18) Futures Contracts

Vitor Kamada

March 2019

Reference

Tables, Graphics, and Figures from

https://www.quantopian.com/lectures

Lecture 51 Futures Contracts

Futures Contract

Legal agreement to buy or sell a particular commodity or asset at a predetermined price at a specified time in the future

- Oil producer plans to produce 1M barrels of oil over the next year. It will be ready for delivery in 12 months.
- Current price is $75 per barrel.
- One-year oil futures contracts are priced at $78 per barrel.
- In one year, the producer is obligated to deliver 1M barrels of oil and is guaranteed to receive $78 million.
- The $78 price per barrel is received regardless of where spot market prices are at the time.

https://www.quantopian.com/help#available-futures

Name	Root Symbol	Exchange
Corn	CN	CBOT
S&P 500 E-Mini	ES	CME
Japanese Yen	JY	CME
Gold	GC	COMEX
Light Sweet Crude Oil	CL	NYMEX
Natural Gas	NG	NYMEX
Silver mini-sized	YS	ICE
Russell 2000 Mini	ER	ICE

Delivery Months and Codes

Code	Delivery Month
F	January
G	February
H	March
J	April
K	May
M	June
N	July
Q	August
U	September
V	October
X	November
Z	December

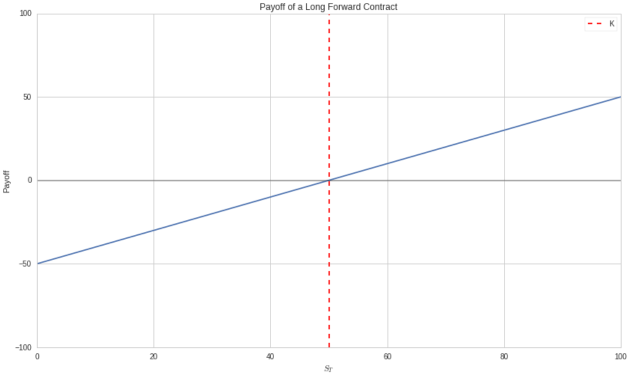
Long Position: $S_t - K$

Short Position: $K - S_t$

```
K = 50
# Here we look at various different
# values that S_T can have
S_T = np.linspace(0, 100, 200)
# Calculate the long and short payoffs
long_payoff = S_T - K
short_payoff = K - S_T
```

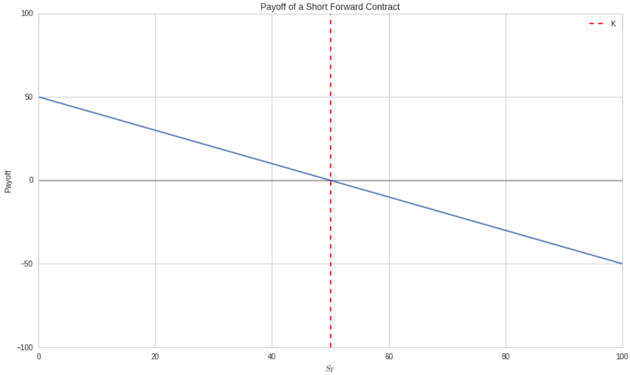
Payoff of a Long Forward Contract

```
plt.plot(S_T, long_payoff)
plt.axhline(0, color='black', alpha=0.3)
plt.axvline(0, color='black', alpha=0.3)
plt.xlim(0, 100)
plt.ylim(-100, 100)
plt.axvline(K, linestyle='dashed', color='r', label='K')
plt.ylabel('Payoff')
plt.xlabel('$S_T$')
plt.title('Payoff of a Long Forward Contract')
plt.legend();
```



Payoff of a Short Forward Contract

```
plt.plot(S_T, short_payoff);
plt.axhline(0, color='black', alpha=0.3)
plt.axvline(0, color='black', alpha=0.3)
plt.xlim(0, 100)
plt.ylim(-100, 100)
plt.axvline(K, linestyle='dashed', color='r', label='K')
plt.ylabel('Payoff')
plt.xlabel('$S_T$')
plt.title('Payoff of a Short Forward Contract')
plt.legend();
```



Corn Futures

```
contract = symbols('CNH17')
futures_position_value = get_pricing(contract, start_date = '2017-01-19'
, end_date = '2017-02-15', fields = 'price')
futures_position_value.name = futures_position_value.name.symbol
futures_position_value.plot()
plt.title('Corn Futures Price')
plt.xlabel('Date')
plt.ylabel('Price');
```

Maintenance Margin

The profit or loss of the position fluctuates in the account as the price of the futures contract moves. If the loss gets too big, the broker will ask the trader to deposit more money to cover the loss

- January-April contracts are trading at \$4 (5,000 bushels of corn).
- Trader is not required to pay \$4 (\$4 x 5,000 bushels) for this privilege.
- Broken only requires an initial margin payment (\$990), and maintenance margin (\$900).

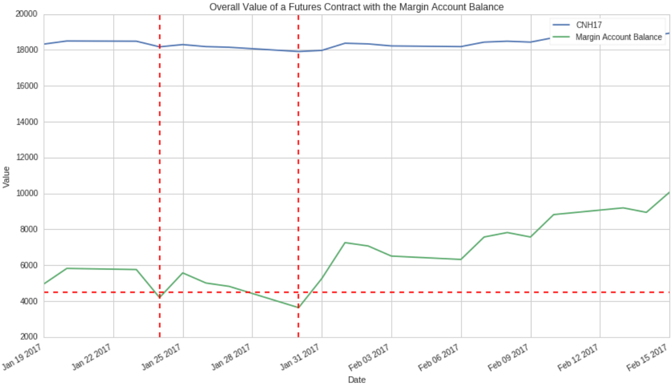
Margin Call

```python
initial_margin = 990
maintenance_margin = 900
contract_count = 5
```

```python
margin_account_changes = \
  futures_position_value.diff()*contract.multiplier*contract_count
margin_account_changes[0] = initial_margin*contract_count
margin_account_balance = margin_account_changes.cumsum()
margin_account_balance.name = 'Margin Account Balance'
# First margin call
margin_call_idx = np.where(margin_account_balance < \
                           maintenance_margin*contract_count)[0][0]
margin_deposit = initial_margin*contract_count - \
              margin_account_balance[margin_call_idx]
margin_account_balance[margin_call_idx+1:] = \
  margin_account_balance[margin_call_idx+1:] + margin_deposit
# Second margin call
second_margin_call_idx = np.where(margin_account_balance < \
                                  maintenance_margin*contract_count)[0][1]
second_margin_deposit = initial_margin*contract_count - \
  margin_account_balance[second_margin_call_idx]
margin_account_balance[second_margin_call_idx+1:] = \
  margin_account_balance[second_margin_call_idx+1:] + second_margin_deposit
```

Futures Contract with the Margin Account Balance

```
(futures_position_value*contract.multiplier).plot()
margin_account_balance.plot()
plt.axvline(margin_account_balance.index[margin_call_idx], color='r', linestyle='--')
plt.axvline(margin_account_balance.index[second_margin_call_idx], color='r', linestyle='--')
plt.axhline(maintenance_margin*contract_count, color='r', linestyle='--')
plt.title('Overall Value of a Futures Contract with the Margin Account Balance')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend();
```



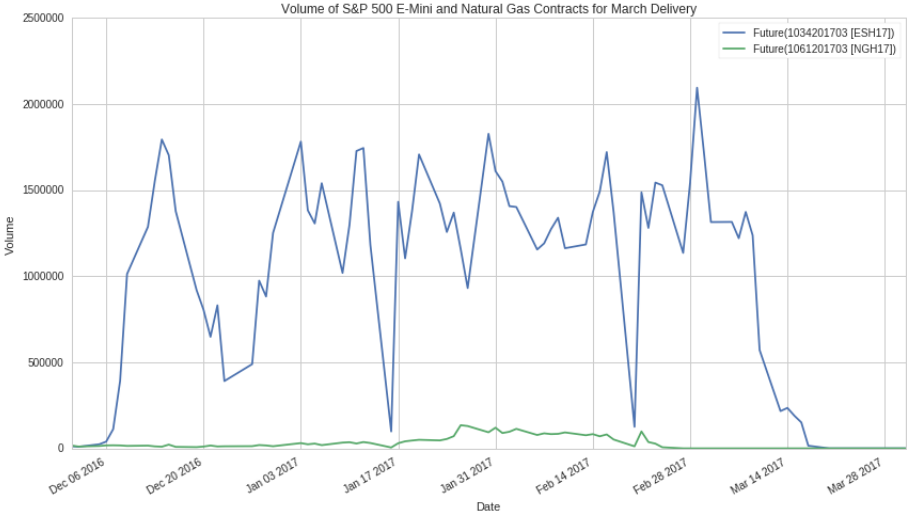
Financial vs Commodity Futures

```
contracts = symbols(['ESH17', 'NGH17'])
volume_comparison = get_pricing(contracts,
     start_date = '2016-12-01',
     end_date = '2017-04-01', fields = 'volume')
volume_comparison.plot()
plt.title('Volume of S&P 500 E-Mini and \
   Natural Gas Contracts for March Delivery')
plt.xlabel('Date')
plt.ylabel('Volume');
```

```
print volume_comparison.max()
```

```
Future(1034201703 [ESH17])    2095150.0
Future(1061201703 [NGH17])     134561.0
dtype: float64
```

Volume of S&P 500 E-Mini and Natural Gas Contracts for March Delivery



Closing a Futures Position

```
cls = symbols(['CLF16', 'CLG16', 'CLH16'])
contract_volume = get_pricing(cls, start_date='2015-10-01',
            end_date='2016-04-01', fields='volume')
contract_volume.plot()
plt.title('Volume of Contracts with Different Expiry')
plt.xlabel('Date')
plt.ylabel('Volume');
```

expiration_date: when the contract will stop trading

```
cl_january_contract = symbols('CLF16')
print cl_january_contract.expiration_date
```

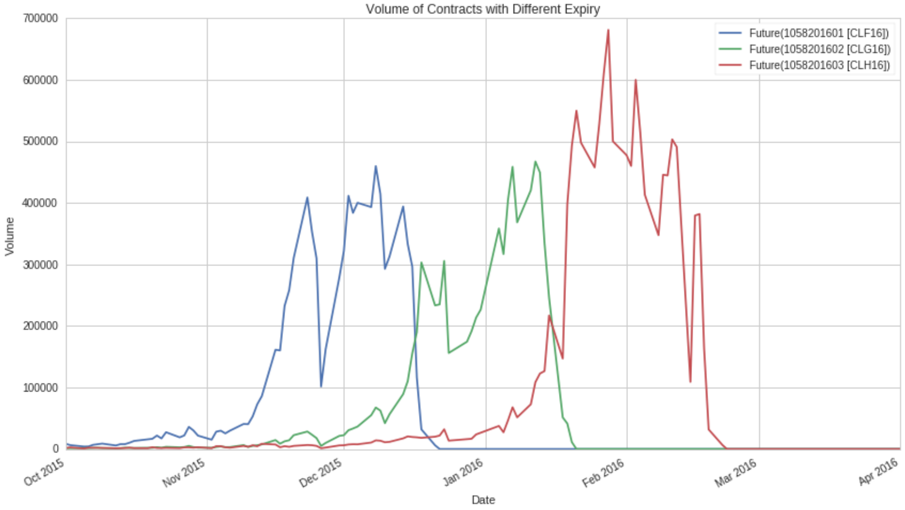
```
2015-12-21 00:00:00+00:00
```

```
es_march_contract = symbols('ESH17')
print es_march_contract.expiration_date
```

```
2017-03-17 00:00:00+00:00
```

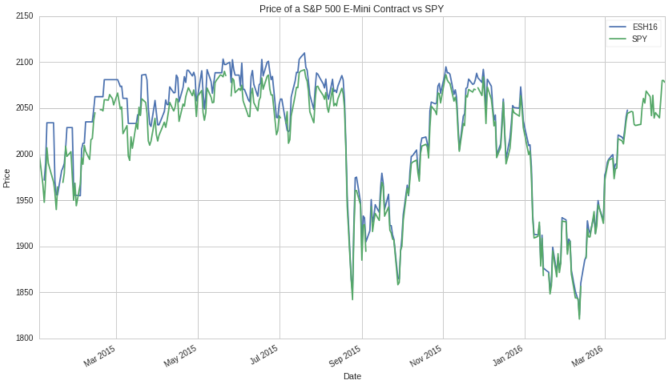
Volume of Contracts with Different Expiry

"Light Sweet Crude Oil"



Spot Prices and Futures Prices

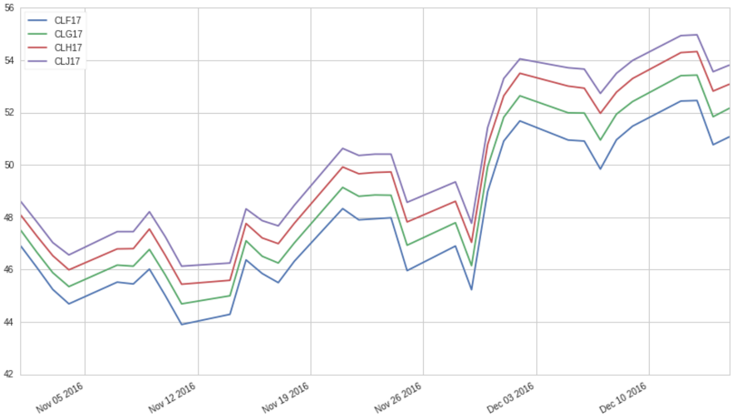
```
assets = ['SPY', 'ESH16']
prices = get_pricing(assets, start_date = '2015-01-01', end_date = '2016-04-15'
, fields = 'price')
prices.columns = map(lambda x: x.symbol, prices.columns)
prices['ESH16'].plot()
(10*prices['SPY']).plot()
plt.legend()
plt.title('Price of a S&P 500 E-Mini Contract vs SPY')
plt.xlabel('Date')
plt.ylabel('Price');
```



$F(t, T) = S(t)(1 + c)^{T-t} = S(t)e^{c(T-t)}$

```
contracts = symbols(['CLF17', 'CLG17', 'CLH17', 'CLJ17'])
prices = get_pricing(contracts, start_date='2016-11-01', end_date='2016-12-15'
, fields='price')
prices.columns = map(lambda x: x.symbol, prices.columns)
```

```
prices.plot();
```

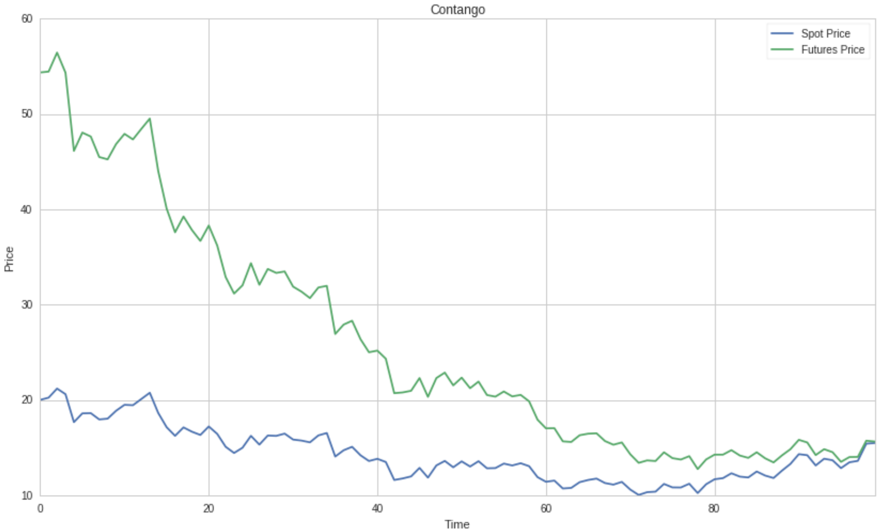


Toy Example 1

```python
N = 100 # Days to expiry of futures contract
cost_of_carry = 0.01
spot_price = pd.Series(np.ones(N), name = "Spot Price")
futures_price = pd.Series(np.ones(N), name = "Futures Price")
spot_price[0] = 20
futures_price[0] = spot_price[0]*np.exp(cost_of_carry*N)
for n in range(1, N):
    spot_price[n] = spot_price[n-1]*(1 + np.random.normal(0, 0.05))
    futures_price[n] = spot_price[n]*np.exp(cost_of_carry*(N - n))

spot_price.plot()
futures_price.plot()
plt.legend()
plt.title('Contango')
plt.xlabel('Time')
plt.ylabel('Price');
```

Contango


Contango
60
50
40
30
20
10
Price
0
20
40
60
80
Time
Spot Price
Futures Price

Toy Example 2

```
N = 100 # Days to expiry of futures contract
cost_of_carry = -0.01
spot_price = pd.Series(np.ones(N), name = "Spot Price")
futures_price = pd.Series(np.ones(N), name = "Futures Price")
spot_price[0] = 20
futures_price[0] = spot_price[0]*np.exp(cost_of_carry*N)
for n in range(1, N):
    spot_price[n] = spot_price[n-1]*(1 + np.random.normal(0, 0.05))
    futures_price[n] = spot_price[n]*np.exp(cost_of_carry*(N - n))

spot_price.plot()
futures_price.plot()
plt.legend()
plt.title('Backwardation')
plt.xlabel('Time')
plt.ylabel('Price');
```

Backwardation