EzPyZ

Release 0.1.9

Ethan Guthrie

Dec 03, 2020

CONTENTS

CHAPTER

ONE

CONTENTS OF EZPYZ

1.1 EzPyZ

1.1.1 DataFrame

class EzPyZ.**DataFrame**(*data*, *columns=None*, *subset=None*)

Bases: object

A DataFrame object will be used to utilize all other functionality in this package.

If you would prefer to pass a pandas dataframe directly to the class:

```
>>> import EzPyZ as ez
>>> import pandas as pd
>>> raw_data = {
...     'height_cm': [134, 168, 149, 201, 177, 168],
...     'weight_kg': [32.2, 64.3, 59.9, 95.4, 104.2, 63.1]
... }
>>> pandas_df = pd.DataFrame(raw_data)
>>> df = ez.DataFrame(data=pandas_df)
```

Or if you'd like to provide the data in a more raw format (similar to what would be passed to a pandas dataframe):

```
>>> import EzPyZ as ez
>>> raw_data = {
...     'height_cm': [134, 168, 149, 201, 177, 168],
...     'weight_kg': [32.2, 64.3, 59.9, 95.4, 104.2, 63.1]
... }
>>> df = ez.DataFrame(data=raw_data)
```

Or if you'd like to provide the data directly from an Excel of CSV file:

```
>>> import EzPyZ as ez
>>> from EzPyZ.tools import read_file
>>> df = ez.DataFrame(data=read_file("bmi_data.csv")) # A bmi_data.xlsx would␣
↪also work here.
```

__init__(*data*, *columns=None*, *subset=None*)

Constructs a DataFrame object.

Parameters

- **data** (`Union[pd.DataFrame, Dict[str, List[Any]]]`) – Either a pandas DataFrame object, or a dictionary where the keys are column titles and the values are lists of associated values (in order).
- **columns** (`List[str]`) – (optional) A list of strings containing the titles of columns to be included in the dataframe. All others will be excluded. If this option is left blank or set to `NoneType`, then all columns will be included.
- **subset** (`str`) – String containing rules to exclude certain rows from the `DataFrame`. This string must be composed with standard comparison operators ('==', '!=', '<', '>', '<=', '>='). "And" statements must be separated by the word 'and' character, and "or" statements must be separated by the word 'or'. Parenthesis are allowed as well. Defaults to `None`.

Returns A new `EzPyZ.DataFrame` object.

Return type ``EzPyZ.DataFrame`

__repr__()

Returns basic `DataFrame` information.

Returns Basic `DataFrame` information for debugging.

Return type `str`

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv") A bmi_data.xlsx would also work␣
↪here.
>>> df = ez.DataFrame(data=data)
>>> print(repr(df))
DataFrame(df=Column(title=height_cm, values=[134, 168, 149, 201, 177, ...]),
          Column(title=weight_kg, values=[32.2, 64.3, 59.9, 95.4, 104.2, ...
↪]))
```

__str__()

Returns the `DataFrame` as a string.

Returns A print-friendly string representing the `DataFrame` object.

Return type `str`

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv") A bmi_data.xlsx would also work␣
↪here.
>>> df = ez.DataFrame(data=data)
>>> print(df)
height_cm      weight_kg
1    134            32.2
2    168            64.3
3    149            59.9
4    201            95.4
5    177            104.2
6    168            63.1
```

get_columns()

Returns columns as a list.

Returns Columns as a list.

Return type `List[EzPyZ.Column]`

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv") A bmi_data.xlsx would also work␣
↪here.
>>> df = ez.DataFrame(data=data)
>>> print(df.get_columns())
[Column(title=height_cm, values=[134, 168, 149, 201, 177, ...]),
 Column(title=weight_kg, values=[32.2, 64.3, 59.9, 95.4, 104.2, ...])]
```

get_titles()

Returns a list of all column titles.

Returns A list of all column titles.

Return type `List[str]`

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv") A bmi_data.xlsx would also work␣
↪here.
>>> df = ez.DataFrame(data=data)
>>> print(df.get_titles())
['height_cm', 'weight_kg']
```

head(*count=5*)

Returns the first `count` rows of the dataframe.

Parameters **count** (`int`) – (optional) The number of rows to return. Defaults to `5`.

Returns The first `count` rows of the dataframe.

Return type `str`

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv") A bmi_data.xlsx would also work␣
↪here.
>>> df = ez.DataFrame(data=data)
>>> print(df.head())
height_cm      weight_kg
0   134            32.2
1   168            64.3
2   149            59.9
3   201            95.4
4   177            104.2
5   168            63.1
```

length_columns()

Returns the number of columns in the `DataFrame`.

Returns Number of columns.

Return type `int`

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv") A bmi_data.xlsx would also work␣
↪here.
>>> df = ez.DataFrame(data=data)
>>> df.length_columns()
2
```

length_rows()
Returns the number of rows in the DataFrame.

Returns Number of rows.

Return type int

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv") A bmi_data.xlsx would also work␣
↪here.
>>> df = ez.DataFrame(data=data)
>>> df.length_rows()
6
```

subset(*criterion*)
Returns a new DataFrame object that meets the filter criterion provided.

Returns A new, filtered DataFrame object.

Return type EzPyZ.DataFrame

write_csv(*filename='out.csv'*, *header=True*)
Writes the dataframe to a CSV file.

Parameters

- **filename** (str) – (optional) The qualified name of the file to write to. Defaults to out.csv.
- **header** (bool) – (optional) Boolean. Specifies whether or not the column titles should be written to the CSV. Defaults to True.

Returns Nothing.

Return type NoneType

Usage:

```
>>> import EzPyZ as ez
>>> raw_data = {
>>>     'height (cm)': [134, 168, 149, 201, 177, 168],
>>>     'weight (kg)': [32.2, 64.3, 59.9, 95.4, 104.2, 63.1]
>>> }
>>> df = ez.DataFrame(data=raw_data)
>>> df.write_csv("bmi_data.csv")
```

1.1.2 Column

class EzPyZ.**Column**(*title*, *values*)
Bases: object

A Column object. Column objects will make up EzPyZ.DataFrame objects in this module. This class is NOT intended for exernal use!

__init__(*title*, *values*)
Constructs a Column object.

Parameters

- **title** (str) – A string containing the title of the column.
- **values** (List[Any]) – A list containing the values in the column, in order.

Returns Nothing.

Return type NoneType

__repr__()
Returns basic Column information.

Returns Basic Column information for debugging.

Return type str

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(repr(col))
Column(title=height_cm, values=[134, 168, 149, 201, 177, ...])
```

__str__()
Returns the Column as a string.

Returns The Column as a string.

Return type str

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col)
height_cm
134
168
149
201
177
168
```

get_values()
Returns self.values.

Returns The values in the column.

Return type List[Any]

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.get_values())
[134, 168, 149, 201, 177, 168]
```

length()

Returns the length of `self.values`.

Returns The number of values in the column.

Return type `int`

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.length())
6
```

mean()

Returns the mean of `self.values`.

Returns The mean of the values in the column.

Return type `float`

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.mean())
166.16666666666666
```

median()

Returns the median of `self.values`.

Returns The median of the values in the column.

Return type `float`

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.median())
168.0
```

mode()

Returns the mode of `self.values`.

Returns The mode of the values in `Column`.

Return type `float`

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.mode())
168
```

set_values(*values*)
Sets `self.values`.

Parameters **values** (`List[Any]`) – A list containing the values in the column, in order.

Returns Nothing.

Return type `NoneType`

stdev()
Returns the standard deviation of `self.values`.

Returns The standard deviation of the values in the column.

Return type `float`

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.stdev())
23.094732444145496
```

title()
Returns `self.col_title`

Returns The title of the column.

Return type `str`

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.title())
height_cm
```

variance()
Returns the variance of `self.values`.

Returns The variance of the values in the column.

Return type `float`

Usage:

```
>>> import EzPyZ as ez
>>> col = ez.column.Column("height_cm", [134, 168, 149, 201, 177, 168])
>>> print(col.variance())
533.3666666666667
```

1.2 t-test

1.2.1 t_test()

`EzPyZ.`**t_test**(*x*, *y=None*, *alternative='two-tailed'*, *mu=None*, *data=None*, *paired=False*, *conf_level=0.05*, *subset=None*)
Conducts a t-test.

Parameters

- **x** (`EzPyZ.column.Column` or `str`) – The column of the first sample. If `data` is not `None`, then a string providing the column title may be provided.
- **y** (`EzPyZ.column.Column` or `str`) – (optional) The column of the second sample. If `data` is not `None`, then a string providing the column title may be provided. If performing a one-sample t-test, this should not be used. Defaults to `None`.
- **alternative** (`str`) – (optional) String. Whether the x column is being tested to be greater than, less than, or not equal to the y column. Must be one of "two-tailed", "less", or "greater". Defaults to "two-tailed".
- **mu** (`float`) – (optional) Float. The population mean for a one-sample t-test. Defaults to `None`.
- **data** (`EzPyZ.DataFrame`) – (optional) The dataframe containing the values. Defaults to `None`
- **paired** (`bool`) – (optional) Boolean. Whether the t-test is a paired-samples t-test. If `False`, an independent-samples t-test is conducted. Defaults to `False`.
- **conf_level** (`float`) – (optional) Float. The confidence interval. Defaults to `0.05`.
- **subset** (`str`) – (optional) String containing rules to exclude cetain rows from the analysis. See `EzPyZ.DataFrame` for more information on writing these strings. **This parameter may only be used when** `data` **is set to a valid** `EzPyZ.DataFrame`! Defaults to `None`.

Returns The results of the t-test.

Return type `EzPyZ.TResult`

Example one-sample t-test:

```
>>> import EzPyZ as ez
>>> data = {
...     'score': [15, 17, 16, 16, 19, 14, 17]
... }
>>> df = ez.DataFrame(data)
>>> # Let's conduct a two-tailed, one-sample t-test between the scores and a␣
↪population mean,
>>> # in this case well say 12.
>>> # We'll also use the standard confidence level of 0.05.
>>> t_res = ez.t_test(data=df, x='score', mu=15)
>>> print(t_res)

                        One-sample t-test

data:   score
t = 7.0711, df = 6, p-value = 0.000401
null hypothesis:                          true difference in means is equal to 0
alternative hypothesis:                   true difference in means is not equal to 0
resolution:                               reject null hypothesis with confidence␣
↪level of 0.05
95.0 percent confidence interval for x: [13.14278, 19.428649]
mean of the differences ( - x):        -4.285714
```

Example independent-samples t-test:

```
>>> import EzPyZ as ez
>>> data = {
...     'before': [1, 3, 4, 2, 3, 4, 6],
...     'after': [3, 4, 6, 9, 8, 7, 11]
```

(continues on next page)

(continued from previous page)

```
... }
>>> df = ez.DataFrame(data)
>>> # Let's conduct a two-tailed, independent-samples t-test between the before␣
↪and after
>>> # scores.
>>> # We'll also use the standard confidence level of 0.05.
>>> t_res = ez.t_test(data=df, x='before', y='after', paired=True)
>>> print(t_res)

                        Welch Two-Sample t-test

data:   before and after
t = -2.9327, df = 9.5647, p-value = 0.015663
null hypothesis:                        true difference in means is equal to 0
alternative hypothesis:                 true difference in means is not equal to 0
resolution:                             reject null hypothesis with confidence␣
↪level of 0.05
95.0 percent confidence interval for x: [0.14278, 6.428649]
mean of the differences (y - x):        3.571429
```

Example paired-samples t-test:

```
>>> import EzPyZ as ez
>>> data = {
...     'before': [1, 3, 4, 2, 3, 4, 6],
...     'after': [3, 4, 6, 9, 8, 7, 11]
... }
>>> df = ez.DataFrame(data)
>>> # Let's conduct a two-tailed, paired-samples t-test between the before and␣
↪after scores.
>>> # We'll also use the standard confidence level of 0.05.
>>> t_res = ez.t_test(data=df, x='before', y='after', paired=True)
>>> print(t_res)

                                        Paired t-test

data:                                                   before (m = 3.29) and␣
↪after (m = 6.86)
output:                                                 t = -4.3966, df = 6, p-
↪value = 0.004585
null hypothesis:                                        true difference in means␣
↪is equal to 0
alternative hypothesis:                                 true difference in means␣
↪is not equal to 0
resolution:                                             reject null hypothesis␣
↪with confidence level of 0.05
95.0 percent confidence interval for x:                 [0.14278, 6.428649]
mean of the differences (y - x):                        3.571429
```

1.2.2 TResult

class EzPyZ.t_test.**TResult**(*info*)
Bases: object

A TResult object will be generated and returned by t-tests. It will contain the following attributes:

TResult.desc A description of the t-test run (i.e. one-sample, paired-samples, etc.).

TResult.x The EzPyZ.Column object for the x column.

TResult.y The EzPyZ.Column object for the y column.

TResult.mu The population mean.

TResult.conf_level The confidence level.

TResult.conf_perc The percentage confidence level. For conf_level = .05, this would be 95.

TResult.t The t-score.

TResult.df The degrees of freedom.

TResult.p The p-value.

TResult.resolution A brief statement saying whether the null hypothesis was rejected.

TResult.alt The alternative hypothesis.

TResult.null The null hypothesis.

TResult.conf_interval The confidence interval of the x column.

TResult.mean_diff The mean difference (y - x) or (- x).

__init__(*info*)
Constructs a TResult object.

Parameters **info** (Dict[str, Union[str, Dict[str, Any]]]) – Dictionary. The data from the t-test.

Returns Nothing.

Return type NoneType

__repr__()
Returns basic TResult information.

Returns Basic TResult information.

Return type str

Usage:

```
>>> import EzPyZ as ez
>>> data = {
...     'before': [1, 3, 4, 2, 3, 4, 6],
...     'after': [3, 4, 6, 9, 8, 7, 11]
... }
>>> df = ez.DataFrame(data)
>>> t_res = ez.t_test(data=df, x='before', y='after')
>>> print(repr(t_res))
TResult(x=before, y=after, paired=False, t=-2.9327, df=9.5647, p=0.015663)
```

__str__()
Returns the `TResult` as a string.

Returns A print-friendly string representing the `TResult` object.

Return type `str`

Usage:

```
>>> import EzPyZ as ez
>>> data = {
...     'score': [15, 17, 16, 16, 19, 14, 17]
... }
>>> df = ez.DataFrame(data)
>>> # Let's conduct a two-tailed, one-sample t-test between the scores and a␣
↪population
>>> # mean, in this case well say 12.
>>> # We'll also use the standard confidence level of 0.05.
>>> t_res = ez.t_test(data=df, x='score', mu=15)
>>> # t_res now contains a ``TResponse``object.
>>> print(t_res)

                    One-sample t-test

data:   score
t = 7.0711, df = 6, p-value = 0.000401
null hypothesis:                          true difference in means is equal to 0
alternative hypothesis:                   true difference in means is not equal␣
↪to 0
resolution:                               reject null hypothesis with␣
↪confidence level of 0.05
95.0 percent confidence interval for x: [13.14278, 19.428649]
mean of the differences ( - x):        -4.285714
```

apa_style()
Generates and returns an APA-style string. This string is compliant to the APA 7th edition standard.

Returns An APA-style string describing the results of the t-test.

Return type `str`

1.3 EzPyZ.tools

1.3.1 read_files.py

read_file()

EzPyZ.tools.**read_file**(*filename: str*, *return_pandas_df=False*)
Reads the provided Excel or CSV data file. Returns a pandas `DataFrame` object if `return_pandas_df` is `True`, or a dictionary where the keys are column titles and the values are lists of associated values (in order) otherwise.

Parameters

- **filename** (`str`) – The qualified path to the data file to read.
- **return_pandas_df** – (optional) Boolean. Whether the data should be returned as a `pandas.DataFrame`. Defaults to `False`.

Returns A formatted version of the data in `filename`.

Return type `pandas.DataFrame` or `Dict[str, List[Any]]`

Usage:

```
>>> import EzPyZ as ez
>>> data = ez.tools.read_file("bmi_data.csv")
```

is_excel()

`EzPyZ.tools.`**`is_excel`**(*filename*)

Returns `True` if the file provided in `filename` is a valid Excel file, and `False` otherwise.

Parameters **`filename`** (`str`) – The qualified name of the file to be checked.

Returns Boolean. Whether `filename` is a valid Excel file.

Return type `bool`

Usage:

```
>>> import EzPyZ as ez
>>> ez.tools.read_files.is_excel("bmi_data.xlsx")
True
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX