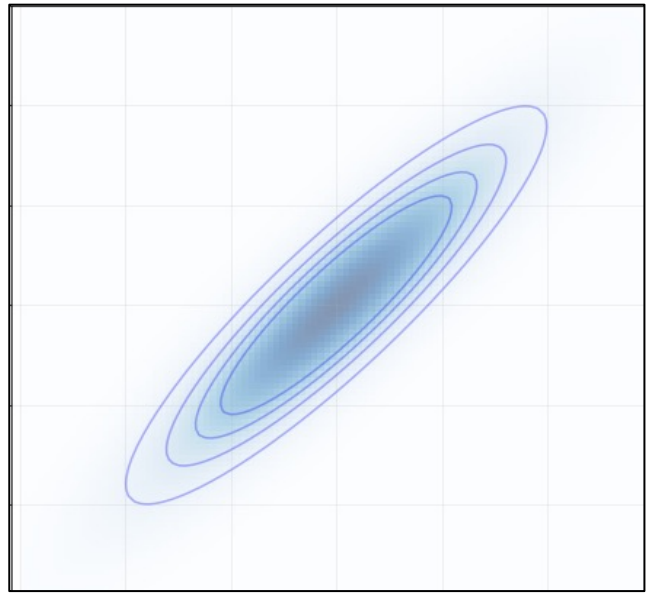
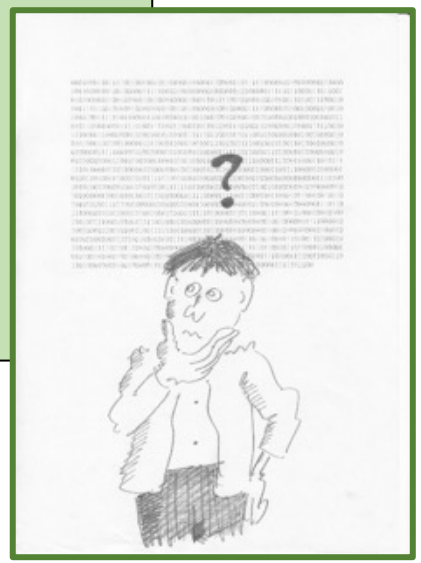
Statistical assessment of correlations : a practical recap



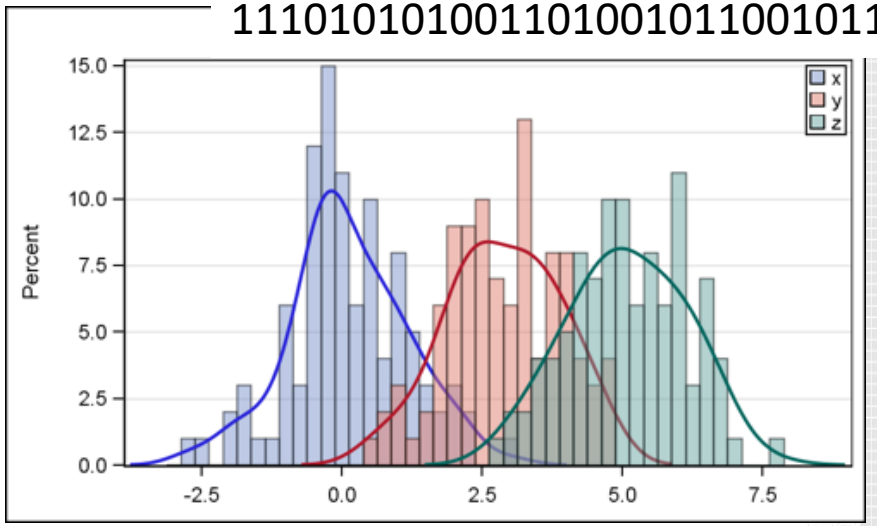
Statistics and data analysis

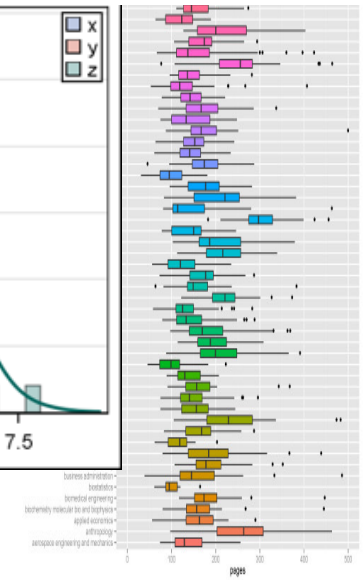
Zohar Yakhini,
Leon Anavy,
Ben Galili

IDC, Herzeliya



0010011101010100101010100100100010
1010100010101111101011010011001001
1110101010011010010110010110110010





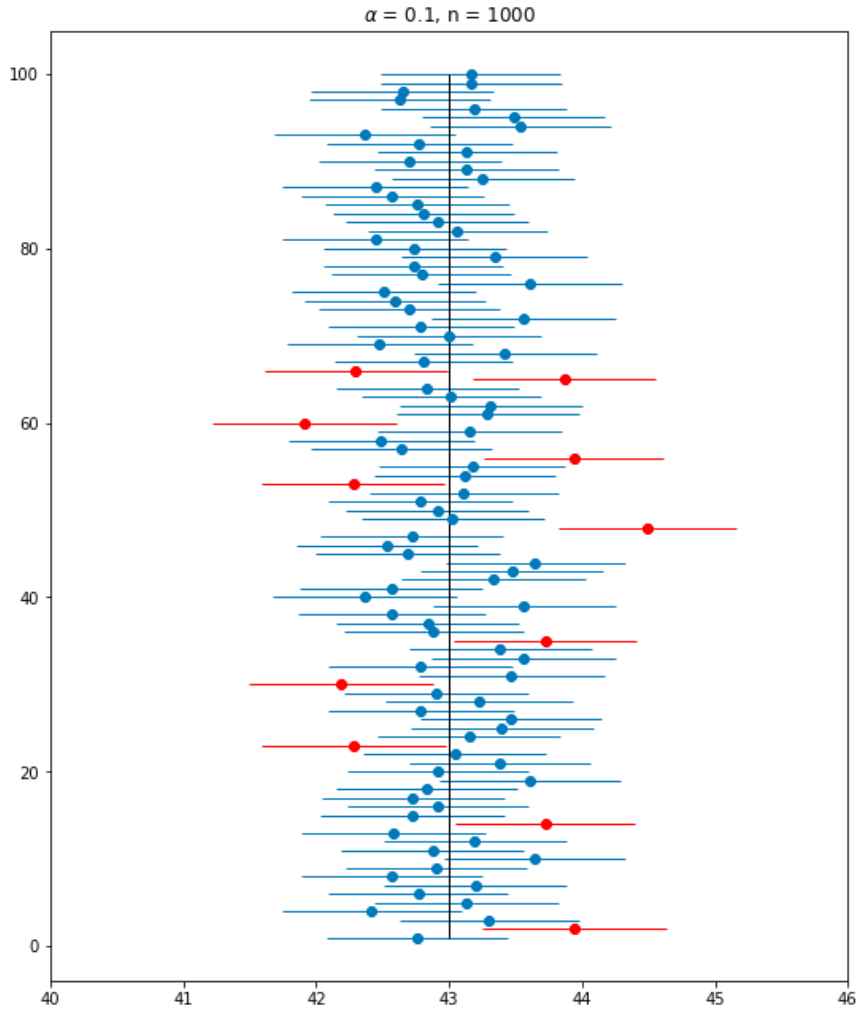
Confidence intervals for proportions

$$P(\hat{p} - \gamma \cdot \hat{\sigma} \leq p \leq \hat{p} + \gamma \cdot \hat{\sigma}) \approx 2\Phi(\gamma) - 1$$

where $\hat{p}$ is the empirical proportion

and $\hat{\sigma} = \frac{\sqrt{\hat{p}(1-\hat{p})}}{\sqrt{n}}$

$$P\left(p \in \hat{p} \pm \Phi^{-1}\left(1 - \frac{\alpha}{2}\right) \cdot \hat{\sigma}\right) \approx 1 - \alpha$$



CIs for correlations? based on the Fisher Transform

$$\Sigma = \begin{pmatrix} V(X) & Cov(X,Y) \\ Cov(X,Y) & V(Y) \end{pmatrix}$$

$$Cov(X,Y) = \rho(X,Y)\sigma(X)\sigma(Y)$$

When drawing n samples from a bivariate normal distribution with correlation coefficient ρ and computing the empirical Pearson correlation on the sample, $\hat{\rho}_n$, we have:

$$P\left(F(\rho) \in F(\hat{\rho}_n) \pm \frac{1}{\sqrt{n-3}}\,\Phi^{-1}\left(1-\frac{\alpha}{2}\right)\right) = 1-\alpha$$

And also:

$$P\left(F(\rho) \geq F(\hat{\rho}_n) - \frac{1}{\sqrt{n-3}}\,\Phi^{-1}(1-\alpha)\right) = 1-\alpha$$

How do we use this?

Suppose that we obtained $\hat{\rho} = 0.8$ when calculated for grades in two exams in a class of 20 students.

What can you say about the correlation between the grades in these two exams, in the entire world population?
Assume that the grades in these two exams follow a bivariate normal joint distribution.
Also, assume, of course, that our 20 students were sampled from the relevant (entire world) population.

How do we use this?

we obtained $\hat{\rho} = 0.8$ when calculated for grades in two exams in a class of 20 students.

Therefore, working with $\alpha = 0.05$, say, we have

$$F(\rho) \in F(0.8) \pm \frac{1}{\sqrt{17}} \Phi^{-1}(0.975) = 1.099 \pm 0.243 \cdot 1.96 = [0.624 , 1.574]$$

To be useful, we now want to convert this to a CI for ρ .
Since the Fisher Transform, F, is monotone and invertible we get:

$$\rho \in F^{-1}([0.624 , 1.574]) = [F^{-1}(0.624) , F^{-1}(1.574)] = [0.554 , 0.918] .$$

And so we can state that $0.554 \leq \rho \leq 0.918$ with 95% confidence.

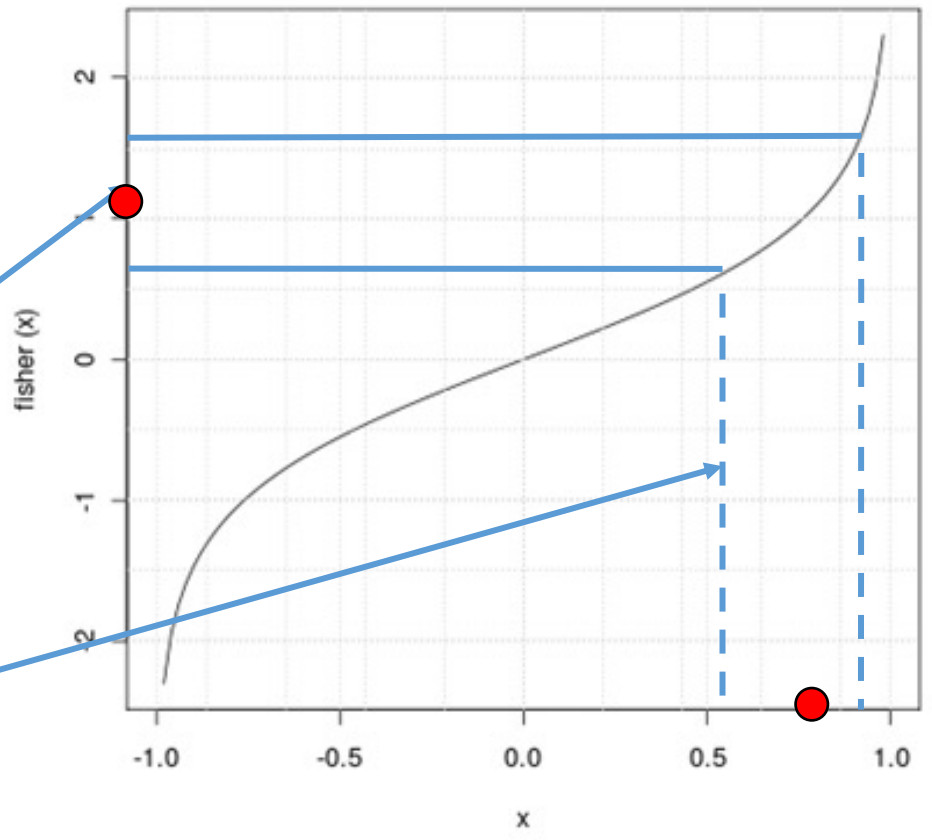
Inverting the Fisher Transform

Note: the CI around ρ is NOT symmetric

$$r = F^{-1}(u) = \frac{e^{2u} - 1}{e^{2u} + 1}$$

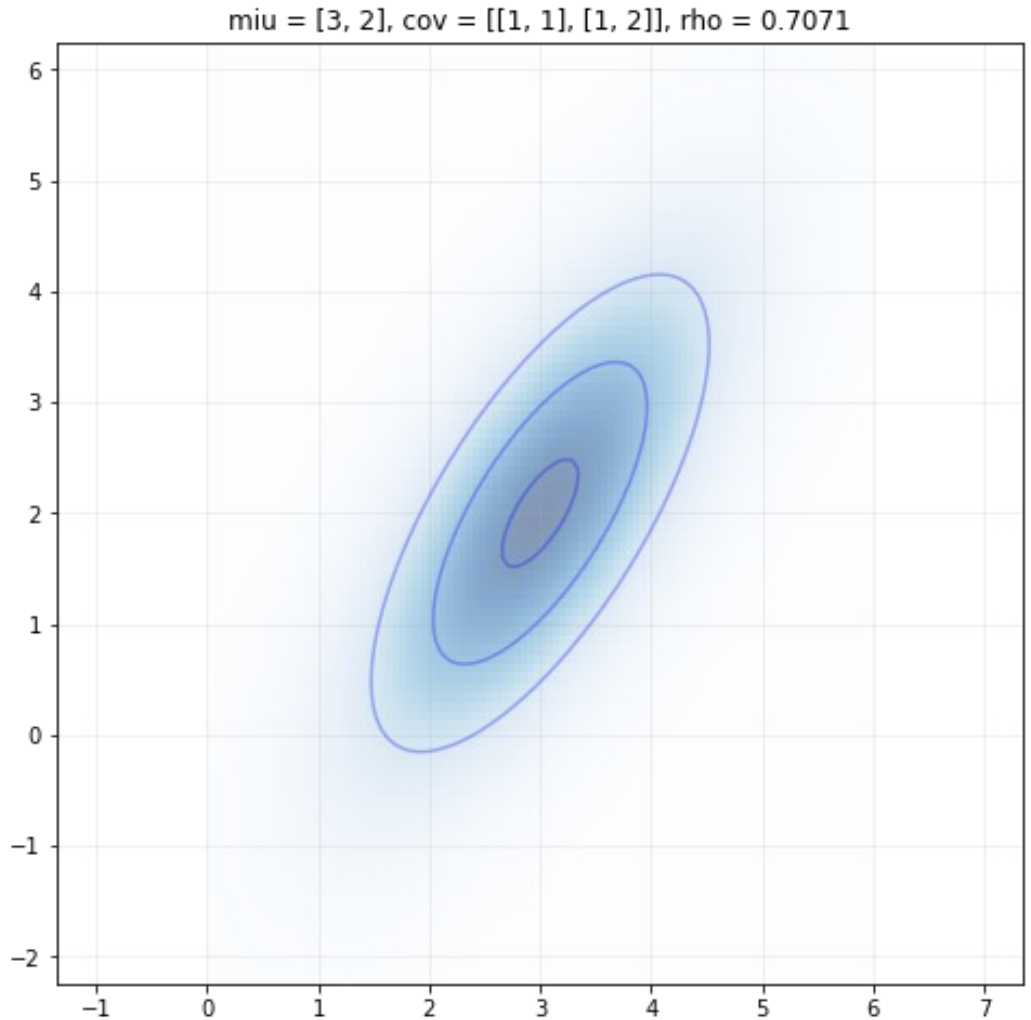
CI for $F(\rho)$, obtained from a normal cdf

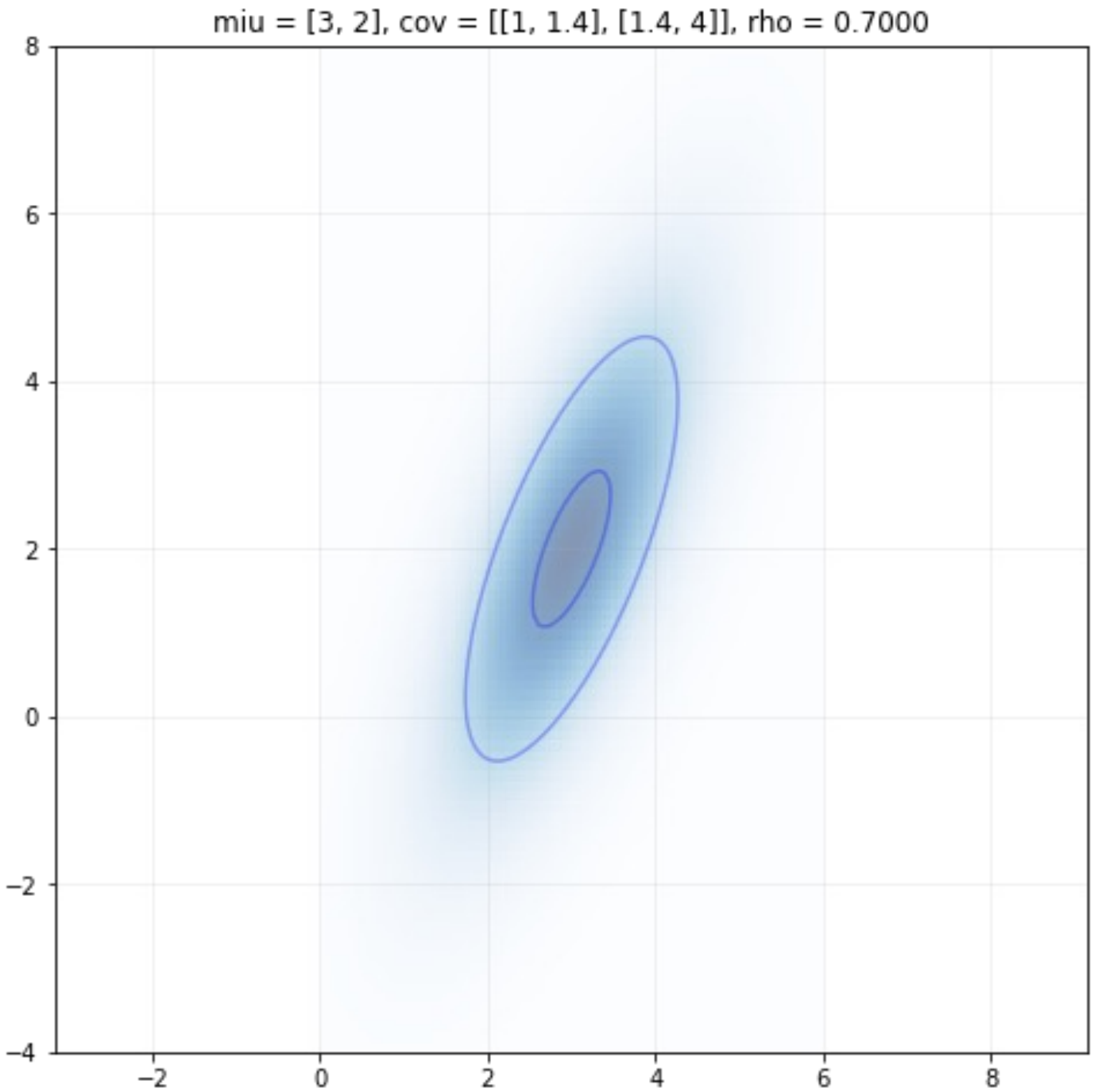
CI for ρ , obtained from the above by taking the inverse image, under F.

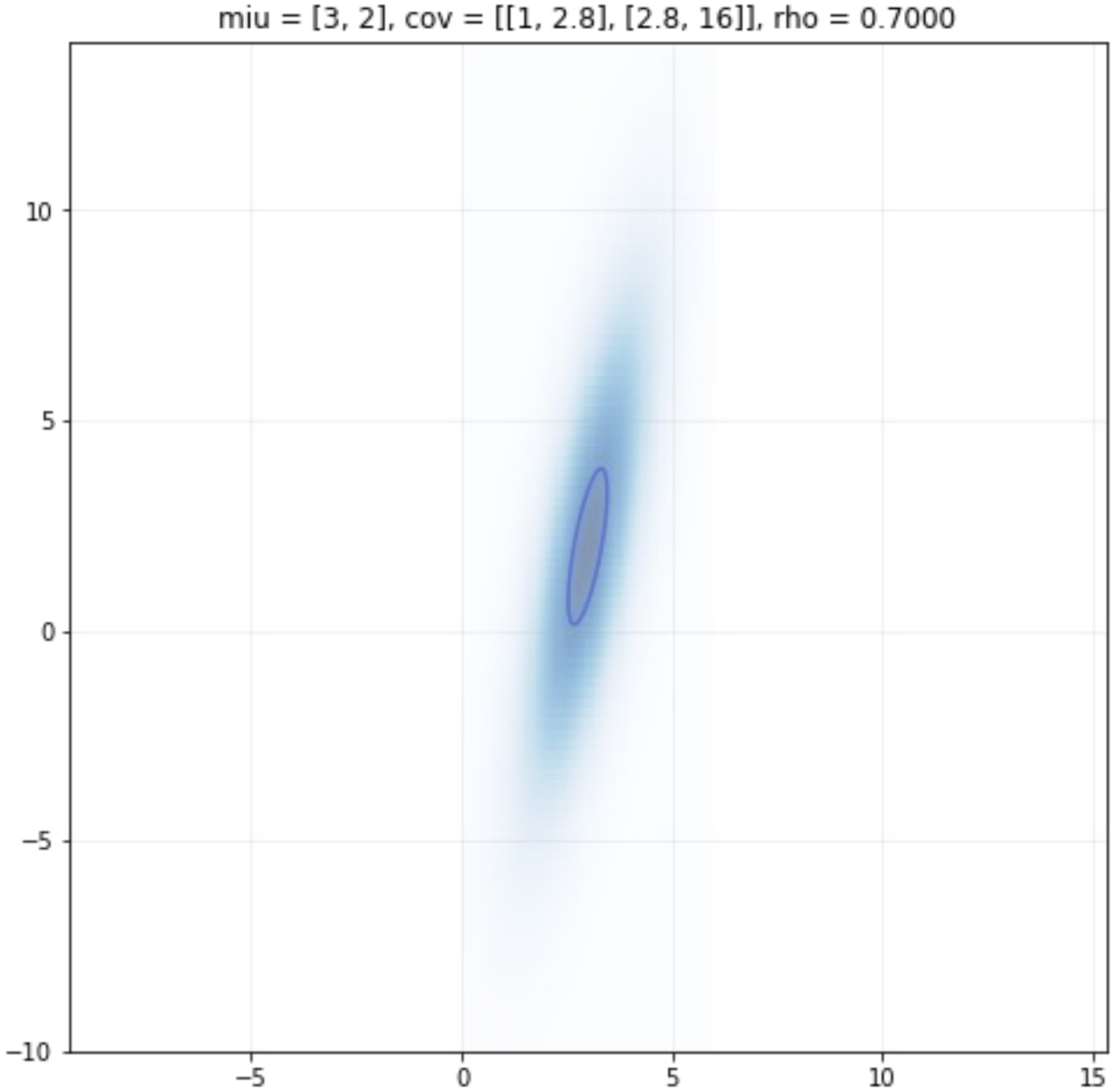


Bi-Var Gauss plotting

```
def plot_2d_Gaussian_pdf(means, cov):
    n = 100
    x1 = np.linspace(means[0] - 3 * np.sqrt(cov[0][0]), means[0] + 3 * np.sqrt(cov[0][0]), n)
    x2 = np.linspace(means[1] - 3 * np.sqrt(cov[1][1]), means[1] + 3 * np.sqrt(cov[1][1]), n)
    x1_v, x2_v = np.meshgrid(x1, x2)
    Xgrid = np.vstack([x1_v.ravel(), x2_v.ravel()]).T
    Y = mn.pdf(Xgrid, means, cov)
    fig, ax = plt.subplots(figsize= (8,8))
    ax.pcolorfast(x1, x2, Y.reshape(x1_v.shape), alpha=0.5, cmap='Blues')
    ax.contour(x1_v, x2_v, Y.reshape(x1_v.shape),
               levels=[0.05, 0.1, 0.15, 0.2], alpha=0.3, colors='b')
    ax.axis('equal')
    ax.grid(alpha=0.2)
    real_r = cov[0][1] / np.sqrt(cov[0][0] * cov[1][1])
    ax.set_title('miu = {}, cov = {}, rho = {:.4f}'.format(means, cov, real_r))
    plt.show()
```


miu = [3, 2], cov = [[1, 1], [1, 2]], rho = 0.7071
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4
5
6
7

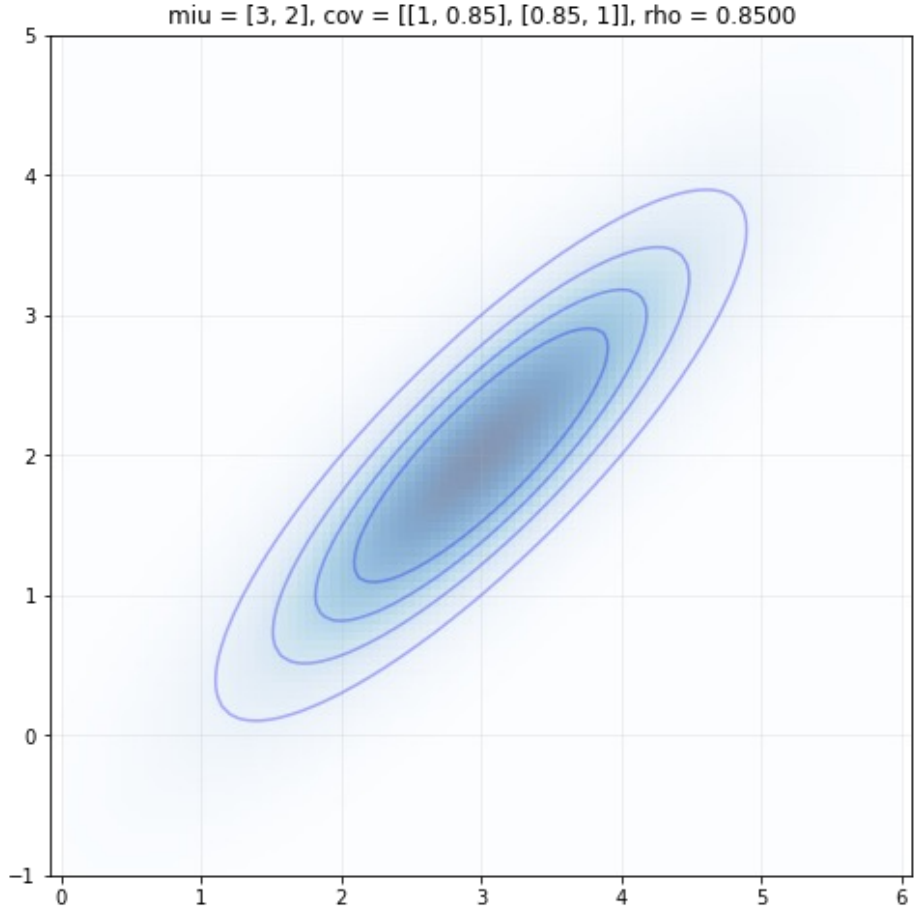

miu = [3, 2], cov = [[1, 1.4], [1.4, 4]], rho = 0.7000
8
6
4
2
0
−2
−4
−2
0
2
4
6
8

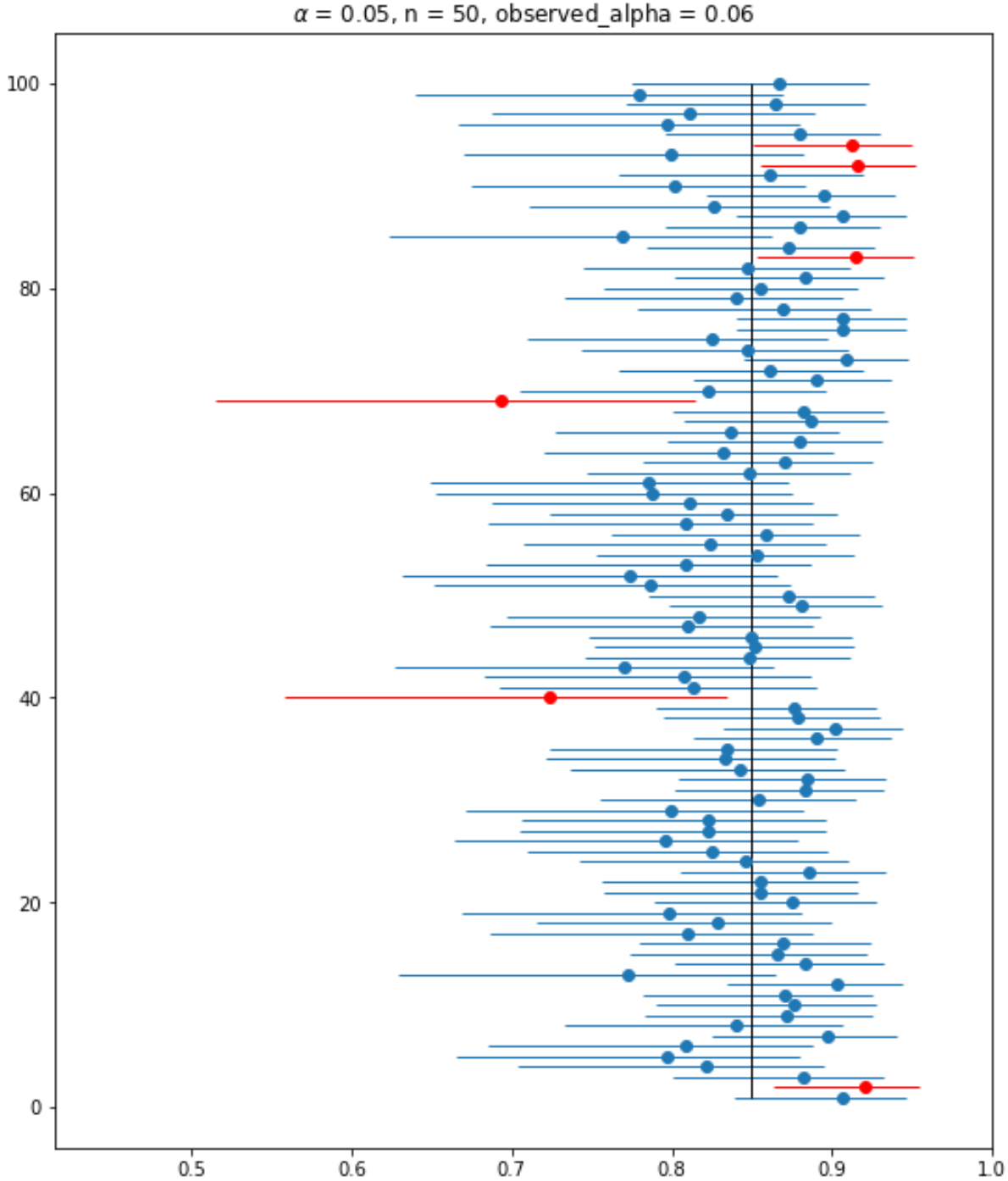

miu = [3, 2], cov = [[1, 2.8], [2.8, 16]], rho = 0.7000
10
5
0
−5
−10
−5
0
5
10
15

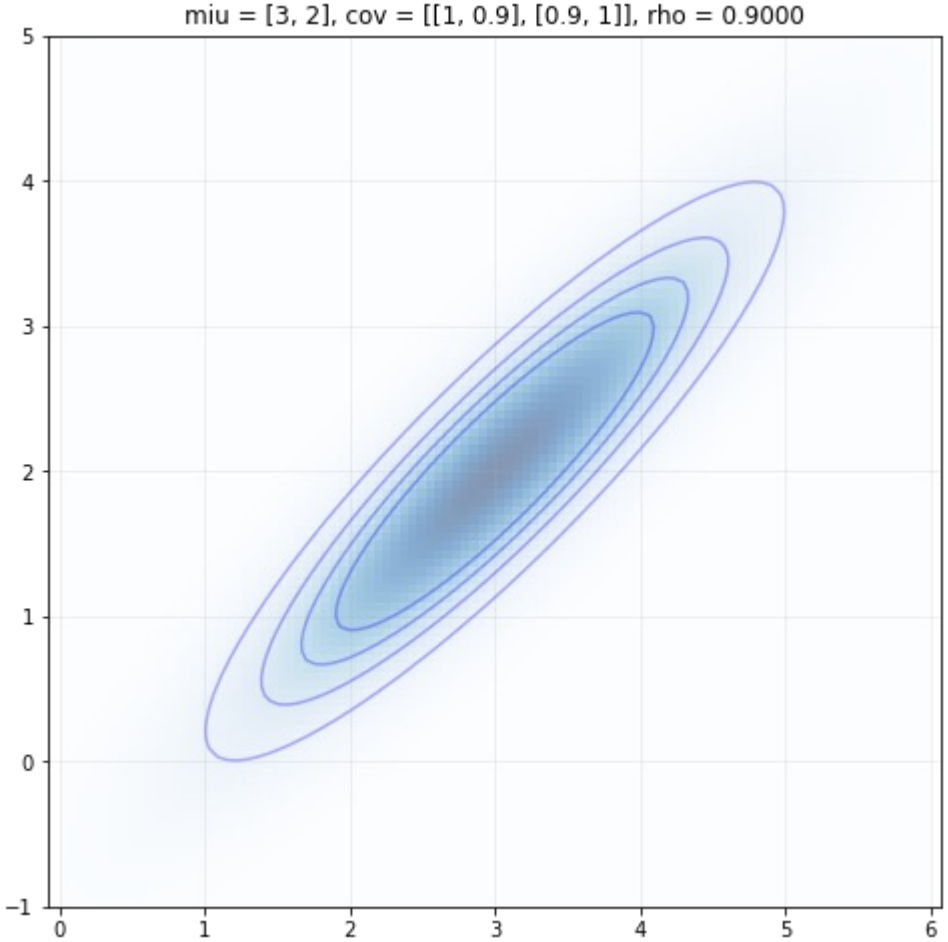
Bi-Var Gauss correlations – conf intervals

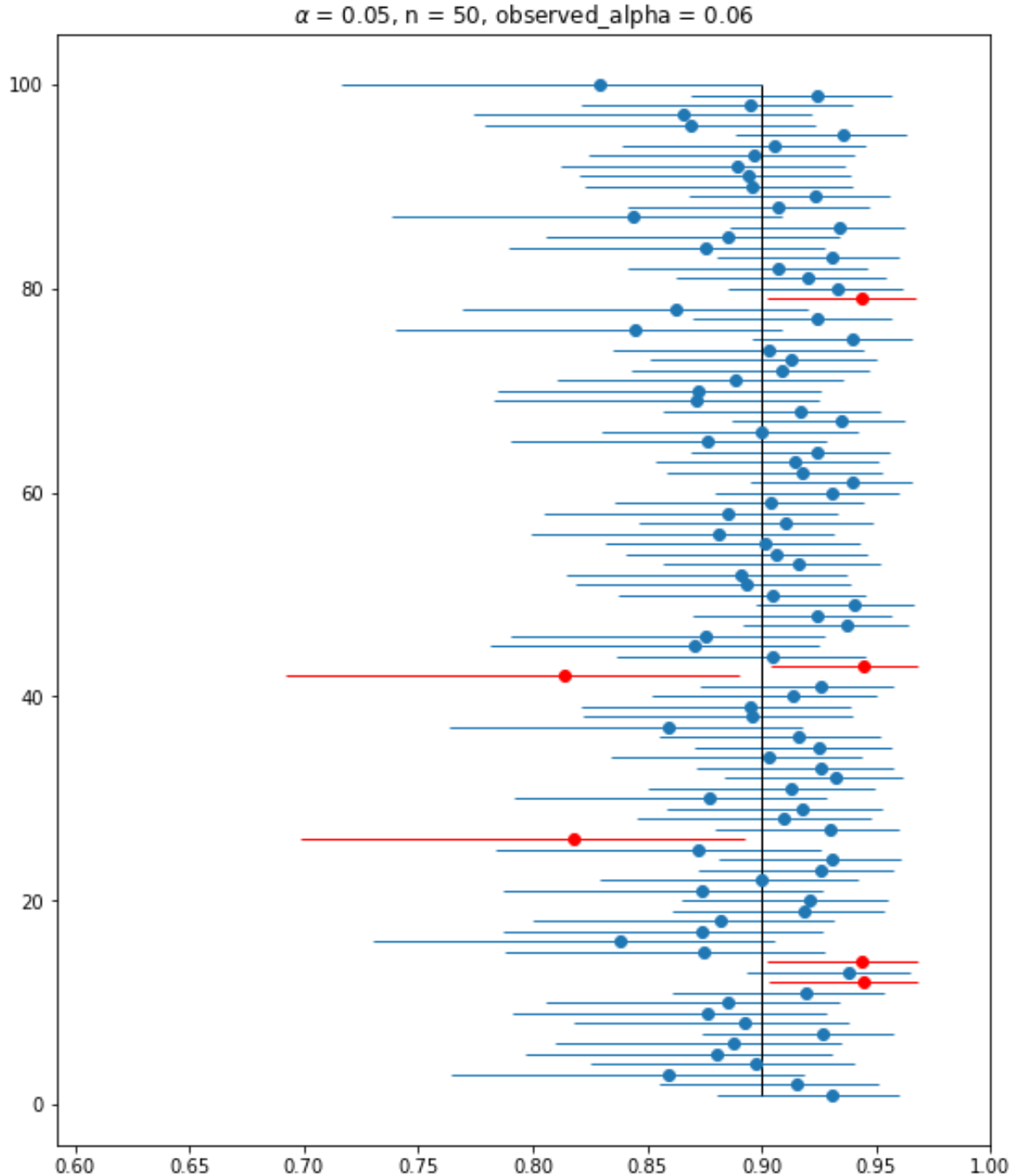
```
def corr_interval(r, z, n):
    # fisher interval
    f = 0.5 * np.log((1 + r) / (1 - r))
    f_L = f - z / np.sqrt(n - 3)
    f_U = f + z / np.sqrt(n - 3)
    # corr interval
    r_L = (np.e ** (2 * f_L) - 1) / (np.e ** (2 * f_L) + 1)
    r_U = (np.e ** (2 * f_U) - 1) / (np.e ** (2 * f_U) + 1)
    return r_L, r_U
```

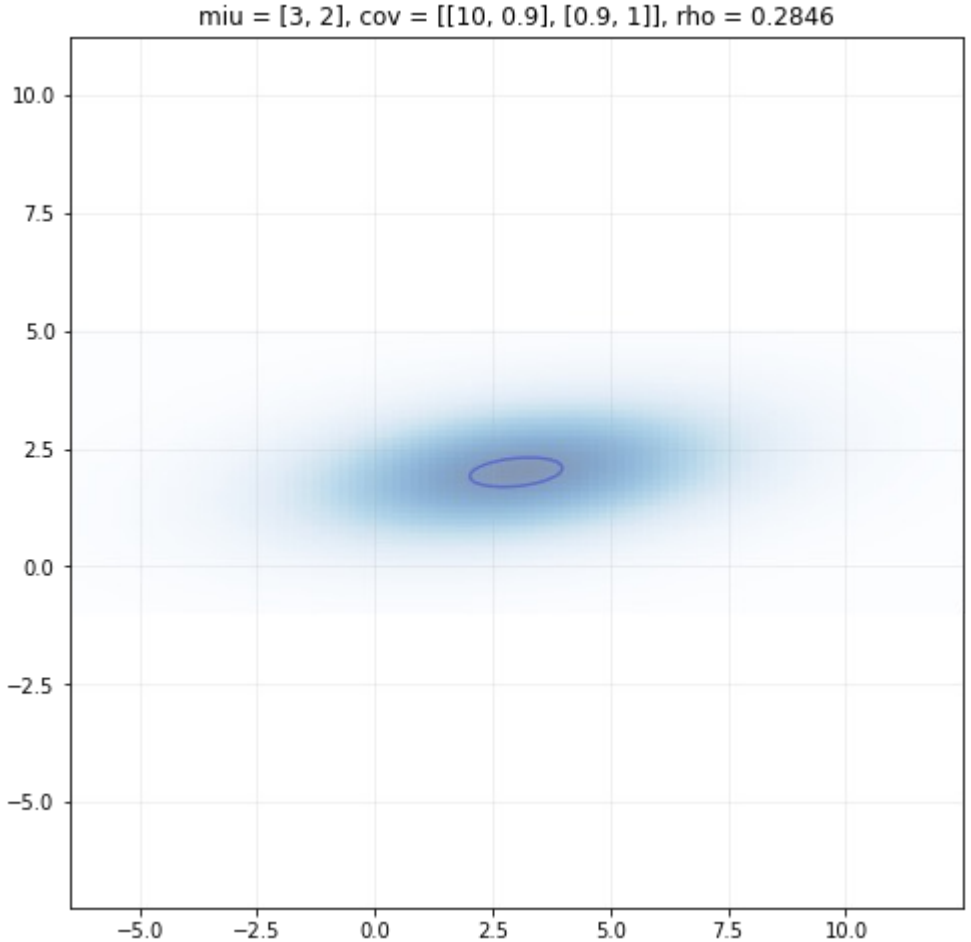
```
    z_critical = norm.ppf(q = 1-alpha/2)
    # Get the z-critical value (two tails)
```

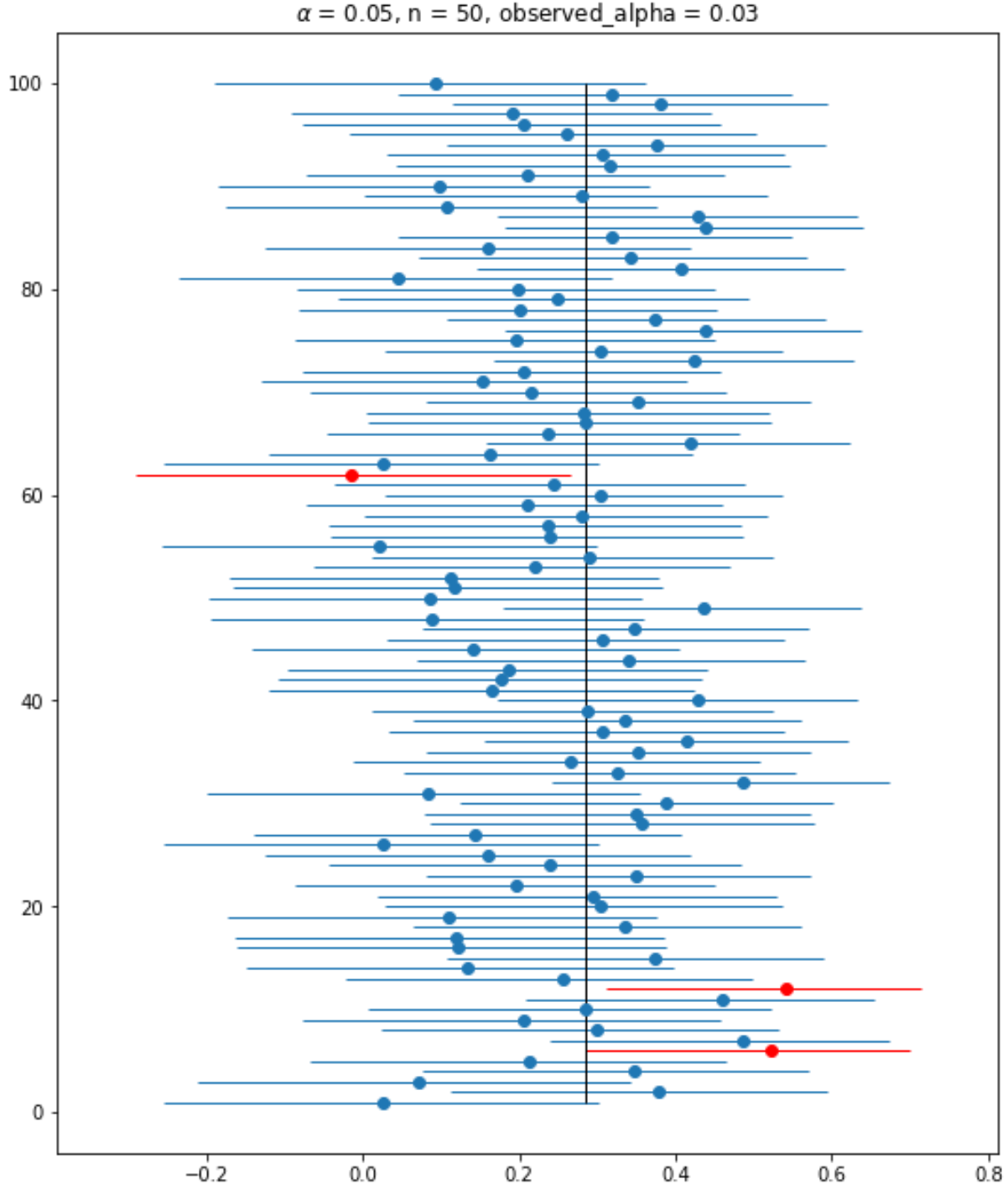

miu = [3, 2], cov = [[1, 0.85], [0.85, 1]], rho = 0.8500
5
4
3
2
1
0
−1
0
1
2
3
4
5
6


α = 0.05, n = 50, observed_alpha = 0.06
100
80
60
40
20
0
0.5
0.6
0.7
0.8
0.9
1.0


miu = [3, 2], cov = [[1, 0.9], [0.9, 1]], rho = 0.9000
5
4
3
2
1
0
−1
0
1
2
3
4
5
6


α = 0.05, n = 50, observed_alpha = 0.06
100
80
60
40
20
0
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00


miu = [3, 2], cov = [[10, 0.9], [0.9, 1]], rho = 0.2846
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


α = 0.05, n = 50, observed_alpha = 0.03
100
80
60
40
20
0
−0.2
0.0
0.2
0.4
0.6
0.8

p-Values for correlations

Built in functions typically compute p-values for Pearson's correlation using normal approximations (+Fisher), under the null model of X and Y following an independent standard bivariate normal distribution.

Built in functions compute p-values for Kendall's tau and Spearman's rho using either the exact permutation distributions (for small sample sizes), or large-sample normal approximations.
(what is the null model?)

In practice …

```
import pandas as pd
X = mvarnorm([0,0,0],1)
d = pd.DataFrame(X.rvs(8),columns=['X','Y','Z'])
print(d)

print('\n\n\n')
print('rho (Pearson):')
print(d.corr())
print('\np-value:')
print(d.corr(method=lambda x, y: pearsonr(x, y)[1]))

print('\n\n\n')
print('rho (Spearman):')
print(d.corr(method='spearman'))
print('\np-value:')
print(d.corr(method=lambda x, y: spearmanr(x, y)[1]))

print('\n\n\n')
print('tau (Kendall):')
print(d.corr(method='kendall'))
print('\np-value:')
print(d.corr(method=lambda x, y: kendalltau(x, y)[1]))
```

```
          X         Y         Z
0 -0.284804 -0.049761  1.072098
1  0.346245  0.464317 -1.467083
2 -1.058019  0.267875  0.508588
3  1.370080 -1.576222 -0.868370
4  1.581979  0.757885 -0.145476
5 -0.395770 -0.270037  0.158660
6  1.062945 -1.097698  0.486185
7  0.321568 -0.338953 -0.641666


rho (Pearson):
          X         Y         Z
X  1.000000 -0.306324 -0.403977
Y -0.306324  1.000000  0.020537
Z -0.403977  0.020537  1.000000

p-value:
          X         Y         Z
X  1.000000  0.460561  0.320918
Y  0.460561  1.000000  0.961503
Z  0.320918  0.961503  1.000000
```

```
rho (Spearman):
          X         Y         Z
X  1.000000 -0.071429 -0.523810
Y -0.071429  1.000000  0.047619
Z -0.523810  0.047619  1.000000

p-value:
          X         Y         Z
X  1.000000  0.866526  0.182721
Y  0.866526  1.000000  0.910849
Z  0.182721  0.910849  1.000000


tau (Kendall):
          X         Y         Z
X  1.000000 -0.142857 -0.357143
Y -0.142857  1.000000  0.071429
Z -0.357143  0.071429  1.000000

p-value:
          X         Y         Z
X  1.000000  0.719544  0.275099
Y  0.719544  1.000000  0.904861
Z  0.275099  0.904861  1.000000
```

In practice …

```
import pandas as pd
sig = np.array([[1,0.8,0],[0.8,1,-0.5],[0,-0.5,1]])
print('Sigma:')
print(sig)
X = mvarnorm([0,0,0],sig)
d = pd.DataFrame(X.rvs(8),columns=['X','Y','Z'])
print(d)
```

```
Sigma:
[[ 1.   0.8  0. ]
 [ 0.8  1.  -0.5]
 [ 0.  -0.5  1. ]]
          X         Y         Z
0 -0.909012 -0.778479 -0.316023
1  1.256040  0.199301  0.508225
2  0.495742  0.200058 -0.154261
3 -0.010702  0.204850 -0.235058
4  1.122977  1.193013  0.408497
5 -0.354397  0.042358 -0.080204
6 -0.260343 -0.472427  0.626067
7 -0.396123 -0.284651 -0.426412
```

```
rho (Spearman):
          X         Y         Z
X  1.000000  0.761905  0.619048
Y  0.761905  1.000000  0.166667
Z  0.619048  0.166667  1.000000

p-value:
          X         Y         Z
X  1.000000  0.028005  0.101733
Y  0.028005  1.000000  0.693239
Z  0.101733  0.693239  1.000000
```

```
rho (Pearson):
         X         Y         Z
X  1.00000  0.809750  0.592410
Y  0.80975  1.000000  0.334095
Z  0.59241  0.334095  1.000000

p-value:
          X         Y         Z
X  1.000000  0.014852  0.121752
Y  0.014852  1.000000  0.418625
Z  0.121752  0.418625  1.000000
```

```
tau (Kendall):
          X         Y         Z
X  1.000000  0.571429  0.500000
Y  0.571429  1.000000  0.071429
Z  0.500000  0.071429  1.000000

p-value:
          X         Y         Z
X  1.000000  0.061012  0.108681
Y  0.061012  1.000000  0.904861
Z  0.108681  0.904861  1.000000
```

Summary

- Correlation measures serve to quantify relationships between different aspects of observed data
- Statistical assessment under a null model
- Pearson correlation is the classical and most popular correlation coefficients. It’s a sample version of the population covariance.
- Pearson correlation affords confidence interval inference, under assumptions of normality.
- Normal assumptions don’t always hold.
- Kendall and Spearman correlations are parameter free correlation measures, based on ranks.
- Kendall and Spearman correlations afford a clear, well-defined null model as well as confidence interval inference.