Problem 1: Descriptive Statistics and Probability Theory: Real Data on CEO Compensation

In [274]:

```
import numpy as np
import pandas as pd
from scipy.stats import trim_mean
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
sns.set_style("darkgrid")
import warnings
warnings.filterwarnings('ignore')

ceo = pd.read_excel("ceo.xls")
num_observations = len(ceo)
salary = ceo.salary
ceo.head()
```

Out[274]:

	salary	totcomp	tenure	age	sales	profits	assets	Unnamed: 7
0	3030	8138	7	61	161315.0	2956.0	257389.0	NaN
1	6050	14530	0	51	144416.0	22071.0	237545.0	NaN
2	3571	7433	11	63	139208.0	4430.0	49271.0	NaN
3	3300	13464	6	60	100697.0	6370.0	92630.0	NaN
4	10000	68285	18	63	100469.0	9296.0	355935.0	NaN

1

In the pre-reading you/we have discussed tools and methods for visualizing data and computing some simple characteristic measures. Our aim here is to apply all the basic techniques and to draw correct conclusions. The file ceo.xls contains data on the CEO compensations and some additional variables listed below

```
salary  = 1999 salary + bonuses in 1000 USD
totcomp = 1999 CEO total compensation
tenure  = # of years as CEO (=0 if less than 6 months)
age     = age of CEO
sales   = total 1998 sales revenue of firm i
profits = 1998 profits for firm i
assets  = total assets of firm i in 1998
```

Our aim is to evaluate the data set with basic tools.

1.a

For the variable salary compute the common location measures:

- [x] mean
- [x] 10%trimmed mean
- [x] median
- [x] upper and lower quartiles
- [x] the upper and lower 10%quantiles.
- [x] Give an economic interpretation for every location measure.

In [275]:

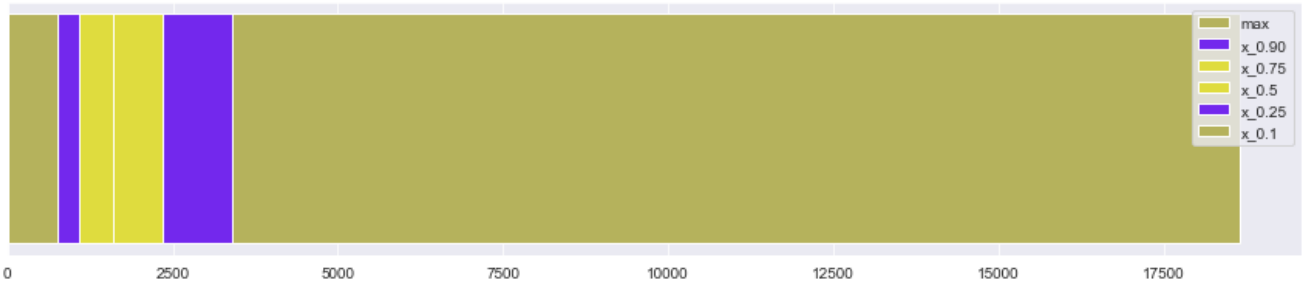
```
def p_quantile(x, p):
    from math import floor
    """
    0 <= p <= 1
```

```
    """
    x = sorted(x)
    n = len(x)

    m = n * p
    if m % 1 == 0:
        m = int(m)
        return (x[m-1] + x[m]) / 2
    else:
        return x[int(floor(m))]

def p_quantile_pandas(x, p):
    from math import floor
    """
    0 <= p <= 1
    """
    x = sorted(x)
    n = len(x)

    m = (n - 1) * p
    print('(n-1)p = {}'.format(m))
    if m % 1 != 0:
        m = int(m)
        return (x[m] + x[m+1]) / 2
    else:
        return x[int(m)]
```

In [276]:

```
median = salary.median()
trimmean = trim_mean(salary, 0.1)
q25 = p_quantile(salary, .25) # salary.quantile(.25, interpolation='midpoint')
q75 = p_quantile(salary, .75) # salary.quantile(.75, interpolation='midpoint')
q10 = p_quantile(salary, .1)  # salary.quantile(.1, interpolation='midpoint'))
q90 = p_quantile(salary, .9)  # salary.quantile(.9, interpolation='midpoint'))
IQR = q75 - q25
# lower_fence = q25 - 1.5 * IQR
# upper_fence = q75 + 1.5 * IQR

# visualize quantiles
mpl.rcParams['figure.figsize'] = (15,3)

def rc():
    return '#{:02X}{:02X}{:02X}'.format(*np.random.randint(0,255,3))

# quantiles_names = ['lower 10%', 'lower (25%)', 'median', 'upper (75%)', 'upper 10% (90%)']
quantiles_names = ['x_0.1', 'x_0.25', 'x_0.5', 'x_0.75', 'x_0.90', 'max']
quantiles = [q10, q25, median, q75, q90, salary.max()]
ind = np.arange(1)
n = len(quantiles)
colors = [rc(), rc(), rc()]
colors.extend(colors[::-1])
p = [0] * n

for i, q in enumerate(quantiles):
    if i == 0:
        p[i] = plt.barh(ind, (q), color=colors[i])
    else:
        p[i] = plt.barh(ind, (q), color=colors[i], left=(quantiles[i-1]))

plt.yticks([])
plt.title('')
plt.legend([p_[0] for p_ in p][::-1], quantiles_names[::-1], loc='upper right')
plt.show()

print('minimum value: {} | maximum value: {}'.format(salary.min(), salary.max()))
print('mean:\t\t\t {:.2f}'.format(salary.mean()))
print('\t as the only location measure it tells a few about a distribution, \n\
       in terms of economics it allows roughly compare salaries of CEOs \n\
       with some other positions, but it should be used with other measures \n\
       (described below) if we want to analyse salaries in details.\n')

print('10%-trimmed mean:\t {:.2f}'.format(trimmean))
print('\t 10 % of salaries extreme values are reduced and the mean is \n\
       significantly diminished reflecting some concentration of very \n\
       big salary values (there should be either a gap between these big \n\
```

```
      big salary values (there should be either a gap between these big \n\
      and less salaries or a sparse allocation of salaries-outliers on big \n\
      scales).\n')


print('median:\t\t\t {:.2f}'.format(median))
print('\t this value is close to the 10%-trimmed mean and since the \n\
      two measures are robust to outliers we can conclude that the \n\
      `center` of salaries is near {:.0f}-{:.0f}.\n'.format(median, trimmean))

print('lower quartile:\t\t {:.2f}'.format(q25))
print('upper quartile:\t\t {:.2f}'.format(q75))
print('\t the quartiles are almost equidistant from the `center` described \n\
      above (with interquartile range {:.0f}). So, the most concentration \n\
      of salaries is between {:.0f} & {:.0f}.\n'.format(IQR, q25, q75))

print('lower 10%-quantile:\t {:.2f}'.format(q10))
print('upper 10%-quantile:\t {:.2f}'.format(q90))
print('\t upper quantile is far away from the maximum salary accepting hypothesis \n\
      about the existence of outliers or very big salaries. Lower quantile \n\
      indicates that there are few/no outliers on the left side. So, usually \n\
      CEOs have salaries between {:.0f} & {:.0f}.'.format(q10, q90))
```




```
minimum value: 100 | maximum value: 15250
mean:    2027.52
  as the only location measure it tells a few about a distribution,
      in terms of economics it allows roughly compare salaries of CEOs
      with some other positions, but it should be used with other measures
      (described below) if we want to analyse salaries in details.

10%-trimmed mean:  1710.09
  10 % of salaries extreme values are reduced and the mean is
      significantly diminished reflecting some concentration of very
      big salary values (there should be either a gap between these big
      and less salaries or a sparse allocation of salaries-outliers on big
      scales).

median:    1600.00
  this value is close to the 10%-trimmed mean and since the
      two measures are robust to outliers we can conclude that the
      `center` of salaries is near 1600-1710.

lower quartile:   1083.00
upper quartile:   2350.00
  the quartiles are almost equidistant from the `center` described
      above (with interquartile range 1267). So, the most concentration
      of salaries is between 1083 & 2350.

lower 10%-quantile:  750.00
upper 10%-quantile:  3400.00
  upper quantile is far away from the maximum salary accepting hypothesis
      about the existence of outliers or very big salaries. Lower quantile
      indicates that there are few/no outliers on the left side. So, usually
      CEOs have salaries between 750 & 3400.
```

1.b

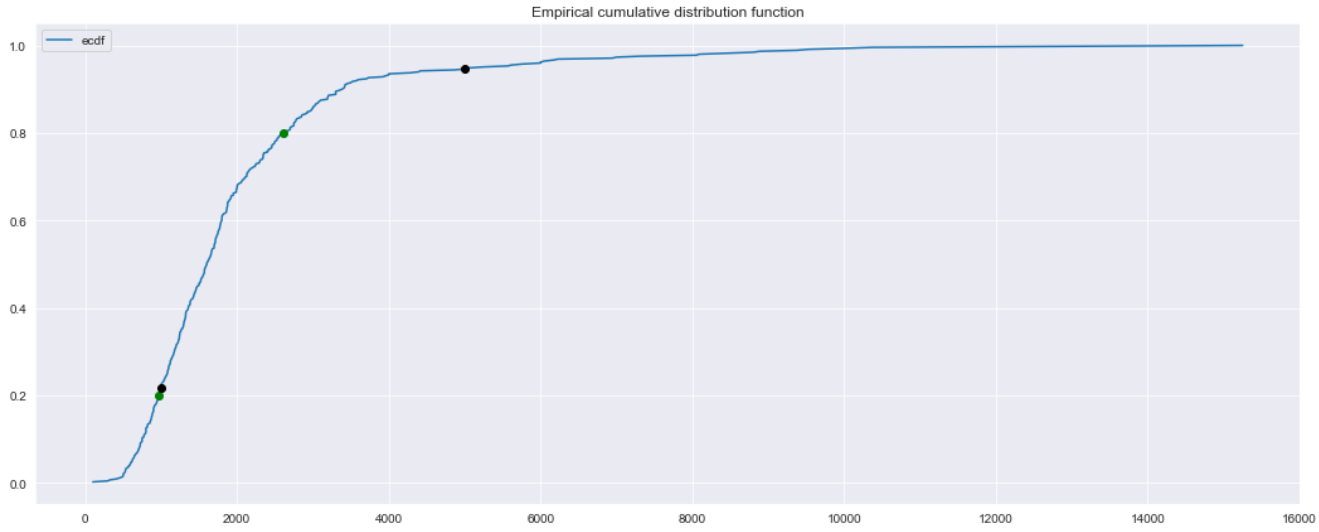
Plot the empirical cumulative distribution function.

- [x] Compute and explain in economic terms the following quantities
- [x] $F^{-1}(0.2)$ and $F^{-1}(0.8)$
- [x] $F(1000)$ and $1-F(5000)$

In [277]:

```
mpl.rcParams['figure.figsize'] = (18, 7)

from statsmodels.distributions.empirical_distribution import ECDF
cdf = ECDF(salary)
plt.plot(cdf.x, cdf.y, label='ecdf')
plt.legend()
plt.title('Empirical cumulative distribution function')


def CDF(cdf, i):
    return cdf.y[list(cdf.x).index(i)]

F_1000 = CDF(cdf, 1000)
F_5000 = CDF(cdf, 5000)

F_inverse_02 = p_quantile(salary, 0.2)
F_inverse_08 = p_quantile(salary, 0.8)

print('F^(-1)(0.2) = {:.2f}'.format(F_inverse_02))
print('F^(-1)(0.8) = {:.2f}'.format(F_inverse_08))
print('\tthe values of the inverse ecdf or the quantiles are close to \n\
       quantiles described above (these are between 10% and 25%) and may \n\
       also represent the boundaries of the concentration of the data.\n\
       So it is between {:.0f} & {:.0f} (black points).'.format(F_inverse_02, F_inverse_08))
print('F(1000) = {:.3f}'.format(F_1000))
print('\tthis means that {:.0f}% of salaries data is less than 1000.'.format(F_1000 * 100))
print('1 - F(5000) = {:.3f}'.format(1 - F_5000))
print('\tthis means that {:.0f}% of salaries data is more than 5000.'.format((1 - F_5000) * 100))
plt.plot(F_inverse_02, 0.2, 'go')
plt.plot(F_inverse_08, 0.8, 'go')
plt.plot(1000, F_1000, 'ko')
plt.plot(5000, F_5000, 'ko')
plt.show()
# bins = 100
# salary.hist(cumulative=True, density=1, bins=bins)
# plt.show()

# pdf, bin_edges = np.histogram(salary, normed=True, bins=bins)
# plt.plot(bin_edges[:-1], pdf)
```

```
F^(-1)(0.2) = 975.00
F^(-1)(0.8) = 2615.00
 the values of the inverse ecdf or the quantiles are close to
       quantiles described above (these are between 10% and 25%) and may
       also represent the boundaries of the concentration of the data.
       So it is between 975 & 2615 (black points).
F(1000) = 0.217
 this means that 22% of salaries data is less than 1000.
1 - F(5000) = 0.054
 this means that 5% of salaries data is more than 5000.
```



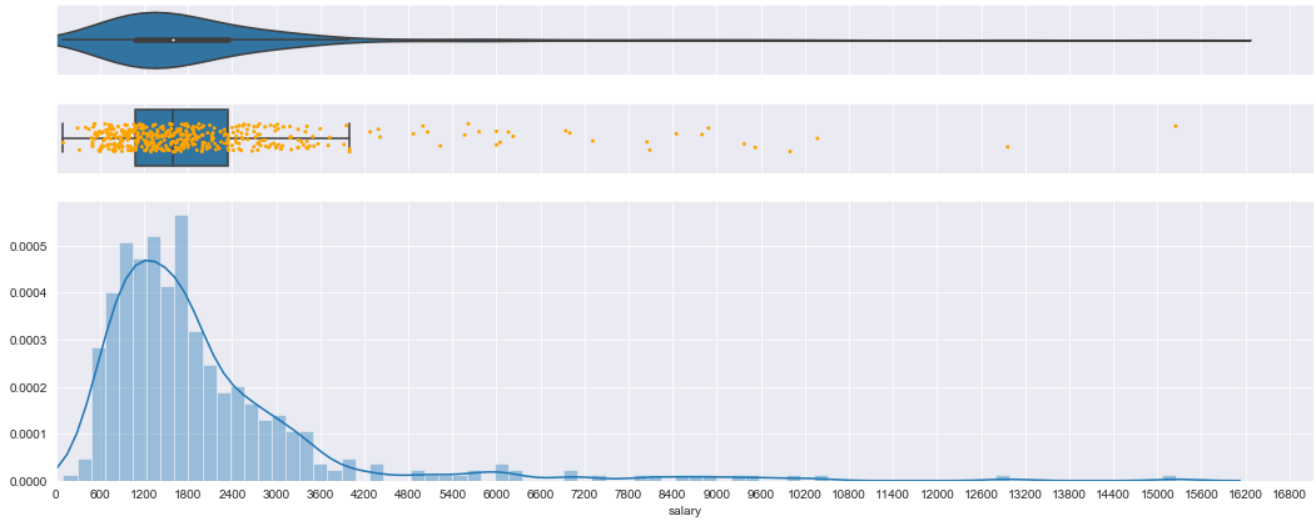
1.c

- [x] Plot the histogram of salary and the Box-plot (or violin-plot).

In [278]:

```
def represent_distribution(sample, xmin=None, nx=None):
    mpl.rcParams['figure.figsize'] = (18, 7)

    f, (ax_viol, ax_box, ax_hist) = plt.subplots(3, sharex=True,
                                                 gridspec_kw={"height_ratios": (.15, .15, .60)})

    sns.violinplot(sample, showmeans=True, ax=ax_viol)
    ax_viol.set(xlabel='')

    sns.boxplot(sample, fliersize=0, whis=1.5, ax=ax_box)
    sns.stripplot(sample, color="orange", jitter=0.2, size=3, ax=ax_box)
    ax_box.set(xlabel='')

    sns.distplot(sample, ax=ax_hist, bins=80)
    if nx is not None:
        ax_hist.xaxis.set_major_locator(plt.MaxNLocator(nx))
    if xmin is not None:
        ax_hist.set_xlim(left=xmin)

    print('Number of observations: {}'.format(len(salary)))
    plt.show()

represent_distribution(salary, 0, 30)

salary_skew = salary.skew()
print('Sample skewness: {:.3f} {}'.format(salary_skew, '> 0 '
                                          if salary_skew > 0 else '< 0'))
```

```
Number of observations: 447
```




```
Sample skewness: 3.402 > 0
```

- [x] What can be concluded about the distribution of the data?

> The salaries are really concentrated between 400 & 3600 and have a heaavy tail of extremely big salaries. The distribution form is similar to Chi-Square.

- [x] Are the location measures computed above still appropriate?

> In our case the mean, which is equal to 2028, is inappropriate, because of the outliers it's shifted to greater value than the real "center" of the data. 10%-trimmed mean and median (1710 and 1600 accordingly) are relatively close and represent, where the data is really concentrated. Other quantiles also give a useful information about the location of data (10% upper and lower quantiles are close to the data concantraion boundaries; the same with quartiles but with aa very high concentration).

- [x] Compute and discuss an appropriate measure of symmetry.

> The sample skewness is equal to $3.4 \geq 0$ meaning that the distribution is right-skewed which is clear from the representation below (the peak is shifted to the left part). One can interpret it as (since there are more CEOs with salary a little less than the peak than CEOs with salary more than the peak) some factors preventing from getting a larger salary after achieving the median may exist, and it's getting stronger with moving the salary forward.

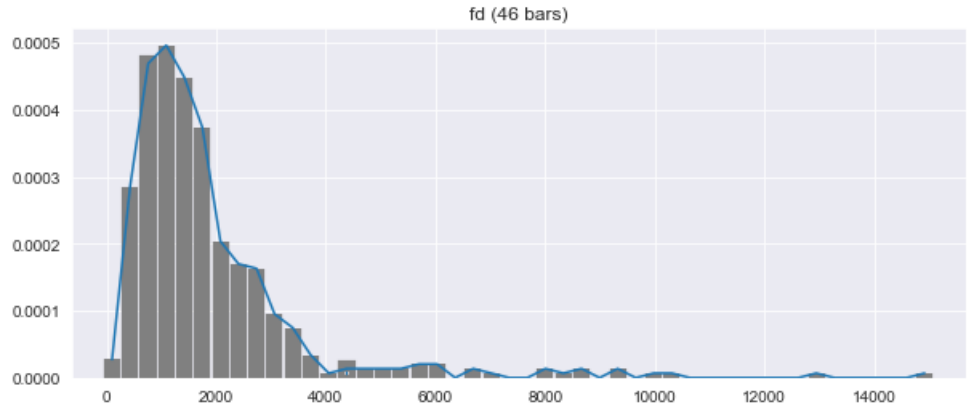
1.d

- [x] Check which method is used in your software to compute the optimal bandwidth (or the number of bars) in the histogram.

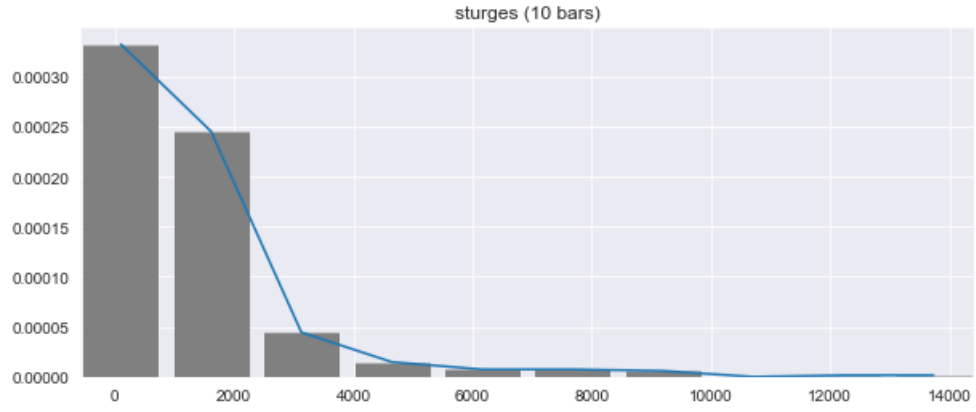
In [279]:

```
mpl.rcParams['figure.figsize'] = (10, 4)
for bins in ['fd', 'sturges', 'doane', 'scott', 'rice', 'sqrt',
             500, 8, 30]:
    plt.title(bins)
    pdf, bin_edges = np.histogram(salary, normed=True, bins=bins)
    n_bins = len(bin_edges) - 1
    if isinstance(bins, int):
        print('Bandwidth = {}. Histogram:'.format(bins))
        plt.title('{} bars'.format(n_bins))
    else:
        print('For the `{}` method the optimal bandwidth is {}. Histogram:'.format(bins, n_bins))
        plt.title('{} ({} bars)'.format(bins, n_bins))
    plt.plot(bin_edges[:-1], pdf)
    plt.bar(bin_edges[:-1], pdf, linewidth=10*47/n_bins, edgecolor='gray')
    plt.show()
```

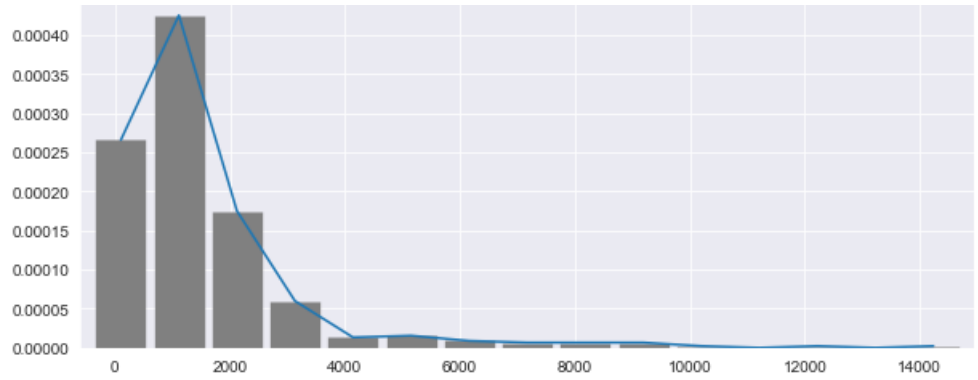
```
For the `fd` method the optimal bandwidth is 46. Histogram:
```



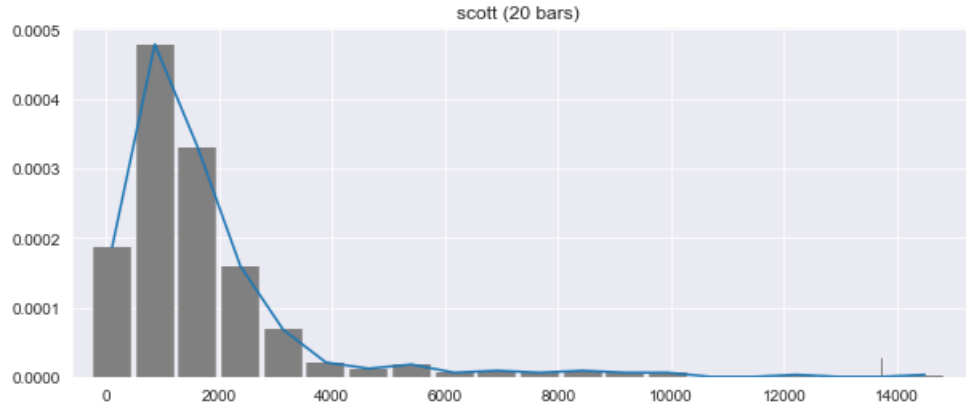

```
For the `sturges` method the optimal bandwidth is 10. Histogram:
```



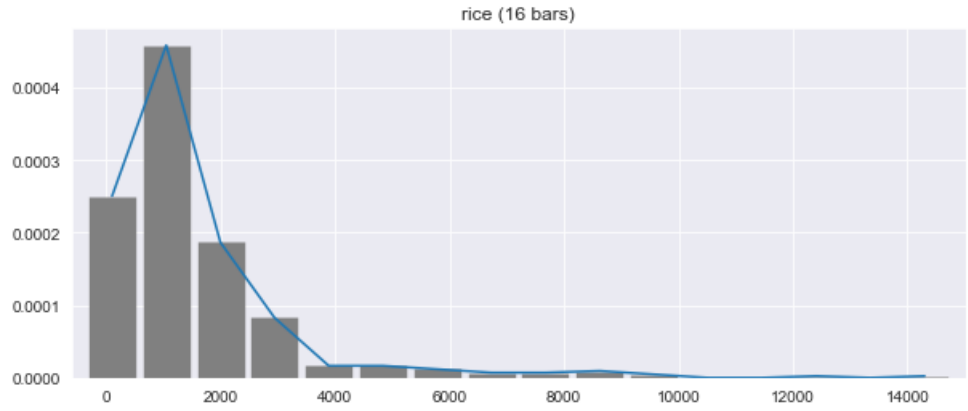

```
For the `doane` method the optimal bandwidth is 15. Histogram:
```



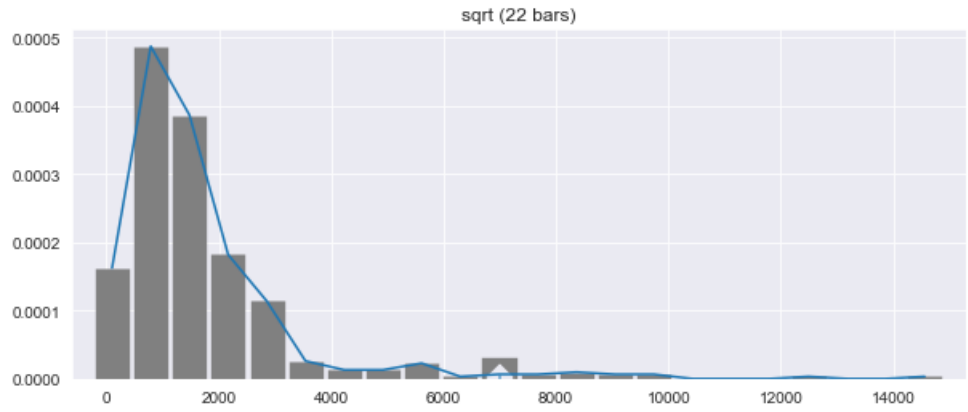
For the `scott` method the optimal bandwidth is 20. Histogram:



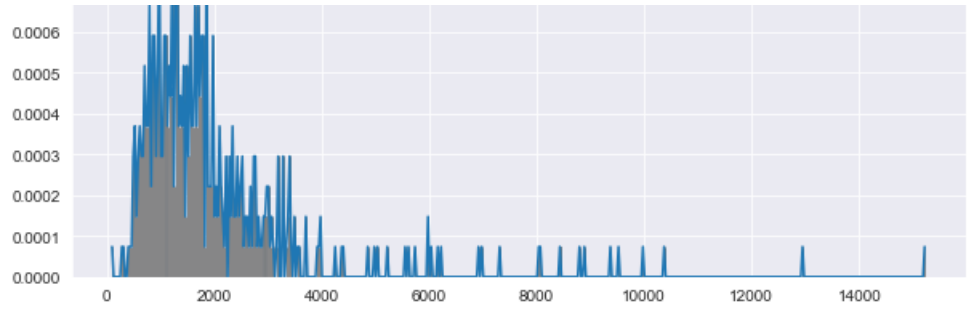
For the `rice` method the optimal bandwidth is 16. Histogram:



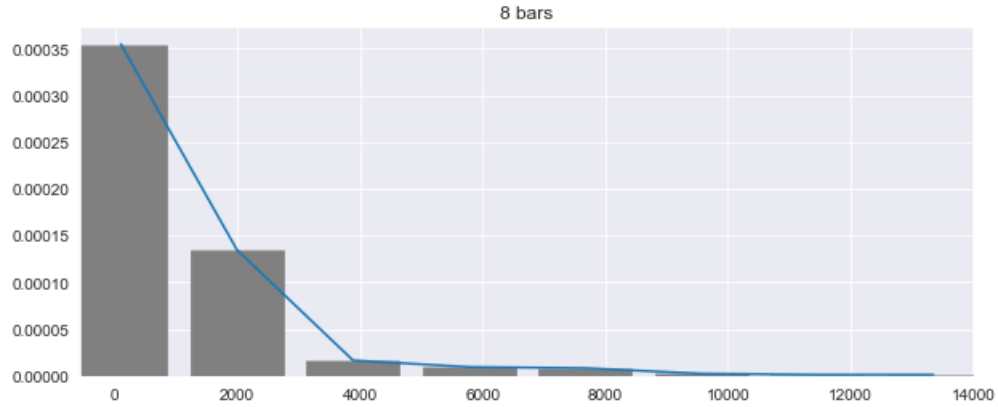
For the `sqrt` method the optimal bandwidth is 22. Histogram:



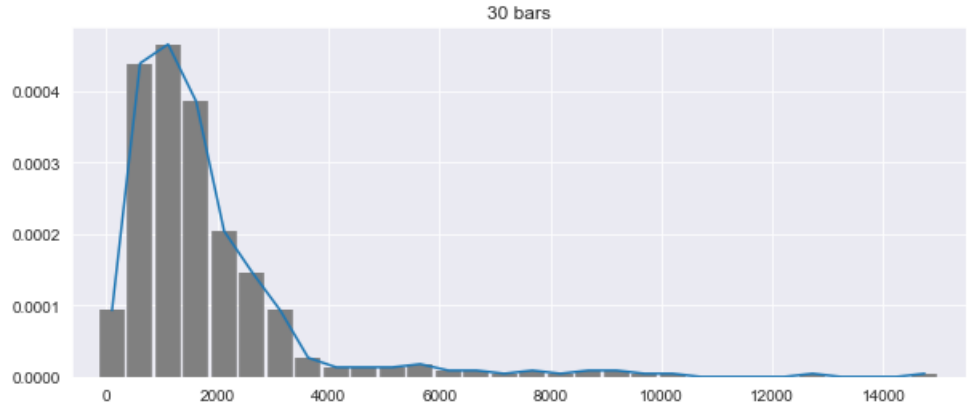
Bandwidth = 500. Histogram:




```
Bandwidth = 8. Histogram:
```




```
Bandwidth = 30. Histogram:
```



- [x] Describe it shortly here. Make plots of too detailed and too rough histograms. What can we learn from these figures?

> In numpy there are 6 methods ('fd': 46, 'sturges': 10, 'doane': 15, 'scott': 20, 'rice': 16, 'sqrt': 22) and 1 auto which just chooses maximum of 'sturges' and 'fd'. In my opinion the optimal bandwidth should be something between 22, defined by 'sqrt' method (square root of data size) and 46, defined by 'fd' method (Robust Freedman Diaconis estimator which takes into account data variability and data size), e.g. 30 -- it is neither too detailed nor too rough.
>
> For too detailed histogram I took the bandwidth = 500. With such "detailed" view the sense of the distribution is lost. The relative frequency of very small groups (or unique values) won't give us the useful information due to the restricted sample size and the sample basically (in the real world it will never be ideally representative).
>
> For too rough histogram I took the bandwidth = 8. It accumulates too much information in one, meaning that the ranges of bins are so big that we are getting a really rough picture.

1.e

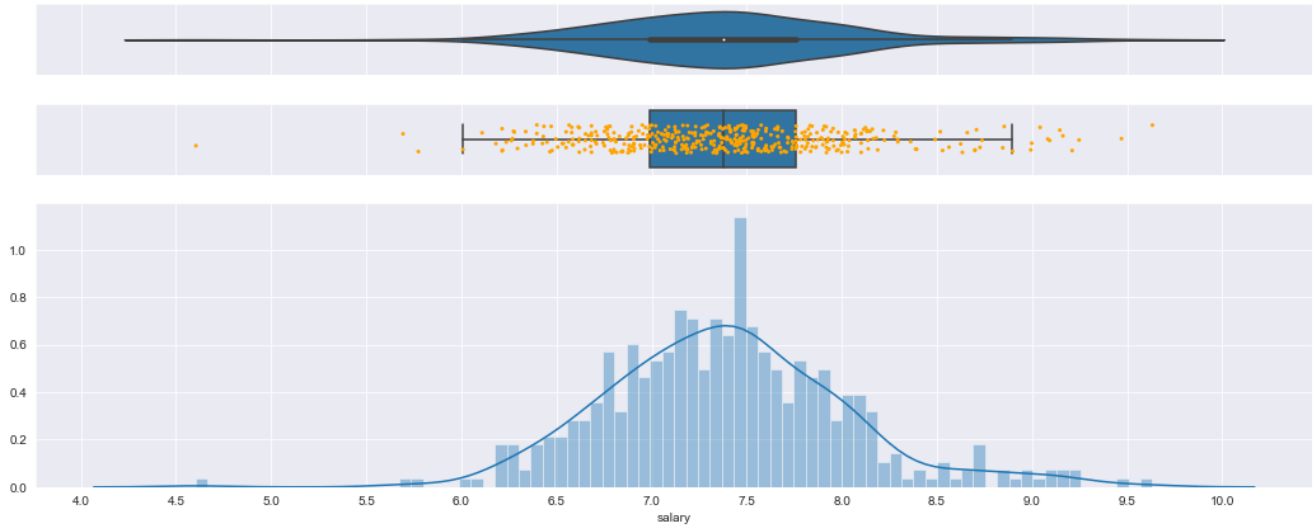
There are methods which help us make the distribution of the sample more symmetric. Consider the natural logarithm of the salary: ln(salary).

- [x] Plot the histogram (and Box-plot) and compare it with the figures for original data.
- [x] Compute the mean and the median.

In [280]:

```
salary_ln = np.log(salary)
represent_distribution(salary_ln, nx=15)

salary_skew = salary_ln.skew()
print('Sample skewness: {:.3f} {}'.format(salary_skew, '> 0 '
                                          if salary_skew > 0 else '< 0'))
print('mean:\t {:.2f}'.format(salary_ln.mean()))
print('median:\t {:.2f}'.format(salary_ln.median()))
```

```
Number of observations: 447
```




```
Sample skewness: 0.306 > 0
mean:  7.39
median:  7.38
```

- [x] What can be concluded from the new values. Provide economic interpretation.

> The mean and the median are now very close and represent where the $\ln(salary)$ data is concentrated. The logarithm function helps to reduce tails and somewhat smoothen the distribution, so we can focus on the characteristics of the distribution.

2

2.a

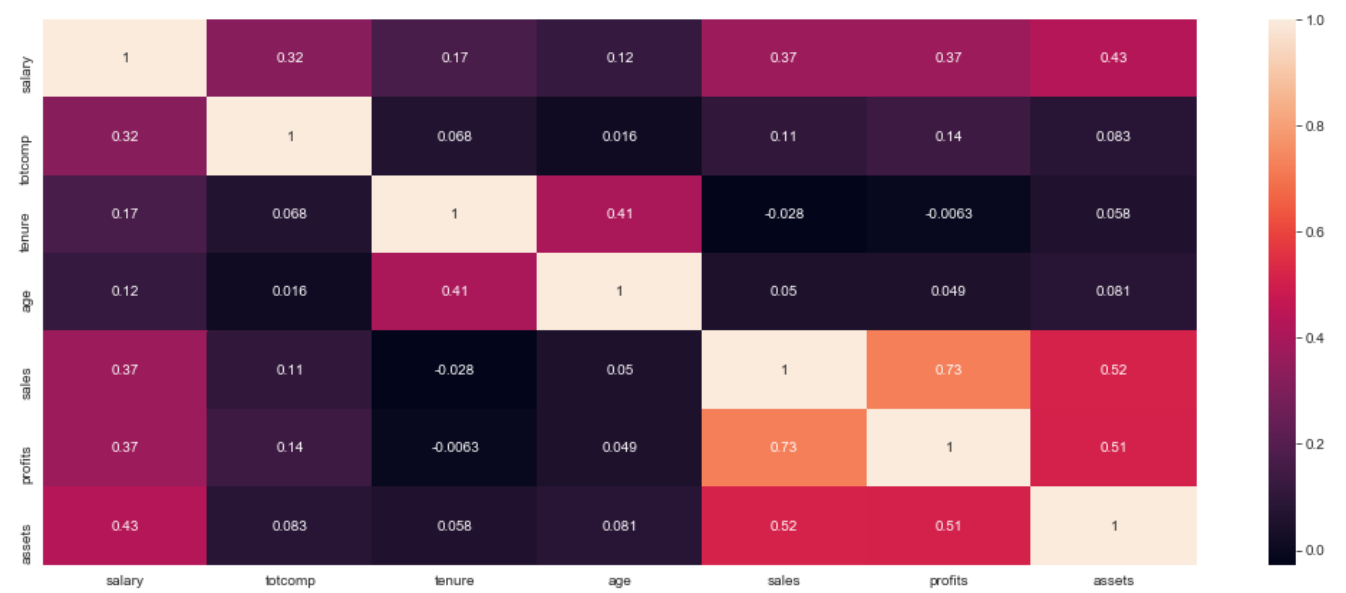
We suspect that the salary of the CEO and other variables are related.

- [x] Compute the correlation coefficients of Pearson and plot them as a heatmap (correlation map).
- [x] Discuss the strength of the correlations.

> From the correlation map we can see that the salary in 1999 may be related with assets, sales and profits in 1998, and with total compensation since their values are close to 0.5 (considering salary as an obligatory variable of the two). The number of years as a CEO and age should not be related to salary since their coefficients are close to zero.
>
> Regarding the full table of relations, one should notice the correlation coefficient 0.73 for sales and profits that tells us about very possible dependency between these two variables. Also with coefficients close to 0.5 there may be a relation between assets and sales & profits.

In [281]:

```
sns.heatmap(ceo.corr('pearson'), annot=True)
plt.show()
```

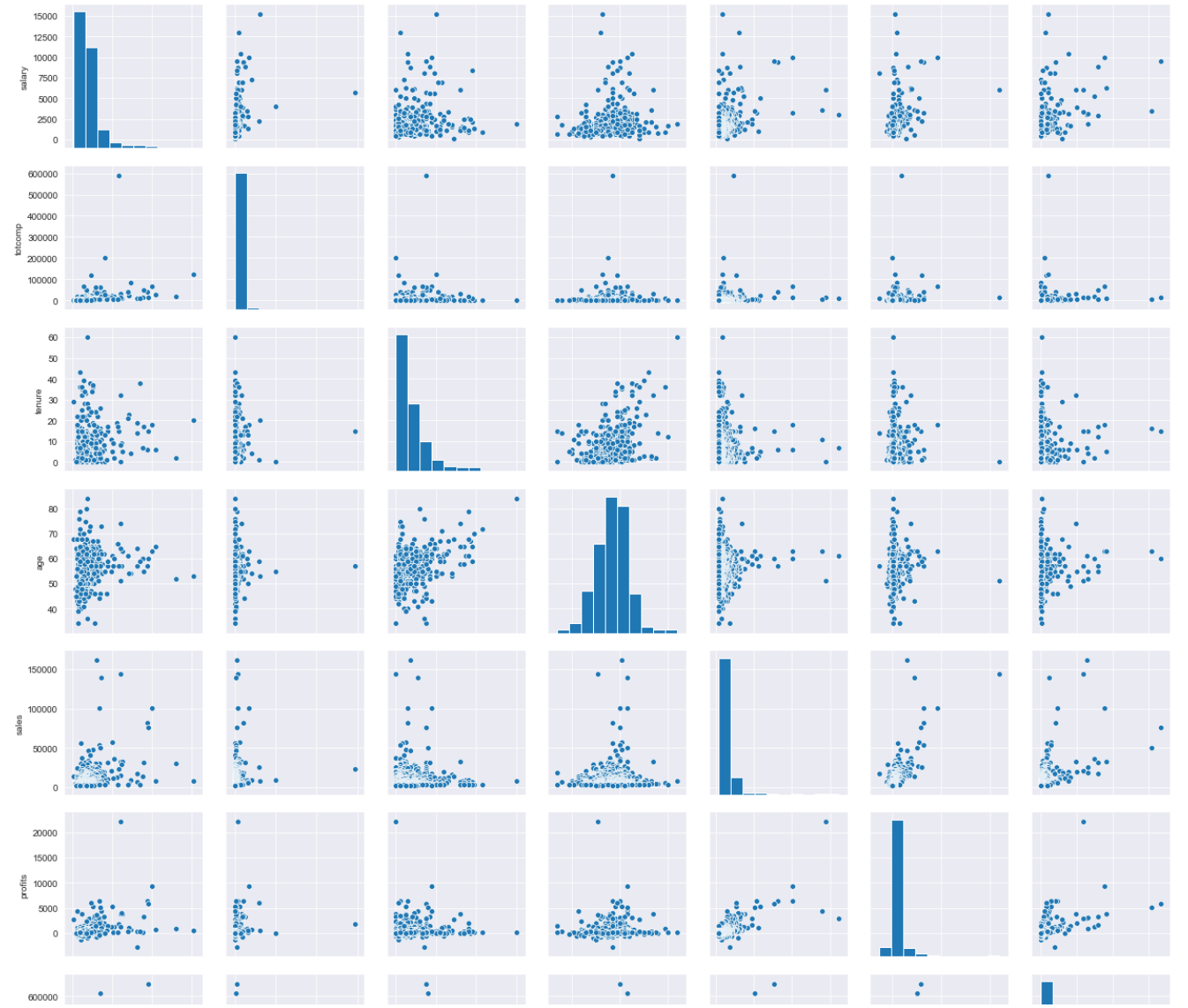


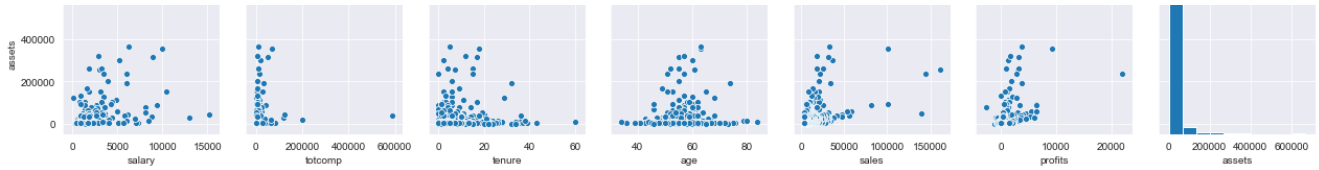
2.b

- [x] Plot the scatter plots.

In [282]:

```
sns.pairplot(ceo)
plt.show()
```





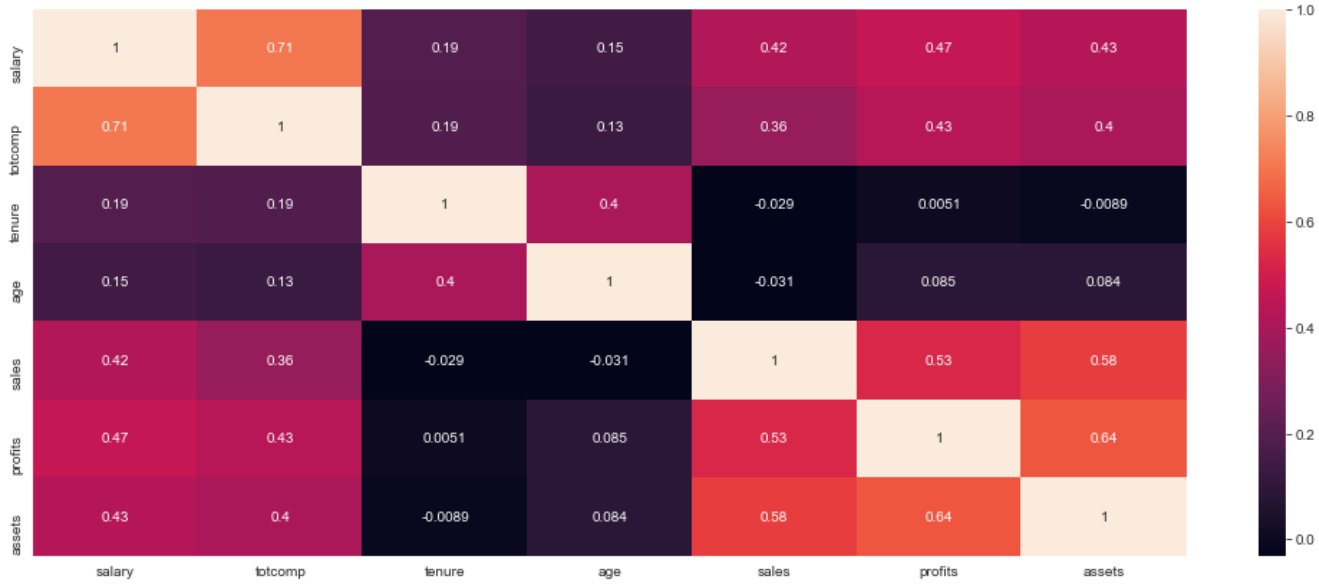
- [x] Conclude if the linear correlation coefficients are appropriate here.

> The scatter plots shows that the linear correlation is not approriate in this case, the relations mostly either very complex or absent at all. The only correlation, that can have linear component, is between sales and profits (for smaller values it is better observed) therefore in this case the linear correlation coefficient may be appropriate. In other cases it's not.

- [x] Compute now the Spearman's correlations and make a heatmap.

In [283]:

```
sns.heatmap(ceo.corr('spearman'), annot=True)
plt.show()
z = sorted(set(list(salary)))
print('The rank of the observation salary=6000 is {}'.format(dict(zip(z, range(len(z))))[6000]))
```




```
The rank of the observation salary=6000 is 373
```

- [x] Compare the results with Pearson.

> There are several significant differences: the salary in 1999 are very correlated with total compensation (corr. coef. = 0.71, by Spearman) and more correlated with sales & profits; the correlation between total compensation and all the other variables is stronger (especially with assets); profits and sales are less correlated; profits and assets are more correlated.

- [x] What is the rank of the observation $salary=6000$?

> 373

2.c

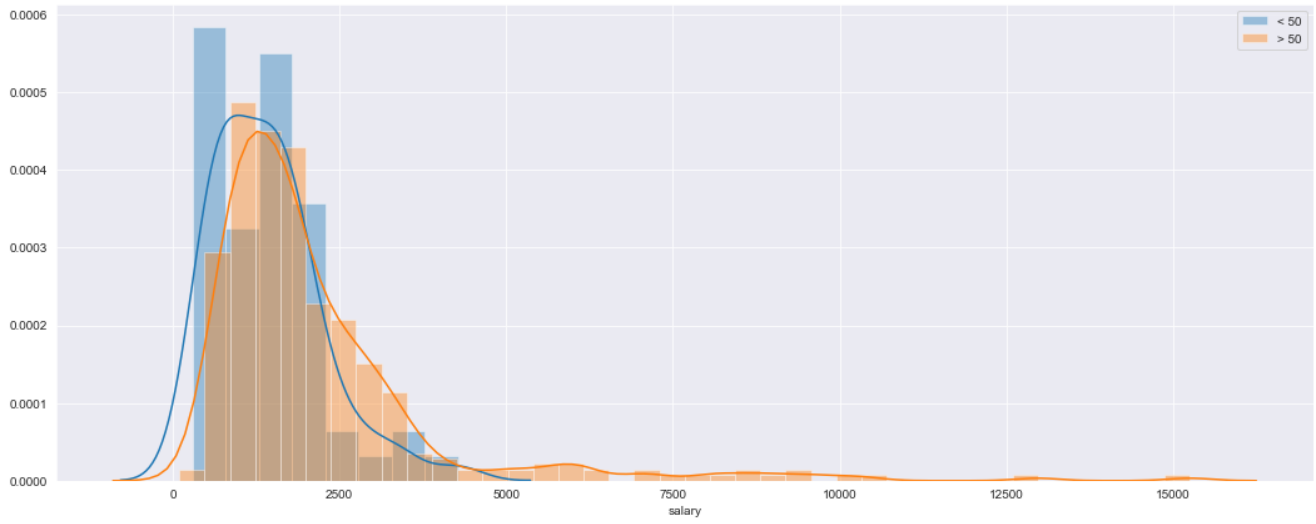
Consider the two subsamples: CEOs younger than 50 and older than 50.

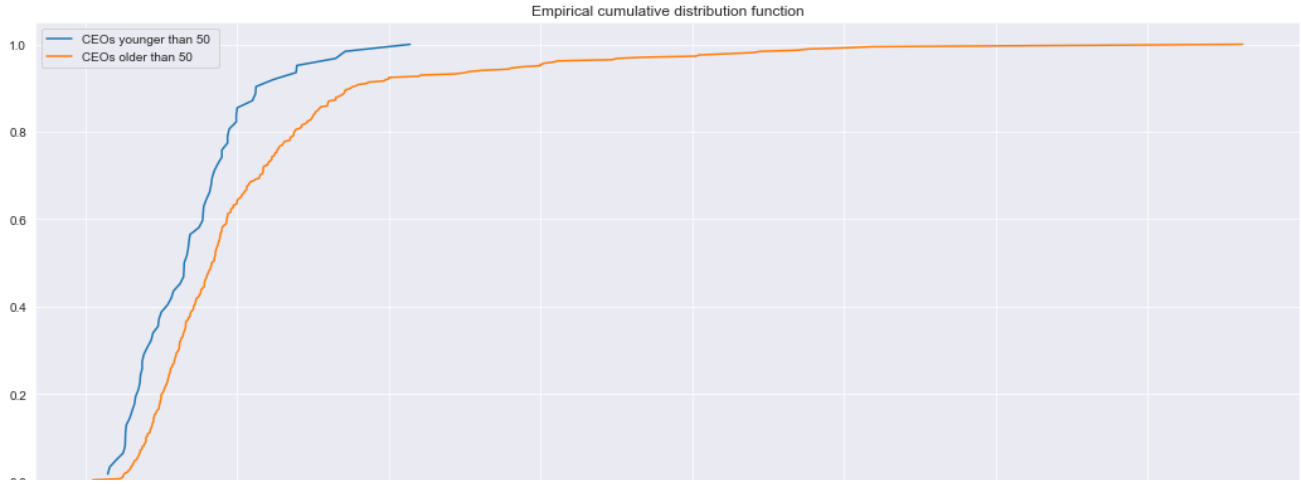
- [x] Plot for both subsamples overlapping histograms/ecdf's and discuss the results.

In [284]:

```
salary_younger50 = ceo[ceo.age<50].salary
salary_older50 = ceo[ceo.age>50].salary
print('observations for < 50:\t\t   {:.2f} | \
observations for > 50:\t\t  {:.2f}'.format(len(salary_younger50), len(salary_older50)))
print('min-max for < 50:\t {:.2f} - {:.2f}| \
min-max for > 50:\t\t {:.2f} - {:.2f}'.format(salary_younger50.min(), salary_younger50.max(),
                                    salary_older50.min(), salary_older50.max()))
print('mean for < 50:\t\t\t {:.2f} | \
mean for > 50:\t\t\t {:.2f}'.format(salary_younger50.mean(), salary_older50.mean()))
print('10%-trimmed mean for < 50:\t {:.2f} | \
10%-trimmed mean for > 50:\t\t {:.2f}'.format(trim_mean(salary_younger50, 0.1),
trim_mean(salary_older50, 0.1)))
print('median for < 50:\t\t {:.2f} | \
median for > 50:\t\t\t {:.2f}'.format(salary_younger50.median(), salary_older50.median()))
print('dispersion for < 50:\t\t {:.2f}   | \
dispersion for > 50:   \t\t {:.2f}'.format(salary_younger50.std(), salary_older50.std()))
sns.distplot(salary_younger50)
sns.distplot(salary_older50)
plt.legend(['< 50', '> 50'])
plt.show()

cdf = ECDF(salary_younger50)
plt.plot(cdf.x, cdf.y, label='CEOs younger than 50 ')
cdf = ECDF(salary_older50)
plt.plot(cdf.x, cdf.y, label='CEOs older than 50 ')
plt.legend()
plt.title('Empirical cumulative distribution function')
plt.show()
```

```
observations for < 50:      62.00 | observations for > 50:     369.00
min-max for < 50:  297.00 - 4280.00| min-max for > 50:   100.00 - 15250.00
mean for < 50:     1406.55 | mean for > 50:     2157.34
10%-trimmed mean for < 50:  1311.56 | 10%-trimmed mean for > 50:   1801.00
median for < 50:   1321.00 | median for > 50:     1675.00
dispersion for < 50:   805.55   | dispersion for > 50:      1838.61
```





- [x] What can we learn from the corresponding location and dispersion (!) measures?

> For CEOs older than 50 salaries are much more volatilized than for younger ones. Most of the outliers are those CEOs older than 50 and the tail of bigger salaries for them is much heavier therefore the mean is highly shifted towards greater values and dispersion is much larger than for CEOs younger than 50. But for these younger CEOs the bigger concentration of values near the median and absence of outliers may be not correct due to the much smaller number of observations.

3

Consider another grouping of the data. Define the groups:

```
S1 if salary < 2000
S2 if salary ≥ 2000 but < 4000
S3 if salary ≥ 4000

A1 if age < 50
A2 if age ≥ 50
```

In [285]:

```
ceo["S"] = list(map(lambda x: 1 if x < 2000 else 2 if x < 4000 else 3, ceo.salary))
ceo["A"] = list(map(lambda x: 1 if x < 50 else 2, ceo.age))
```

3.a

- [x] Aggregate the data to a 2×3 contigency table with absolute and with relative frequencies.

In [286]:

```
contingency_table = pd.crosstab(ceo.A, ceo.S)
short_contingency_table = contingency_table.copy()
contingency_table['marginal'] = contingency_table.aggregate('sum', 1)
contingency_table = contingency_table.append(pd.Series(dict(zip([1,2,3,'marginal'],
                                                                contingency_table.aggregate('sum')))
                                                       name='marginal'))
relative_contingency_table = contingency_table / num_observations
```

In [287]:

```
contingency_table
```

Out[287]:

S	1	2	3	marginal
A				
1	52	9	1	62
2	248	107	30	385
marginal	300	116	31	447

In [288]:

```
relative_contingency_table
```

Out[288]:

S	1	2	3	marginal

S \ A	1	2	3	marginal
1	0.116331	0.020134	0.002237	0.138702
2	0.554810	0.239374	0.067114	0.861298
marginal	0.671141	0.259508	0.069351	1.000000

3.b

- [x] Give interpretation for the values of n_{12} , h_{12} , $n_{1\cdot}$ and $h_{1\cdot}$.

> $n_{12} = 9$ of 447 CEOs (relatively $h_{12} = 0.020$ or 2% of all) are younger than 50 having salary between 2000 & 4000. It is one the least groups among aggregated.
>
> $n_{1\cdot} = 62$ of 447 CEOs (relatively $h_{1\cdot} = 0.139$ or 14% of all) are younger than 50. This age group is relatively small and not very representative.

3.c

- [x] Compute an appropriate dependence measure for S_i and A_j.

In [294]:

```
def compute_chi2(contingency_table):
    arr = np.array(contingency_table)
    num_observations = arr.sum()
    n, m = arr.shape
    s = 0
    for i in range(n):
        for j in range(m):
            n_ij = arr[i, j]
            n_i = arr[i,:].sum()
            n_j = arr[:,j].sum()
            s += ((n_ij - n_i * n_j / num_observations) ** 2) / (n_i * n_j / num_observations)
    return s

def cont_coef_pearson(contingency_table):
    """Corrected contingency coefficient of Pearson
    """
    chi2 = compute_chi2(contingency_table)
    min_kl = min(contingency_table.shape)
    C_max = np.sqrt((min_kl - 1) / min_kl)
    C = np.sqrt(chi2 / (chi2 + num_observations))
    C_corr = C / C_max
    return C_corr
```

In [293]:

```
#import scipy.stats as st
# chi2, p, dof, expected = st.chi2_contingency(short_contingency_table)
# print(chi2, p, dof)
# print('p = {:.3f} {}'.format(p, '> 0.05 ' if p > 0.05 else '< 0.05'))

coef = cont_coef_pearson(short_contingency_table)
print('Corrected contingency coefficient of Pearson = {:.4f}'.format(coef))
```

```
Corrected contingency coefficient of Pearson = 0.2048
```

- [x] What can be concluded from its value?

> Since the corrected contingency coefficient of Pearson $\in [0,1]$, we have the value closer to zero indicating "weak" dependence between variables S & A. So we didn't notice significant correlation between two age groups and three salary groups. Perhaps, the other aggregating method would help find strong relations.

Problem2

~~0.1~~ Problem 2: Descriptive Statistics and Probability Theory: Silated Data

It is obvious that the descriptive tools are very sensitive to contamination or outliers in the data. The objective of this problem is to assess the sensitivity of these measures/tools to outliers or very heterogenous data.

```
In [1]: import numpy as np
        import pandas as pd
        from scipy import stats
        from IPython.display import HTML
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        from matplotlib import animation, rc
        import seaborn as sns
        mpl.rcParams['figure.figsize'] = (10, 3)
        sns.set_style("darkgrid")

        import warnings
        warnings.filterwarnings('ignore')

        from utils import *

        %load_ext autoreload
        %autoreload 2
```

~~1~~ 1

- [x] Simulate (with a xed seed) a sample of size $n = 100$ from the normal distribution with $_1 = 10$ and $\sigma_1^2 = 9$.

```
In [2]: np.random.seed(0)

        n = 100
        mu1 = 10
        sigma1 = 3

        rv_norm = stats.norm(loc=mu1, scale=sigma1)
        sample_norm1 = rv_norm.rvs(size=n)
```

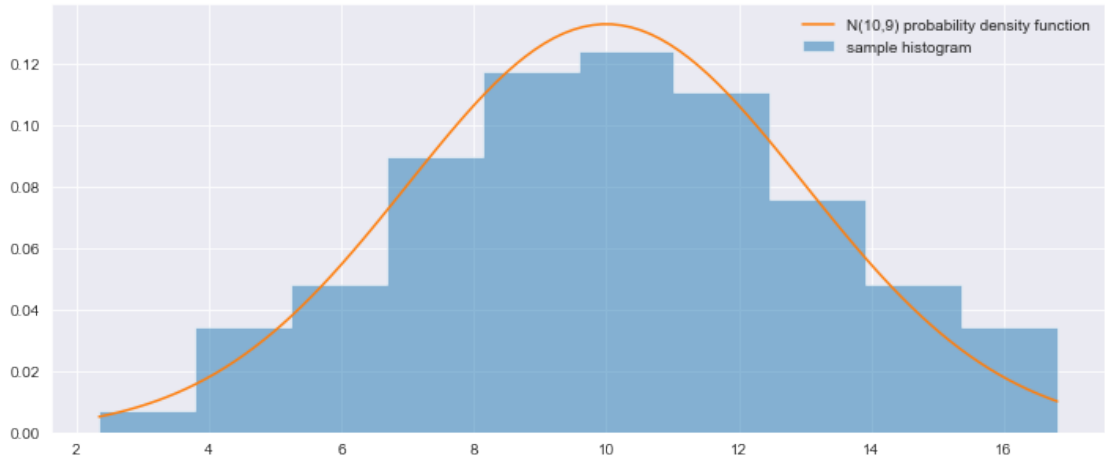
```
        print('N({},{}) | Sample: {:.2f} {:.2f} {:.2f} {:.2f} {:.2f}...'\
              .format(mu1, sigma1**2, *sample_norm1))
```

```
N(10,9) | Sample: 15.29 11.20 12.94 16.72 15.60...
```

~~1.1~~ 1.a

- [x] Plot the histogram and compare it to the density of $N(10, 9)$.

```
In [4]: mpl.rcParams['figure.figsize'] = (12, 5)
        xs = np.linspace(sample_norm1.min(), sample_norm1.max(), 100)
        ys = rv_norm.pdf(xs)

        plt.hist(sample_norm1, bins='sqrt', density=True, histtype='stepfilled',
                 alpha=0.5, label='sample histogram')
        plt.plot(xs, ys, label='N(10,9) probability density function')
        plt.legend(loc='best', frameon=False)
        plt.show()
```



~~1.2~~ 1.b

- [x] Now draw a sample y_i of size $n = 100$ from t_5. Transform it as follows: $10 + 3\sqrt{3/5} \cdot y_i$.
- [x] Plot the histogram and compare the density of $N(10, 9)$.
- [x] What can be concluded and why this example might be relevant for empirical studies?

  We can conclude that we may approximate Normal distribution using Student distribution. Indeed, if $X \sim N(a, \sigma^2)$ so $EX = a$ & $Var(X) = \sigma^2$, and $Y \sim t_n$ which means that $EY = 0$ & $Var(Y) = \frac{n}{n-2}$ then we can approximate:

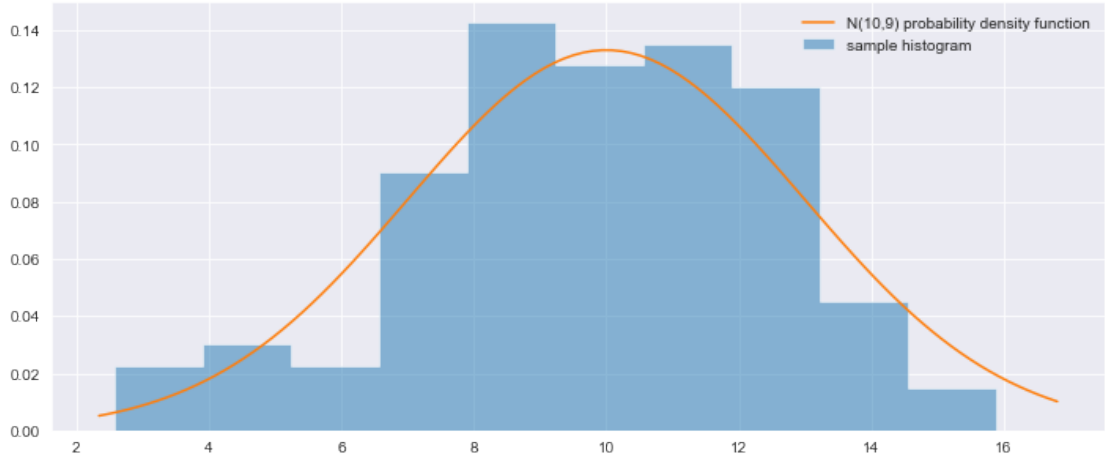
$$X \approx \sigma_X \left(\frac{Y - EY}{\sigma_Y} \right) + EX = \sigma_X \sqrt{\frac{n-2}{n}} Y + a \tag{1}$$

In our case $n = 5$, $EX = 10$, $\sigma_X = 3$ therefore $X \approx 3\sqrt{\frac{3}{5}}Y + 10$

This example reveals the closeness of Normal and Student distributions. Therefore many different empirical studies that conclude with the normal distribution of one or anothet variable may be wrong although we would never know the truth. Anyway the t-distribution might also be considered.

```
In [81]: mpl.rcParams['figure.figsize'] = (12, 5)
         np.random.seed(0)
         df=5
         rv_st = stats.t(df=df)
         sample_st = rv_st.rvs(size=100)
         sample_st_transformed = 10 + 3 * np.sqrt(3/5) * sample_st
         print('Student_{} | Sample: {:.2f} {:.2f} {:.2f} {:.2f} {:.2f}...'\
               .format(df, *sample_st))
         print('Transformed sample: {:.2f} {:.2f} {:.2f} {:.2f} {:.2f}...'\
               .format(*sample_st_transformed))
         plt.hist(sample_st_transformed, bins='sqrt', density=True,
                  histtype='stepfilled', alpha=0.5, label='sample histogram')
         plt.plot(xs, ys, label='N(10,9) probability density function')
         plt.legend(loc='best', frameon=False)
         plt.show()
```

```
Student_5 | Sample: 1.66 -1.64 1.08 1.63 0.78...
Transformed sample: 13.87 6.19 12.50 13.79 11.82...
```



~~2~~ 2

In practice the data is always very heterogenous. To reect it we contaminate the data by adding an outlier or a subsample with dierent characteristics.

~~2.1~~ 2.a

- [x] To obtain a realistic heterogenous sample add to the original normal data a new sample of size m simulated from $N(20, 2^2)$, i.e. $_2 = 20$ and $\sigma_2^2 = 4$. The size m will obviously inuence the above measures.
- [x] Vary m from 10 to 200. (The resulting sample is said to stem from a mixture normal distribution).

```
In [83]: delta_m = 5
         ms = np.arange(0, 200 + delta_m, delta_m)
         print('m values are: {} {} ... {} {}.'.format(*ms[:2], *ms[-2:]))
         mu2 = 20
         sigma2 = 2

         rv_norm2 = stats.norm(loc=mu2, scale=sigma2)
         samples_norm2 = {}
         samples_h = {}
         sample_norm2 = rv_norm2.rvs(size=ms[-1])
         for m in ms:
         #     samples_norm2[m] = rv_norm2.rvs(size=m)
             samples_norm2[m] = sample_norm2[:m]
             samples_h[m] = np.hstack((sample_norm1, samples_norm2[m]))
         print('Generated {} samples for different m values'.format(len(ms)))
```

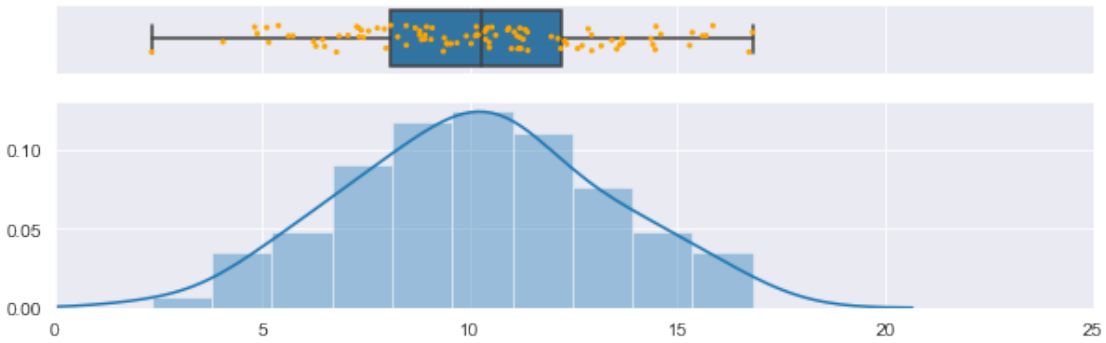
```
m values are: 0 5 ... 195 200.
Generated 41 samples for different m values
```

~~2.2~~ 2.b

- [x] Plot Box-plots (or `violin` plots) and histograms for each subsample individually and for the sample for a few dierent values of m.

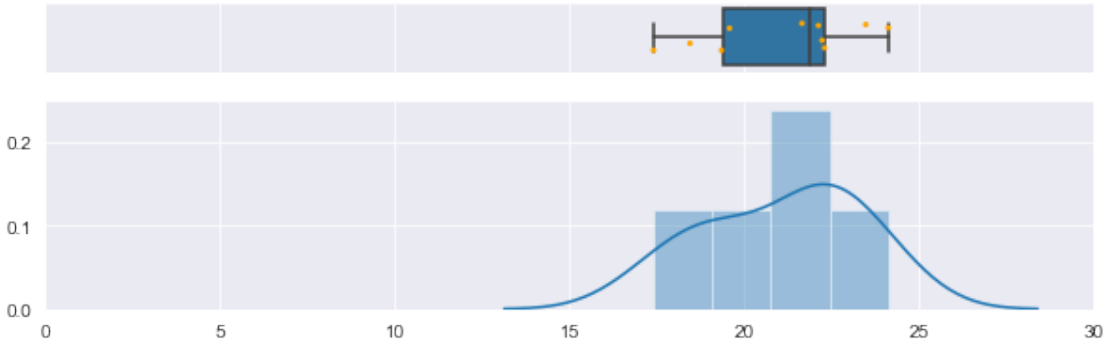
```
In [84]: print('FIRST SUBSAMPLE from N({},{})'.format(mu1, sigma1**2))
         represent_distribution(sample_norm1, 0, 25)
         for m in [10,60,120,200]:
             print('================================================================================')
             print('SECOND SUBSAMPLE from N({},{}); m = {}'.format(mu2, sigma2**2, m))
             represent_distribution(samples_norm2[m], 0, 30)
             print('TOTAL SAMPLE')
             represent_distribution(samples_h[m])
```

```
FIRST SUBSAMPLE from N(10,9)
(number of observations: 100)
```

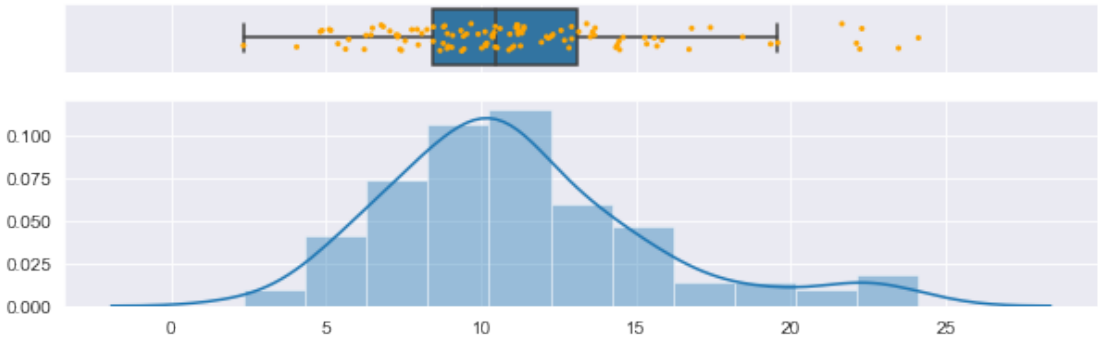


==========

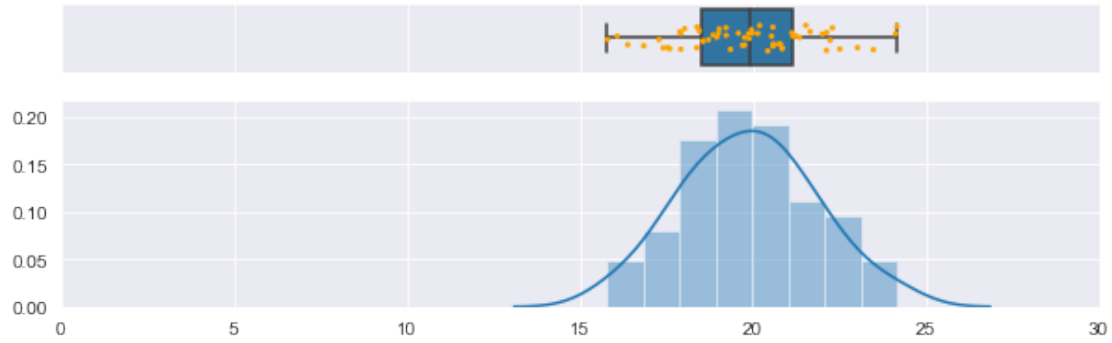
SECOND SUBSAMPLE from N(20,4); m = 10
(number of observations: 10)



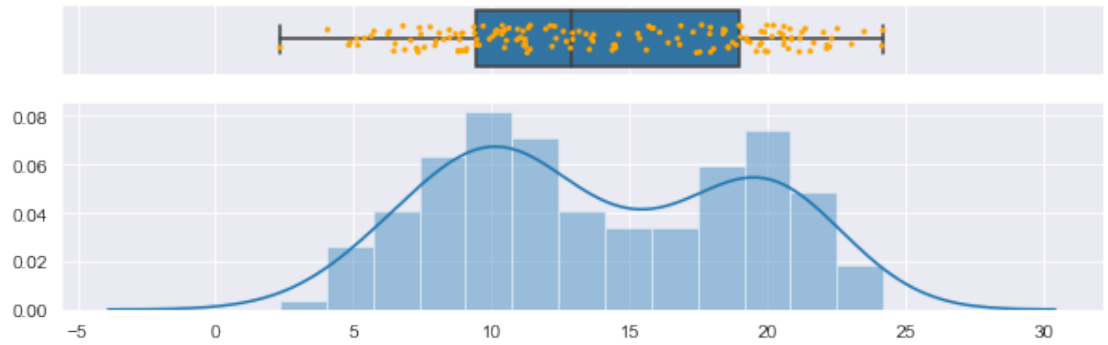
TOTAL SAMPLE
(number of observations: 110)



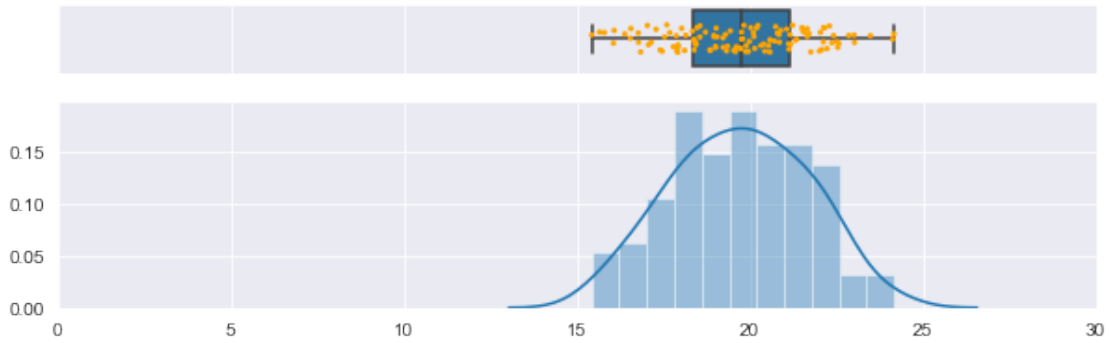
```
================================================================================
SECOND SUBSAMPLE from N(20,4); m = 60
(number of observations: 60)
```



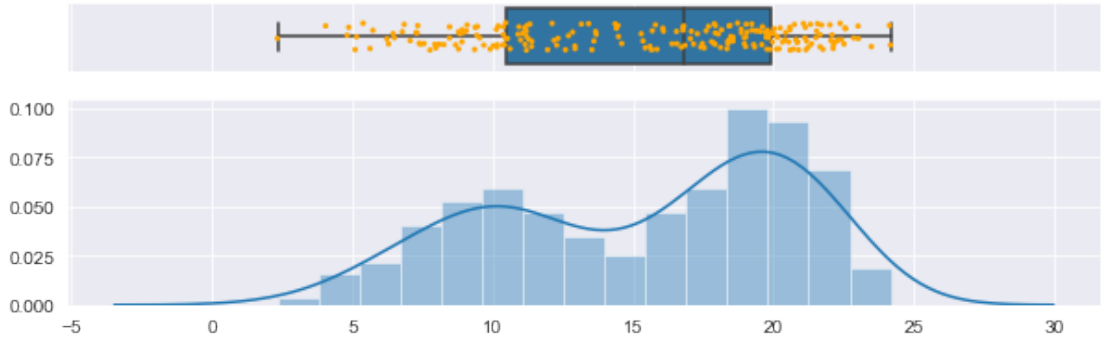

```
TOTAL SAMPLE
(number of observations: 160)
```




```
================================================================================
SECOND SUBSAMPLE from N(20,4); m = 120
(number of observations: 120)
```

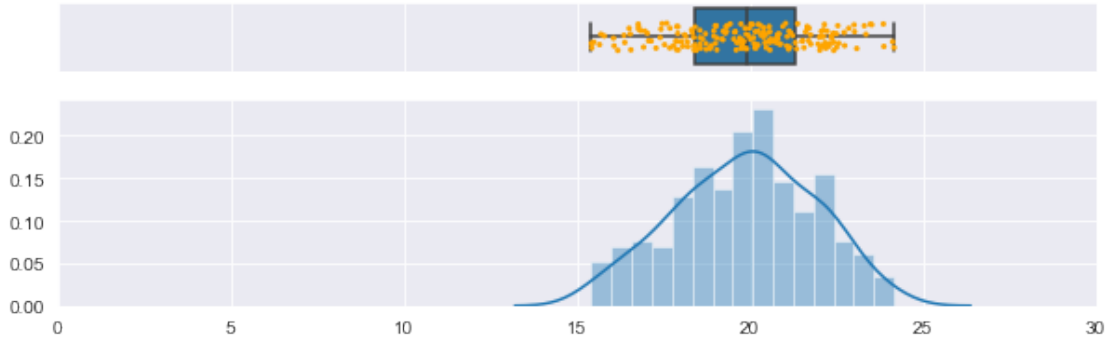


TOTAL SAMPLE
(number of observations: 220)

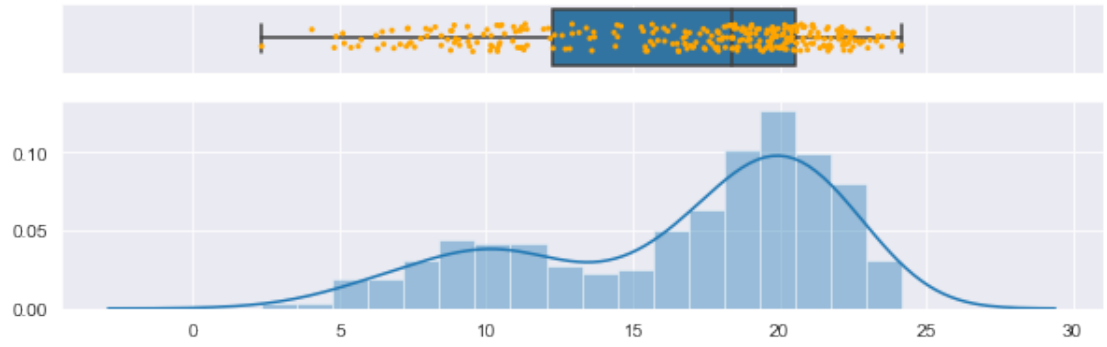


==========

SECOND SUBSAMPLE from N(20,4); m = 200
(number of observations: 200)



```
TOTAL SAMPLE
(number of observations: 300)
```

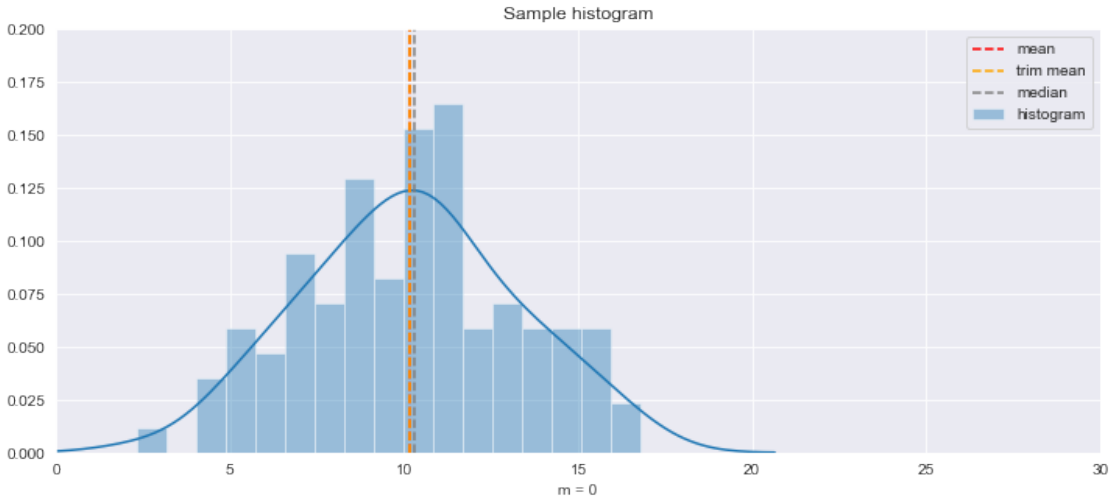


~~2.3~~ 2.c

- [x] Make animated or interactive graphics (with `manipulate`, `plotly`, `ggplot`, etc.) to visualize the impact of m on the histogram and location measures (added as vertical lines in the graph) of the data.

```
In [87]: mpl.rcParams['figure.figsize'] = (12, 5)
         nbins = int(np.sqrt(len(samples_h[ms[-1]])))
         def update_hist(i, data):
             m = ms[i]
             sample = data[m]
             mean = sample.mean()
             trim_mean = stats.trim_mean(sample, 0.1)
             median = p_quantile(sample, 0.5)
             plt.cla()
             sns.distplot(sample, bins=nbins, label='histogram')
             plt.axvline(x=mean, linestyle='--', color='red', label='mean')
             plt.axvline(x=trim_mean, linestyle='--', color='orange', label='trim mean')
             plt.axvline(x=median, linestyle='--', color='gray', label='median')
             plt.legend()
             plt.title('Sample histogram')
             plt.xlabel('m = {}'.format(m))
             plt.xlim(0, 30)
             plt.ylim(0, 0.2)

In [88]: fig = plt.figure()
         hist = plt.hist(samples_h[ms[0]])
         ani = animation.FuncAnimation(fig, update_hist, len(ms), fargs=(samples_h, ))
```

```
ani.save('animation.mp4', writer=animation.writers['ffmpeg'](fps=15, metadata=dict(ar
# HTML(ani.to_html5_video())
```



Please refer:

https://drive.google.com/open?id=17XOVoPnz9jZaBdzPPRdGLcopLjFZ-53R

~~3~~ 3

Next step is to simulate two dependent data sets. We simulate two samples with a given value of the correlation coecient.

~~3.1~~ 3.a

Let $U \sim N(0,1)$ and $V \sim N(0,1)$.
Let $U = U$ and $V = \rho U + \sqrt{1\rho^2}V$.

- [x] Prove that $Corr(U,V) = \rho$ and the variances of both variables U and V equal one.

  $Var(U^*) = Var(U) = 1$

In [ ]:

  $Var(V^*) = Var\left(\rho U + \sqrt{1\rho^2}V\right) = \| U \text{ and } V \text{ are independent } \| = \rho^2 Var(U) + (1 - \rho^2)Var(V) = \rho^2 \cdot 1 + (1-\rho^2)\cdot 1 = 1$

In [ ]:

  $Corr(U^*, V^*) = \frac{Cov(U^*, V^*)}{\sqrt{Var(U^*), Var(V^*)}} = Cov(U^*, V^*) = Cov\left(U, \rho U + \sqrt{1\rho^2}V\right) = \rho Var(U) + \sqrt{1\rho^2}Cov(U,V) = | U \text{ and } V \text{ are independent } | = \rho + \sqrt{1\rho^2}\cdot 0 = \rho$

~~3.2~~ 3.b

- [x] Use the above idea to simulate two samples of size $n = 100$ from a normal distribution with dierent values of ρ.

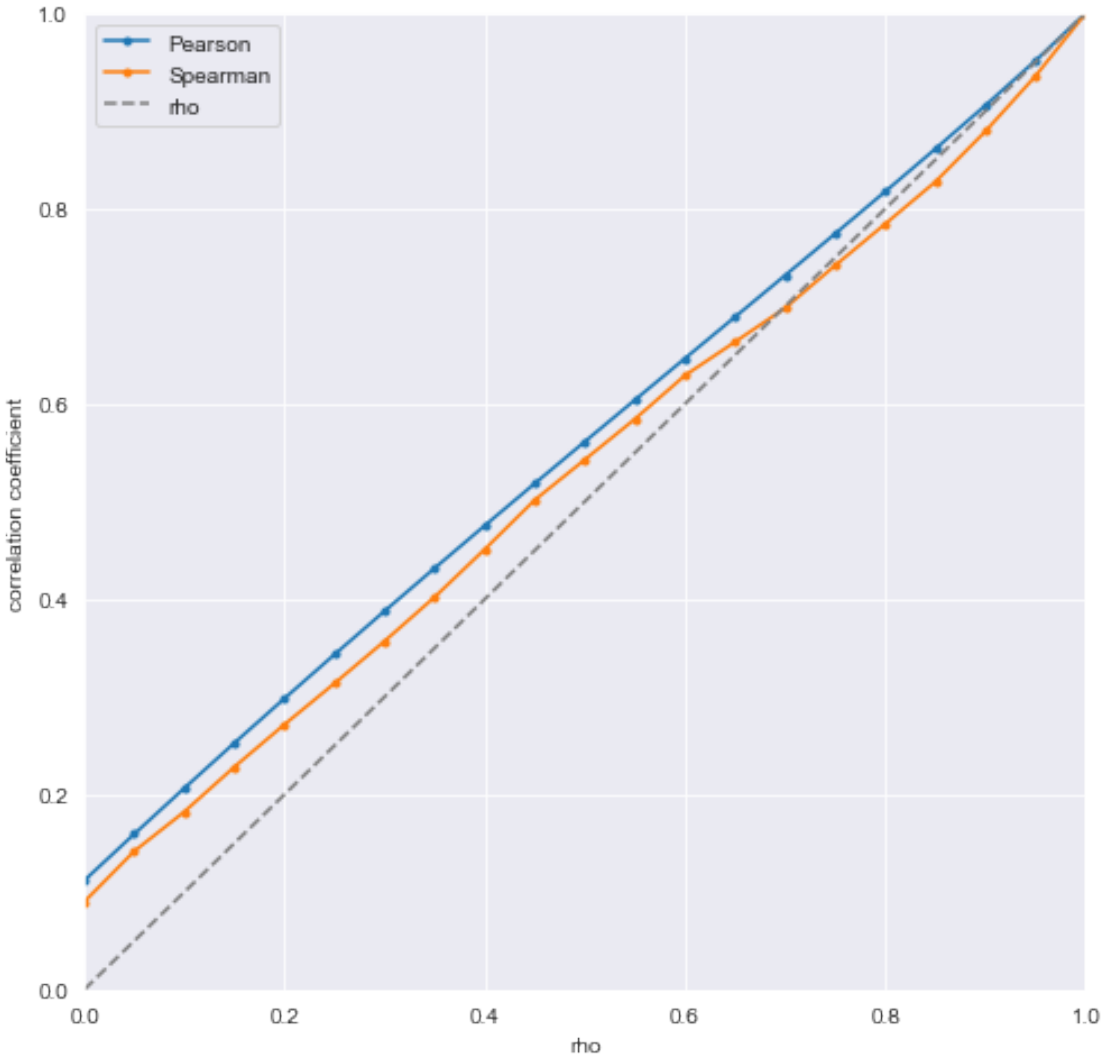
```
In [70]: np.random.seed(0)
         n = 100
         delta_rho = 0.05
         rhos = np.arange(0, 1 + delta_rho, delta_rho)
         print('rho values are: {:.2f} {:.2f} ... {:.2f} {:.2f}'\
               .format(*rhos[:2], *rhos[-2:]))

         rv_U = stats.norm(loc=0, scale=1)
         rv_V = stats.norm(loc=0, scale=1)

         sample_u = rv_U.rvs(size=n)
         sample_v = rv_V.rvs(size=n)
         print('U  ~ N(0,1) | Sample: {:.2f} {:.2f} {:.2f} {:.2f} {:.2f}...'\
               .format(*sample_u))
         print('V  ~ N(0,1) | Sample: {:.2f} {:.2f} {:.2f} {:.2f} {:.2f}...'\
               .format(*sample_v))

         sample_u2 = sample_u[:]
         print('U* ~ N(0,1) | Sample: {:.2f} {:.2f} {:.2f} {:.2f} {:.2f}...'\
               .format(*sample_u))
         # represent_distribution(sample_u2, -5, 5)

         samples_v2 = {}
         for rho in rhos:
             samples_v2[rho] = rho * sample_u + np.sqrt(1 - rho**2) * sample_v

         # represent_distribution(sample_u, -5, 5)
         # represent_distribution(sample_v, -5, 5)
         # represent_distribution(samples_v2[rhos[-1]], -5, 5)
         print('V* | Generated {} samples for different values of rho.'.format(len(rhos)))
```

```
rho values are: 0.00 0.05 ... 0.95 1.00
U  ~ N(0,1) | Sample: 1.76 0.40 0.98 2.24 1.87...
V  ~ N(0,1) | Sample: 1.88 -1.35 -1.27 0.97 -1.17...
U* ~ N(0,1) | Sample: 1.76 0.40 0.98 2.24 1.87...
V* | Generated 21 samples for different values of rho.
```

- [x] Compute the correlation coecients of Pearson and of Spearman.
- [x] Compare the correlation to the original parameter ρ (for example, plot Pearson vs. ρ and Spearman vs. ρ).

```
In [127]: mpl.rcParams['figure.figsize'] = (8, 8)
          corrs = {'pearson': {}, 'spearman': {}}
```

```
for rho in rhos:
    corrs['pearson'][rho] = stats.pearsonr(sample_u2, samples_v2[rho])[0]
    corrs['spearman'][rho] = stats.spearmanr(sample_u2, samples_v2[rho])[0]
plt.plot(rhos, [corrs['pearson'][rho] for rho in rhos], '.-')
plt.plot(rhos, [corrs['spearman'][rho] for rho in rhos], '.-')
plt.plot(rhos, rhos, '--', color='gray')
plt.legend(['Pearson', 'Spearman', 'rho'])
plt.axis([0, 1, 0, 1])
plt.xlabel('rho')
plt.ylabel('correlation coefficient')
plt.show()
```



When rho is close to zero, which means that there is (almost) no corelation, both coefficients have higher values, which may be caused by stochasticity. The more rho is, the closer are coefficients to it. Pearson coefficient is growing linearly while

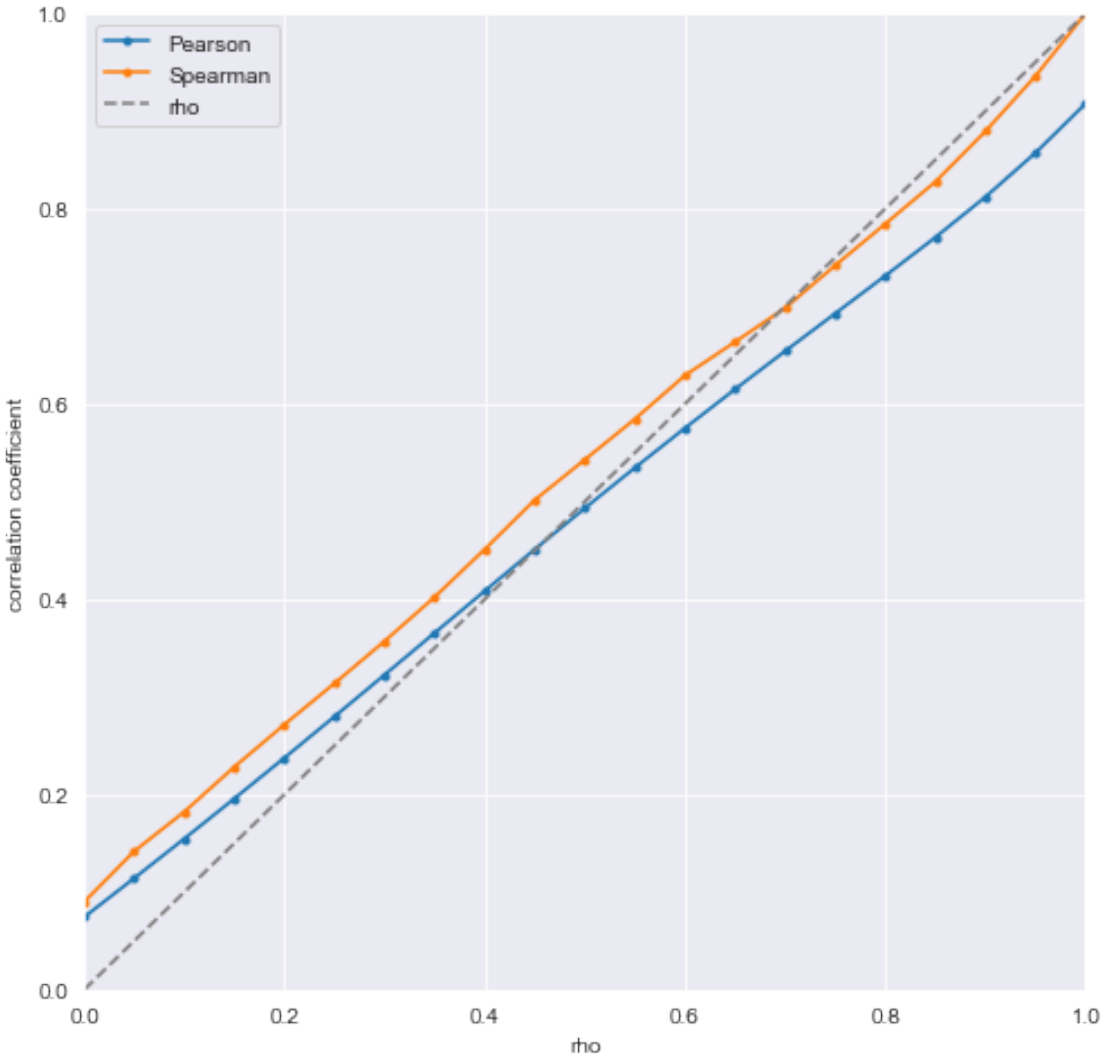
~~3.3~~ 3.c

- [x] Make a nonlinear but monotone transformation of V , say V^2, ln or exp. > Let $V^{**} = exp(V^*) + 0.5V^*$ - monotone transformation of V^*

```
In [128]: samples_v3 = {}
          for rho in rhos:
              samples_v3[rho] = np.exp(samples_v2[rho]) + 0.5 * samples_v2[rho]
```

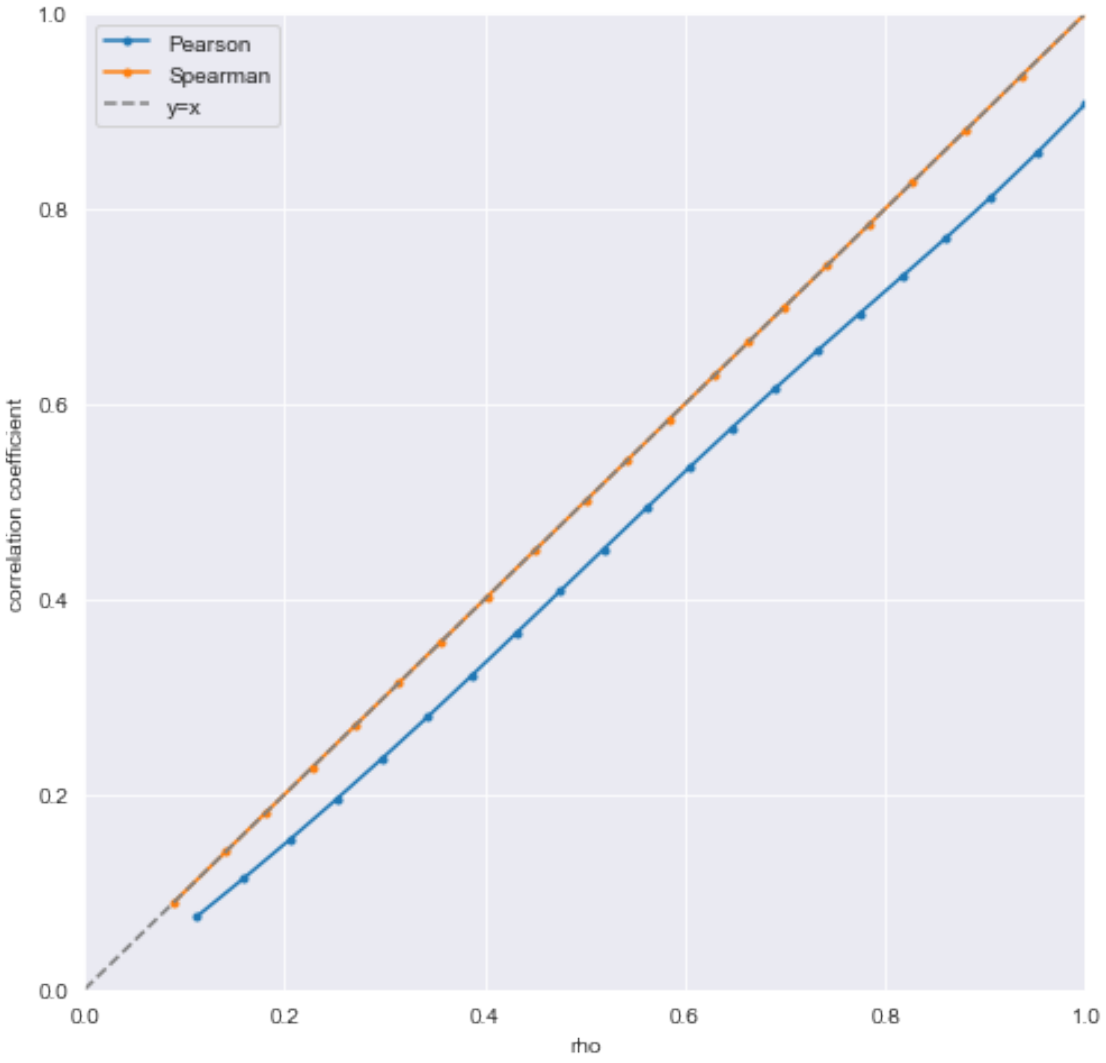
- [x] Check the impact of this transformation on the correlation coecients of Spearman and Pearson.
- [x] Think about an appropriate visualization of the ndings.

```
In [129]: corrs2 = {'pearson': {}, 'spearman': {}}
          for rho in rhos:
              corrs2['pearson'][rho] = stats.pearsonr(sample_u2, samples_v3[rho])[0]
              corrs2['spearman'][rho] = stats.spearmanr(sample_u2, samples_v3[rho])[0]
          plt.plot(rhos, [corrs2['pearson'][rho] for rho in rhos], '.-')
          plt.plot(rhos, [corrs2['spearman'][rho] for rho in rhos], '.-')
          plt.plot(rhos, rhos, '--', color='gray')
          plt.legend(['Pearson', 'Spearman', 'rho'])
          plt.axis([0, 1, 0, 1])
          plt.xlabel('rho')
          plt.ylabel('correlation coefficient')
          plt.show()
```



If we look at the functions of correlation coeffitients with respect to ρ we wouldn't conclude much but if look at the function of the correlation coefficients of the transformed V* with respect to the same coefficients of the original V*:

```
In [131]: plt.plot([corrs['pearson'][rho] for rho in rhos], [corrs2['pearson'][rho] for rho in
          plt.plot([corrs['spearman'][rho] for rho in rhos], [corrs2['spearman'][rho] for rho
          plt.plot(rhos, rhos, '--', color='gray')
          plt.legend(['Pearson', 'Spearman', 'y=x'])
          plt.axis([0, 1, 0, 1])
          plt.xlabel('rho')
          plt.ylabel('correlation coefficient')
          plt.show()
```



We can see that the correlation coefficient of Spearman hasn't changed. Indeed, since it deals not with variable values but with its ranks, the monotone (!) transformation wouldn't affect the rank values. The coefficient of Pearson got lower since the dependence between variables is no longer linear but more complex.