[5] for row in rows] )
plt.vlines( 0, -1, len( rows )-1, linestyles='dashed' )
plt.gca().set_yticks( range( len( rows ) ) )
plt.gca().set_yticklabels( [ f'{x[0]}-{x[1]}' for x in rows ] )
plt.show()
```

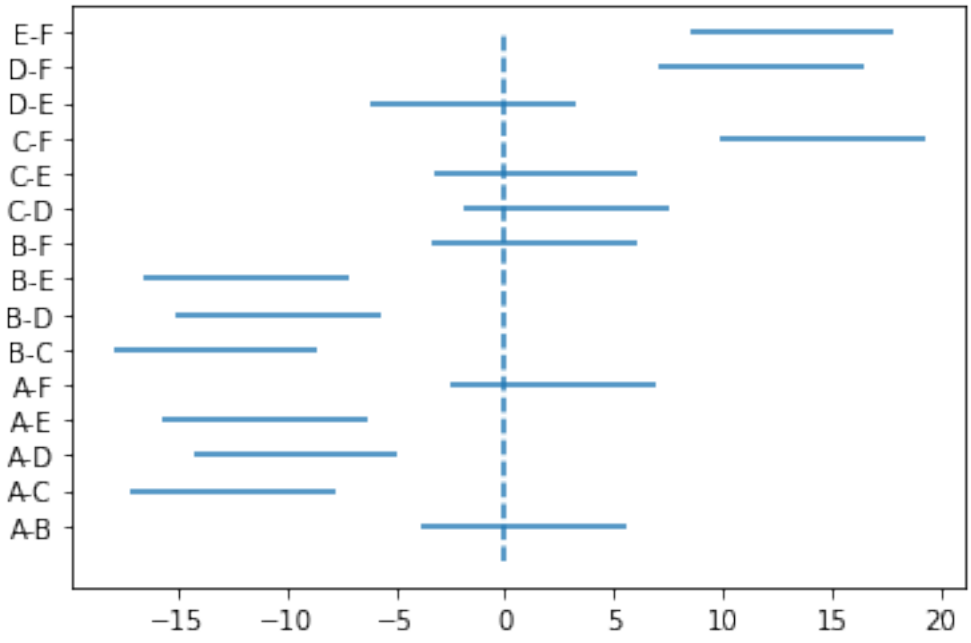

Figure 7: png

Confidence intervals that cross the vertical, dashed line at $x = 0$ are those in which the means across those groups may be equal. Other intervals have mean differences whose 95% confidence intervals do not include zero.

Content last modified on 10 September 2021.

See a problem? Tell us or edit the source.

How to test for a treatment effect in a single factor design

Description

Suppose you are given a dataset that has more than one treatment level and you wish to see if there is a unit-level treatment effect. How would you check that?

Solution in Python

The solution below uses an example dataset about the teeth of 10 guinea pigs at three Vitamin C dosage levels (in mg) with two delivery methods (orange juice vs. ascorbic acid). (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('ToothGrowth')
```

In this dataset, there are only two treatments (orange juice and ascorbic acid, in the variable `supp`). We can therefore perrform a two-sample t test. But first we must filter the outcome variable `len` (tooth length) based on `supp`.

```
subjects_receiving_oj = df[df['supp']=='OJ']['len']
subjects_receiving_vc = df[df['supp']=='VC']['len']

import scipy.stats as stats
stats.ttest_ind( subjects_receiving_oj, subjects_receiving_vc, equal_var=False )
```

```
Ttest_indResult(statistic=1.91526826869527, pvalue=0.06063450788093387)
```

At the 5% significance level, we see that the length of the tooth does not differ between the two delivery methods. We assume that the model assumptions are met, but do not check that here.

If there are multiple levels (two or more), you can apply the parametric ANOVA test which in this case will provide a similar p value.

```
from statsmodels.formula.api import ols
model = ols('len ~ supp', data = df).fit()

import statsmodels.api as sm
sm.stats.anova_lm(model, typ=1)
```

	df	sum_sq	mean_sq	F	PR(>F)
supp	1.0	205.350000	205.350000	3.668253	0.060393
Residual	58.0	3246.859333	55.980333	NaN	NaN

We see the p value in the final column is very similar.

However, if the assumptions of ANOVA are not met, we can utilize a nonparametric approach via the Kruskal-Wallis Test. We use the filtered variables defined above and import the `kruskal` function from SciPy.

```
from scipy.stats import kruskal
kruskal( subjects_receiving_oj, subjects_receiving_vc )
```

```
KruskalResult(statistic=3.4453580631407035, pvalue=0.06342967639688878)
```

Similar to the previous results, the length of the tooth does not differ between the delivery methods at the 5% significance level.

Content last modified on 24 October 2021.

See a problem? Tell us or edit the source.

How to plot interaction effects of treatments

Description

When there are multiple treatment conditions with multiple levels and you wish to undertsand the interaction effects of each of them, a plot can be useful. How can we create the right kind of plot for that situation?

- How to create basic plots (on website)
- How to add details to a plot (on website)
- How to create a histogram (on website)
- How to create a box (and whisker) plot (on website)
- How to change axes, ticks, and scale in a plot (on website)
- How to create bivariate plots to compare groups

Solution in Python

The solution below uses an example dataset about the teeth of 10 guinea pigs at three Vitamin C dosage levels (in mg) with two delivery methods (orange juice vs. ascorbic acid). (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('ToothGrowth')
```

To plot the interaction effects among tooth length, supplement, and dosage, we can use the `pointplot` function in the Seaborn package.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pointplot(x='dose',y='len',hue='supp',data=df)
plt.legend(loc='lower right')  # Default is upper right, which overlaps the data here.
plt.show()
```

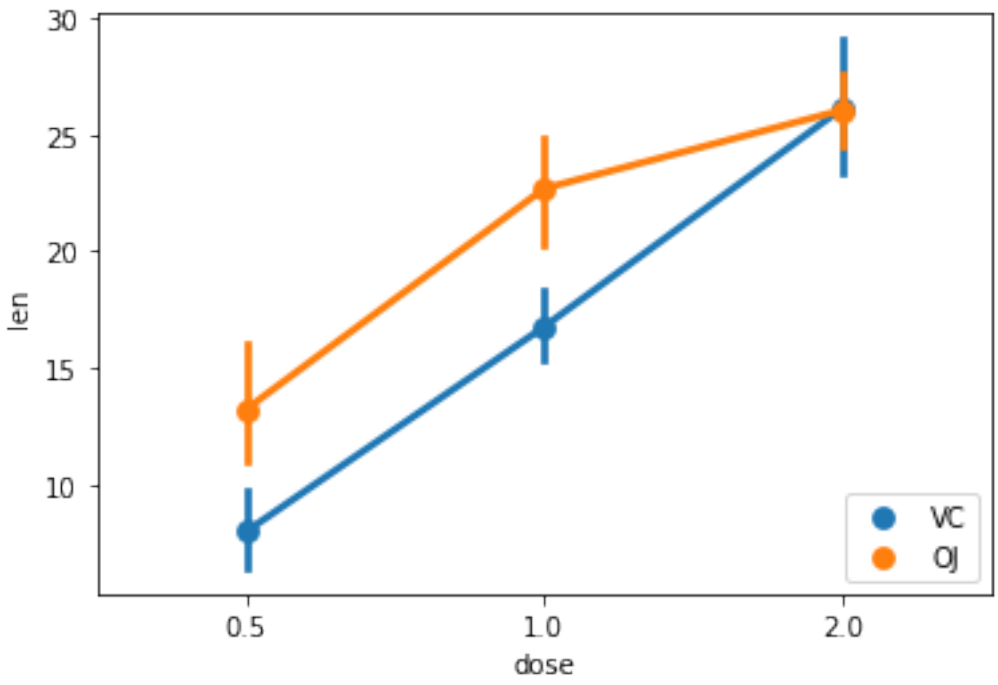

Figure 8: png

Looking at the output, we first see that there is an interaction effect because the two supp lines intersect. We also see that there is a difference in length when giving 0.5mg and 1mg dosage of either of the two delivery methods. However, there is barely any difference between the delivery methods when the dosage level is 2mg.

Content last modified on 24 October 2021.

See a problem? Tell us or edit the source.

How to analyze the sample means of different treatment conditions

Description

In a single-factor experiment with three or more treatment levels, how can we compare them to see which one impacts the outcome variable the most?

Solution in Python

The solution below uses an example dataset about the teeth of 10 guinea pigs at three Vitamin C dosage levels (in mg) with two delivery methods (orange juice vs. ascorbic acid). (See how to quickly load some sample data (on website).)

```
from rdatasets import data
df = data('ToothGrowth')
```

To visually plot the means of the length of the tooth based on the Vitamin C dosage levels we can create a pointplot. We will have to import the `seaborn` and `matplotlib.pyplot` packages to be able to create it.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pointplot( x = 'dose', y = 'len', data = df,
               ci = 95,         # ci stands for Confidence Interval
               capsize = 0.1 )  # the width of the "caps" on error bars
plt.show()
```

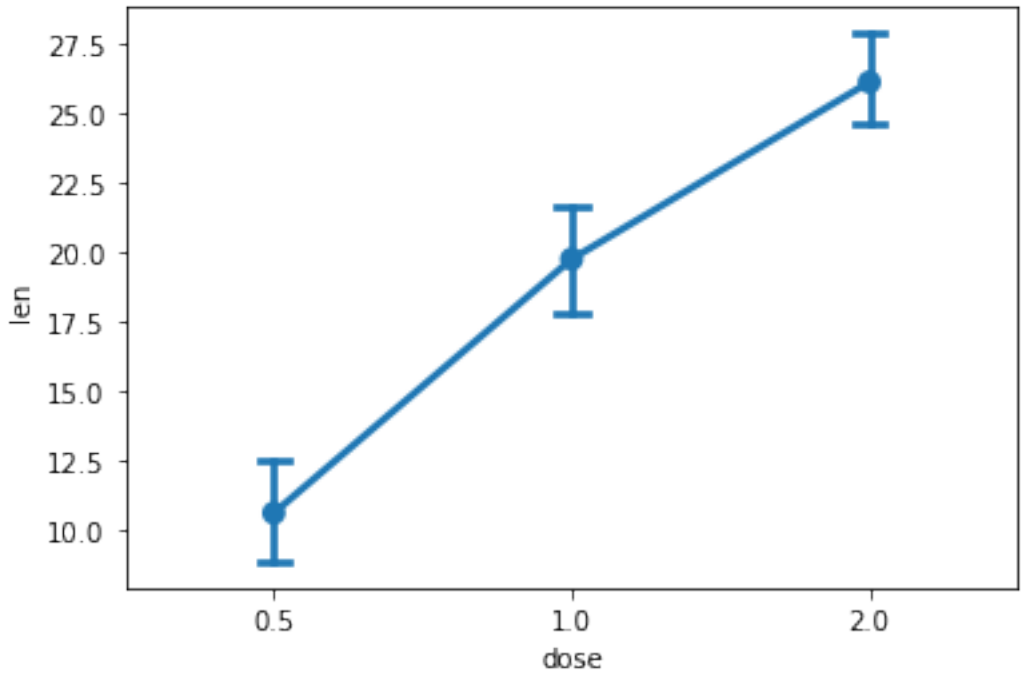

Figure 9: png

The point plot informs us that as the dosage levels increase, the tooth length also increases.

To obtain the actual numbers, we can use the `groupby` function to compute the treatment level means, and the `mean` function to compute the mean for the entire column.

```
df.groupby('dose')['len'].mean()
```

```
dose
0.5    10.605
1.0    19.735
2.0    26.100
Name: len, dtype: float64
```

```
df['len'].mean()
```

```
18.813333333333336
```

If you wish to display the difference between the overall mean and the group means, you can subtract the overall mean from the treatment level means.

```
df.groupby('dose')['len'].mean() - df['len'].mean()
```

```
dose
0.5   -8.208333
1.0    0.921667
2.0    7.286667
Name: len, dtype: float64
```

Content last modified on 26 July 2021.

See a problem? Tell us or edit the source.

How to compare two nested linear models

Description

Model A is said to be "nested" in model B if the predictors included in A are a subset of those included in B. In such a situation, how can we determine if the larger model (in this case B) is significantly better than the smaller (reduced) model? We can use an Extra Sums of Squares test, also called a partial F-test, to compare two nested linear.

This technique will also help us with another question. If we have a multivarate linear model,

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k,$$

how can we test the influence of only some of the coefficients? If we remove some of the coefficients, we have a smaller model nested in the larger one, so the question is the same.

Related tasks:

- How to do a one-way analysis of variance (ANOVA) (on website)
- How to conduct a mixed designs ANOVA
- How to conduct a repeated measures ANOVA
- How to perform an analysis of covariance (ANCOVA)

Solution in Python

The solution below uses an example dataset about car design and fuel consumption from a 1974 Motor Trend magazine. (See how to quickly load some sample data (on website).)

We will create two models, one nested inside the other, in a natural way in this example. But this is not the only way to create nested models; it is just an example.

```
from rdatasets import data
df = data('mtcars')
```

Consider a model using number of cylinders (cyl) and weight of car (wt) to predict its fuel efficiency (mpg). We create this model and perform an ANOVA to see if the predictors are significant. We use the Ordinary Least Squares module from `statsmodels`.

```
from statsmodels.formula.api import ols
add_model = ols('mpg ~ cyl + wt', data = df).fit()

import statsmodels.api as sm
sm.stats.anova_lm(add_model, typ= 1)
```

	df	sum_sq	mean_sq	F	PR(>F)
cyl	1.0	817.712952	817.712952	124.043687	5.424327e-12
wt	1.0	117.162269	117.162269	17.773034	2.220200e-04
Residual	29.0	191.171966	6.592137	NaN	NaN

In the final column of output we see that all numbers are below 0.05, which suggests that both predictors are significant. A natural question to ask is whether the two predictors have an interaction effect. Let's create a model containing the interaction term.

```
int_model = ols('mpg ~ cyl*wt', data = df).fit()
sm.stats.anova_lm(int_model, typ= 1)
```

	df	sum_sq	mean_sq	F	PR(>F)
cyl	1.0	817.712952	817.712952	145.856269	1.280635e-12
wt	1.0	117.162269	117.162269	20.898350	8.942713e-05
cyl:wt	1.0	34.195767	34.195767	6.099533	1.988242e-02
Residual	28.0	156.976199	5.606293	NaN	NaN

As seen in the final column of output, there is a significant interaction between the two predictors (bottom number being below 0.05).

We now have one model (`add_model`) nested inside a larger model (`int_model`). To check which model is better, we can conduct an ANOVA comparing the two models. We use the `anova_lm` function from `statsmodels`.

```
from statsmodels.stats.anova import anova_lm
anova_lm(add_model, int_model)
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	29.0	191.171966	0.0	NaN	NaN	NaN
1	28.0	156.976199	1.0	34.195767	6.099533	0.019882

We have just performed this hypothesis test:

H_0 = the two models are equally useful for predicting the outcome

H_a = the larger model is significantly better than the smaller model

In the final column of the output, called **Pr(>F)**, the only number in that column is our test statistic, 0.019882. Since is below our chosen threshold of 0.05, we reject the null hypothesis, and prefer to use the second model.

This method can be used to check if covariates should be included in the model, or if additional variables should be added as well.

Content last modified on 30 November 2021.

See a problem? Tell us or edit the source.

How to conduct a mixed designs ANOVA

Description

When you have a dataset that includes the responses of a mixed design test, where one factor is a within-subjects factor and the other is a between-subjects factor, and you wish check if there is a significant difference for both factors, this requires a Mixed Design ANOVA. How can we conduct one?

Related tasks:

- How to do a one-way analysis of variance (ANOVA) (on website)
- How to do a two-way ANOVA test with interaction (on website)
- How to do a two-way ANOVA test without interaction (on website)
- How to compare two nested linear models using ANOVA
- How to conduct a repeated measures ANOVA
- How to perform an analysis of covariance (ANCOVA)

Solution in Python

We create the data for a hypothetical 2×2 mixed design with the following attributes.

- Between-subjects treatment factor: Type of music played (classical vs. rock)
- Within-subjects treatment factor: Type of room (light vs. no light)
- Outcome variable: Heart rate of subject

```
import pandas as pd
df = pd.DataFrame( {
    'Subject'    : [1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10],
    'Music'      : ['Classical','Rock','Classical','Rock','Classical','Rock','Classical',
                    'Rock','Classical','Rock','Classical','Rock','Classical','Rock','Classical',
                    'Rock','Classical','Rock','Classical','Rock'],
    'Room Type'  : ['Light','Light','Light','Light','Light','Light','Light','Light','Light',
                    'Light','No Light','No Light','No Light','No Light','No Light','No Light',
                    'No Light','No Light','No Light','No Light'],
    'Heart Rate' : [78,60,85,75,99,94,75,84,100,76,90,109,99,94,113,92,91,88,89,90]
} )
df.head()
```

	Subject	Music	Room Type	Heart Rate
0	1	Classical	Light	78
1	2	Rock	Light	60
2	3	Classical	Light	85
3	4	Rock	Light	75
4	5	Classical	Light	99

We will use the `pingouin` statistics package to conduct a two-way mixed-design ANOVA. The parameters are as follows:

1. `dv`: name of the column containing the dependant variable
2. `within`: name of the column containing the within-group factor
3. `between`: name of the column containing the between-group factor
4. `subject`: name of the column identifying each subject
5. `data`: the pandas DataFrame containing all the data

```
import pingouin as pg
pg.mixed_anova( dv='Heart Rate', within='Room Type', between='Music', subject='Subject', data=df )
```

	Source	SS	DF1	DF2	MS	F	p-unc	np2	eps
0	Music	162.45	1	8	162.45	1.586813	0.243288	0.165520	NaN
1	Room Type	832.05	1	8	832.05	6.416426	0.035088	0.445077	1.0
2	Interaction	76.05	1	8	76.05	0.586466	0.465781	0.068301	NaN

The output informs us that, on average, the subjects that listened to classical music did not significantly differ ($p = 0.243288 > 0.05$) from those that listened to rock music. However, there is, on average, a significant difference ($p = 0.035088 < 0.05$) between each of the subject's heart rate when put in a room with or without light. Additionally, since the interaction term is not significant ($p = 0.465781 > 0.05$), we can use the additive (no interaction) model.

Content last modified on 24 October 2021.

See a problem? Tell us or edit the source.

How to conduct a repeated measures ANOVA

Description

In a repeated measures test, the same subject receives multiple treatments. When you have a dataset that includes the responses of a repeated measures test where the measurements are dependent (within subjects design), you may wish to check if there is a difference in the treatment effects. How would you conduct a repeated measures ANOVA to answer that question?

Related tasks:

- How to do a one-way analysis of variance (ANOVA) (on website)
- How to do a two-way ANOVA test with interaction (on website)
- How to do a two-way ANOVA test without interaction (on website)
- How to compare two nested linear models using ANOVA
- How to conduct a mixed designs ANOVA
- How to perform an analysis of covariance (ANCOVA)

Solution in Python

We create a hypothetical repeated measures dataset where the 5 subjects undergo all 4 skin treatments and their rating of the treatment is measured.

```
import pandas as pd
df = pd.DataFrame( {
    'Subject':        [1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5],
    'Skin Treatment': ['W','X','Y','Z','W','X','Y','Z','W','X',
                       'Y','Z','W','X','Y','Z','W','X','Y','Z'],
    'Rating':         [7,5,8,4,8,10,7,5,7,6,5,4,7,7,4,5,8,8,6,6]
} )
df.head()
```

	Subject	Skin Treatment	Rating
0	1	W	7
1	1	X	5
2	1	Y	8
3	1	Z	4
4	2	W	8

Before we conduct a repeated measures ANOVA, we need to decide which approach to use - Univariate or Multivariate. We decide this using Mauchly's test of sphericity. If we fail to reject the null hypothesis then we use the univariate approach.

- H_0 = the sphericity assumption holds
- H_A = the sphericity assumption is violated

We use the `pingouin` statistics package to conduct the test. Most of the parameters below are self-explanatory, except that `dv` stands for dependent variable.

```
import pingouin as pg
pg.sphericity( dv='Rating', within='Skin Treatment', subject='Subject', method='mauchly', data=df )
```

```
/opt/conda/lib/python3.9/site-packages/outdated/utils.py:14: OutdatedPackageWarning: The package pingouin is out o
Set the environment variable OUTDATED_IGNORE=1 to disable these warnings.
  return warn(
```

```
SpherResults(spher=True, W=0.06210054956238558, chi2=7.565056754547507, dof=5, pval=0.20708214225927316)
```

Since the p value of `skin_treatment` is about 0.2071, we fail to reject the sphericity assumption at a 5% significance level and use the univariate approach to conduct the repeated measures ANOVA.

```
# Compute a repeated measures ANOVA using a function pingouin adds to our DataFrame:
df.rm_anova( dv='Rating', within='Skin Treatment', subject='Subject', detailed=False )
```

	Source	ddof1	ddof2	F	p-unc	np2	eps
0	Skin Treatment	3	12	5.117647	0.016501	0.56129	0.541199

Since the p value of about 0.017 is less than 0.05, we conclude that there is significant evidence of a treatment effect.

Note: If there is more than 1 repeated measures factor, you can add a list of them to the `within` parameter and conduct the test.

Content last modified on 24 October 2021.

See a problem? Tell us or edit the source.

How to perform a planned comparison test

Description

Suppose that ANOVA reveals a significant difference between treatment levels, and you wish to explore further through post hoc analysis by comparing two specific treatment levels. How can we perform perform planned comparisons, also called a contrast test?

Solution in Python

How to Data does not yet contain a solution for this task in Python.