In [3]:
```
%matplotlib inline
import pylab as plt
import jax.numpy as jnp
import seaborn as snb

from scipy.stats import binom as binom_dist
from scipy.stats import beta as beta_dist
from scipy.special import beta as beta_fun
from scipy.special import binom

snb.set_style('darkgrid')
snb.set(font_scale=1.5)
plt.rcParams['lines.linewidth'] = 3
```

Part 1

Consider the following regression model

$$y(x) = f(x) + e = w_0 + w_1 x^2 + w_2 \sin x + w_3 x + e,$$

such that $y_n = f(x_n) + e_n$, where $x_n, y_n \in \mathbb{R}$ are input and targets, respectively. The additive noise $e_n \in \mathbb{R}$ is assumed to be i.i.d from a zero-mean Gaussian distribution, i.e. $e_n \sim \mathcal{N}(0, \beta^{-1})$ for $\beta > 0$.

Let

$$\mathbf{x} = [2.29, -1.8, -0.06, 3.72, 2.6, -5.93, -0.15]$$

and

$$\mathbf{y} = [3.17, -4.53, -0.78, 3.15, 4.76, -1.96, -1.32]$$

denote the vector of inputs and targets, respectively, for a dataset with $N = 5$ observations.

Let

$$\mathbf{w} = [w_0, w_1, w_2, w_3] \in \mathbb{R}^4$$

denote the parameter vector.

Question 1.1:

> Compute and report a maximum likelihood estimate for $\mathbf{w}$ and β.

In [28]:
```
X = jnp.array([2.29, -1.8, -0.06, 3.72, 2.6, -5.93, -0.15])[: , None]

y = jnp.array([3.17, -4.53, -0.78, 3.15, 3.76, -1.96, -1.32])[:, None]

def design_matrix(x):
```

```
    return jnp.column_stack([jnp.ones(len(x)), x**2, jnp.sin(x), x])

design = design_matrix(X)
print(design)


w_map = jnp.linalg.solve(design.T @ design, design.T @ y )

print(w_map)
```

```
[[ 1.0000000e+00  5.2440996e+00  7.5233060e-01  2.2900000e+00]
 [ 1.0000000e+00  3.2399998e+00 -9.7384763e-01 -1.8000000e+00]
 [ 1.0000000e+00  3.5999999e-03 -5.9964005e-02 -5.9999999e-02]
 [ 1.0000000e+00  1.3838400e+01 -5.4669106e-01  3.7200000e+00]
 [ 1.0000000e+00  6.7599993e+00  5.1550144e-01  2.5999999e+00]
 [ 1.0000000e+00  3.5164898e+01  3.4588841e-01 -5.9299998e+00]
 [ 1.0000000e+00  2.2500001e-02 -1.4943814e-01 -1.5000001e-01]]
[[-0.871454  ]
 [ 0.11179117]
 [ 2.1343005 ]
 [ 0.97465134]]
```

In [39]:
```
rss = jnp.sum((y- design @ w_map)**2)

beta_mle = len(x) / rss
```

In [40]:
```
print(beta_mle)
```

```
22.028181
```

In [ ]: