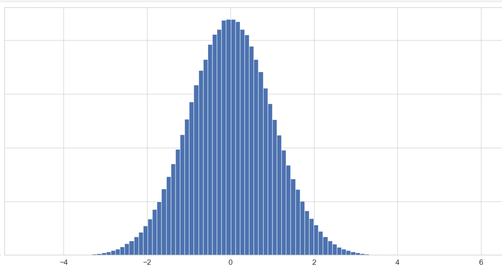
Examples

In [1]:
```
import numpy as np
import pandas as pd
import scipy.stats as sts

import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid", font_scale=1.5)
sns.despine()

%matplotlib inline
```

In [2]:
```
def get_ax():
    fig, ax = plt.subplots(figsize=(20,10))
    return ax
```

Example 1

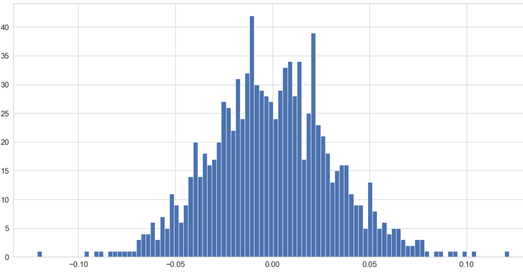
In [3]:
```
N = 1_000_000
theta = 0
population = np.random.randn(N) + theta
get_ax().hist(population, bins=100);
```



In [4]:
```
n = 1_000
sample = np.random.choice(population, size=n, replace=False)
estimate = sample.mean()
print(f"mean-estimate: {estimate:5f}")
```

```
mean-estimate: -0.004028
```

In [5]:
```
m = 1_000
samples = np.array([np.random.choice(population, size=n, replace=False) for _ in range(m)])
sampling_distribution = estimates = np.array([sample.mean() for sample in samples])
get_ax().hist(sampling_distribution, bins=100);
```



In [6]:
```
mean_estimate = sampling_distribution.mean()
numerical_bias = mean_estimate - theta
expected_estimate = theta
analytical_bias = expected_estimate - theta
print(f"Estimator bias (numerical): {numerical_bias:5f}")
print(f"Estimator bias (analytical): {analytical_bias:5f}")
```

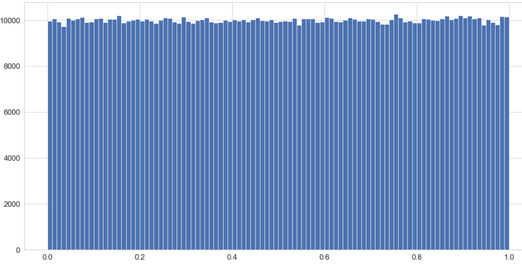
```
Estimator bias (numerical): -0.000180
Estimator bias (analytical): 0.000000
```

In [7]:
```
sem_numerical = np.sqrt(sampling_distribution.var())
std = sample.std()
sem_numerical_std = std / np.sqrt(n)
sem_scipy = sts.sem(sample)
sem_analytical = np.sqrt(1 / n)
print(f"Estimator standard error (numerical): {sem_numerical:5f}")
print(f"Estimator standard error (numerical via std): {sem_numerical_std:5f}")
print(f"Estimator standard error (numerical via scipy): {sem_scipy:5f}")
print(f"Estimator standard error (analytical): {sem_analytical:5f}")
```

```
Estimator standard error (numerical): 0.031657
Estimator standard error (numerical via std): 0.031115
Estimator standard error (numerical via scipy): 0.031131
Estimator standard error (analytical): 0.031623
```

Example 2

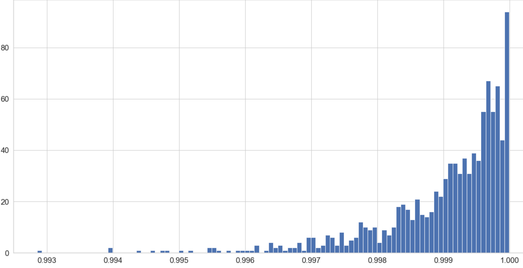
In [8]:
```
N = 1_000_000
theta = 1
population = np.random.rand(N) * theta
get_ax().hist(population, bins=100);
```



In [9]:
```
n = 1_000
sample = np.random.choice(population, size=n, replace=False)
estimate = sample.max()
print(f"max-estimate: {estimate:5f}")
```

```
max-estimate: 0.999887
```

In [10]:
```
m = 1_000
samples = np.array([np.random.choice(population, size=n, replace=False) for _ in range(m)])
sampling_distribution = estimates = np.array([sample.max() for sample in samples])
get_ax().hist(sampling_distribution, bins=100);
```



In [11]:
```
mean_estimate = sampling_distribution.mean()
numerical_bias = mean_estimate - theta
expected_estimate = n / (n + 1) * theta ** (n + 1)
analytical_bias = expected_estimate - theta
print(f"Estimator bias (numerical): {numerical_bias:5f}")
print(f"Estimator bias (analytical): {analytical_bias:5f}")
```

```
Estimator bias (numerical): -0.000973
Estimator bias (analytical): -0.000999
```

In [12]:
```
sem_numerical = np.sqrt(sampling_distribution.var())
sem_analytical = np.sqrt(n / (n + 2) * theta ** 2 - expected_estimate ** 2)
print(f"Estimator standard error (numerical): {sem_numerical:5f}")
print(f"Estimator standard error (analytical): {sem_analytical:5f}")
```

```
Estimator standard error (numerical): 0.000992
Estimator standard error (analytical): 0.000998
```

Can not use std. Why?

std is for estimates of mean only.

Example 3

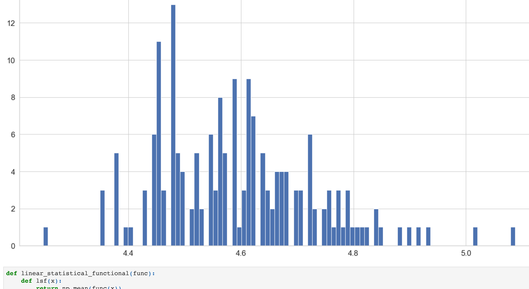
In [13]:
```
import sklearn.datasets as ds
```

In [14]:
```
wine_ds = ds.load_wine()
df = pd.DataFrame(wine_ds["data"], columns=wine_ds["feature_names"])
df.head()
```

Out[14]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wine
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.9
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.4
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.1
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.4
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.9

In [15]:
```
sample = df["magnesium"].transform(np.log).to_numpy()
get_ax().hist(sample, bins=100);
```



In [16]:
```
def linear_statistical_functional(func):
    def laf(x):
        return np.mean(func(x))

    return laf
```

In [17]:
```
sample_mean = sample.mean()
plug_in_estimator_for_mean = linear_statistical_functional(lambda x: x)
print(f"Sample mean: {sample_mean:5f}")
print(f"Plug-in estimator for mean: {plug_in_estimator_for_mean(sample):5f}")
```

```
Sample mean: 4.593042
Plug-in estimator for mean: 4.593042
```

In [18]:
```
sample_var = sample.var()
plug_in_estimator_for_var = linear_statistical_functional(lambda x: (x - sample_mean) ** 2)
print(f"Sample std: {sample_var:5f}")
print(f"Plug-in estimator for std: {plug_in_estimator_for_var(sample):5f}")
```

```
Sample std: 0.018562
Plug-in estimator for std: 0.018562
```