computer science
pandas

https://pandas.pydata.org/pandas-docs/stable/index.html

pandas for data (science)

- syntax errors: raised while the interpreter while executing the program reaches an instruction that does not match syntax rules (the first time the instruction is executed)

- unexpected behavior (exceptions): raised while the interpreter executes the program, due to a situation (data) not taken into account while writing the program

dataframes

- The primary pandas data
- Two-dimensional, size-mutable, potentially heterogeneous tabular data.
- Data structure also contains labeled axes (rows and columns).
- Arithmetic operations align on both row and column labels.
- Can be thought of as a dictionary-like container for Series objects.

dataframes 2

- tidy data
  - each variable is saved in its own column
  - each observation is saved in its own row

... might seem not to compat ... easy to manipulate

dataframes 3

- generic way to store information

it is necessary to identify classifying elements & values

elements:

- n
- `type of test: TTT/OTD`
- `alpha: 0.05. 0.01`

Critical Values of the Wilcoxon Signed Ranks Test

n	Two-Tailed Test		One-Tailed Test	
	α = .05	α = .01	α = .05	α = .01
5	--	--	0	--
6	0	--	2	--
7	2	--	3	0
8	3	0	5	1
9	5	1	8	3
10	8	3	10	5
11	10	5	13	7
12	13	7	17	9
13	17	9	21	12
14	21	12	25	15
15	25	15	30	19
16	29	19	35	23
17	34	23	41	27

get data from csv file

`read_csv(csvfilename)`: returns a dataframe from with the data from the csvfilename

```
df = pd.read_csv("SBP.csv")
```

	SBP_before	SBP_after
0	125	118
1	132	134
2	138	130
3	120	124
4	125	105
5	127	130
6	136	130
7	139	132
8	131	123
9	132	128
10	135	126
11	136	140
12	128	135
13	127	126
14	130	132

```
SBP_before,SBP_after
125,118
132,134
138,130
120,124
125,105
127,130
136,130
139,132
131,123
132,128
135,126
136,140
128,135
127,126
130,132
```

`len(df)`

returns the number of row in the dataframe

get data from other file types

```
read_excel('myfile.xlsx',sheet_name='Sheet1',
                         index_col=None, na_values=['NA'])

read_stata('myfile.dta')

read_sas('myfile.sas7bdat')

read_hdf('myfile.h5','df')
```

dataframe data types

Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs(see below), pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	Values meant to hold time data. Look into these for time series experiments.

handling missing data

fillna(_value_)

```
>>> df=df.fillna(0.0)
>>> df
    SBP_before  SBP_after
0          125      118.0
1          132      134.0
2          138      130.0
3          120      124.0
4          125      105.0
5          127      130.0
6          136      130.0
7          139      132.0
8          131      123.0
9          132      128.0
10         135      126.0
11         136      140.0
12         128      135.0
13         127      126.0
14         130      132.0
15         134        0.0
```

replace NaN with a value

```
    SBP_before  SBP_after
0          125      118.0
1          132      134.0
2          138      130.0
3          120      124.0
4          125      105.0
5          127      130.0
6          136      130.0
7          139      132.0
8          131      123.0
9          132      128.0
10         135      126.0
11         136      140.0
12         128      135.0
13         127      126.0
14         130      132.0
15         134        NaN
```

remove when incomplete

dropna()

```
>>> df = df.dropna()
>>> df
    SBP_before  SBP_after
0          125      118.0
1          132      134.0
2          138      130.0
3          120      124.0
4          125      105.0
5          127      130.0
6          136      130.0
7          139      132.0
8          131      123.0
9          132      128.0
10         135      126.0
11         136      140.0
12         128      135.0
13         127      126.0
14         130      132.0
```

hands on …

- Find how many records there are in a dataframe
- How many elements are there?
- What are the column names?
- What types of columns are there in a dataframe?

preparing dataframes

- reorganize information ...

```
n,type,alpha,value
5,TTT,0.05,NaN
5,TTT,0.01,NaN
5,OTT,0.05,0
5,OTT,0.01,NaN
6,TTT,0.05,0
6,TTT,0.01,NaN
6,OTT,0.05,2
6,OTT,0.01,NaN
7,TTT,0.05,2
7,TTT,0.01,NaN
7,OTT,0.05,3
7,OTT,0.01,0
8,TTT,0.05,3
8,TTT,0.01,0
8,OTT,0.05,5
```

```
n,TTT05,TTT01,OTT05,OTT01
5,NaN,NaN,0,NaN
6,0,NaN,2,NaN
7,2,NaN,3,0
8,3,0,5,1
9,5,1,8,3
10,8,3,10,5
11,10,5,13,7
12,13,7,17,9
13,17,9,21,12
```

"quick" data analysis

`describe()`	applies basic statistical evaluations (count, quantiles, …)
`min() max()`	minimum and maximum values
`mean() median() mode()`	arithmetic average, median and mode
`var() std()`	variance and standard deviation
`sem()`	standard error mean
`skew()`	sample skewness
`kurt()`	kurtosis

"quick" data analysis 2

`describe()`

applies basic statistical evaluations (count, quantiles, …)

```
>>> df
   student  mathematics  science
0        1           20       21
1        2           13       13
2        3           30       36
3        4           14       19
4        5           16       20
5        6           12       15
6        7           25       23
7        8           29       27
8        9           17       14
9       10           26       29
```

```
>>> df.describe()
         student  mathematics    science
count  10.00000    10.000000  10.000000
mean    5.50000    20.200000  21.700000
std     3.02765     6.795423   7.288499
min     1.00000    12.000000  13.000000
25%     3.25000    14.500000  16.000000
50%     5.50000    18.500000  20.500000
75%     7.75000    25.750000  26.000000
max    10.00000    30.000000  36.000000
```

"quick" data analysis 3

```
>>> df.min()
student         1
mathematics    12
science        13
dtype: int64
```

```
>>> df.mean()
student         5.5
mathematics    20.2
science        21.7
dtype: float64
```

```
>>> df.std()
student        3.027650
mathematics    6.795423
science        7.288499
dtype: float64
```

selecting rows (subset observation)

- by position

  `df.iloc[fromrow:torow]`

```
>>> dfw = pd.read_csv("./WSRTdf.csv")
>>> dfw
      n type  alpha  value
0     5  TTT   0.05    NaN
1     5  TTT   0.01    NaN
2     5  OTT   0.05    0.0
3     5  OTT   0.01    NaN
4     6  TTT   0.05    0.0
..   ..  ...    ...    ...
99   29  OTT   0.01  110.0
100  30  TTT   0.05  137.0
101  30  TTT   0.01  109.0
102  30  OTT   0.05  151.0
103  30  OTT   0.01  120.0

[104 rows x 4 columns]
```

```
>>> dfw.iloc[1:5]
   n type  alpha  value
1  5  TTT   0.01    NaN
2  5  OTT   0.05    0.0
3  5  OTT   0.01    NaN
4  6  TTT   0.05    0.0
```

selecting rows (subset observation) 2 *filtering*

- by value in one …

```
df[(df[colname] == val)]
```

```
>>> dfw = pd.read_csv("./WSRTdf.csv")
>>> dfw
      n type  alpha  value
0     5  TTT   0.05    NaN
1     5  TTT   0.01    NaN
2     5  OTT   0.05    0.0
3     5  OTT   0.01    NaN
4     6  TTT   0.05    0.0
..   ..  ...    ...    ...
99   29  OTT   0.01  110.0
100  30  TTT   0.05  137.0
101  30  TTT   0.01  109.0
102  30  OTT   0.05  151.0
103  30  OTT   0.01  120.0

[104 rows x 4 columns]
```

```
>>> dfw[(dfw['alpha'] == ALPHA05)]
      n type  alpha  value
0     5  TTT   0.05    NaN
2     5  OTT   0.05    0.0
4     6  TTT   0.05    0.0
6     6  OTT   0.05    2.0
8     7  TTT   0.05    2.0
10    7  OTT   0.05    3.0
12    8  TTT   0.05    3.0
14    8  OTT   0.05    5.0
16    9  TTT   0.05    5.0
18    9  OTT   0.05    8.0
```

selecting rows (subset observation) 3
filtering

... or more columns

```
df[(df[colname] == val)
  & df[colname2] == val2)
```

```
>>> dfw = pd.read_csv("./WSRTdf.csv")
>>> dfw
     n type  alpha  value
0    5  TTT   0.05    NaN
1    5  TTT   0.01    NaN
2    5  OTT   0.05    0.0
3    5  OTT   0.01    NaN
4    6  TTT   0.05    0.0
..  ..  ...    ...    ...
99  29  OTT   0.01  110.0
100 30  TTT   0.05  137.0
101 30  TTT   0.01  109.0
102 30  OTT   0.05  151.0
103 30  OTT   0.01  120.0

[104 rows x 4 columns]
```

```
>>> dfw[(dfw['alpha'] == ALPHA05) &
...      (dfw['type'] == TWOTAILED)]
     n type  alpha  value
0    5  TTT   0.05    NaN
4    6  TTT   0.05    0.0
8    7  TTT   0.05    2.0
12   8  TTT   0.05    3.0
16   9  TTT   0.05    5.0
20  10  TTT   0.05    8.0
24  11  TTT   0.05   10.0
28  12  TTT   0.05   13.0
32  13  TTT   0.05   17.0
```

selecting values

select row within the dataframe by using index

```
>>> df.iloc[0]
SBP_before    127.0
SBP_after     126.0
Difference      1.0
AbsDiff         1.0
Ranks           1.0
R+              1.0
Name: 13, dtype: float64
```

```
>>> df.iloc[2]
SBP_before    130.0
SBP_after     132.0
Difference     -2.0
AbsDiff         2.0
Ranks           2.5
R+              0.0
Name: 14, dtype: float64
```

```
>>> df
    SBP_before  SBP_after  Difference  AbsDiff  Ranks    R+
13         127        126           1        1    1.0   1.0
1          132        134          -2        2    2.5   0.0
14         130        132          -2        2    2.5   0.0
5          127        130          -3        3    4.0   0.0
3          120        124          -4        4    6.0   0.0
11         136        140          -4        4    6.0   0.0
9          132        128           4        4    6.0   6.0
6          136        130           6        6    8.0   8.0
12         128        135          -7        7   10.0   0.0
0          125        118           7        7   10.0  10.0
7          139        132           7        7   10.0  10.0
2          138        130           8        8   12.5  12.5
8          131        123           8        8   12.5  12.5
10         135        126           9        9   14.0  14.0
4          125        105          20       20   15.0  15.0
```

selecting values 2

1. specify row and column
   uncommon, based on a position …

```
>>> df.iloc[2].SBP_after
132.0
>>> df.SBP_after.iloc[2]
132
```

row, column

```
>>> df.iloc[2,1]
132
```

```
>>> df
    SBP_before  SBP_after  Difference  AbsDiff  Ranks   R+
0           13        127         126        1    1.0  1.0
1            1        132         134       -2    2.5  0.0
2           14        130         132       -2    2.5  0.0
5          127        130          -3        3    4.0  0.0
3          120        124          -4        4    6.0  0.0
11         136        140          -4        4    6.0  0.0
9          132        128           4        4    6.0  6.0
6          136        130           6        6    8.0  8.0
12         128        135          -7        7   10.0  0.0
0          125        118           7        7   10.0 10.0
7          139        132           7        7   10.0 10.0
2          138        130           8        8   12.5 12.5
8          131        123           8        8   12.5 12.5
10         135        126           9        9   14.0 14.0
4          125        105          20       20   15.0 15.0
```

0 1 2 …

selecting values 3

unless we have a selected row ... and we want a value

```
>>> dfw[(dfw['type'] == TWOTAILED) & (dfw['alpha'] == ALPHA05) & (dfw['n'] == number_of_samples)]
      n type  alpha  value
40   15  TTT   0.05   25.0
```

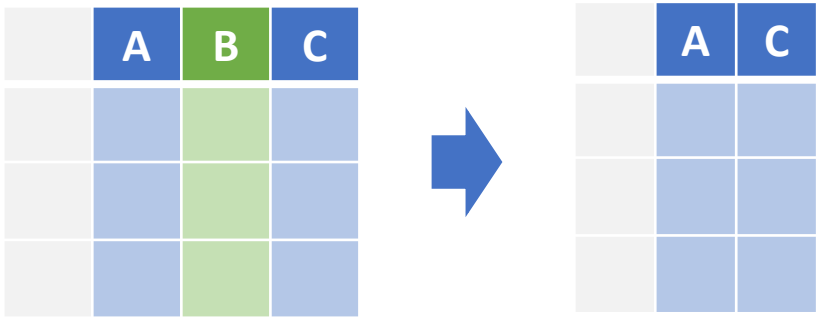
this column

```
>>> dfw[(dfw['type'] == TWOTAILED) & (dfw['alpha'] == ALPHA05) & (dfw['n'] == number_of_samples)]['value'].iloc[0]
25.0
```

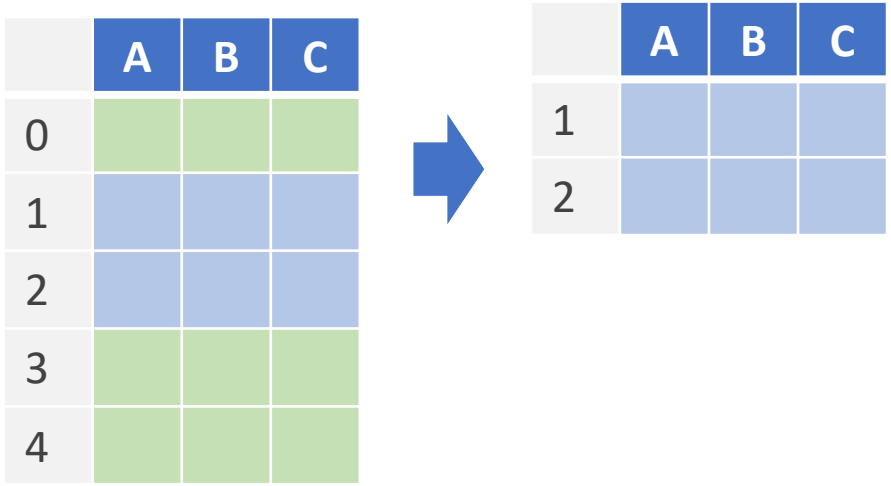
there is only one row

slicing

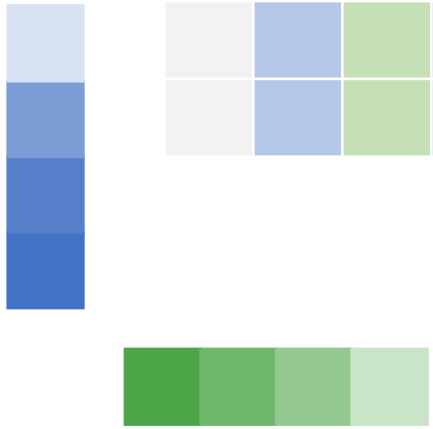
- secting a subset of rows/columns
  - by name
  - by position




```
df[['A', 'C']]
```




```
df[2:3]
```



slicing 2

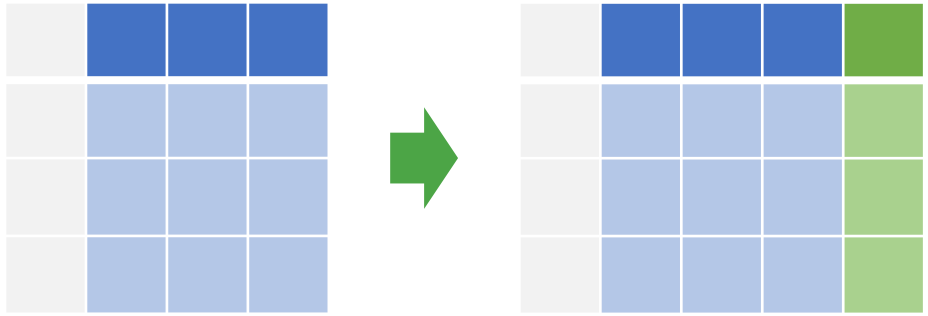
	A	B	C
0			
1			
2			
3			
4			
5			
6			

	A	B
1		
2		
3		
4		

```
df.loc[2:6,['A', 'B']]
```

creating new data

```
df['NewColumnName'] = df.ExistingColumn operator variable
```



```
>>> df['Difference'] = df.SBP_before - df.SBP_after
>>> df
    SBP_before  SBP_after  Difference
0          125        118           7
1          132        134          -2
2          138        130           8
3          120        124          -4
4          125        105          20
5          127        130          -3
6          136        130           6
7          139        132           7
8          131        123           8
9          132        128           4
10         135        126           9
11         136        140          -4
12         128        135          -7
13         127        126           1
14         130        132          -2
```

```
df['Difference'] = df.SBP_before - df.SBP_after
```

creating new data 2

```
df['NewColumnName'] = df.ExistingColumn operator variable
```

```
df['AbsDiff'] = abs(df.Difference)
```

```
>>> df['AbsDiff'] = abs(df.Difference)
>>> df
    SBP_before  SBP_after  Difference  AbsDiff
0          125        118           7        7
1          132        134          -2        2
2          138        130           8        8
3          120        124          -4        4
4          125        105          20       20
5          127        130          -3        3
6          136        130           6        6
7          139        132           7        7
8          131        123           8        8
9          132        128           4        4
10         135        126           9        9
11         136        140          -4        4
12         128        135          -7        7
13         127        126           1        1
14         130        132          -2        2
```

creating new data, selectively

```
df['R+'] = df['Ranks']*(df['Difference'] > 0)
```

```
>>> df['R+'] = df['Ranks']*(df['Difference'] > 0)
>>> df
    SBP_before  SBP_after  Difference  AbsDiff  Ranks   R+
13         127        126           1        1    1.0  1.0
1          132        134          -2        2    2.5  0.0
14         130        132          -2        2    2.5  0.0
5          127        130          -3        3    4.0  0.0
3          120        124          -4        4    6.0  0.0
11         136        140          -4        4    6.0  0.0
9          132        128           4        4    6.0  6.0
6          136        130           6        6    8.0  8.0
12         128        135          -7        7   10.0  0.0
0          125        118           7        7   10.0 10.0
7          139        132           7        7   10.0 10.0
2          138        130           8        8   12.5 12.5
8          131        123           8        8   12.5 12.5
10         135        126           9        9   14.0 14.0
4          125        105          20       20   15.0 15.0
```

1 when true
0 when false

sorting

list of columns

`sort_values(by,ascending)`: sorts with respect to values in columns, in ascending (default) or descending order

it is not "in place"

```
df = df.sort_values("AbsDiff")
```

sorting 2

```
df = df.sort_values("AbsDiff")
```

```
>>> df = df.sort_values("AbsDiff")
>>> df
    SBP_before  SBP_after  Difference  AbsDiff
13         127        126           1        1
1          132        134          -2        2
14         130        132          -2        2
5          127        130          -3        3
3          120        124          -4        4
9          132        128           4        4
11         136        140          -4        4
6          136        130           6        6
0          125        118           7        7
7          139        132           7        7
12         128        135          -7        7
2          138        130           8        8
8          131        123           8        8
10         135        126           9        9
4          125        105          20       20
```

```
>>> df = df.sort_values("Difference")
>>> df
    SBP_before  SBP_after  Difference  AbsDiff
12         128        135          -7        7
3          120        124          -4        4
11         136        140          -4        4
5          127        130          -3        3
1          132        134          -2        2
14         130        132          -2        2
13         127        126           1        1
9          132        128           4        4
6          136        130           6        6
0          125        118           7        7
7          139        132           7        7
2          138        130           8        8
8          131        123           8        8
10         135        126           9        9
4          125        105          20       20
```

```
df = df.sort_values("Difference")
```

sorting 3

sort by this

then by this

```
df = df.sort_values(["AbsDiff", "Difference")
```

```
>>> df = df.sort_values(["AbsDiff", "Difference"])
>>> df
    SBP_before  SBP_after  Difference  AbsDiff
13         127        126           1        1
1          132        134          -2        2
14         130        132          -2        2
5          127        130          -3        3
3          120        124          -4        4
11         136        140          -4        4
9          132        128           4        4
6          136        130           6        6
12         128        135          -7        7
0          125        118           7        7
7          139        132           7        7
2          138        130           8        8
8          131        123           8        8
10         135        126           9        9
4          125        105          20       20
```

group of data / aggregation

```
groupby(colnames).aggr()
```

- organize data into groups based on criteria
- perform computation on grouped data
  (eg. minimum with respect to grouped information)
- similar to `dplyr()` function in R

group of data / aggregation 2

groupby(colnames).aggr()

```
>>> df.groupby('AbsDiff').size()
AbsDiff
1     1
2     2
3     1
4     3
6     1
7     3
8     2
9     1
20    1
dtype: int64
```

```
>>> df
    SBP_before  SBP_after  Difference  AbsDiff  Ranks    R+
13         127        126           1        1    1.0   1.0
1          132        134          -2        2    2.5   0.0
14         130        132          -2        2    2.5   0.0
5          127        130          -3        3    4.0   0.0
3          120        124          -4        4    6.0   0.0
11         136        140          -4        4    6.0   0.0
9          132        128           4        4    6.0   6.0
6          136        130           6        6    8.0   8.0
12         128        135          -7        7   10.0   0.0
0          125        118           7        7   10.0  10.0
7          139        132           7        7   10.0  10.0
2          138        130           8        8   12.5  12.5
8          131        123           8        8   12.5  12.5
10         135        126           9        9   14.0  14.0
4          125        105          20       20   15.0  15.0
```

group of data / aggregation 3

groupby(colnames).aggr()

```
>>> df.groupby('AbsDiff')[['SBP_before','SBP_after']].mean()
          SBP_before   SBP_after
AbsDiff
1         127.000000  126.000000
2         131.000000  133.000000
3         127.000000  130.000000
4         129.333333  130.666667
6         136.000000  130.000000
7         130.666667  128.333333
8         134.500000  126.500000
9         135.000000  126.000000
20        125.000000  105.000000
```

```
>>> df
    SBP_before  SBP_after  Difference  AbsDiff  Ranks    R+
13         127        126           1        1    1.0   1.0
1          132        134          -2        2    2.5   0.0
14         130        132          -2        2    2.5   0.0
5          127        130          -3        3    4.0   0.0
3          120        124          -4        4    6.0   0.0
11         136        140          -4        4    6.0   0.0
9          132        128           4        4    6.0   6.0
6          136        130           6        6    8.0   8.0
12         128        135          -7        7   10.0   0.0
0          125        118           7        7   10.0  10.0
7          139        132           7        7   10.0  10.0
2          138        130           8        8   12.5  12.5
8          131        123           8        8   12.5  12.5
10         135        126           9        9   14.0  14.0
4          125        105          20       20   15.0  15.0
```

group of data / aggregation 4

groupby(colnames).aggr()

```
>>> df.groupby('AbsDiff')[['SBP_before','SBP_after']].describe()
        SBP_before                                                    SBP_after
             count        mean       std    min     25%    50%     75%    max      count        mean       std    min     25%    50%     75%    max
AbsDiff
1              1.0  127.000000       NaN  127.0  127.00  127.0  127.00  127.0        1.0  126.000000       NaN  126.0  126.00  126.0  126.00  126.0
2              2.0  131.000000  1.414214  130.0  130.50  131.0  131.50  132.0        2.0  133.000000  1.414214  132.0  132.50  133.0  133.50  134.0
3              1.0  127.000000       NaN  127.0  127.00  127.0  127.00  127.0        1.0  130.000000       NaN  130.0  130.00  130.0  130.00  130.0
4              3.0  129.333333  8.326664  120.0  126.00  132.0  134.00  136.0        3.0  130.666667  8.326664  124.0  126.00  128.0  134.00  140.0
6              1.0  136.000000       NaN  136.0  136.00  136.0  136.00  136.0        1.0  130.000000       NaN  130.0  130.00  130.0  130.00  130.0
7              3.0  130.666667  7.371115  125.0  126.50  128.0  133.50  139.0        3.0  128.333333  9.073772  118.0  125.00  132.0  133.50  135.0
8              2.0  134.500000  4.949747  131.0  132.75  134.5  136.25  138.0        2.0  126.500000  4.949747  123.0  124.75  126.5  128.25  130.0
9              1.0  135.000000       NaN  135.0  135.00  135.0  135.00  135.0        1.0  126.000000       NaN  126.0  126.00  126.0  126.00  126.0
20             1.0  125.000000       NaN  125.0  125.00  125.0  125.00  125.0        1.0  105.000000       NaN  105.0  105.00  105.0  105.00  105.0
```

ranking data

`rank(method="…")`:

returns a rank of every respective index of a series passed.
The rank is returned on the basis of position after sorting

there are a few methods available

```
df["AbsDiff"].rank()
```

ranking data 2

extra column

```
df["AbsDiff"].rank()  ➡  df["Ranks"] = df["AbsDiff"].rank()
```

with sorting

	SBP_before	SBP_after	Difference	AbsDiff	Ranks
13	127	126	1	1	1.0
1	132	134	-2	2	2.5
14	130	132	-2	2	2.5
5	127	130	-3	3	4.0
3	120	124	-4	4	6.0
11	136	140	-4	4	6.0
9	132	128	4	4	6.0
6	136	130	6	6	8.0
12	128	135	-7	7	10.0
0	125	118	7	7	10.0
7	139	132	7	7	10.0
2	138	130	8	8	12.5
8	131	123	8	8	12.5
10	135	126	9	9	14.0
4	125	105	20	20	15.0

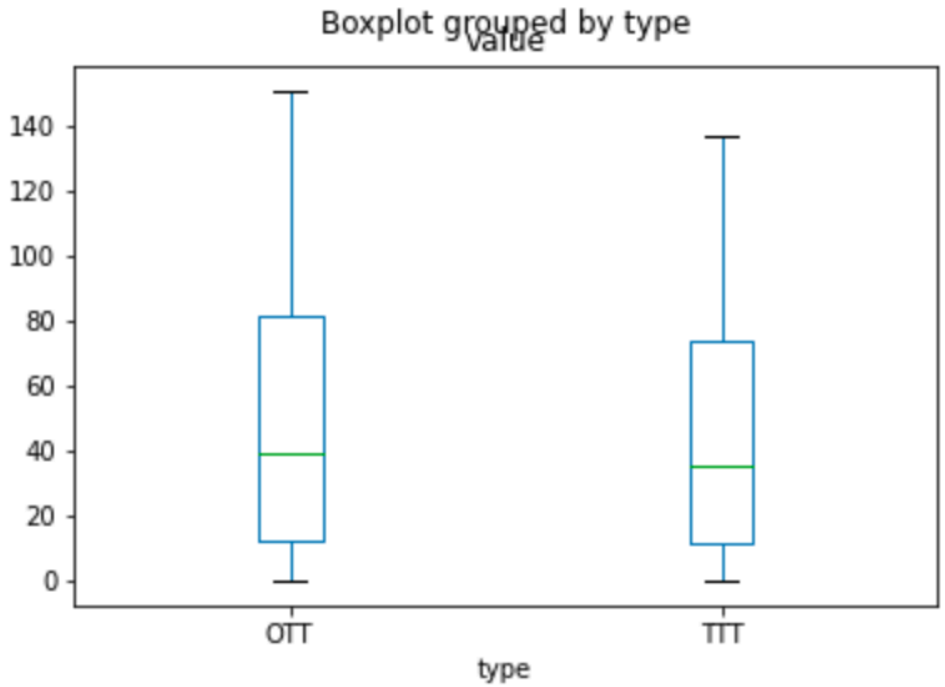
without sorting

	SBP_before	SBP_after	Difference	AbsDiff	Ranks
0	125	118	7	7	10.0
1	132	134	-2	2	2.5
2	138	130	8	8	12.5
3	120	124	-4	4	6.0
4	125	105	20	20	15.0
5	127	130	-3	3	4.0
6	136	130	6	6	8.0
7	139	132	7	7	10.0
8	131	123	8	8	12.5
9	132	128	4	4	6.0
10	135	126	9	9	14.0
11	136	140	-4	4	6.0
12	128	135	-7	7	10.0
13	127	126	1	1	1.0
14	130	132	-2	2	2.5

to visualization ...

pandas methods

```
dfw = pd.read_csv("./WSRTdf.csv")
boxplot = dfw.boxplot(by='type', column='value')
```




```
>>> dfw = pd.read_csv("./WSRTdf.csv")
>>> dfw
      n type  alpha  value
0     5  TTT   0.05    NaN
1     5  TTT   0.01    NaN
2     5  OTT   0.05    0.0
3     5  OTT   0.01    NaN
4     6  TTT   0.05    0.0
..   ..  ...    ...    ...
99   29  OTT   0.01  110.0
100  30  TTT   0.05  137.0
101  30  TTT   0.01  109.0
102  30  OTT   0.05  151.0
103  30  OTT   0.01  120.0

[104 rows x 4 columns]
```

to seaborn ...