Week 1

Wednesday 28.03.18 - v. 0.1

- **Pre-Training**: `tangency-portfolio`
- **Market Simulation**: baseline models
- **Environment**: observation & OpenAI new API

Risk Averse Portfolio with Transaction Costs

Determine portfolio vector $\mathbf{w}$, such that:

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{maximize}} & \mathbf{w}^T\mathbf{r} - \alpha\mathbf{w}^T\mathbf{\Sigma}\mathbf{w} - \mathbf{1}^T(\mathbf{w_0} - \mathbf{w}) \\ \text{subject to} & \mathbf{w}^T\mathbf{1} = 1 \end{array}$$

where:

- M: number of assets in portfolio
- $\alpha \geq 0$: risk-aversion coefficient
- $\mathbf{\Sigma} \in \mathbb{R}^{M \times M}$: portfolio returns covariance
- $\mathbf{r} \in \mathbb{R}^{M}$: portfolio returns mean
- $\mathbf{w_0}$: initial portfolio weights

Empirical Estimators

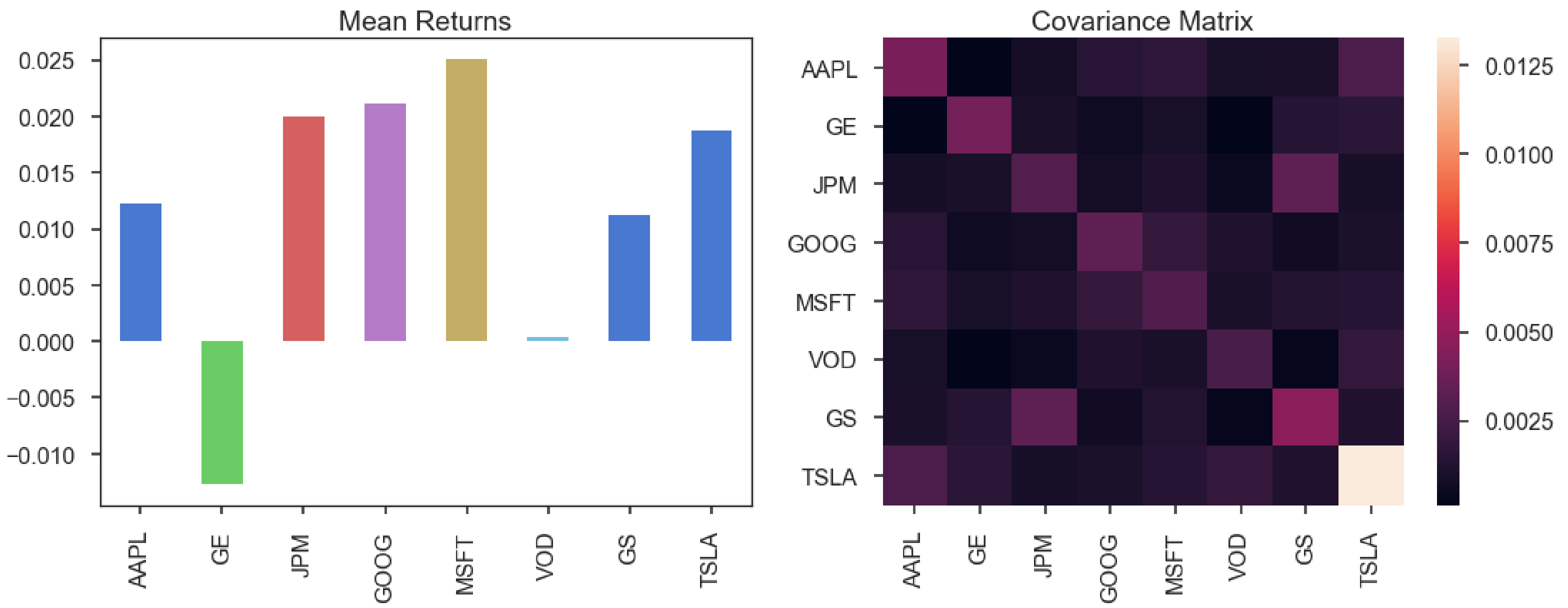
In [3]:

```
# mean returns
mu_r = returns.mean()
# returns covariance
Sigma_r = returns.cov()

fig, axes = plt.subplots(ncols=2, figsize=(18.0, 6.0))

mu_r.plot.bar(ax=axes[0])
axes[0].set_title('Mean Returns')

sns.heatmap(Sigma_r, ax=axes[1])
axes[1].set_title('Covariance Matrix');
```



Experiments

In [6]:
```
np.sqrt(sigmas).shape
```

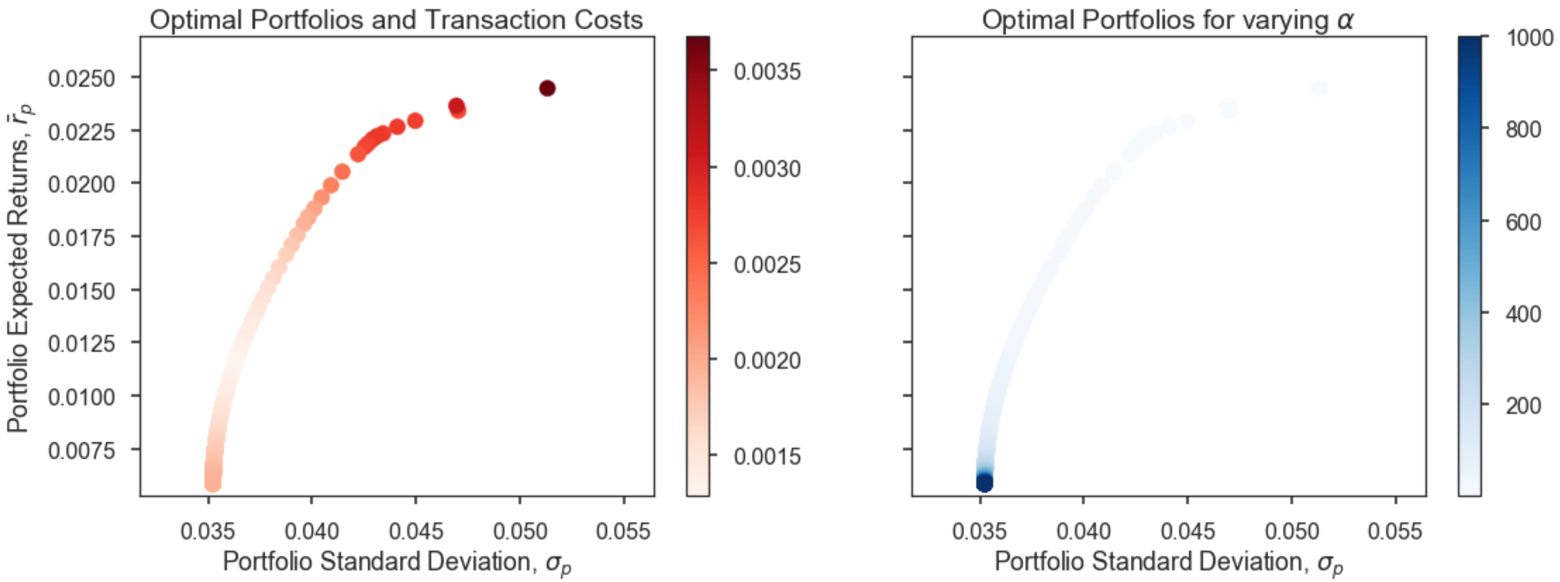
Out[6]:
```
(1000,)
```

In [7]:

```
fig, axes = plt.subplots(ncols=2, sharey=True, figsize=(18.0, 6.0))
sc0 = axes[0].scatter(np.sqrt(sigmas), mus, c=trans, cmap=plt.cm.Reds)

axes[0].set_title('Optimal Portfolios and Transaction Costs')
axes[0].set_xlabel('Portfolio Standard Deviation, $\\sigma_{p}$')
axes[0].set_ylabel('Portfolio Expected Returns, $\\bar{r}_{p}$')
axes[0].set_ylim([mus.min() * 0.9, mus.max() * 1.1])
axes[0].set_xlim([np.sqrt(sigmas).min() * 0.9, np.sqrt(sigmas).max() * 1.1])
fig.colorbar(sc0, ax=axes[0])

sc1 = axes[1].scatter(np.sqrt(sigmas), mus, c=alphas, cmap=plt.cm.Blues)
axes[1].set_title('Optimal Portfolios for varying $\\alpha$')
axes[1].set_xlabel('Portfolio Standard Deviation, $\\sigma_{p}$')
axes[1].set_ylim([mus.min() * 0.9, mus.max() * 1.1])
axes[1].set_xlim([np.sqrt(sigmas).min() * 0.9, np.sqrt(sigmas).max() * 1.1])
fig.colorbar(sc1, ax=axes[1]);
```



Market Simulation: Baseline Models

Surrogates: Amplitude Adjusted Fourier Transform (AAFT)

Preserve first and second order statistical moments by:

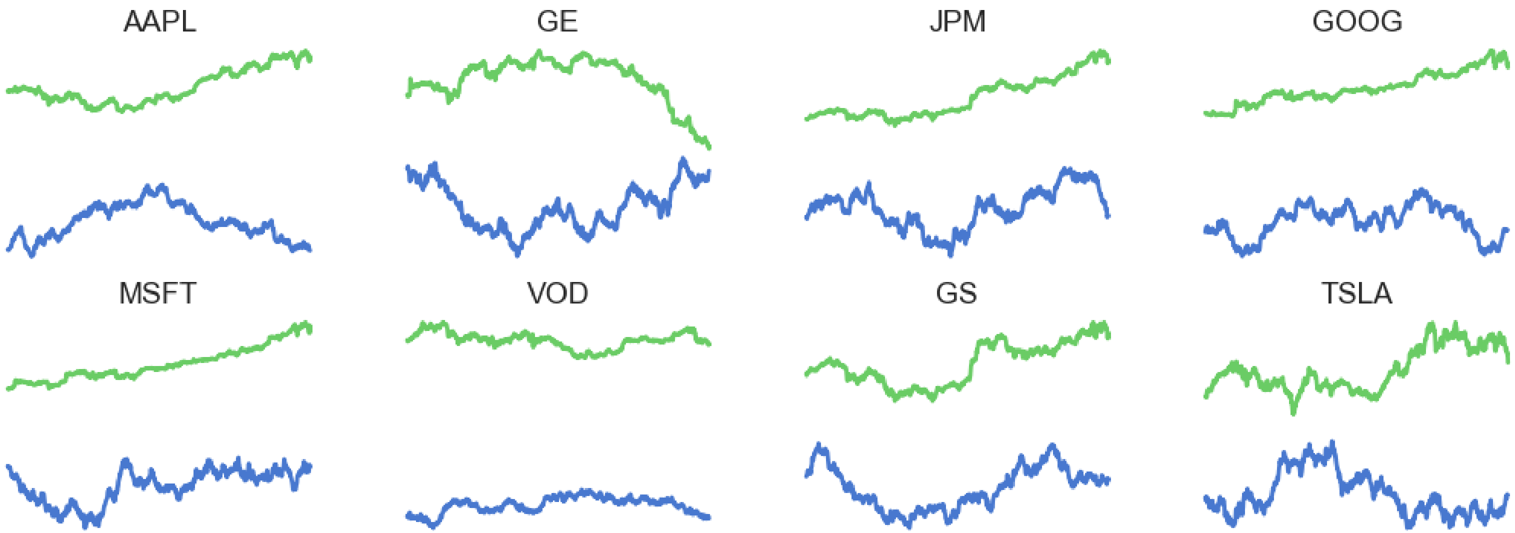
1. Fourier Transformation of multivariate time-series
2. Randomisation of Phase
3. Inverse Fourier Transaformation

The identity of the autocorrelation functions is based on the fact that the original time series and the surrogate have per construction the same power spectrum, which in turn is linked to the autocorrelation function via the Wiener–Khinchin theorem (https://en.wikipedia.org/wiki/Wiener%E2%80%93Khinchin_theorem).

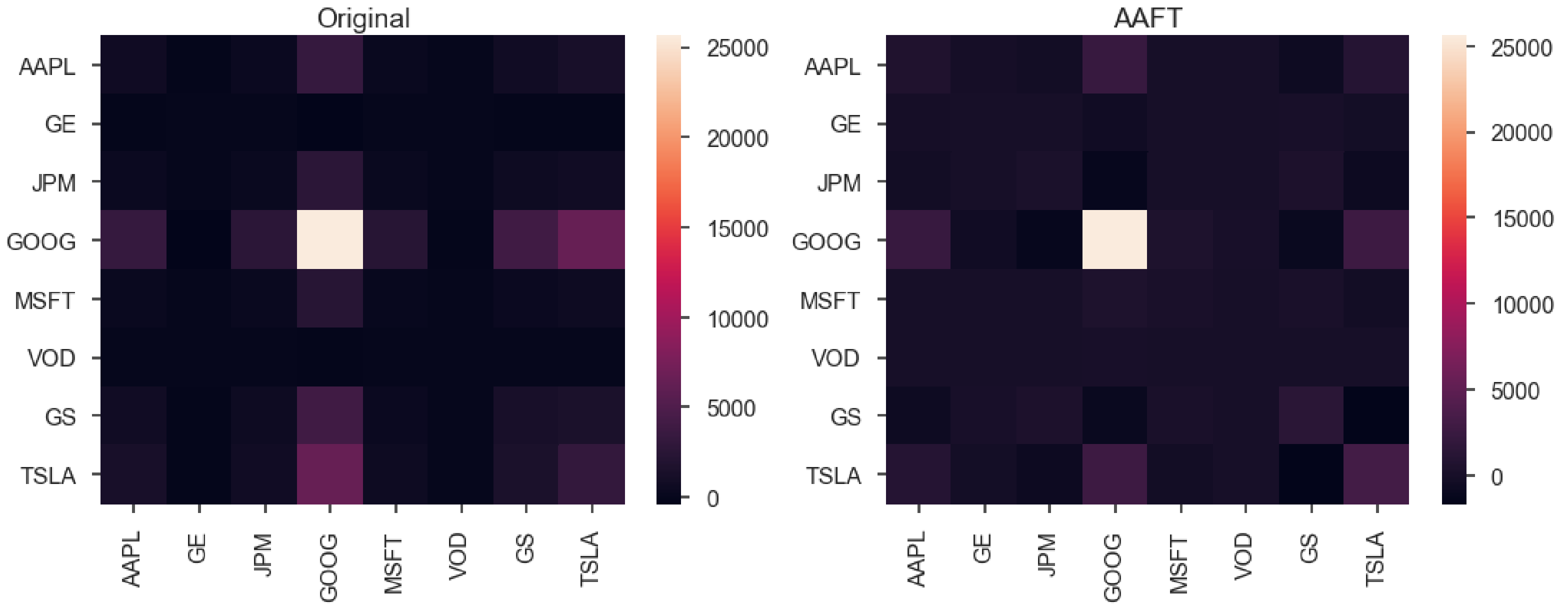
Experiments

In [10]:

```
# time-series plots
fig, axes = plt.subplots(nrows=2, ncols=int(prices.shape[1] / 2), figsize=(18.0,
6.0))
for j, (ax, ts) in enumerate(zip(axes.flatten(), prices_gen.T)):
    ax.plot(ts, label="AAFT")
    ax.plot(prices.values[:, j], label="Original")
    ax.set_title(prices.columns[j])
    ax.set_xticks([])
    ax.set_yticks([])
    ax.axis('off');
```

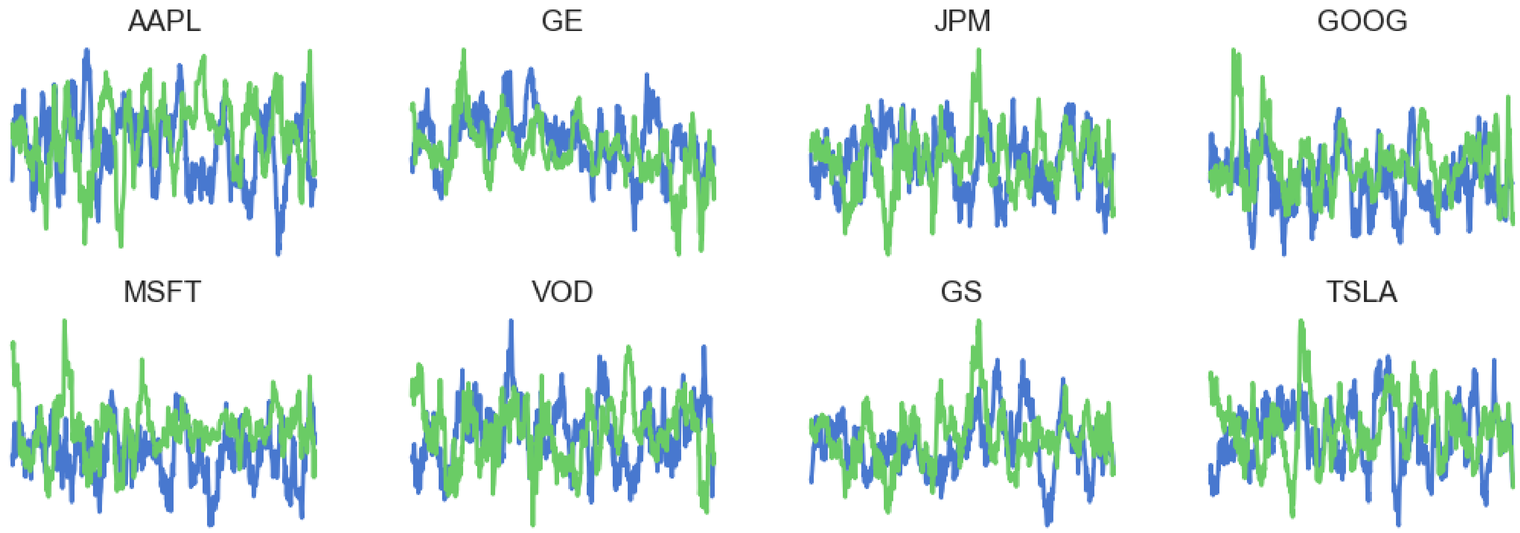


In [11]:
```
# statistical tests: covariances
fig, axes = plt.subplots(ncols=2, figsize=(18.0, 6.0))
sns.heatmap(np.cov(prices.values.T), ax=axes[0])
axes[0].set_title("Original")
axes[0].set_xticklabels(prices.columns, rotation=90)
axes[0].set_yticklabels(prices.columns, rotation=0)
sns.heatmap(np.cov(prices_gen.T), ax=axes[1])
axes[1].set_title("AAFT")
axes[1].set_xticklabels(prices.columns, rotation=90)
axes[1].set_yticklabels(prices.columns, rotation=0);
```

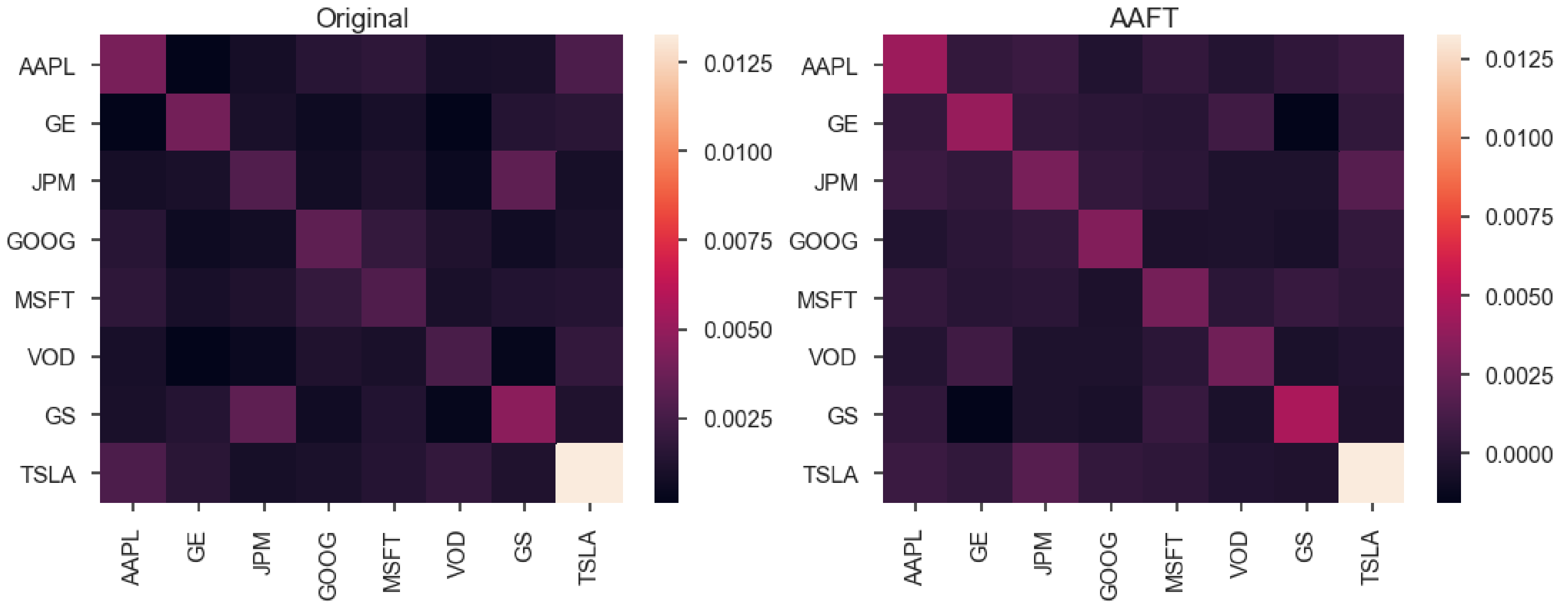


In [12]:

```
# time-series plots
fig, axes = plt.subplots(nrows=2, ncols=int(returns.shape[1] / 2), figsize=(18.0,
6.0))
for j, (ax, ts) in enumerate(zip(axes.flatten(), returns_gen.T)):
    ax.plot(ts, label="VAR")
    ax.plot(returns.values[:, j], label="Original")
    ax.set_title(returns.columns[j])
    ax.set_xticks([])
    ax.set_yticks([])
    ax.axis('off');
```



In [13]:
```
# statistical tests: covariances
fig, axes = plt.subplots(ncols=2, figsize=(18.0, 6.0))
sns.heatmap(np.cov(returns.values.T), ax=axes[0])
axes[0].set_title("Original")
axes[0].set_xticklabels(returns.columns, rotation=90)
axes[0].set_yticklabels(returns.columns, rotation=0)
sns.heatmap(np.cov(returns_gen.T), ax=axes[1])
axes[1].set_title("AAFT")
axes[1].set_xticklabels(returns.columns, rotation=90)
axes[1].set_yticklabels(returns.columns, rotation=0);
```



Vector Autoregressive Process (VAR)

We are interested in modeling a $T \times K$ multivariate time series Y, where T denotes the number of observations and K the number of variables. One way of estimating relationships between the time series and their lagged values is the **Vector Autoregression (VAR) Process**:

$$\begin{aligned} Y_t &= A_1 Y_{t-1} + A_2 + Y_{t-2} + \ldots + A_p + Y_{t-p} + u_t \\ &= \sum_{i=1}^{p} A_i Y_{t-i} + u_t \\ u_t &\sim \text{Normal}(0, \Sigma_u) \end{aligned}$$

where $A_i \in R^{K \times K}$ a coefficient matrix.

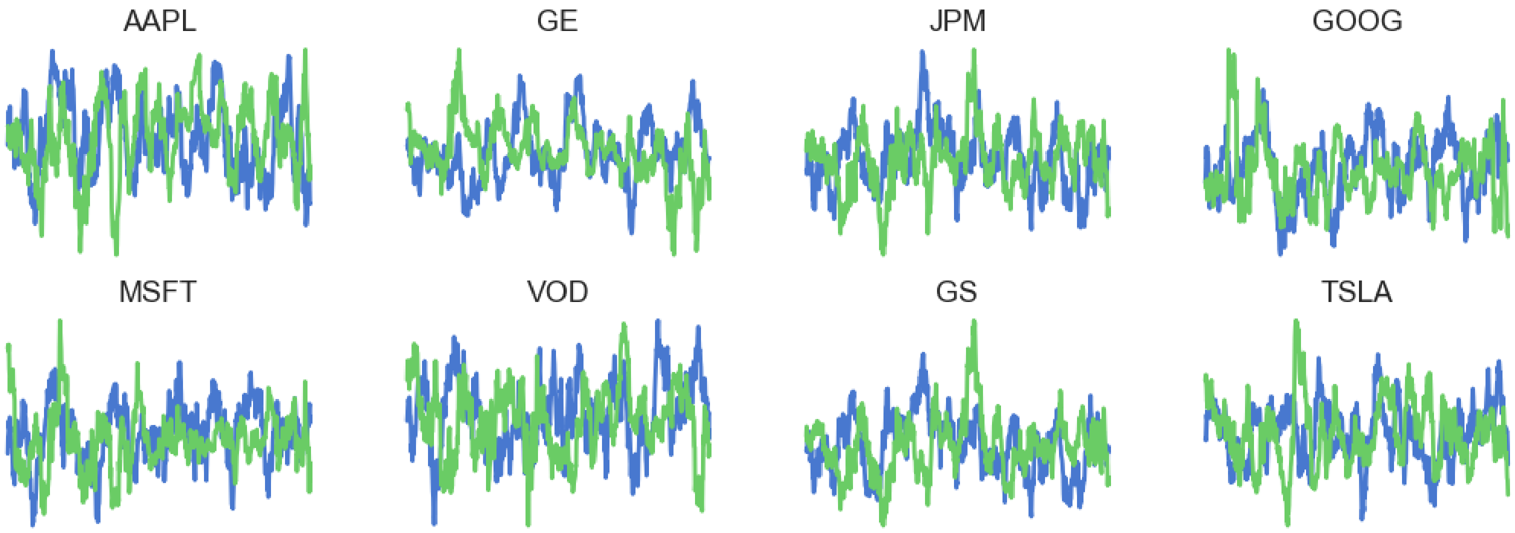
In [14]:

```
# VAR model
var = VAR(returns.values)
# optimal order
order = var.select_order(5)['aic']
# fit model
model = var.fit(order)
# Simulation
returns_gen = var_util.varsim(model.coefs, model.intercept, model.sigma_u, steps=l
en(returns.values))
```

```
                 VAR Order Selection
=====================================================
           aic          bic          fpe          hqic
-----------------------------------------------------
0       -47.50       -47.45    2.350e-21       -47.48
1      -65.14*      -64.69*   5.125e-29*      -64.97*
2       -65.14       -64.28    5.153e-29       -64.80
3       -65.12       -63.86    5.217e-29       -64.64
4       -65.06       -63.40    5.550e-29       -64.42
5       -65.03       -62.96    5.715e-29       -64.23
=====================================================
* Minimum
```

In [15]:

```
# time-series plots
fig, axes = plt.subplots(nrows=2, ncols=int(returns.shape[1] / 2), figsize=(18.0, 6.0))
for j, (ax, ts) in enumerate(zip(axes.flatten(), returns_gen.T)):
    ax.plot(ts, label="VAR")
    ax.plot(returns.values[:, j], label="Original")
    ax.set_title(returns.columns[j])
    ax.set_xticks([])
    ax.set_yticks([])
    ax.axis('off');
```



In [16]:

```
# statistical tests: covariances
fig, axes = plt.subplots(ncols=2, figsize=(18.0, 6.0))
sns.heatmap(np.cov(returns.values.T), ax=axes[0])
axes[0].set_title("Original")
axes[0].set_xticklabels(returns.columns, rotation=90)
axes[0].set_yticklabels(returns.columns, rotation=0)
sns.heatmap(np.cov(returns_gen.T), ax=axes[1])
axes[1].set_title("VAR")
axes[1].set_xticklabels(returns.columns, rotation=90)
axes[1].set_yticklabels(returns.columns, rotation=0);
```