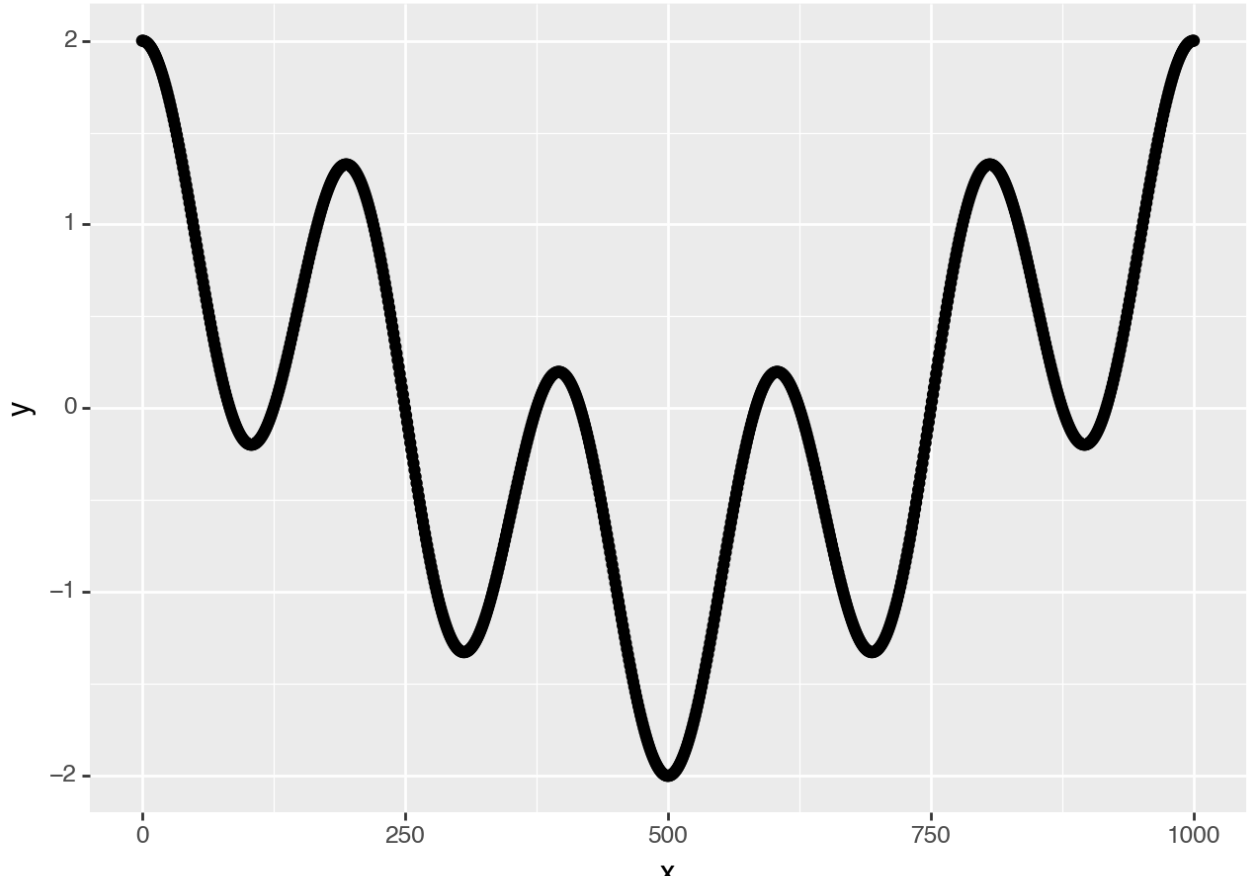
Nonlinear Correlation: beyond Spearman

In [2]:

```
# Set up: x denotes a predictor and y is a nonlinear function of x

np.random.seed(0)
x = np.arange(1001)
df = pd.DataFrame({'x': x,
                   'y': np.cos(x * np.pi / 100) +
                        np.cos(x * np.pi / 500)})
ggplot(data=df, mapping=aes(x="x", y="y")) + geom_point()
```



In [3]:

```
# Calculate Pearson and Spearman correlation coefficients.
# As expected, these correlations are close to zero

print(f"Rp(x, y) = {pearsonr(df['x'], df['y'])}")
print(f"Rs(x, y) = {spearmanr(df['x'], df['y'])}")
```

```
Rp(x, y) = PearsonRResult(statistic=-1.1129220571718668e-16, pv
alue=0.9999999999999948)
Rs(x, y) = SignificanceResult(statistic=-0.0001206458320702504
8, pvalue=0.9969582368382301)
```

In [4]:

```
# But we know that y is a deterministic function of x!
# Enter Chatterjee's correlation coefficient, XICOR.
# Unlike Spearman R, XICOR does not assume monotonicity.

print(f"Rs(x, y) = {xicor(df['x'], df['y'])}")
```

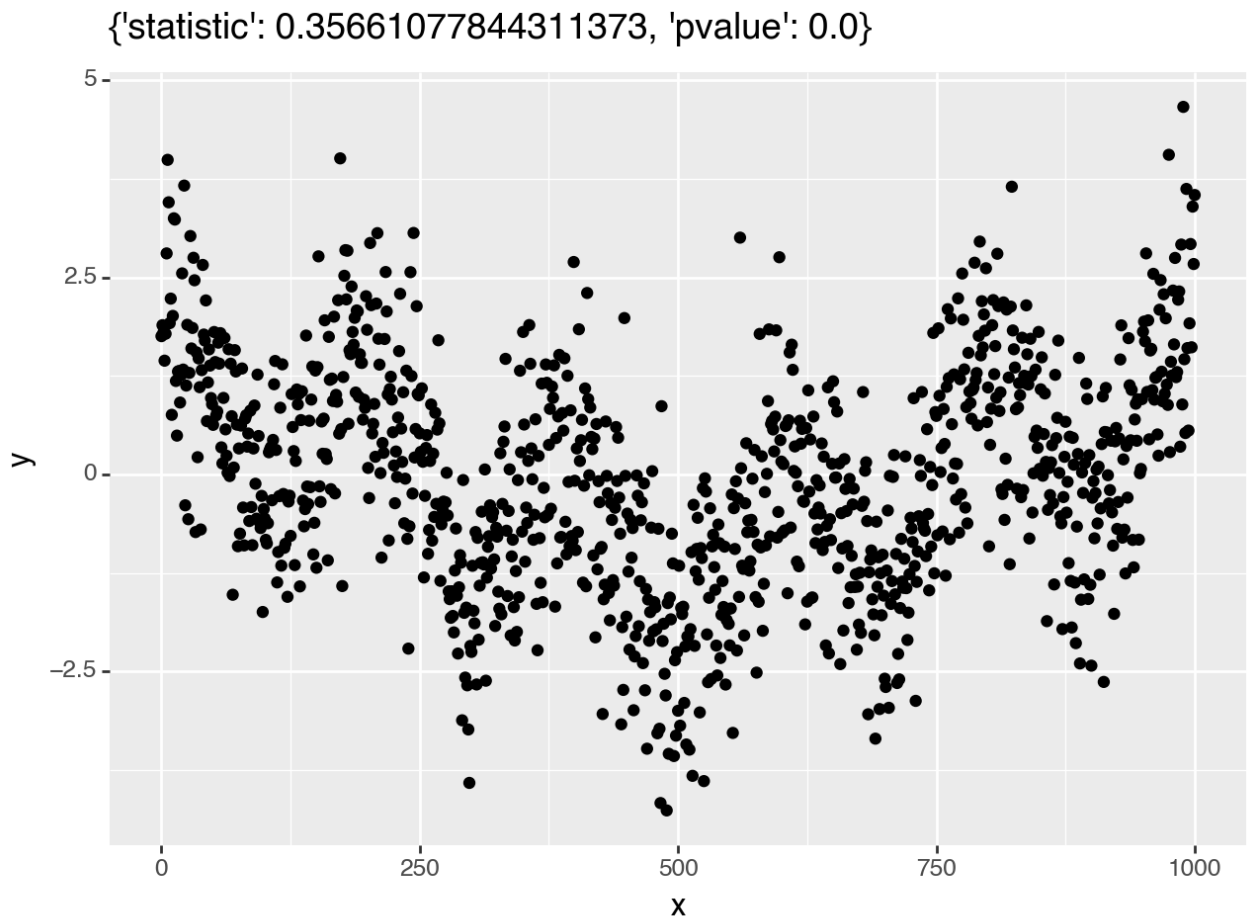
```
Rs(x, y) = {'statistic': 0.9807934131736527, 'pvalue': 0.0}
```

In [5]:

```
# let us now add noise to y and see what happens to XICOR

def add_noise_to_y(y, sd):
    return y + sd * np.random.normal(size=len(y))

def average_xicor(x, y, sd, reps):
    a = [xicor(x, add_noise_to_y(y, sd)) for _ in range(reps)]
    avg_statistic = np.mean([t['statistic'] for t in a])
    avg_pvalue = np.mean([t['pvalue'] for t in a])
    return avg_statistic, avg_pvalue

def get_stats(x, y, sd):
    avg_statistic, avg_pvalue = average_xicor(x, y, sd, 2000)
    return pd.DataFrame({'noise_sd': sd,
                         ' ': ['statistic', 'pvalue'],
                         'value': [avg_statistic, avg_pvalue]})

noise_sd = np.linspace(0, 10, 100)
noise_df = pd.concat([get_stats(df['x'], df['y'], sd)
                      for i, sd in enumerate(noise_sd)],
                     axis=0, ignore_index=True)

ggplot(data=noise_df) + \
    geom_line(aes(x="noise_sd", y="value", color=" ", group=" "))
```

In [6]:

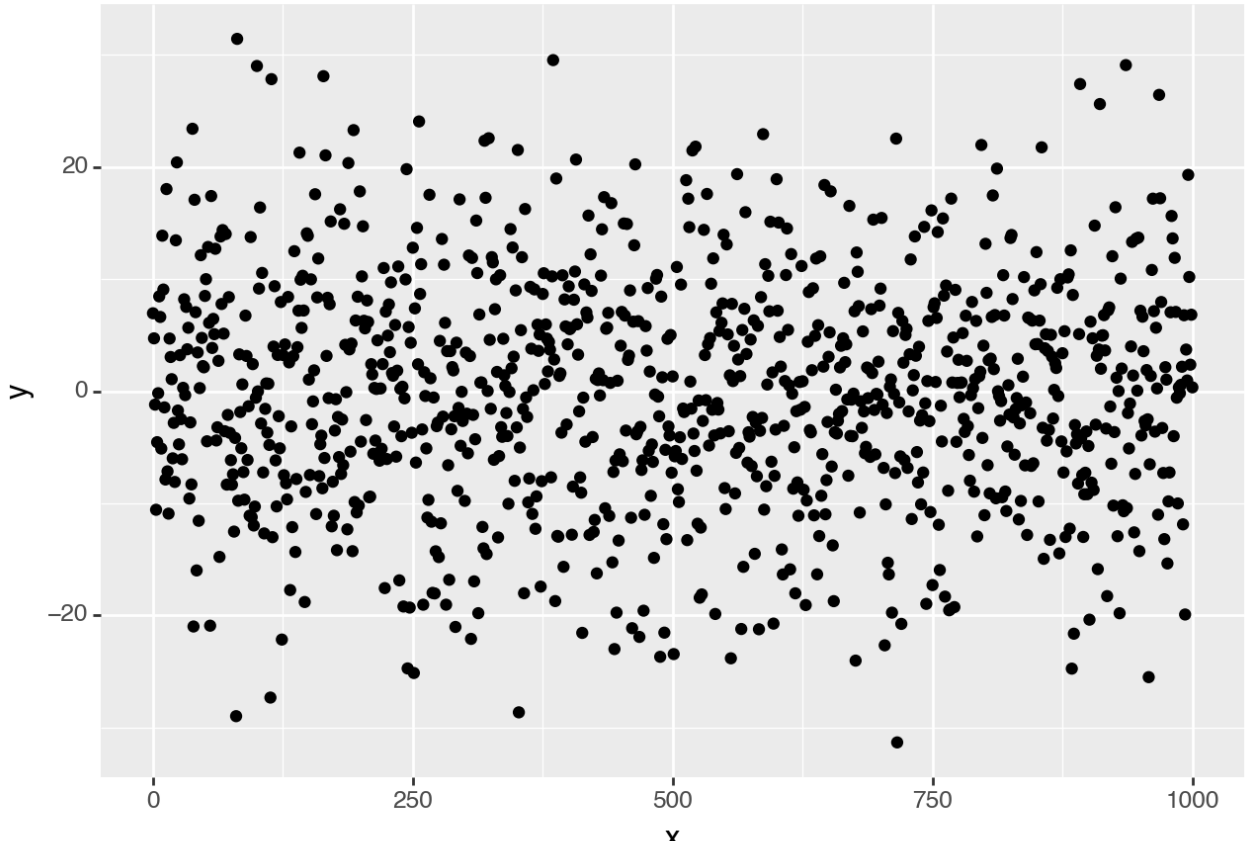
```
# When noise_sd = 1, we still discern the effect of x on y

y = add_noise_to_y(df['y'], 1)
ggplot(data=pd.DataFrame({'x': x, 'y': y})) + \
    geom_point(aes(x="x", y="y")) + \
    ggtitle(xicor(x, y))
```



In [7]:

```
# In contrast, when noise_sd=10, R ~ 0 (possibly negative)
# With XICOR, R < 0 does not imply anti-correlation;
# rather, it implies no correlation.

y = add_noise_to_y(df['y'], 10)
ggplot(data=pd.DataFrame({'x': x, 'y': y})) + \
    geom_point(aes(x="x", y="y")) + \
    ggtitle(xicor(x, y))
```

{'statistic': -0.029922155688622665, 'pvalue': 0.9327845890243527}



Conclusion

This notebook presents a novel nonlinear correlation coefficient which, unlike Spearman's R, does not assume monotonicity. My repo contains the Python version of the original R code in package XICOR (using asymptotic p-values only).

References:

[1] Chatterjee S (2021). A new coefficient of correlation. JASA 116:536, 2009-2022, DOI: 10.1080/01621459.2020.1758115

[2] Chatterjee S, Holmes S (2023). XICOR: Robust and generalized correlation coefficients. https://github.com/spholmes/XICOR, https://CRAN.R-project.org/package=XICOR.