In [1]:

```
import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))

import numpy as np
import pandas as pd
```

Regression

Thomas Schmelzer

Linear Regression

Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{b} \in \mathbb{R}^{n}$. Solve the unconstrained least squares problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^m} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

The ith column of $\mathbf{A}$ may represent the time series of returns for asset i.

Portfolio Optimisation is about all about clever (linear) combinations of assets.

Examples:

- Tracking an index (index in $\mathbf{b}$, assets in $\mathbf{A}$)
- Constructing an indicator, factor analyis, ...
- Approximation...
- ...

Regression is the **Swiss army knife** of professional quant finance.

The normal equations

As we (probably) all know

$$\mathbf{x}^* = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{x}$$

solves

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

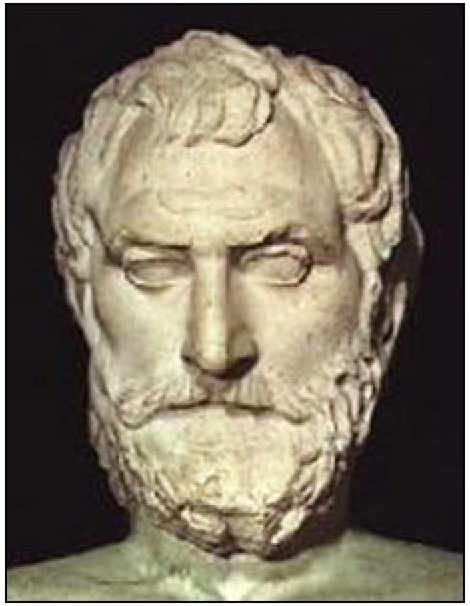
You may see here already

Constrained regression

Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{b} \in \mathbb{R}^n$. We solve the constrained least squares problem:

$$\begin{aligned} \mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^m} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \\ \text{s.t. } \Sigma\, x_i = 1 \\ \mathbf{x} \geq 0 \end{aligned}$$

The Sculptor method



Thales of Miletus (c. 624 BC - c. 546 BC)

Shall we apply the sculptor method?

- We could delete the negative entries (really bad if they are all negative)
- We could scale the surviving entries to enforce the $\Sigma x_i = 1$.

Done?

Conic Programming

We introduce an auxiliary scalar z:

$$
\begin{aligned}
\min_{z \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^m} \quad & z \\
\text{s.t.} \quad & z \geq \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \\
& \sum x_i = 1 \\
& \mathbf{x} \geq 0
\end{aligned}
$$

We introduce an auxiliary vector $\mathbf{y} \in \mathbb{R}^n$:

$$\begin{aligned}
\min_{z \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n} \quad & z \\
\text{s.t.} \quad & z \geq \|y\|_2 \\
& \mathbf{y} = \mathbf{Ax} - \mathbf{b} \\
& \Sigma x_i = 1 \\
& \mathbf{x} \geq 0
\end{aligned}$$

We **lifted** the problem from a m dimensional space into a $m+n+1$ dimensional space.

Nerd alarm:

$$z \geq \|y\|_2 \Leftrightarrow [z, y] \in \mathcal{Q}_{n+1}$$

Application: Implementing a minimum variance portfolio

The ith column of $\mathbf{A}$ is the time series of returns for the ith asset. Hence to minimize the variance of a portfolio (a linear combination of assets) we solve:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \|\mathbf{A}\mathbf{w} - \mathbf{0}\|_2$$
$$\text{s.t. } \Sigma w_i = 1$$
$$\mathbf{w} \geq 0$$

In [2]:

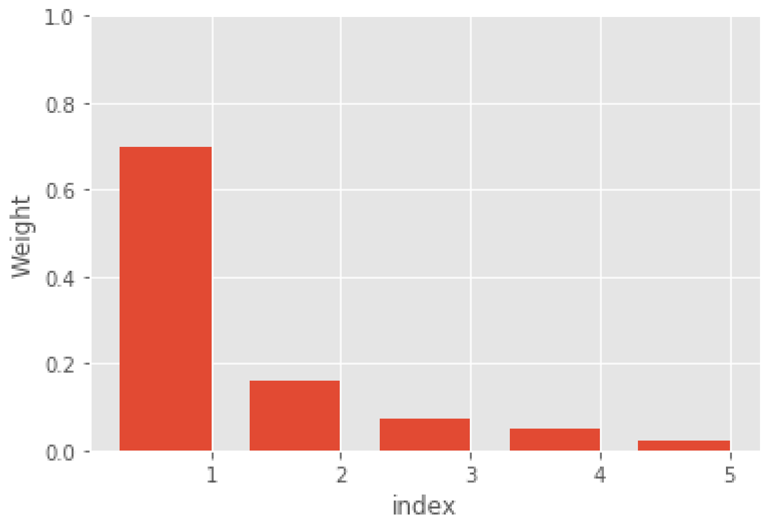
```python
from cvx.util import minimize, cvx

def min_var(matrix, lamb=0.0):
    """
    min 2-norm (matrix*w) + lamb*2-norm(w)
    s.t. e'w = 1, w >= 0
    """
    w = cvx.Variable(matrix.shape[1])
    minimize(objective=cvx.norm(matrix*w,2)+lamb*cvx.norm(w,2),
             constraints=[0 <= w, cvx.sum(w) == 1])
    return w.value
```

In [3]:
```python
def plot(ax, data, width=0.35, title=""):
    ax.bar(np.arange(5)+1-width, data, 2*width)
    ax.set_ylabel("Weight"), ax.set_xlabel("index"), ax.set_title(title)
    ax.set_ylim([0,1])
    return ax

random_data = np.dot(np.random.randn(250,5), np.diag([1,2,3,4,5]))
data = min_var(random_data)

fig, ax = plt.subplots()
plot(ax, data)
plt.show()
```

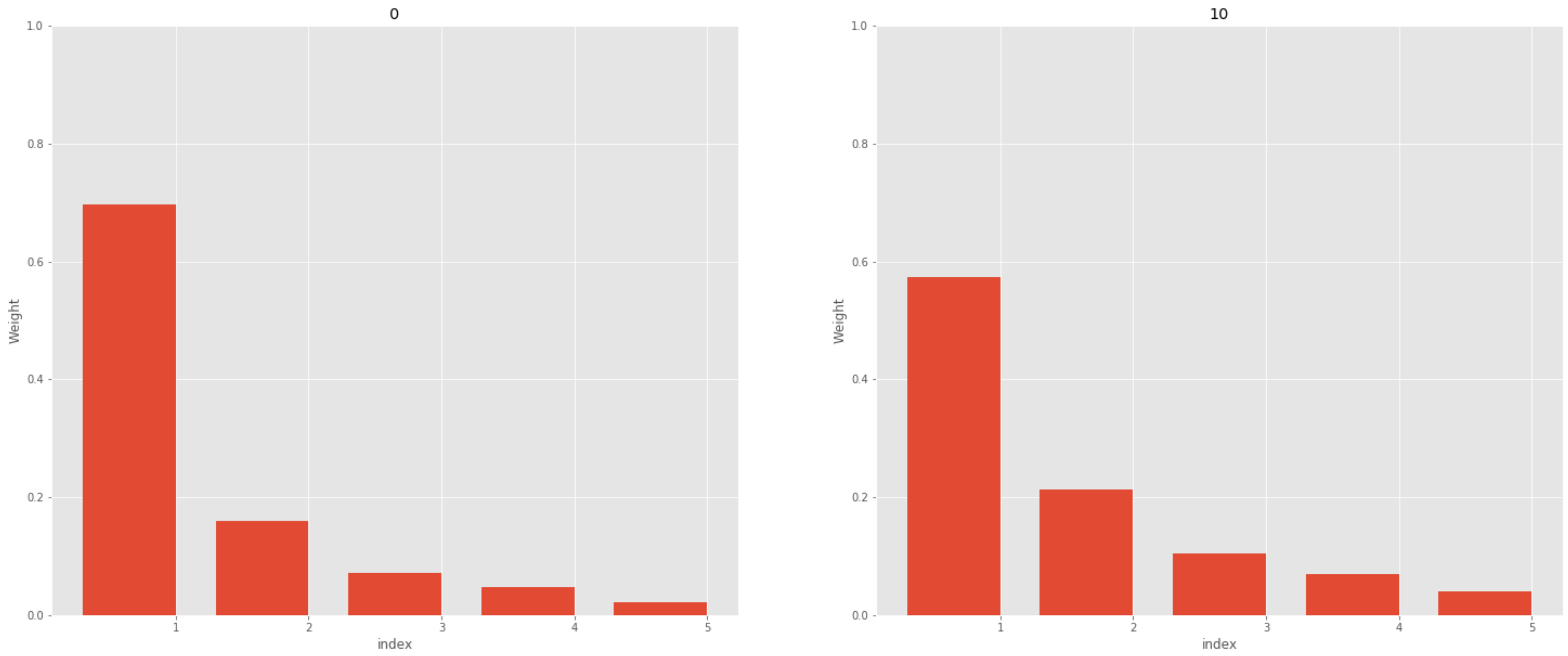


Balance?

- Bounds
- **Tikhonov regularization** (penalty by the 2-norm of the weights in the objective), also known as **Ridge Regression** or **Shrinkage to the mean**

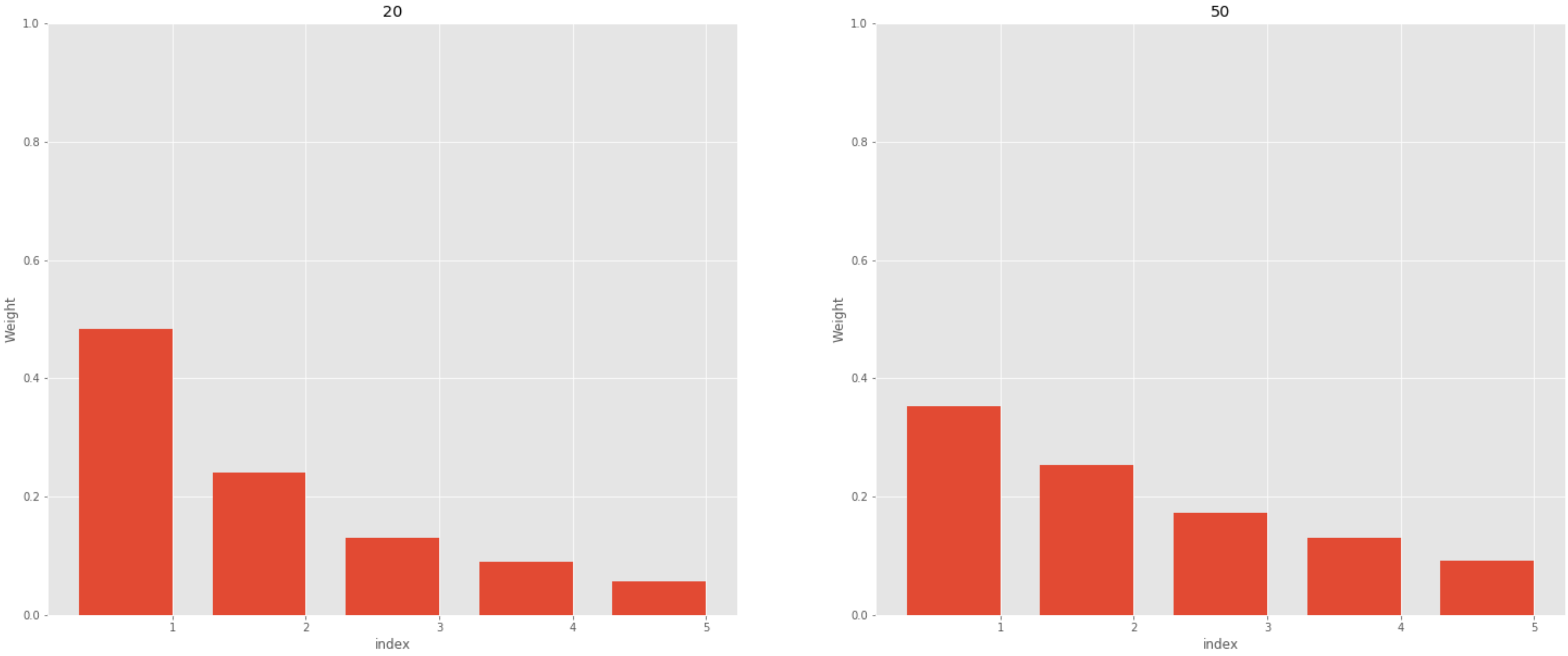
$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \|\mathbf{A}\mathbf{w}\|_2 + \lambda\|\mathbf{w}\|_2$$
$$\text{s.t. } \Sigma w_i = 1$$
$$\mathbf{w} \geq 0$$

In [4]:
```
f, axs = plt.subplots(1,2,figsize=(25,10))
plot(axs[0], data=min_var(random_data, lamb=0), title="0")
plot(axs[1], data=min_var(random_data, lamb=10), title="10")
plt.show()
```



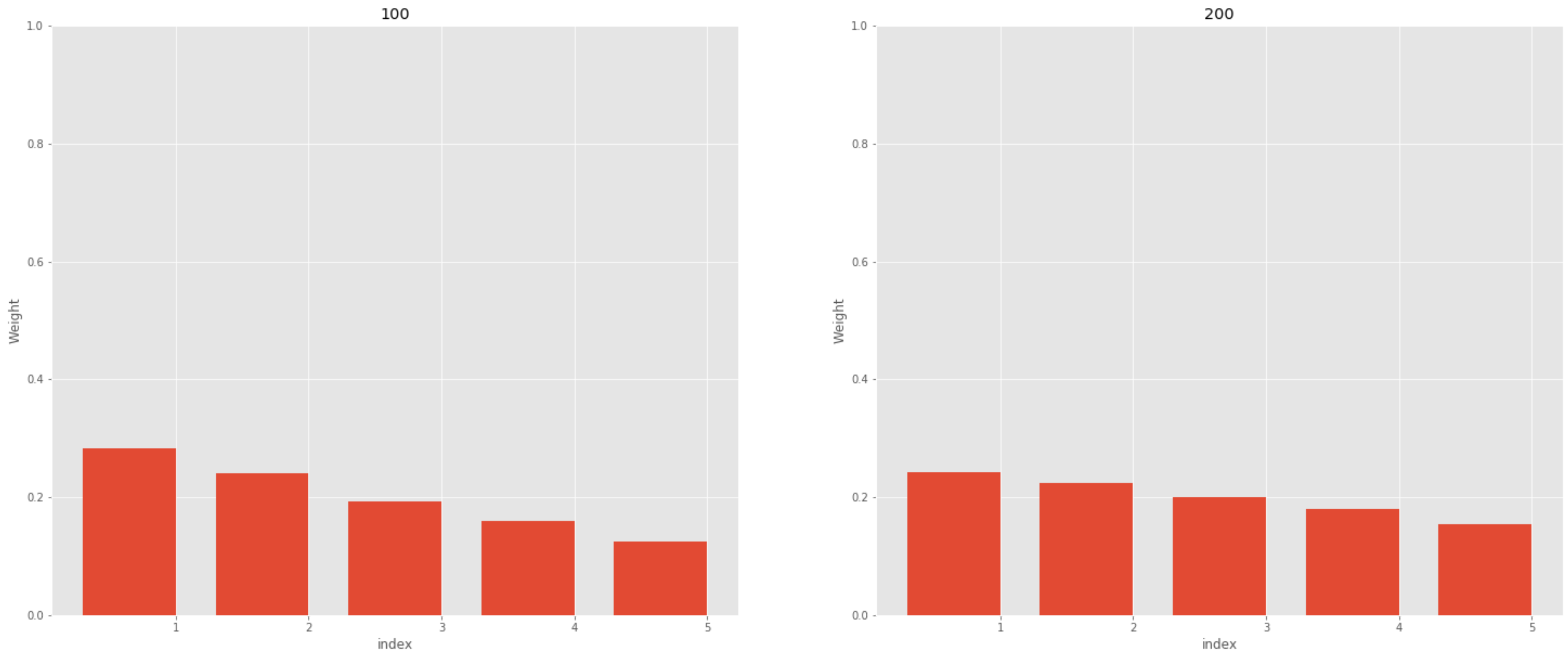
In [5]:

```
f, axs = plt.subplots(1,2,figsize=(25,10))
plot(axs[0], data=min_var(random_data, lamb=20), title="20")
plot(axs[1], data=min_var(random_data, lamb=50), title="50")
plt.show()
```



In [6]:

```
f, axs = plt.subplots(1,2,figsize=(25,10))
plot(axs[0], data=min_var(random_data, lamb=100), title="100")
plot(axs[1], data=min_var(random_data, lamb=200), title="200")
plt.show()
```



Summary