Python in Finance

ΤΜΗΜΑ ΛΟΓΙΣΤΙΚΗΣ ΚΑΙ ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΗΣ

Αξία του χρήματος

Η έννοια της αξίας του χρήματος αποτελέι ένα από τα πιο σημαντικά κομμάτια στην ανάλυση τον επενδύσεων

Καθότι η αξία του χρήματος μεταβάλλεται με την πάροδο του χρόνου,η λήψη απόφασης σχετικά με μια επένδυση σήμερα εξαρτάται από την μελλοντική αξία του χρήματος

Γενικά η αξία μειώνεται με την πάροδο του χρόνου. Για πάραδειγμα η αξία 1€ σήμερα είναι μεγαλύτερη από αυτή σε κάποια μελλοντική χρονική στιγμή.

Υπάρχουν 2 κύριοι λόγοι σχετικά με αυτην την αλλαγή στην αξία

Αξία του χρήματος

- Οι ταμειακές ροές σε διαφορες χρονικές περιόδους έχουν διαφορετική αξία:

Όπως αναφέραμε και προηγουμένως η αξία 1€ δεν είναι το ίδιο πολύτιμή με αυτή σε έναν χρόνο. Επίσης θα μπορούσε κάποιος να επενδύσει ένα συγκεκριμένο ποσό σήμερα, και μέσω του τόκου το επόμενο έτος η αξία της επένδυσης του να έχει αυξηθεί. Με λίγα λόγια πρέπει να λαμβάνουμε υπόψιν την αξία του χρήματος για να αξιολογήσουμε τις ταμειακές ροές στην πάροδο του χρόνου

Αξία του χρήματος

- Οι ταμειακές ροές είναι αβέβαιες

Ορισμένες ταμειακές ροές ενδέχεται να μην πραγματοποιηθούν στο μέλλον. Η αβεβαιότητα πηγάζει σχετικά με την πρόβλεψη σχετικά με το χρονοδιάγραμμα και το ποσο των χρηματικών ροών. Με λίγα λογια δεν γνωρίζουμε με βεβαιότητα, τον χρόνο η το ποσό της ταμειακής ροής που θα αποπληρωθεί στο μέλλον. Αυτή η αβεβαιότητα σχετικά με την ταμειακή ροή, πρέπει με κάποιον τρόπο να ληφθεί υπόψιν σχετικά με την αξιολόγηση μια επένδυσης (π.χ Δάνειο)

Αξία του χρήματος

- Προεξόφληση(Discounting)

Με τον όρο προεξόφληση εννοούμε την διαδικασία οπου χρησιμοποιόντας μελλοντικές αξίες(Future Value) υπολογίζουμε την καθαρή παρούσα αξία μιας επένδυσης(Present Value)

- Ανατοκισμός(Compounding)

Αντίστοιχα με την έννοια του ανατοκισμού μετατρέπουμε την παρούσα αξία μιας επέδυσης(Present Value) σε μελλοντική(Future Value)

Αξία του χρήματος

Έστω δανείζουμε ένα ποσό π.χ. 100€ με την συμφωνία αποπληρωμής του ποσού σε 1 μήνα. Αν μας επιστρεφόταν το ποσό των 100€ θα ήταν δίκαιο ; Πιθανόν όχι.

Αρχικά , το γεγονός πως παρέχουμε ένα συγκεκριμένο ποσό σε κάποιον δανειολήπτη αποτελεί το κόστος ευκαιρίας του χρήματος(τόκος).Επίσης ποια είναι η πιθανότητα αθέτησης της αποπληρωμής μετά από 1 μήνα ;

Εκτός από το κόστος ευκαιρίας , πρέπει να λάβουμε υποψιν και την αβεβαιότητα της μη επιστροφής του κεφαλαίου μας την χρονική περίοδο που έχει συμφωνηθεί

Αξία του χρήματος

- Ετσι λοιπόν σε περίπτωση δανείου 100€ έχουμε :

1. Το αρχικο κεφάλαιο 100€ (Principal)
2. Ένα είδος «αποζημίωσης» για το κόστος ευκαιρίας, και την αβεβαιότητα της αποπληρωμής του δανείου στην συμφωνηθέντα χρονική περίοδο(interest rate)

Το ποσό που έχουμε σκοπό να δανείσουμε σήμερα αποτελεί την παρούσα αξία του δανείου και αντίστοιχα το πόσο που αναμένουμε να εισπράξουμε αποτελεί την μέλλουσα αξία

Αξία του χρήματος

Παράδειγμα:

Εστω κάνουμε μια κατάθεση 1000€ στην τράπεζα ‘ΑΒΓ’ με 5% επιτόκιο ανα έτος.

Στο τέλος του έτους το κεφάλαιο μας θα ισούται με 1050€.Ουσιαστικά αποτελείται από το αρχικό μας κεφάλαιο (1000€) συν τον τόκο των 50€.

- Παρούσα αξία της επένδυσης μας 1000€ (PV)
- Μέλλουσα αξία της επένδυσης 1050€ (FV)
- Τόκος 5% επι του αρχικού μας κεφαλαίου

Αξία του χρήματος :Μέλλουσα Αξία

Αν πραγματοποιήσουμε μια κατάθεση σήμερα με ετήσιο επιτόκιο 10%

Ποια είναι η αξία του χρήματος σε 1 ετος από σήμερα ;

PV=100

R=0.10

n=1

FV=PV*(1+R)**n

FV

$$FV = PV(1+R)^n$$

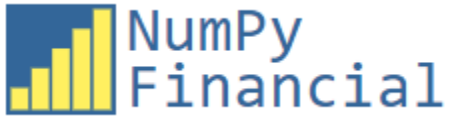
Αξία του χρήματος : Παρούσα Αξία

Αντιστοιχά αν θέλαμε να υπολογίσουμε την παρούσα αξία του χρήματος θα εργαζόμασταν ως εξής :

```
FV=100
R=0.10
n=2
PV=FV/(1+R)**n
PV
```

$$PV = \frac{FV}{(1+R)^n}$$

NumPy-Financial



Το πακέτο NumPy-Financial αποτελεί ένα οικονομικό πακέτο της Python το οποίο μας παρέχει κάποιες βασικές οικονομικές συναρτήσεις. Γενικά αυτές οι συναρτήσεις ήταν διαθέσιμες από την NumPy , όμως από το 2013 δημιουργήθηκε το ξεχωριστό πακέτο NumPy-Financial.

Ένας από τους κύριους λόγους είναι πως αυτές οι συναρτήσεις θεωρήθηκαν ειδικά εξιδεικευμένες για την NumPy

NEP 32 — Remove the financial functions from NumPy — NumPy Enhancement Proposals

Στις επόμενες διαφάνειες θα δούμε εφαρμογές αυτών των συναρτήσεων, για περισσότερες πληροφορίες μπορείτε να ανατρέξετε και στο numpy-financial 1.0.0 — numpy-financial documentation

NumPy-Financial

- Πριν ξεκινήσουμε με τις συναρτήσεις ας ξεκαθαρίσουμε κάτι. Γενικά αν ανατρέξετε στο official site της NumPy-Financial στις συναρτήσεις fv,pv,pmt,rate θα δείτε

```
fv +
pv*(1 + rate)**nper +
pmt*(1 + rate*when)/rate*((1 + rate)**nper - 1) == 0
```

or, when `rate == 0`:

```
fv + pv + pmt * nper == 0
```

Καθώς η παρούσα αξία και οι πληρωμές θεωρούνται κινήσεις κεφαλαίου προς τα έξω, θα τα εισάγουμε με αρνητικό πρόσημο.

NumPy-Financial

pv

`numpy_financial.pv(rate, nper, pmt, fv=0, when='end')`

Compute the present value.

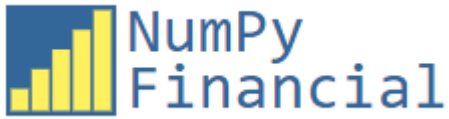
Given:

- a future value, *fv*
- an interest *rate* compounded once per period, of which there are
- *nper* total
- a (fixed) payment, *pmt*, paid either
- at the beginning (*when* = {'begin', 1}) or the end (*when* = {'end', 0}) of each period

Return:

the value now

NumPy-Financial



Με την συνάρτηση pv υπολογίζουμε την παρούσα αξία μιας επένδυσης.

Ποια είναι η παρουσα αξία μιας επένδυσης οπου χρειάζεται $15692.93 μετα από 10 έτη αν αποταμιεύουμε μηνιαία $100 ;Υποθέστε ότι το ετήσιο επιτόκιο ισούται με 5% και ο ανατοκισμός γίνεται μηνιαία.

```
import numpy_financial as npf
npf.pv(0.05/12, 10*12, -100, 15692.93)
```

NumPy-Financial

Η πρώτη συνάρτηση που θα δούμε είναι η fv (future value)

fv

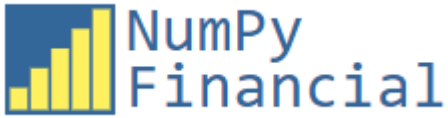
`numpy_financial.fv(rate, nper, pmt, pv, when='end')`

Compute the future value.

Given:

- a present value, *pv*
- an interest *rate* compounded once per period, of which there are
- *nper* total
- a (fixed) payment, *pmt*, paid either
- at the beginning (*when* = {'begin', 1}) or the end (*when* = {'end', 0}) of each period

NumPy-Financial



Αντίστοιχα με το προηγούμενο μας παράδειγμα θα υπολογίσουμε την μέλλουσα αξία της επένδυσης μας.

Οπότε ξεκινάμε με παρούσα αξία στα $100 και αποταμίευση στα $100/μήνα με ετησιο επιτοκιο 5% και μηνιαίο ανατοκισμό.

```
import numpy_financial as npf
npf.fv(0.05/12, 12*10,-100,-100)
```

NumPy-Financial

rate

`numpy_financial.rate(nper, pmt, pv, fv, when='end', guess=None, tol=None, maxiter=100)`

Compute the rate of interest per period.

Parameters:

nper : *array_like*
Number of compounding periods

pmt : *array_like*
Payment

pv : *array_like*
Present value

fv : *array_like*
Future value

when : *{{'begin', 1}, {'end', 0}}, {string, int}, optional*
When payments are due ('begin' (1) or 'end' (0))

guess : *Number, optional*
Starting guess for solving the rate of interest, default 0.1

tol : *Number, optional*
Required tolerance for the solution, default 1e-6

maxiter : *int, optional*
Maximum iterations in finding the solution

NumPy-Financial

Τέλος με την χρήση της Rate θα μπορούσαμε να υπολογίσουμε το επιτόκιο.

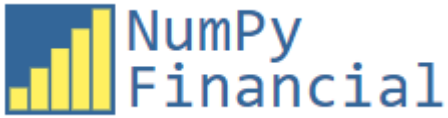
Έχουμε λοιπόν :

Αρχική επένδυση στα $100 με μηνιαίες καταθέσεις επίσης στα $100.Αν θέλουμε να έχουμε μέλλουσα αξία στα $ 15692.93 σε πάροδο 10 ετών με μηνιαίο ανατοκισμό, πόσο ισούται το επιτόκιο ;

```
Import numpy_financial as npf
npf.rate(12*10,-100,-100,15692.9288)
```

Βλέπουμε ότι το αποτέλεσμα πράγματι ισούται με 0.05/12≈0,0041

NumPy-Financial



irr

`numpy_financial.irr(values)`

Return the Internal Rate of Return (IRR).

This is the "average" periodically compounded rate of return that gives a net present value of 0.0; for a more complete explanation, see Notes below.

`decimal.Decimal` type is not supported.

Parameters:

values : *array_like, shape(N,)*

Input cash flows per time period. By convention, net "deposits" are negative and net "withdrawals" are positive. Thus, for example, at least the first element of `values`, which represents the initial investment, will typically be negative.

Returns:

out : *float*

Internal Rate of Return for periodic input values.

NumPy-Financial

Η συνάρτηση irr υπολογίζει τον εσωτερικό βαθμό απόδοσης μιας επένδυσης.

Έστω επένδυση $100 με μελλοντικές ταμειακές ροές(σε σταθερή περίοδο) $39,$59,$55,$20.

Υποθέτοντας ότι η τελική αξία ισούται με το 0 βλέπουμε πως η αρχική επένδυση των $100 μας απέδοσε συνολικά $173 λόγω του ανατοκισμού και των περιοδικών ταμειακών ροών .Το «μέσο» επιτόκιο της επένδυσης ισούται με $\frac{0,73}{4}$

$$-100+\frac{39}{1+r}+\frac{59}{(1+r)^2}+\frac{55}{(1+r)^3}+\frac{20}{(1+r)^4}=0$$

NumPy-Financial

Ουσιαστικά ο εσωτερικός βαθμός απόδοσης (IRR) μετράει την απόδοση μιας μακροχρόνιας απόδοσης εξισώνοντας την παρούσα αξία των μελλοντικών ταμειακών ροων(πλέον τελικής αξίας) με την αγοραία αξία της επένδυσης

- ```
  import numpy_financial as npf
  ```
- ```
  round(npf.irr([-100, 39, 59, 55, 20]), 5)
  ```

Ο εσωτερικός βαθμός απόδοσης ισούται με 28%

Σε επόμενες διαφάνειες θα επανέλθουμε στον εσωτερικό βαθμό απόδοσης μέσω του IRR rule , και θα παρουσιάσουμε μια πιο αλγοριθμική προσέγγιση αυτου του δείκτη.

NumPy-Financial

npv

`numpy_financial.npv(rate, values)`

Returns the NPV (Net Present Value) of a cash flow series.

Parameters:

rate : *scalar*

The discount rate.

values : *array_like, shape(M,)*

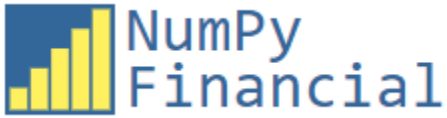
The values of the time series of cash flows. The (fixed) time interval between cash flow "events" must be the same as that for which `rate` is given (i.e., if `rate` is per year, then precisely a year is understood to elapse between each cash flow event). By convention, investments or "deposits" are negative, income or "withdrawals" are positive; `values` must begin with the initial investment, thus `values[0]` will typically be negative.

Returns:

out : *float*

The NPV of the input cash flow series `values` at the discount `rate`.

NumPy-Financial



Θεωρείστε μια πιθάνή επένδυση των $40.000 με χρηματοροές $5.000 $8.000 $12.000 $30.000 στο τέλος κάθε έτους με επιτόκιο 8% ανά έτος. Ποια είναι η καθαρή αξία της επένδυσης ;

- import numpy as np
- import numpy_financial as npf
- rate, cashflows = 0.08, [-40.000, 5.000, 8.000, 12.000, 30.000]
- npf.npv(rate, cashflows).round(5)

NumPy-Financial

Εναλλακτικά θα μπορούσαμε να ορίσουμε ξεχωριστά την αρχική επένδυση , με τις μελλοντικές χρηματοροές. Όπως για παράδειγμα:

```
initial_cashflow = cashflows[0]
cashflows[0] = 0
np.round(npf.npv(rate, cashflows) + initial_cashflow, 5)
```

NumPy-Financial

pmt

`numpy_financial.pmt(rate, nper, pv, fv=0, when='end')`

Compute the payment against loan principal plus interest.

Given:

- a present value, *pv* (e.g., an amount borrowed)
- a future value, *fv* (e.g., 0)
- an interest *rate* compounded once per period, of which there are
- *nper* total
- and (optional) specification of whether payment is made at the beginning (*when* = {'begin', 1}) or the end (*when* = {'end', 0}) of each period

Return:

the (fixed) periodic payment.

NumPy-Financial

Πόση είναι η μηνιαία πληρωμή ώστε να αποπληρώσουμε δάνειο των $200.000 σε 15 έτη με ετήσιο επιτόκιο 7.50% ;

```
import numpy_financial as npf
npf.pmt(0.075/12, 12*15, 200000)
```

Η μηνιαία δόση για την αποπληρωμή του δανείου ισούται με $1,854.024

NumPy-Financial

nper

`numpy_financial.nper(rate, pmt, pv, fv=0, when='end')`

Compute the number of periodic payments.

`decimal.Decimal` type is not supported.

Parameters:

rate : *array_like*
Rate of interest (per period)

pmt : *array_like*
Payment

pv : *array_like*
Present value

fv : *array_like, optional*
Future value

when : *{{'begin', 1}, {'end', 0}}, {string, int}, optional*
When payments are due ('begin' (1) or 'end' (0))

NumPy-Financial

Με την συνάρτηση nper υπολογίζουμε το πλήθος των περιοδικών πληρωμων

Υποθέστε πως διαθέτετε μόνο $150/μήνα για την αποπληρωμή ενός δανείου. Πόσο καιρός απαιτείται ώστε να αποπληρώσετε ένα δάνειο των $8.000 με 7% ετήσιο επιτόκιο;

```
import numpy as np
import numpy_financial as npf
np.round(npf.nper(0.07/12, -150, 8000), 5)
```

Βλέπουμε πως απαιτούνται περίπου 64 μήνες

Καθαρη Παρούσα Αξία(NPV)

Με τον όρο καθαρή παρούσα αξία ορίζουμε το εξής :

$$NPV = PV(benefits) - PV(costs)$$

Υποθέστε πως επενδύετε σε ένα 5ετές έργο 100 εκατομμύρια €.Οι μελλοντικές ταμειακές ροές είναι 20εκατ €,40εκατ €,50 εκατ €,20εκατ €,10εκατ € αντίστοιχα. Επίσης το προεξοφλητικό επιτόκιο (Discount rate) ισούται με 5% ανά έτος. Συμφέρει να επενδύσετε στο επενδυτικό έργο ;

Αρχικά θα υπολογίσουμε την παρούσα αξία της επένδυσης μας, και έπειτα θα εφαρμόσουμε τον (απλό) κανόνα απόφασης (NPV rule)

$$\begin{cases} If\ NPV(project) > 0\ accept \\ If\ NPV(project) < 0\ reject \end{cases}$$

Καθαρη Παρούσα Αξία(NPV)

```
-100 + 20/(1+0.05)+40/(1+0.05)**2 +50/(1+0.05)**3+20/(1+0.05)**4+10/
(1+0.05)**5
```

Κάνοντας έναν απλό υπολογισμό πρακτικά βλέπουμε ότι το καθαρό μας κέρδος είναι περίπου 22,8εκατ €. Βέβαια ήταν μια αρκετή περίπλοκή διαδικασία.Πρακτικά δεν χρειαζόμασταν την Python , καθως με έναν υπολογιστή τσέπης θα μπορούσαμε να κάνουμε τον υπολογισμό μόνοι μας.

Όμως στην περίπτωση, για παράδειγμα που είχαμε περισσότερα έτη οι πράξεις θα ήταν ακόμα πιο χρονοβόρες και πιο «ευαίσθητες» σε πιθανό λάθος εκ μέρους μας. Με την χρήση της Python θα μπορούσαμε να ορίσουμε το r=0.5 εξ αρχής , και να απλοποιούσαμε (ελαφρώς) τον υπολογισμό μας

```
r=0.05
-100 + 20/(1+r)+40/(1+r)**2 +50/(1+r)**3+20/(1+r)**4+10/(1+r)**5
```

Καθαρη Παρούσα Αξία(NPV)

Βέβαια Υπάρχει και καλύτερος τρόπος για ευρεση της καθαρής παρούσας αξίας, με την χρήση της Python :

```
def npv_f(rate, cashflows):   #ορίζουμε την συναρτησή μας
        total = 0.0           # Ξεκινάμε από την τιμή 0
        for i, cashflow in enumerate(cashflows):
                total += cashflow / (1 + rate)**i (Γενικά στην Python το x+=y ->x=x+y)
        return total

r=0.05
cashflows=[-100,20,40,50,20,10]
npv_f(r,cashflows)
```

Η enumerate μας επιστρέφει την τιμή καθώς και τον μετρητή για την τιμή της επανάληψης. Η λογική είναι πιο κατανοητή οπτικά στο PythonTutor

https://pythontutor.com/visualize.html#mode=edit

IRR Rule

Το IRR είναι το προεξοφλητικό επιτόκιο που ουσιαστικά εξισώνει την παρούσα άξία με το 0.Για να αποδεχτούμε μια επένδυση ουσιαστικά ο IRR πρέπει να είναι μεγαλύτερο από το επιτόκιο της απόδοσης

$$\begin{cases} If\ IRR(project) > R_{capital} & accept \\ If\ IRR(project) < R_{capital} & reject \end{cases}$$

IRR Rule

Θα χρησιμοποιήσουμε τα δεδομένα που είχαμε εφαρμόσει στην Numpy Financial

Αρχικά ορίζουμε την συνάρτηση για την παρούσα αξία του χρήματος

```
cashflows=[-100,39,59,55,20]
def npv_f(rate, cashflows):
    total = 0.0
    for i, cashflow in enumerate(cashflows):
        total += cashflow / (1 + rate)**i
    return total
```

IRR Rule

Τώρα θα ορίσουμε την συνάρτηση για το IRR

```
def IRR_f(cashflows,interations=100):
    rate=1.0                          #Ξεκινάμε με R=1 (100%)
    investment=cashflows[0]           #Η επένδυση ισούται με το πρώτο στοιχείο στην λίστα μας
    for i in range(1,interations+1):
        rate*=(1-npv_f(rate,cashflows)/investment)
    return rate
```

Το πιο σημαντικό κομμάτι του κώδικα είναι η προτελευταία σειρά του. Ας την δούμε λίγο αναλυτικά

IRR Rule

- Έχουμε λοιπόν

rate*=(1-npv_f(rate,cashflows)/investment) → $R_{i+1} = R_i * (1-k)$

Εάν το R_i μας οδηγήσει σε θετική παρούσα αξία θα αυξήσουμε το προεξοφλητικό μας επιτόκιο σε R_{i+1} το οποίο θα είναι μεγαλύτερο από το R_i , έτσι το k θα είναι ενας μικρός αρνητικός αριθμός.Σε αντίθετη περίπτωση αν το R_i μας αποδώσει αρνητική παρούσα αξία θα μειώσουμε το προεξοφλητικό μας επιτόκιο το οποίο συνεπάγεται σε θετικό αριθμό k

Έτσι έχουμε το ακόλουθο αποτέλεσμα με χρήση της συνάρτησης

- IRR_f(cashflows)

```
0.2809484211599612
```

EAR & APR

Υποθέστε ότι η τράπεζα ‘Α’ μας προσφέρει δάνειο με εξαμηνιαίο ανατοκισμό 5% και η τράπεζα ‘Β’ μας προσφέρει ανατοκισμό 5.1% ανα τετράμηνο. Ποια από τις 2 τράπεζες πρέπει να επιλέξουμε για το δάνειο μας;

EAR: Effective Annual Rate

APR: Annual Percentage Rate

$$EAR = \left(1 + \frac{APR}{m}\right)^{m} - 1$$

Κατόπιν πράξεων βλέπουμε η τράπεζα Α είναι μια πιο συμφέρουσα επιλογή

EAR & APR

```
APR_A=0.05
m=2
EAR_A=(1+APR_A/m)**m-1
round(EAR_A,6)
APR_B=0.051
m=4
EAR_B=(1+APR_B/m)**m-1
print(f"{EAR_A=} {EAR_B=}")  #Απο python 3.8 +++
```

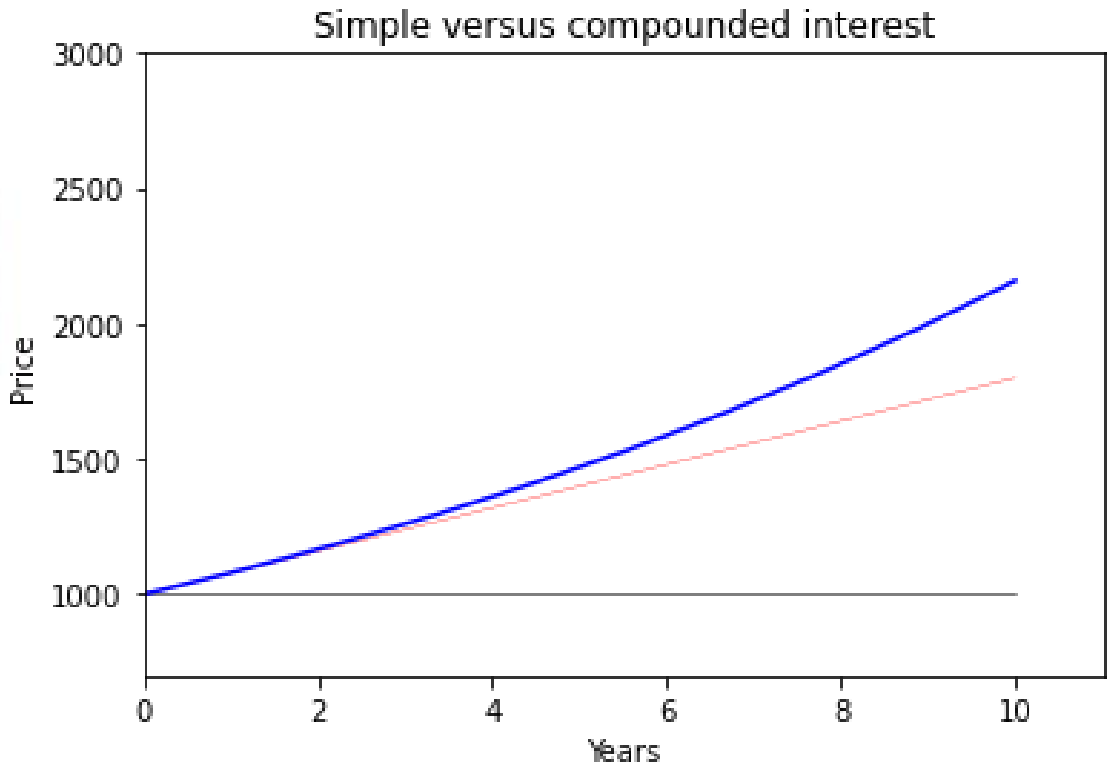
Επιτόκιο και Ανατοκισμός : Οπτικά…

```
import numpy as np
from matplotlib.pyplot import*
from pylab import *
pv=1000
r=0.08
n=10
t=np.linspace(0,n,n)
y1=np.ones(len(t))*pv
y2=pv*(1+r*t)
y3=pv*(1+r)**t
```

```
#Εισαγωγή των απαραίτητων βιβλιοθηκών
#Αρχικό κεφάλαιο
#Τόκος
```

Ανάλυση επιτοκίων

```
title('Simple versus compounded interest')
xlabel('Years')
ylabel('Price')
ylim(700,3000)
xlim(0,n+1)
plot(t,y1,'k',lw=0.5)
plot(t,y2,'r',lw=0.3)
plot(t,y3,'b')
```



Αναζήτηση ιστορικών τιμών με την Python

- Η Python μας παρέχει την δυνατότητα να αντλήσουμε τις ιστορικές τιμές μετοχών με έναν άμεσο τρόπο.
- Γενικα υπάρχουν πολλες βιβλιοθήκες που μας δίνουν αυτην την δινατότητα. Βέβαια πολλές από αυτές απαιτούν εγγραφή σε κάποια πλατφόρμα και την χρήση API key.
- Θα δούμε 2 βιβλιοθήκες όπου δεν απαιτείται εγγραφή καθώς επίσης η διαδικασία που χρειάζεται για να κατεβάσουμε τις τιμές είναι αρκετά απλή

Yahoo Finance

- Η πρώτη μας βιβλιοθήκη είναι η yfinance η οποία αναζητά τις ιστορικές τιμές μέσω του Yahoo Finance
- Πριν προχωρήσουμε στην λήψη των τιμών θα δούμε μερικές ενδιαφέρουσες εφαρμογές αυτού του πακέτου με την χρήση της συνάρτησης yf.Ticker()
- Με την χρήση της Ticker() είμαστε σε θέση να έχουμε πρόσβαση σε διάφορες πληροφορίες σχετικά με την μετοχή που μελετάμε
- Έπειτα θα δούμε την yf.downlad() ώστε να κατεβάσουμε τις τιμες που επιθυμούμε

Yahoo Finance

- Αρχικά όπως και με όλες τις βιβλιοθήκες θα χρειαστεί να κάνουμε import το yfinance

- import yfinance as yf

(Με την χρήση της as δίνουμε μια συντομογραφία στην βιβλιοθήκη για την δικιά μας ευκολία)

```
ford = yf.Ticker("F")
```

Ο όρος Ticker Symbol(Σύμβολο μετοχής) αποτελεί τον κωδικό/συντομογραφία για να προσδιορίσουμε την μετοχή.

Yahoo Finance

Τώρα θα παραθέσουμε μερικές από τις δυνατότητες που έχουμε με την συνάρτηση Ticker().

ford.info	Μας αποδίδει πληροφορίες σχετικα με την μετοχή.
ford.dividends	Μας αποδίδει τα μερίσματα της μετοχής.
ford.recommendations	Λίστα με τις συστάσεις αναλυτών.
ford.cashflow	Ταμειακές ροές της μετοχής.
ford.balance_sheet	Ισολογισμός της μετοχής.
ford.news	Λίστα αρθρων σχετικα με την μετοχή μας(Τιτλος/Site/Link).

Yahoo Finance

```
import yfinance as yf
ford = yf.download("F", start="2020-01-01", end="2021-01-01")
```

Όπως βλέπουμε για να κατεβάσουμε τις ιστορικές τιμές μιας μετοχής είναι αρκετά εύκολο καθότι χρειάζεται να γράψουμε μόνο μια σειρά κώδικα.

Αρχικά κάνουμε χρήση της εντολής yf.download και έπειτα εισάγουμε τα εξής ορίσματα :

```
("Κωδικός μετοχής", start= " YYYY-MM-DD ", end="YYYY-MM-DD")
```

Pandas-datareader

- Ενας εναλλακτικός τρόπος είναι με την χρήσης της βιβλιοθήκης pandas datareader

```
import pandas_datareader as web

ford = web.DataReader('F', 'yahoo', "2020-01-01", "2021-01-01")
```

Εδώ παρατηρούμε πως υπάρχει ένα παραπάνω όρισμα στην συνάρτηση μας. Στο δεύτερο όρισμα εισάγουμε την πηγή των δεδομένων μας. Θα μπορούσαμε να επιλέξουμε και άλλα site για παράδειγμα το Quandl ,όμως θα ήταν υποχρεωτική η χρήση API key για αυτόν τον σκοπό.

Pandas-datareader

Επίσης η pandas_datareader μας παρέχει δεδομένα σχετικά με Eurostat, FRED, World Bank χωρίς χρήση API key.

Για περισσότερες πληροφορίες, μπορείτε να ανατρέξετε και στο official site :

Remote Data Access — pandas-datareader 0.10.0 documentation

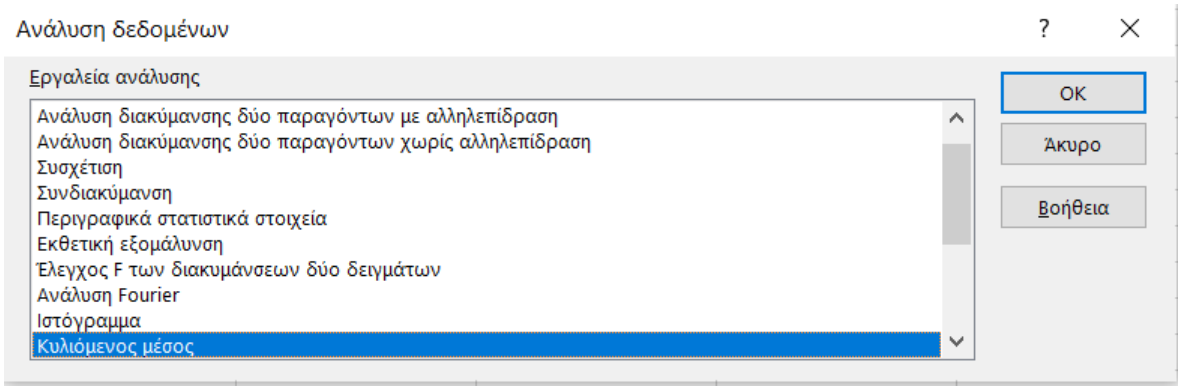
Algorithmic Trading: Moving Averages (SMA)

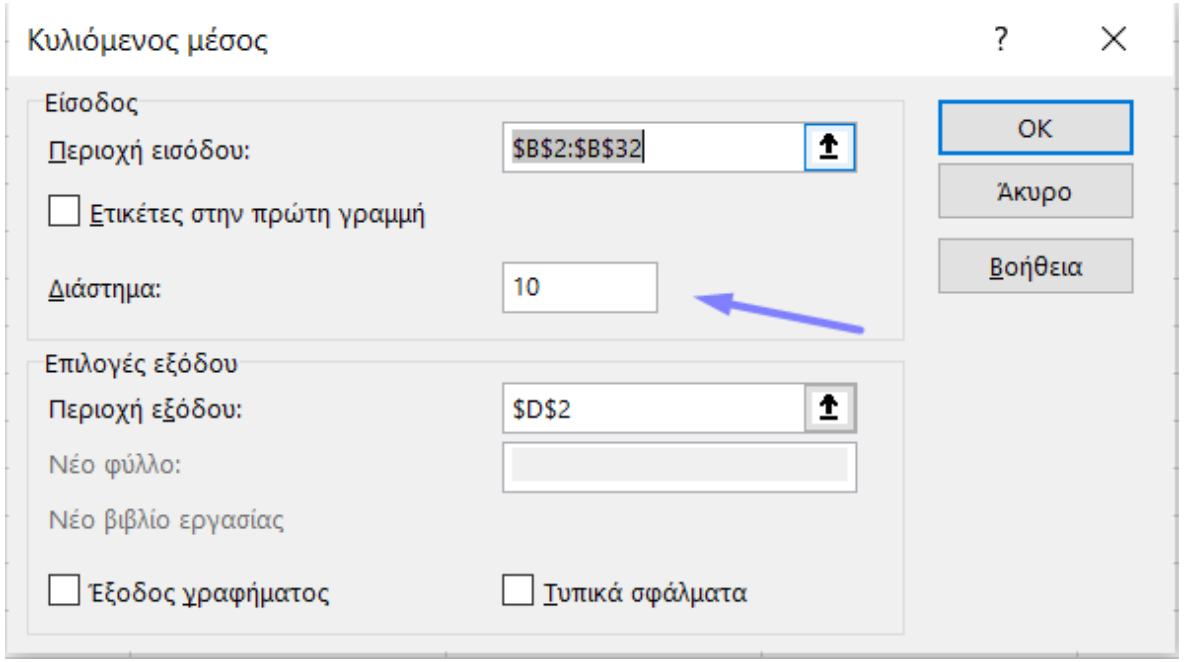
- Ένα από τα πιο γνωστά εργαλεία στον τομέα της τεχνικής ανάλυσης είναι ο κυλιόμενος μέσος όρος ο οποίος εξομαλύνει την χρονοσειρά μας. Ουσιαστικά μας δείχνει την τάση της, παρά τις προσωρινές διακυμάνσεις της τιμής, δημιουργώντας έναν μέσο όρο των τιμών, μιας συγκεκριμένης παρελθοντικής χρονικής περιόδου που ορίζει ο Trader. Για παράδειγμα για να υπολογίσουμε τον SMA(10) προσθέτουμε τις τιμές κλεισίματος των τελευταίων 10 ημερών μια μετοχής και το διαιρούμε με το 10.
- Ένας κυλιόμενος μέσος μικρής περιόδου αντιδρά πιο γρήγορα σε μεταβολές της τιμής απ’ ότι ένας που αναφέρεται σε μεγαλύτερη χρονική διάρκεια.
- Θα εστιάσουμε αρκετά καθότι είναι μια πολύ σημαντική έννοια. Επίσης οι κυλιόμενοι μέσοι χρησιμοποιούνται και σε άλλους δείκτες όπως MACD, Bollinger Bands κ.α

Algorithmic Trading: Moving Averages (SMA)

Για παράδειγμα στο Excel θα κάναμε χρήση της Average για τις 10 προηγούμενες τιμές. Εναλλακτικά μέσω του Data Analysis Tool pack μπορούμε να κάνουμε εύκολα τον ίδιο υπολογισμό.

38		
33		
38		
38		
30		
24		
25		
33		
34		
34	=AVERAGE(B2:B11)	32.7
28	=AVERAGE(B3:B12)	31.7
29	=AVERAGE(B4:B13)	31.3
20	=AVERAGE(B5:B14)	29.5
36	=AVERAGE(B6:B15)	29.3
21	=AVERAGE(B7:B16)	28.4
23	=AVERAGE(B8:B17)	28.3
29	=AVERAGE(B9:B18)	28.7
27	=AVERAGE(B10:B19)	28.1
29	=AVERAGE(B11:B20)	27.6
32	=AVERAGE(B12:B21)	27.4
22	=AVERAGE(B13:B22)	26.8
24	=AVERAGE(B14:B23)	26.3
37	=AVERAGE(B15:B24)	28
37	=AVERAGE(B16:B25)	28.1
31	=AVERAGE(B17:B26)	29.1
30	=AVERAGE(B18:B27)	29.8
29	=AVERAGE(B19:B28)	29.8
23	=AVERAGE(B20:B29)	29.4
36	=AVERAGE(B21:B30)	30.1
24	=AVERAGE(B22:B31)	29.3
35	=AVERAGE(B23:B32)	30.6





Algorithmic Trading: Moving Averages (SMA)

- Ας προχωρήσουμε σε ένα οπτικό παράδειγμα για να καταλάβουμε την χρήση των Moving Averages στην οικονομική ανάλυση. Μάλιστα θα κάνουμε χρήση της Python για να κατεβάσουμε τις τιμές της μετοχής μας ,να υπολογίσουμε τους κυλιόμενους μέσους, καθώς και να οπτικοποιήσουμε τα αποτελέσματα μας.
- Όπως αναφέραμε και προηγουμένως όσο μικρότερο είναι το χρονικό διάστημα που χρησιμοποιείται για την δημιουργία του κυλιόμενου μέσου, τόσο πιο ευαίσθητος είναι στις μεταβολές της τιμής. Αν ορίσουμε ένα μεγάλο χρονικό διάστημα τότε θα δείχνει την μακροχρόνια τάση οπότε θα είναι λιγότερο ευαίσθητος στις πραγματικές τιμές
- Στις επόμενες διαφάνειες θα υπολογίσουμε 2 κυλιόμενους μέσους για την μετοχή της TESLA 10 και 50 ημερών ,και θα δούμε διαγραμματικά την συμπεριφορά τους.

Algorithmic Trading: Moving Averages (SMA)

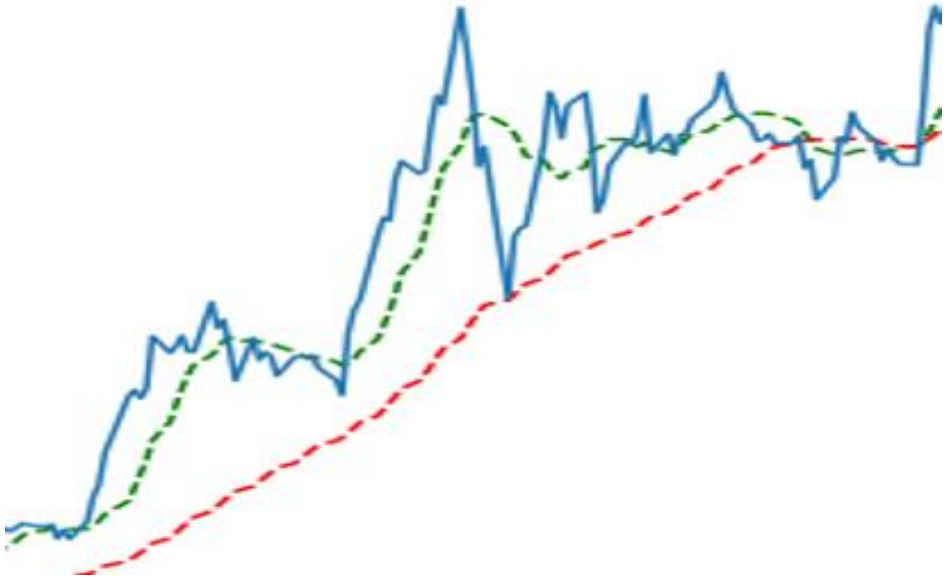
```
import pandas_datareader as web                    #Με την χρήση του pandas_datareader θα κατεβάσουμε τις τιμές μας
import pandas as pd                                #Με την χρήση του pandas θα επεξεργαστούμε το data frame μας
import matplotlib.pyplot as plt                    #Matplotlib για την δημιουργία διαγραμμάτων
start_date = '2020-01-01'
end_date = '2021-01-01'
tesla = web.DataReader('TSLA', 'yahoo', start_date, end_date)    #Κατεβάζουμε τις τιμές της μετοχής μας μέσω Yahoo Finance
tesla["SMA10"] = tesla['Adj Close'].rolling(window=10).mean()    # Υπολογίζουμε  τον κυλιόμενο μέσο 10 ημερών
tesla["SMA50"] = tesla['Adj Close'].rolling(window=50).mean()    # Υπολογίζουμε  τον κυλιόμενο μέσο 50 ημερών
plt.figure(figsize=(10,10))
plt.plot(tesla['SMA10'], 'g--', label="SMA10")
plt.plot(tesla['SMA50'], 'r--', label="SMA50")
plt.plot(tesla['Adj Close'], label="close")
plt.legend()
plt.show()
```

Algorithmic Trading: Moving Averages (SMA)

Algorithmic Trading: Moving Averages (SMA)

Ας δούμε λοιπόν το διάγραμμα μας. Η μπλέ γραμμή απεικονίζει τις τιμές της TESLA για το χρονικό ορίζοντα που ορίσαμε. Η πράσινη γραμμή απεικονίζει τον MA(10) η αλλιώς γρήγορο κυλιόμενο μέσο και η κόκκινη τον MA(50) η εναλλακτικά τον αργό μας κυλιόμενο μέσο.

Όπως παρατηρούμε η πράσινη γραμμή ακολουθεί τις πραγματικές τιμές πιο «γρήγορα». Αντιθέτως η κόκκινη γραμμή όπου εκφράζει τον MA(50) είναι λιγότερο ευαίσθητη. Θα μπορούσε να σκεφτεί κάποιος ότι ίσως ένας μεγάλος MA δεν είναι το ίδιο χρήσιμος με έναν μικρότερο. Ένας μεγάλος MA μας δείχνει την τάση, η οποία σε ορισμένες περιπτώσεις δεν είναι τόσο οφθαλμοφανής.



Algorithmic Trading: Moving Averages (SMA)

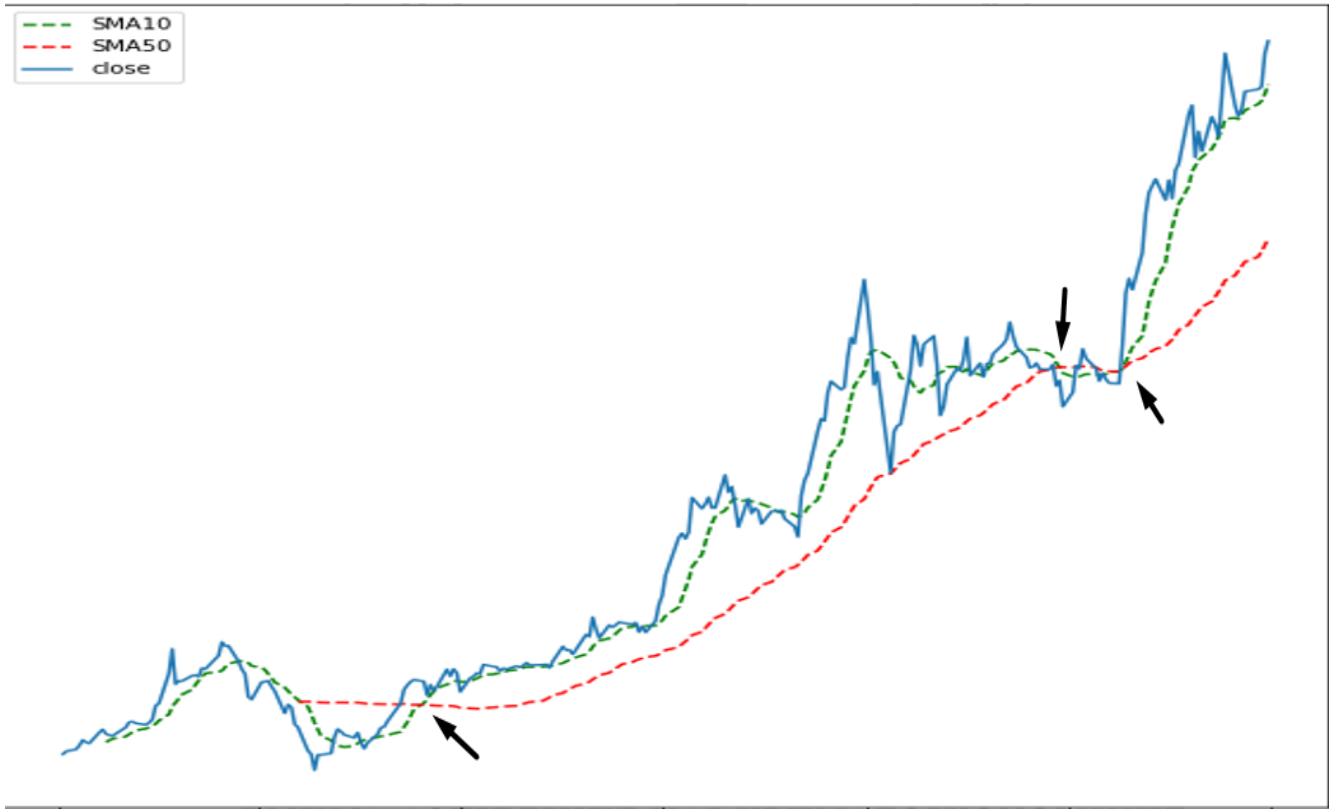
- Ήρθε η ώρα να μιλήσουμε για την πρώτη μας στρατηγική σχετικά με τους Κυλιόμενους μέσου όρους. Εισάγουμε 2 ΜΑ έναν γρήγορό , δηλαδή μικρής διάρκειας και έναν αργό μεγάλης διάρκειας.

- Όταν ο γρήγορος ΜΑ περάσει πάνω από τον αργό ΜΑ είναι σήμα για είσοδο σε θέση αγοράς, καθώς μας δείχνει ότι η τάση θα έχει είναι αυξητική. Αυτό το σημείο είναι γνωστό σαν Golden cross

- Αντίστοιχα σε αντίθετη περίπτωση όταν ο αργός ΜΑ περάσει πάνω από τον γρήγορο είναι ένδειξη για είσοδο σε θέση πώλησης. Αυτό το σημείο είναι γνωστό σαν Death Cross.

Algorithmic Trading: Moving Averages (SMA)

- Βλέπουμε στο πρώτο βελάκι ότι έπειτα από την εμφάνιση του Golden Cross υπήρξε άνοδος της τιμής, οπότε πράγματι η θέση αγοράς θα ήταν μια κερδοφόρα επιλογή. Ακόμα και στο Death Cross πράγματι υπήρξε πτώση τις τιμής η οποία βέβαια διήρκησε λίγο οπότε δεν θα ήταν τόσο κερδοφόρα. Στο τελευταίο Golden Cross βλέπουμε ότι πράγματι υπήρξε άνοδος ξανά



Algorithmic Trading: Moving Averages (EMA)

- Ο Εκθετικός κινητός μέσος όρος(Exponential Moving Average) είναι ένας ακόμα τύπος κινητού μέσου ο οποίος δίνει μεγαλύτερη βαρύτητα στα πιο πρόσφατα σημεία δεδομένων.

$$\text{EMA}=(CP_t - EMA_{t-1}) \times k + EMA_{t-1}$$

CP_t: Τιμή κλεισίματος την στιγμη t(σήμερα)

EMA_{t-1}:EMA της t-1 στιγμής(προηγουμένης μέρας)

k :σταθερά εξομάλυνσης

Algorithmic Trading: Moving Averages (EMA)

Βλέποντας τον προηγούμενο τύπο θα μπορούσε να αναρωτηθεί κάποιος πως υπολογίζουμε τον EMA, ενώ η εξίσωση περιλαμβάνει τον EMA_{t-1} ;

Ουσιαστικά, αν θέλαμε να υπολογίσουμε τον EMA την 11η ημέρα θα χρησιμοποιούσαμε για EMA_{t-1} τον SMA(10).Επίσης σχετικά με το k :

$$\mathrm{k} = \frac{2}{N+1}$$ όπου N:Περίοδοι

Γενικά οι εκθετικοί κυλιόμενοι μέσοι χρησιμοποιούνται και σε διάφορους δείκτες ,όπως ο MACD οπού θα αναλύσουμε μελλοντικά.

Αν και φαίνεται μια περίπλοκή διαδικασία ο υπολογισμός του, με την βοήθεια της βιβλιοθήκης Pandas , μπορούμε με να τον υπολογίσουμε εύκολα.

Algorithmic Trading: Bollinger Bands

- Γενικά υπάρχουν πολλοί συνδυασμοί στρατηγικών σχετικά με τις ζώνες του Bollinger
- Θα δούμε ίσως το πιο απλό παράδειγμα πάνω σε αυτές. Η λογική είναι η εξής, όταν η τιμή του security «χτυπήσει» την πάνω ζώνη του Bollinger σημαίνει ότι υπάρχει Overbought condition και αναμένουμε πτώση τις τιμής, οπότε μπαίνουμε σε θέση πώλησης. Αντίστοιχα όταν η τιμή περάσει την κάτω ζώνη τότε υπάρχει Oversold condition σχετικά με την τιμή του Security ,όποτε αναμένουμε άνοδο της τιμής άρα παίρνουμε θέση αγοράς.
- Σε ορισμένες περιπτώσεις η Signal Line [SMA(20)] μπορεί να αποτελέσει stop-loss condition

Algorithmic Trading: Bollinger Bands

- Ένας επίσης διάσημος και δημοφιλής δείκτης Trading είναι οι Bollinger Bands όπου πήραν το όνομα τους από τον John Bollinger. Οι λωρίδες/ζώνες του Bollinger χρησιμοποιούν έναν SMA (συνήθως 20 ημερών) και επιπλέον 2 ακόμα «ζώνες» οι οποίες ισούνται με:

$$SMA \pm m*\sigma$$

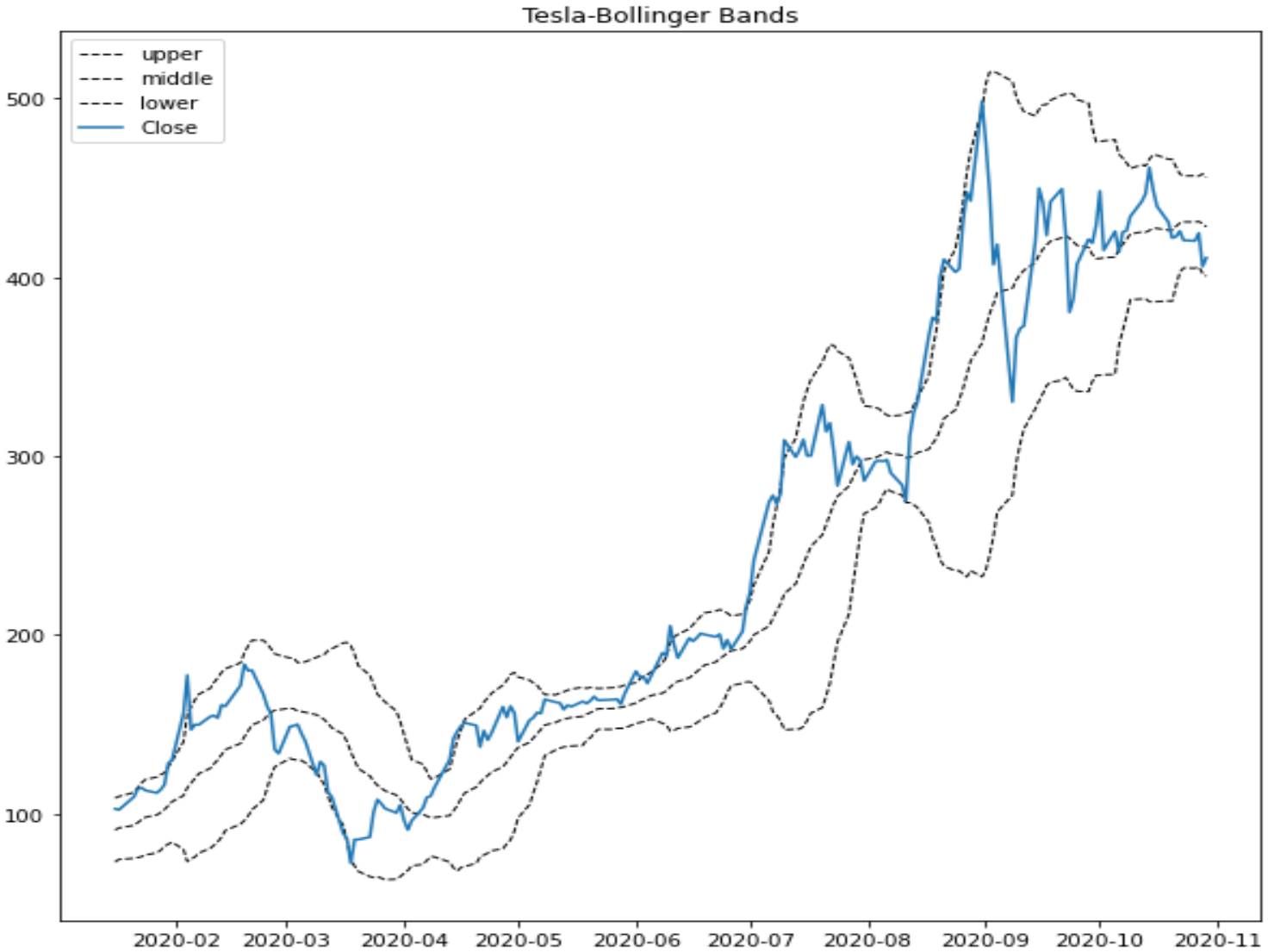
m:Πλήθος τυπικών αποκλίσεων

σ: Τυπική απόκλιση

Algorithmic Trading: Bollinger Bands

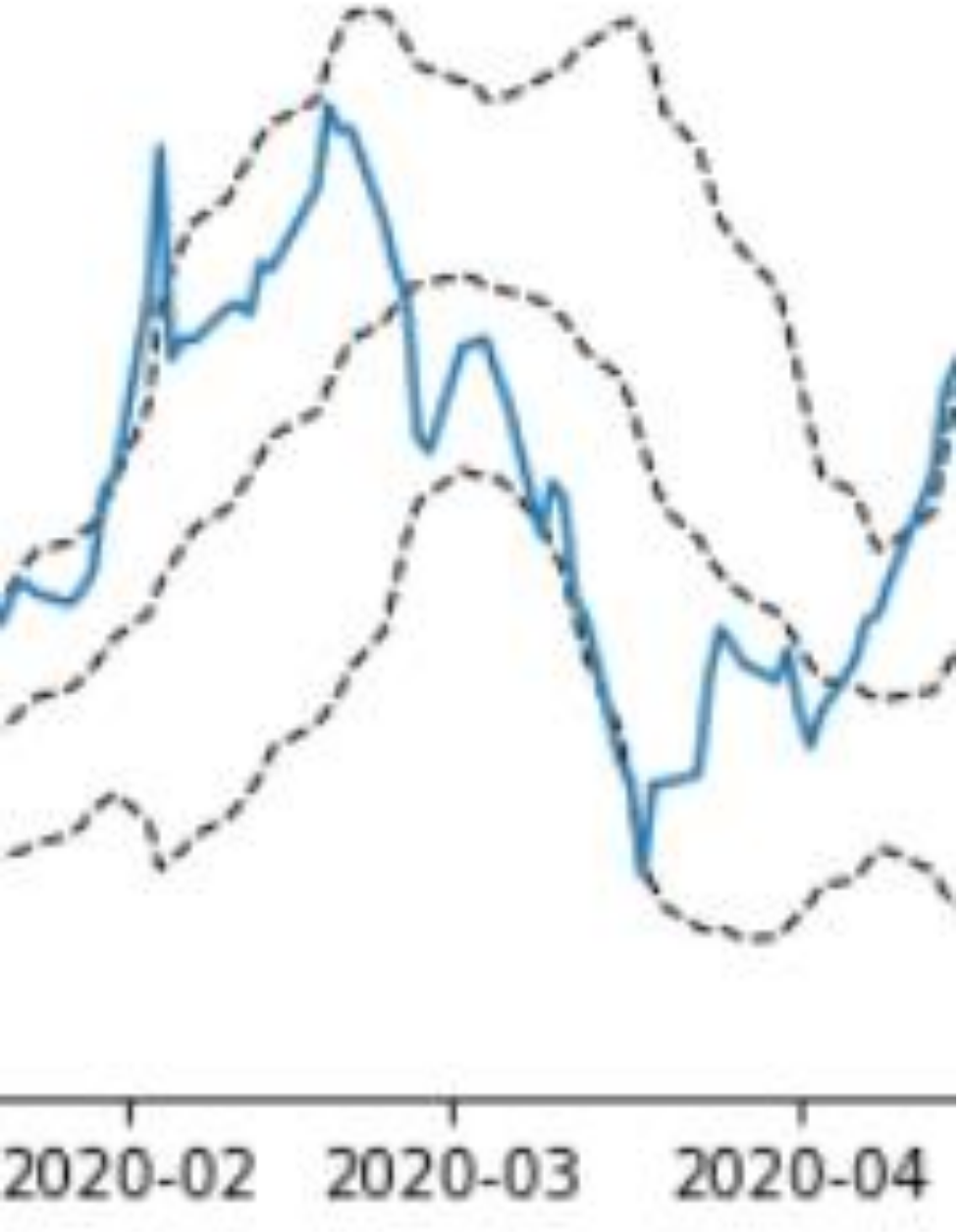
```
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
import pandas_datareader as web
df=web.DataReader('TSLA',data_source='yahoo',start='1/4/2019',end='29/10/2020')
df['middle_band'] = df['Close'].rolling(window=20).mean()
df['upper_band'] = df['Close'].rolling(window=20).mean() + df['Close'].rolling(window=20).std()*2
df['lower_band'] = df['Close'].rolling(window=20).mean() - df['Close'].rolling(window=20).std()*2
plt.figure(figsize=(20,20))
plt.plot(df['upper_band'].iloc[-200:], 'k--',linewidth=1, label="upper")
plt.plot(df['middle_band'].iloc[-200:], 'k--',linewidth=1 ,label="middle")
plt.plot(df['lower_band'].iloc[-200:], 'k--',linewidth=1 ,label="lower")
plt.plot(df['Close'].iloc[-200:], linewidth=1.5,label="Close")
plt.legend()
plt.show()
```

Algorithmic Trading: Bollinger Bands



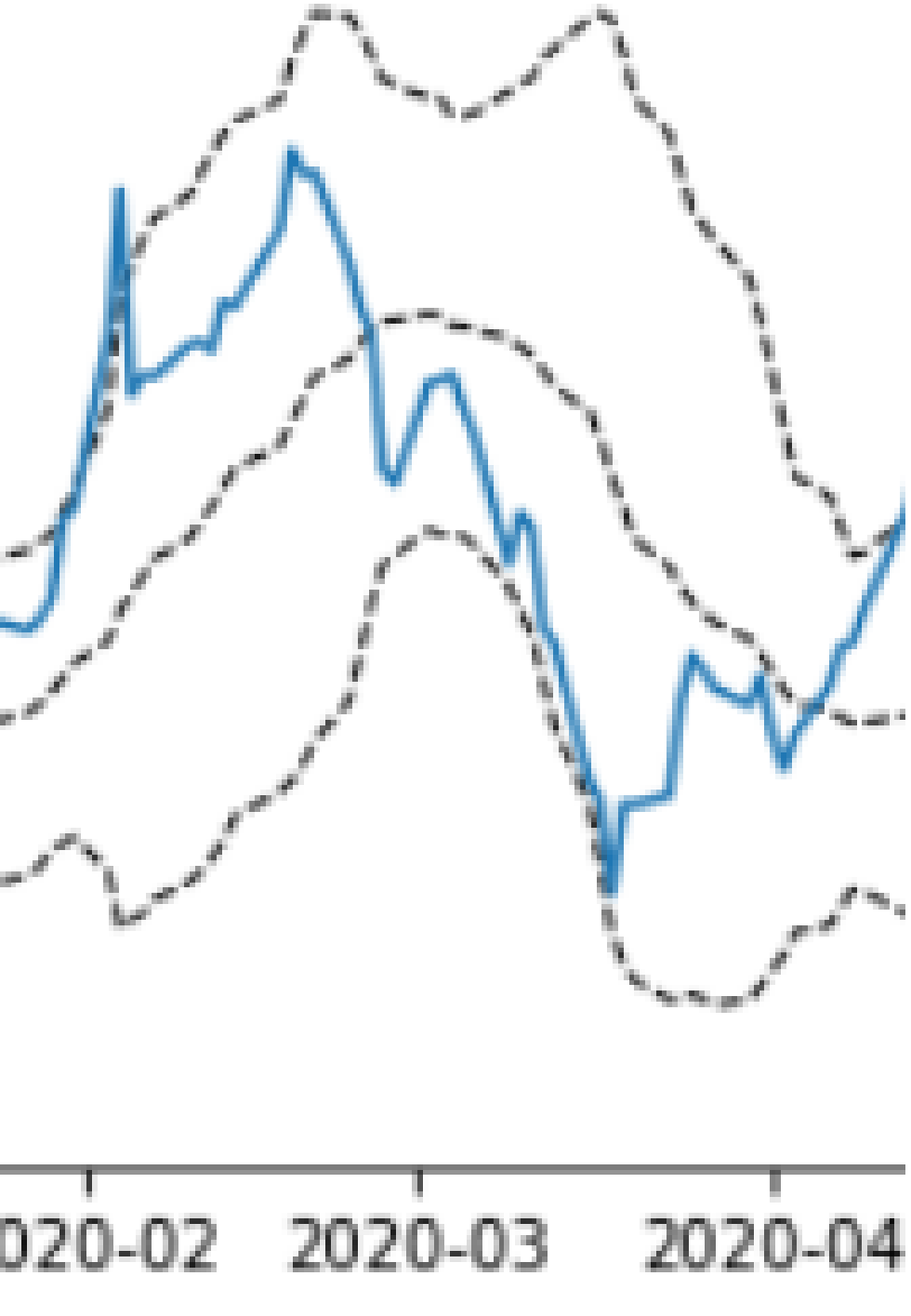
Algorithmic Trading: Bollinger Bands

- Όπως προαναφέραμε σκοπός αυτού του παραδείγματος είναι αρκετά απλοϊκός με σκοπό να αντιληφθούμε την χρήση και την λειτουργία των Ballinger Bands.
- Στο παράδειγμα μας χρησιμοποιήσαμε τον κλασσικό Ballinger Band με έναν SMA 20 ημερών ± 2 τυπικές αποκλείσεις
- Ο Trader πιθανόν να χρησιμοποιήσει άλλες τιμές στις παραμέτρους, έτσι ώστε να βελτιώσει και την απόδοση της τεχνικής του
- Επίσης όπως αναφέραμε και στην αρχή, δεν υπάρχει η τέλεια τεχνική να μας αποφέρει πάντα κέρδος. Θα δούμε λίγο πιο αναλυτικά τώρα το διάγραμμα μας.



Algorithmic Trading: Bollinger Bands

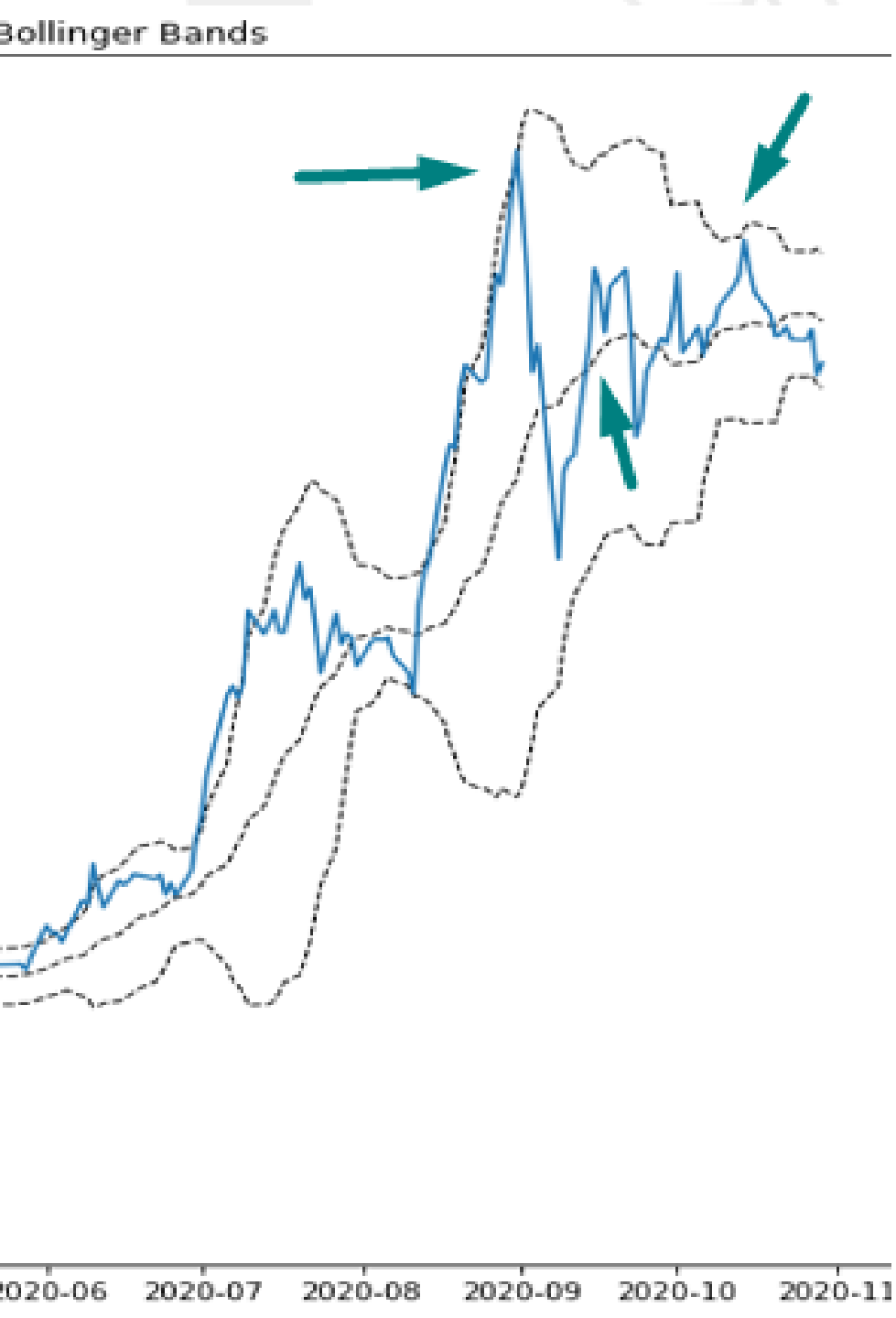
- Σε αυτό το κομμάτι του διαγράμματος βλέπουμε πως η στρατηγική των Bollinger Bands μας οδήγεί σε κέρδος. Αρχικά η τιμή μας είχε προσεγγίσει την άνω ζώνη ,οπότε θα μπαίναμε σε θέση πώλησης. Έπειτα βλέπουμε πράγματι να πέφτει η τιμή και θα μπορούσαμε να κλείσουμε την θέση μας με κέρδος. Παρατηρείστε ότι η χρονοσειρά των τιμών τις μετοχής εφάπτεται για μια συγκεκριμένη περίοδο με την κάτω ζώνη.Θα μπορούσαμε να είχαμε μεγαλύτερα κέρδη αν κλείναμε την θέση μας στο ελάχιστο. Πιθανόν με διαφορετικό SMA ,η διαφορετικό πλήθος τυπικών αποκλίσεων (m) να είχαμε καλυτερα αποτελέσματα.
- Συνήθως το m παίρνει ακέραιες τιμές, βέβαια ας χρησιμοποιήσουμε 2.3 τυπικές αποκλίσεις και ας δούμε εκ νέου το διάγραμμα μας



Algorithmic Trading: Bollinger Bands

- df['upper_band'] = df['Close'].rolling(window=20).mean() + df['Close'].rolling(window=20).std()*2.3
- df['lower_band'] = df['Close'].rolling(window=20).mean() - df['Close'].rolling(window=20).std()*2.3

Πράγματι απ ότι βλέπουμε με 2.3 τυπικές αποκλίσεις. Η τιμή μας εφάπτεται με την κάτω ζώνη την «κατάλληλη στιγμή» οπότε θα είχαμε μεγαλύτερα κέρδη ,με το κλείσιμο της θέσης μας.



Algorithmic Trading: Bollinger Bands

Αναφέραμε ότι ο SMA(20) θα μπορούσε να αποτελέσει ενας stop loss condition.Εστω ότι μπαίναμε σε θέση πώλησης στο πρώτο σημείο που η τιμή της μετοχής έχει φτάσει στην πάνω ζώνη του Bollinger.Θα μπορούσαμε να σκεφτούμε ότι από την στιγμή που υπάρξει πτώση της τιμής μας, η θέση μας θα κλείσει όταν φτάσουμε στο lower band.Βλέπουμε πως πράγματι ενώ η τιμή πέφτει και τείνει να προσεγγίσει την κάτω ζώνη,αρχίζει και ανεβαίνει εκ νέου.Τελικά η χρονοσειρά επανέρχεται στο άνω όριο της ζώνης όπως βλέπουμε και στο 3ο βελάκι.Θα μπορούσαμε να έχουμε κέρδος αν ορίζαμε, κατά την επαναφορά της τιμής στον SMA(20) να κλείσει η θέση μας

Algorithmic Trading: MACD

- Ο MACD είναι ένας δείκτης τάσης που δείχνει την σχέση ανάμεσα σε 2 κυλιόμενους μέσους της τιμής ενός τίτλου. Ισούται με την διαφορά 2 εκθετικών κυλιόμενων μέσων 12 και 26 ημερών

$$MACD=EMA(12)-EMA(26)$$

Η signal line υπολογίζεται από έναν EMA συνήθως 9 του ίδιου του MACD

Η στρατηγική είναι αρκετά απλή:

- Όταν η γραμμή του MACD περάσει πάνω από την signal line τότε παίρνουμε θέση αγοράς
- Αντίθετα όταν η Signal line περάσει πάνω από την γραμμή του MACD τότε θα πάρουμε θέση πώλησης

Algorithmic Trading: MACD

- Γενικά σε διάφορες εφαρμογές θα δείτε τα διαγράμματα του MACD και τις signal line είναι σε ξεχωριστό διάγραμμα, κάτω από αυτό της μετοχής που εφαρμόζουμε την τεχνική.
- Στο παράδειγμα μας θα έχουμε στο ίδιο διάγραμμα την χρονοσειρά της τιμής μας, τον MACD αλλά και την Signal Line μας.
- Θα κάνουμε χρήση της pandas_datareader για να κατεβάσουμε τις τιμές της Ford από 1/1/2019 έως 1/1/2020
- Θα εφαρμόσουμε την κλασσική στρατηγική του MACD.Θέση αγοράς στην περίπτωση που ο MACD περάσει την Signal Line ,και αντιστρόφως θέση πώλησης

Algorithmic Trading: MACD

```
import pandas_datareader as web
import pandas as pd
ford=web.DataReader('F',data_source='yahoo',start='1/1/2019',end='1/1/2020')['Close']
exp1 = ford.ewm(span=12, adjust=False).mean()
exp2 = ford.ewm(span=26, adjust=False).mean()
macd = exp1 - exp2
SigLine = macd.ewm(span=9, adjust=False).mean()
```

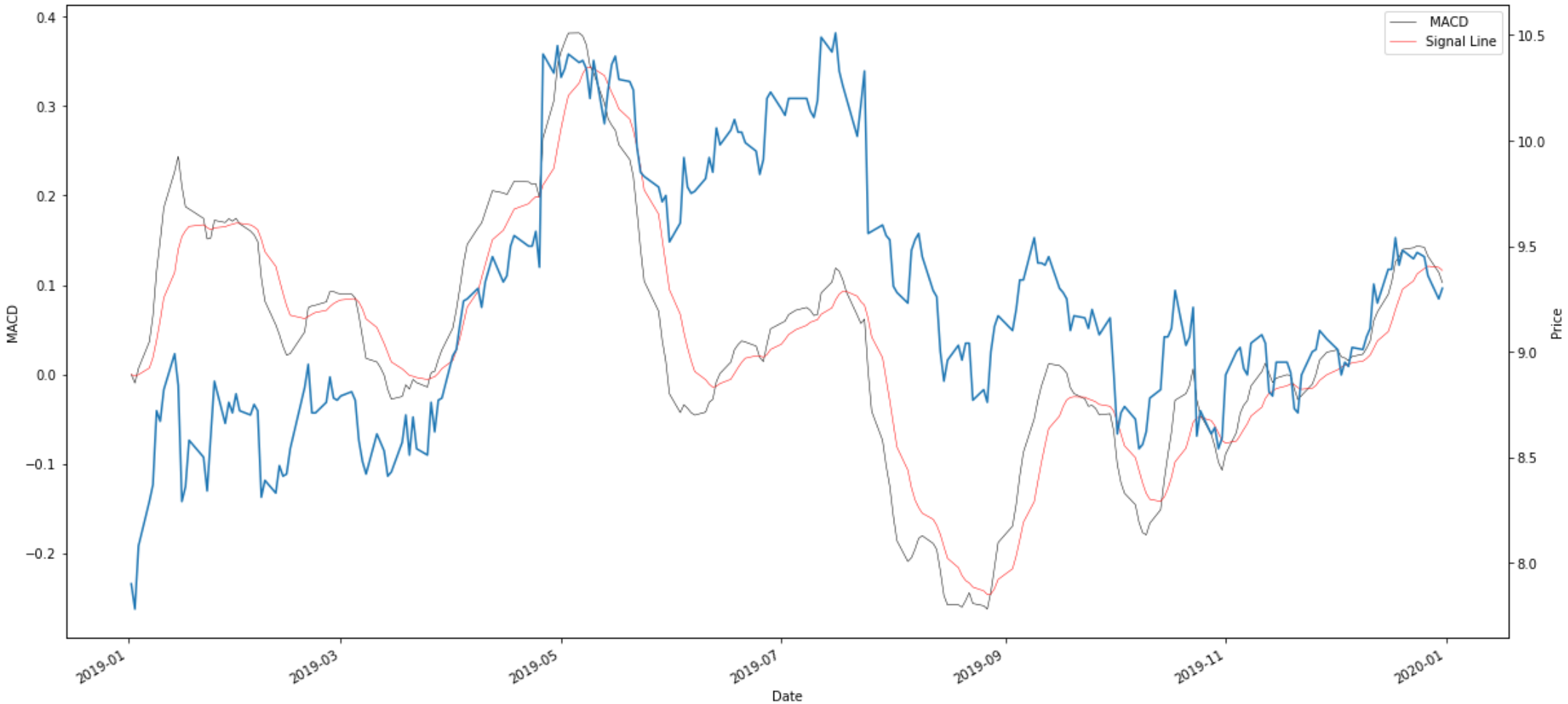
Χάρις στην Pandas βλέπουμε ότι ο υπολογισμός είναι αρκετά απλός. Υπολογίζουμε τους 2 εκθετικούς μέσους 12 & 26 ημερών στην $4^{η}$ και $5^{η}$ σειρά αντίστοιχα. Στην 6ή σειρά, η διαφορά τους μας δίνει τον macd.Τέλος υπολογίζουμε την Signal Line η οποία είναι ένας EMA 9 ημερών του ίδιου του macd.

Algorithmic Trading: MACD

Δημιουργία διαγράμματος με τα ακόλουθα:

```
macd.plot(label=' MACD', color='black',linewidth=0.5)
ax = SigLine.plot(label='Signal Line', color='red',linewidth=0.5)
ford.plot(figsize=(20,10),ax=ax, secondary_y=True)
ax.set_ylabel('MACD')
ax.right_ax.set_ylabel('Price')
ax.set_xlabel('Date')
ax.legend(loc='upper right')
```

Algorithmic Trading: MACD



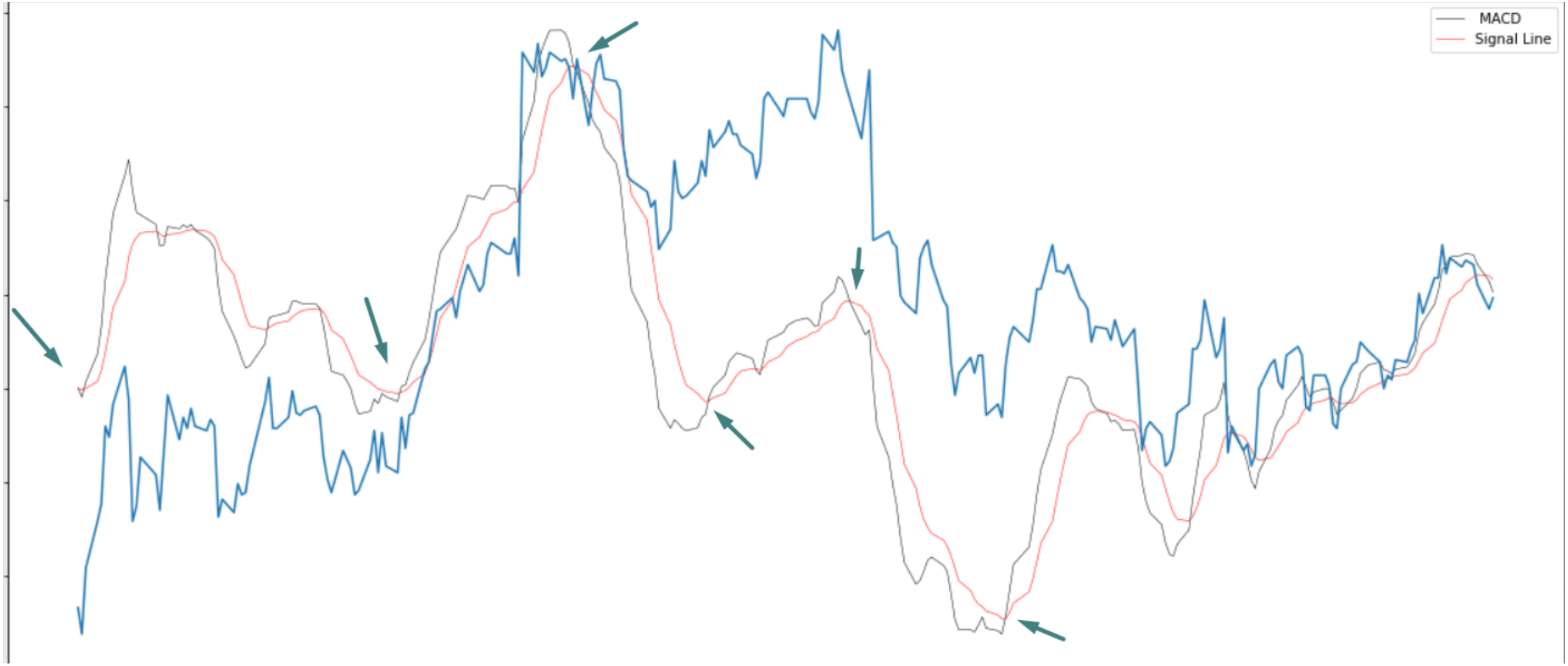
Algorithmic Trading: MACD

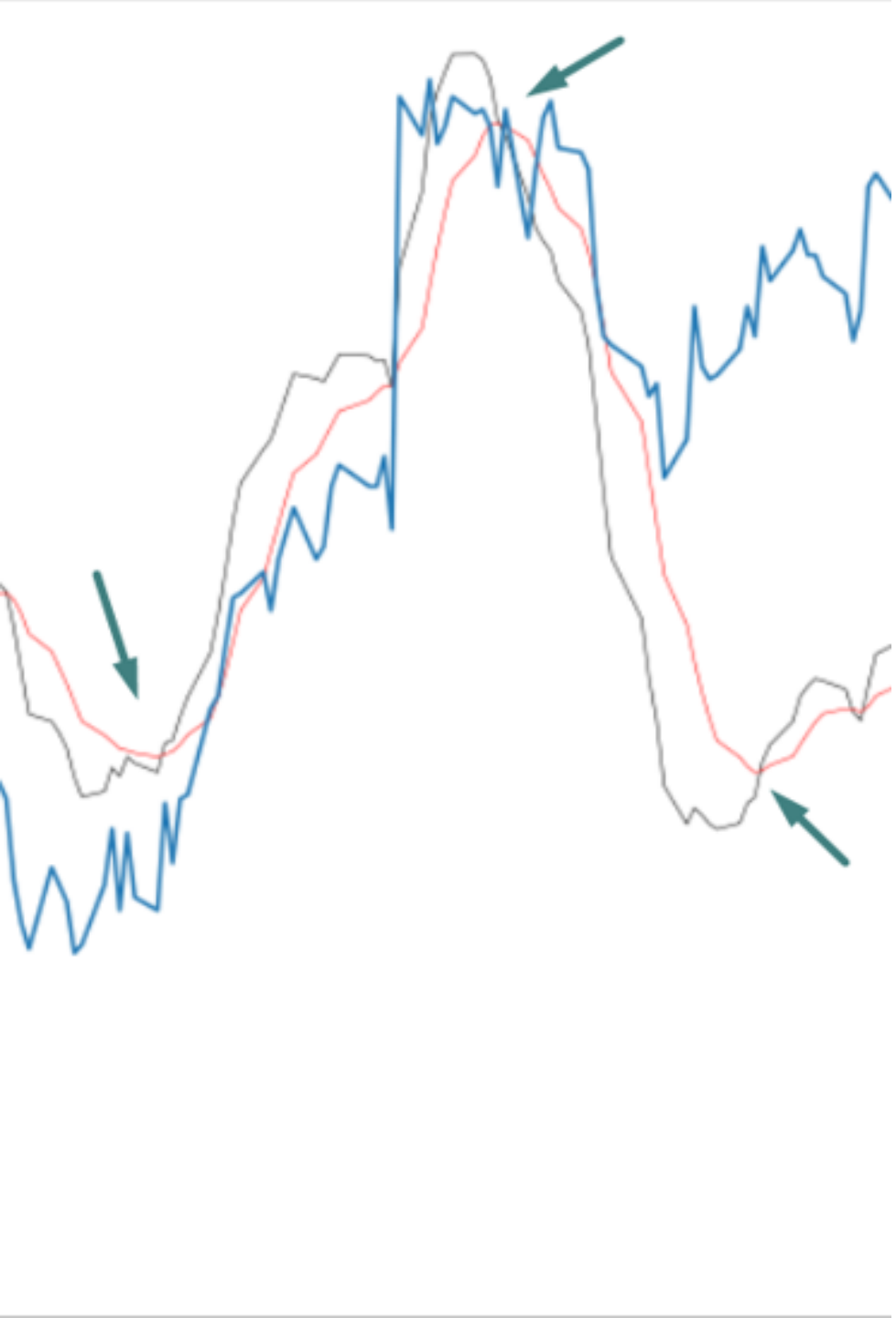
- Έχουμε λοιπόν το διάγραμμα μας όπου η μπλε γραμμή απεικονίζει την τιμή της Ford.

- Η μαύρη γραμμή αποτελεί την τιμή του MACD στην πάροδο του χρόνου και η κόκκινη την Signal Line

- Στην επόμενη διαφάνεια θα πάρουμε ένα τμήμα του διαγράμματος μας και θα δούμε πως υπήρξαν περιπτώσεις, όπου ο MACD μας «προειδοποίησε» για μεγάλες εναλλαγές στην τιμή της FORD στην πάροδο του χρόνου.

Algorithmic Trading: MACD





Algorithmic Trading: MACD

- Βλέπουμε πως στην μετοχή της Ford την συγκεκριμένη χρονική περίοδο υπήρξαν κάποιες έντονες μεταβολές. Την χρονική στιγμή όπου ο MACD πέρασε την Signal Line ,θα παίρναμε θέση αγοράς όπου θα μας απέφερε κέρδος.Έπειτα από μια μεγάλη άνοδο , αλλάζει η τάση της μετοχής με αποτέλεσμα η Signal Line να προσπερνά τον MACD.Σε εκείνη την περίπτωση θα κλείναμε την θέση μας με κέρδος. Στην περίπτωση δε που παίρναμε ταυτόχρονα θέση πώλησης θα οδηγούμασταν εκ νέου σε κέρδος, καθότι σε δεύτερο χρόνο θα κλείναμε ξανά την θέση μας.

Algorithmic Trading: ROC

- Ο δείκτης ROC(Rate Of Change) μας δείχνει την ταχύτητα με την οποία αλλάζει μια μεταβλητή σε μια συγκεκριμένη χρονική περίοδο. Υπολογιστικά είναι ένας απλός δείκτης ο οποίος μας δείχνει το momentum μιας μετοχής μέσα στην πάροδο του χρόνου.
- Για παράδειγμα μια μετοχή με καλό momentum(ορμή) έχει μεγάλο δείκτη ROC, το οποίο είναι ενδειξη για θέση αγοράς. Αντίστοιχα όταν έχουμε χαμηλό η και αρνητικό δείκτη ROC αποτελεί sell signal για έναν επενδυτή.

$$ROC = \frac{CP_t - CP_{t-n}}{CP_{t-n}} \times 100$$

CP_t:Τιμή κλεισίματος την περίοδο t

CP_{t-1}:Τιμή κλεισίματος πριν από n περιόδους

Algorithmic Trading: ROC

- Ο χρόνος των περίοδών είναι ιδιαίτερα σημαντικός για την εφαρμογή αυτής της τεχνικής. Γενικά μπορούμε να δούμε ROC 9,12,25 έως και 200 ημερών. Αυτή η επιλογή εξαρτάται από τον Trader. Ένας short-term Trader θα επιλέξει μια μικρή περίοδο για n και αντίστοιχα ο long-Term Trader θα επιλέξει μια μεγάλη περίοδο μέχρι και 200 ημέρων.
- Βέβαια αυτή η τεχνική εξαρτάται από το security που διαπραγματευόμαστε.Σε περίπτωση που εφαρμόζουμε αυτήν την τεχνική σε ένα Crypto για παράδειγμα, λόγω μεγάλης διακύμανσης των τιμών(volatility) ένας μικρός ROC 5 ημερών είναι πιθανό να μας δώσει λανθασμένα σήματα αγοράς η πώλησης

Algorithmic Trading: ROC

- Ευκαιρία να δούμε πως αυτός ό δείκτης θα μπορούσε να φανεί χρήσιμος στο algorithmic Trading.Θα εφαρμόσουμε αυτήν την τεχνική με δύο διαφορετικά n 110 & 9 ημερών και συγκεκριμένα θα δούμε την εφαρμογή του στο Bitcoin
- Θα παρατηρήσετε πως ο ROC(110) αντιδρά καλύτερα σε μελλοντικές τιμές του Bitcoin έναντι του ROC(9).Όπως αναφέραμε το volatility των cryptos είναι ιδιαίτερα σημαντική παράμετρος στο Algorithmic Trading.
- Στα Cryptos ο Trader πρέπει είναι πολύ προσεκτικός σχετικά με την τεχνική που θα εφαρμόσει.

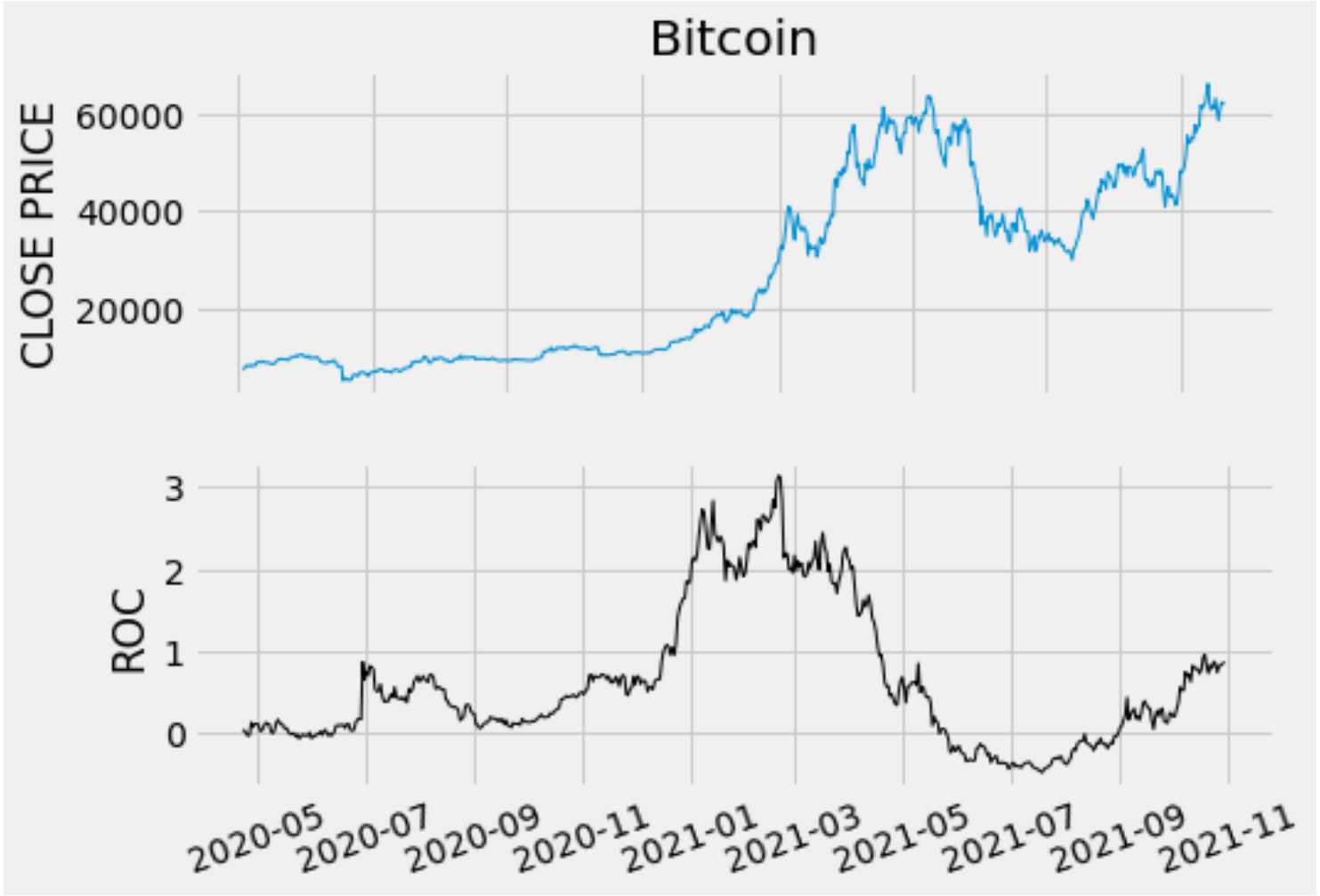
Algorithmic Trading: ROC

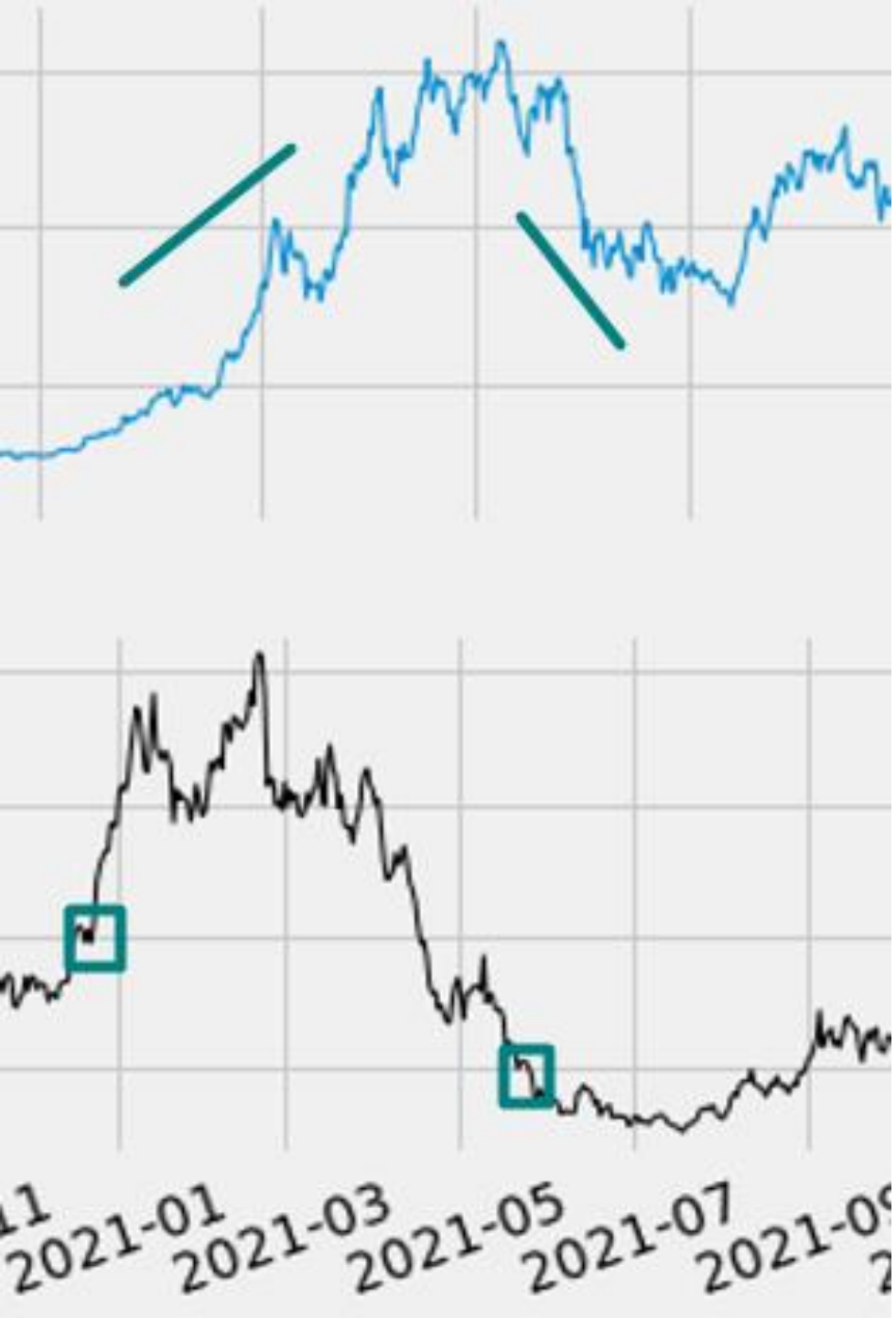
```
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
n=110
BTC=web.DataReader('BTC-USD',data_source='yahoo',start='1/4/2020',end='29/10/2021')
BTC=pd.DataFrame(BTC)
ROC=pd.Series(BTC['Close'].diff(n)/BTC['Close'].shift(n),name='ROC')
BTC=BTC.join(ROC)
```

Algorithmic Trading: ROC

```
fig=plt.figure(figsize=(7,5))
ax=fig.add_subplot(2,1,1)
ax.set_xticklabels([])
plt.plot(data['Close'],lw=1)
plt.title('Bitcoin')
plt.ylabel('CLOSE PRICE')
plt.grid(True)
bx=fig.add_subplot(2,1,2)
plt.plot(ROC,'k',lw=1,linestyle='solid',label='ROC')
plt.ylabel('ROC')
plt.grid(True)
plt.setp(plt.gca().get_xticklabels(),rotation=20)
plt.show()
```

Algorithmic Trading: ROC(110)





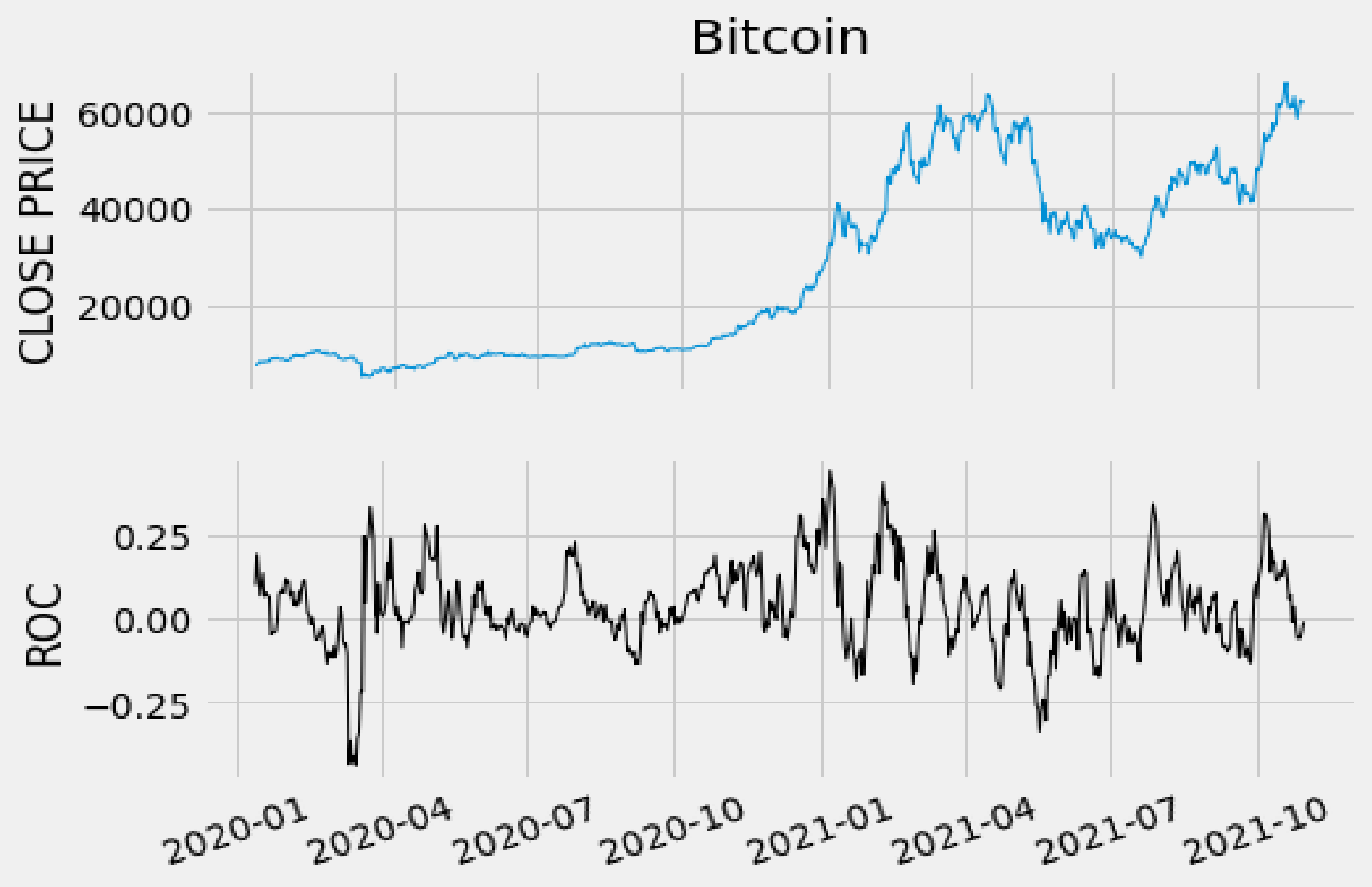
Algorithmic Trading: ROC(110)

Βλέπουμε με το διάγραμμα μας ότι πράγματι ο ROC 110 ημερών «εντοπίζει» την μελλοντική τάση του Bitcoin ιδιαιτέρα ικανοποιητικά.

Ας εστιάσουμε λίγο παραπάνω. Παρατηρήστε πως την στιγμή που ο ROC περνάει το 1 , τότε το Bitcoin αποκτά μια ανοδική τάση στην τιμή του. Αντίστοιχα όταν το ROC περνάει κάτω του 0 αναμένουμε πτώση της τιμής και κάτ. επέκτασιν θα μπαίναμε σε θέση πώλησης. Πράγματι παρόλου που η τιμή του Bitcoin είναι ανοδική βλέπουμε πως ο ROC έκανε μια πολύ καλή πρόβλεψη καθώς υπήρξε πτώση στην τιμή του κρυπτονομίσματος μας

Τώρα ας δούμε τον ROC(9)

Algorithmic Trading: ROC(9)



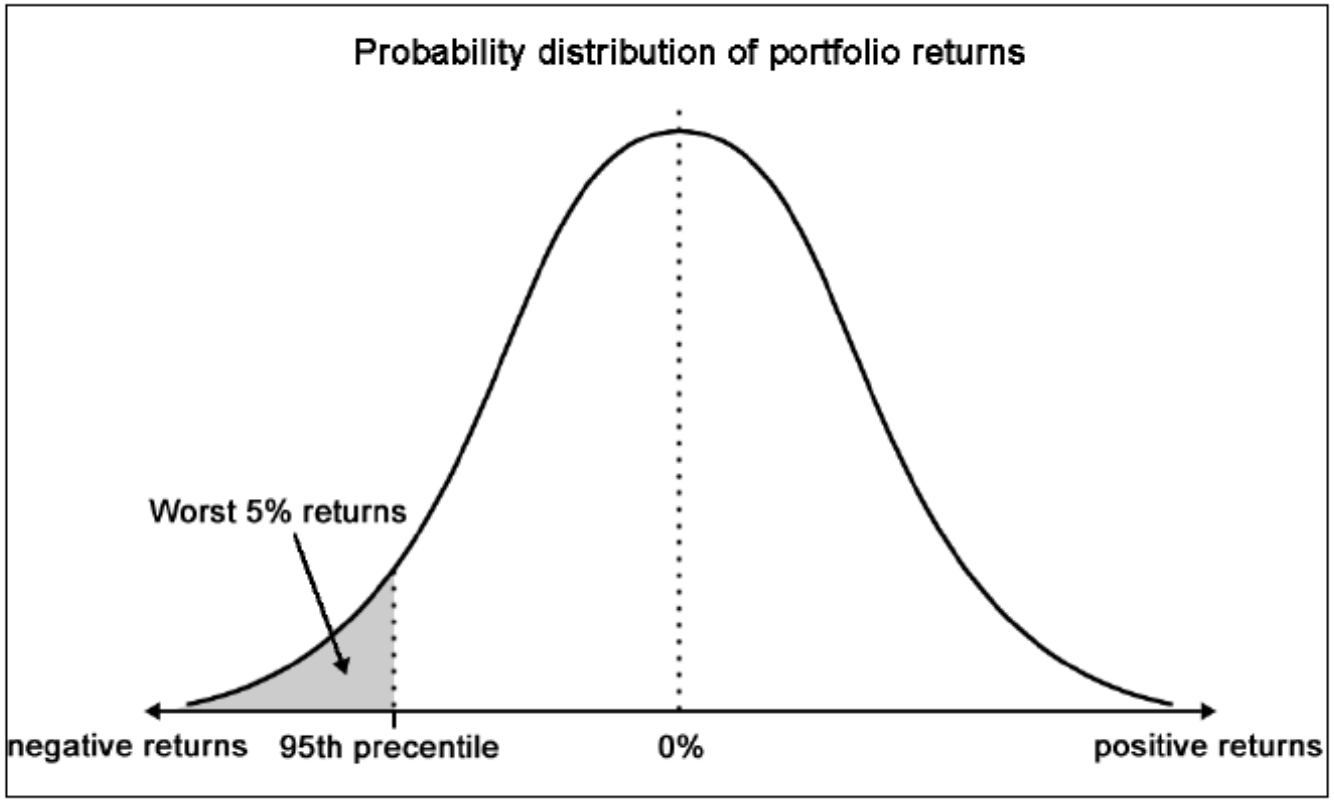
Value at Risk (VaR)

Γενικά ένας επενδυτής εκτίθεται σε πολλαπλούς κινδύνους όπως κίνδυνος μεταβλητότητας η σε κάποιον χρηματοπιστωτικό κίνδυνο.

Ένας από τους πιο διάσημους δείκτες είναι το VaR, ο οποιος μας δείχνει ποια θα είναι η μέγιστη δυνατή απώλεια κεφαλαίων ,σε ένα συγκεκριμένο διάστημα εμπιστοσύνης (π.χ 95%) στην πάροδο του χρόνου.

Επίσης σαν δείκτης μπορεί να εφαρμοστεί σε διαφορετικά επίπεδα από μια απλή επένδυση μέχρι και σε ένα σύνολο χαρτοφυλακίου.

Value at Risk (VaR)



Value at Risk (VaR)

```
import datetime as dt
import numpy as np
import pandas_datareader as rda
from scipy.stats import norm
def calculate_daily_VaR(P, prob, mean, sigma,days_per_year=252.):
    min_ret = norm.ppf(1-prob,
    mean/days_per_year,
    sigma/np.sqrt(days_per_year))
    return P - P*(min_ret+1)
```

Value at Risk (VaR)

```
start = dt.datetime(2020, 12, 1)
end = dt.datetime(2021, 12, 1)
prices = rda.DataReader("AAPL", "yahoo", start, end)
returns = prices["Adj Close"].pct_change().dropna()
portvolio_value = 100000000.00
confidence = 0.95
mu = np.mean(returns)
sigma = np.std(returns)
VaR = calculate_daily_VaR(portvolio_value, confidence,mu, sigma)
print("Value-at-Risk:", round(VaR, 2))
```

Portfolio Risk

Κατά την σύνταξη του χαρτοφυλακίου βρισκόμαστε αντιμέτωποι με τους εξής είδους κίνδυνου :

Συστημικός κίνδυνος : Με τον όρο συστημικό κίνδυνο ορίζουμε τον αναπότρεπτο κίνδυνο ο οποίος μπορεί να επηρεάσει την επένδυση μας. Είναι γνωστός και σαν κίνδυνος αγοράς η μη διαφοροποιησιμος κίνδυνος.

Γενικά επηρεάζει ολοκληρη την αγορα καθως εξαρτάται από την οικονομία μιας χώρας, τον πληθωρισμό ,πολέμους κ.α.

Ανεξαρτήτως της σύστασης του χαρτοφυλακίου μας θα υπάρχει πάντα συστημικός κίνδυνος.

Portfolio Risk

Μη Συστημικός κίνδυνος : Μη συστημικός κίνδυνος η διαφοροποίησης κίνδυνος ορίζεται αυτός ο οποίος επηρεάζει μόνο το αξιόγραφο και όχι ολόκληρη την αγορά

Ο επενδυτής είναι σε θέση να διαφοροποιήσει τον κίνδυνο στο χαρτοφυλάκιο με πολλαπλά αξιόγραφα από διαφορετικούς κλάδους.

Για παράδειγμα η πτώση της μετοχής μια εταιρίας δεν επηρεάζει τον κλάδο δραστηριότητας. Ομως πιθανόν ο κλάδος να υποστεί πλήγμα λόγω μια συγκυρίας

Γενικά ένα πλήρως διαφοροποιήσημο χαρτοφυλάκιο είναι αυτό που περιέχει πολλαπλά αξιόγραφα, και ιδανικά από διαφορετικούς κλάδους.

Portfolio Risk

Από πόσα αξιόγραφα πρέπει να αποτελείται το χαρτοφυλάκιο μας ώστε να να είναι πλήρως διαφοροποιημένο σχετικά με έναν κλάδο;

Ο Statman(1987) σε μια δημοσίευση του υποστηρίζει ότι χρειαζόμαστε τουλάχιστον 30.Θα χρησιμοποιήσουμε τα δεδομένα της δημοσίευσης του, όπου βασίζεται στην σχέση ανάμεσα στο πλήθος των αξιογράφων(n),την ετήσια τυπική απόκλιση της απόδοσης του χαρτοφυλακίου $\overline{\sigma}_p$, καθώς και την μέση τιμή της τυπικής απόκλισης ενός χαρτοφυλακίου με μία μετοχή $\overline{\sigma}_1$

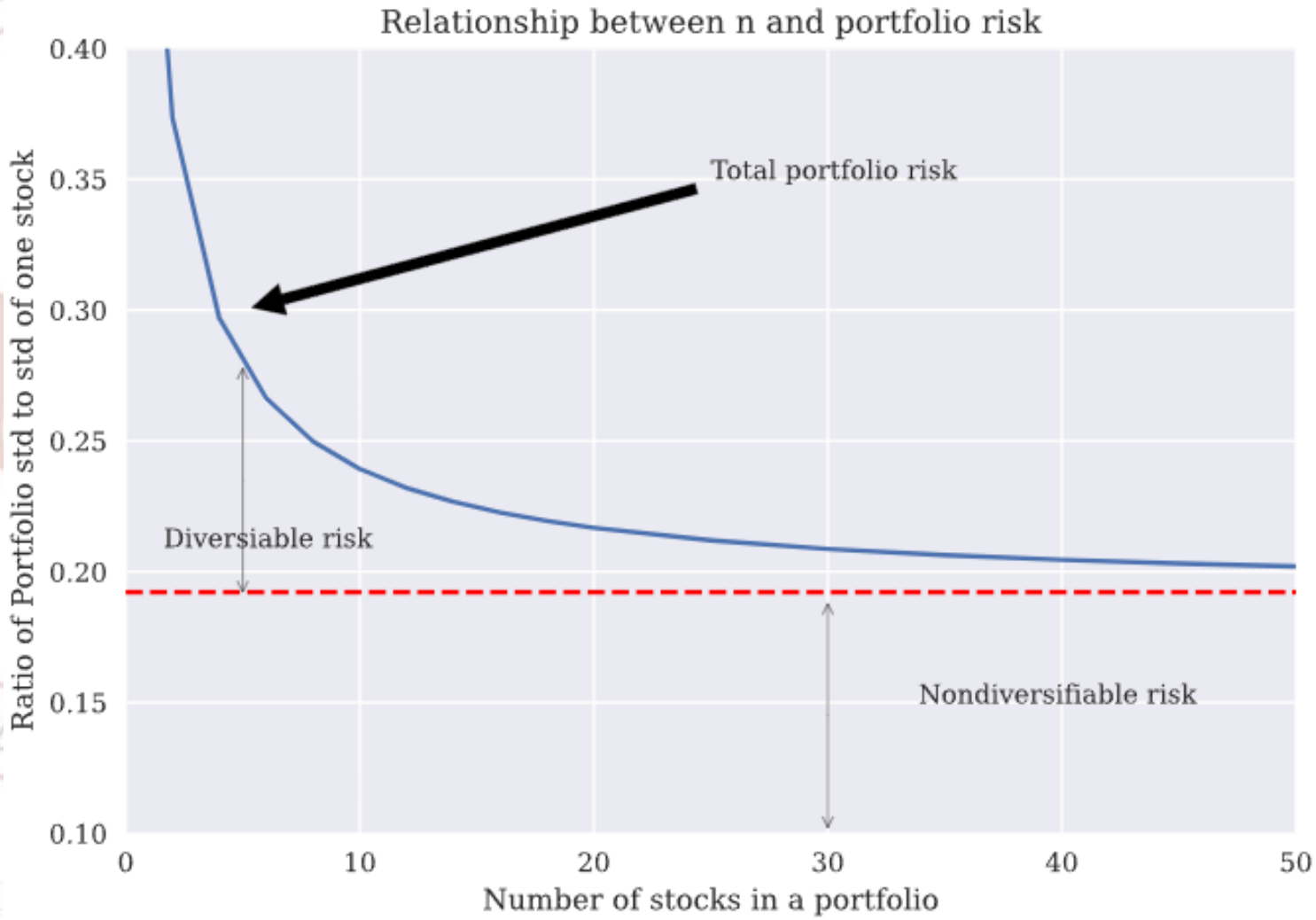
Portfolio Risk

n	$\bar{\sigma}_p$	$\frac{\bar{\sigma}_p}{\bar{\sigma}_1}$	n	$\bar{\sigma}_p$	$\frac{\bar{\sigma}_p}{\bar{\sigma}_1}$
1	49.236	1.00	45	20.316	0.41
2	37.358	0.76	50	20.203	0.41
4	29.687	0.60	75	19.860	0.40
6	26.643	0.54	100	19.686	0.40
8	24.983	0.51	200	19.432	0.39
10	23.932	0.49	300	19.336	0.39
12	23.204	0.47	400	19.292	0.39
14	22.670	0.46	500	19.265	0.39
16	22.261	0.45	600	19.347	0.39
18	21.939	0.45	700	19.233	0.39
20	21.677	0.44	800	19.224	0.39
25	21.196	0.43	900	19.217	0.39
30	20.870	0.42	1000	19.211	0.39
35	20.634	0.42	∞	19.158	0.39
40	20.456	0.42			

Portfolio Risk

- from matplotlib.pyplot import *
- n=[1,2,4,6,8,10,12,14,16,18,20,25,30,35,40,45,50,75,100,200,300,400,500,600,700,800,900,1000]
- port_sigma=[0.49236,0.37358,0.29687,0.26643,0.24983,0.23932,0.23204,0.22670,0.22261,0.21939,0.21677,0.21196,0.20870,
  0.20634,0.20456,0.20316,0.20203,0.19860,0.19686,0.19432,0.1933
  6,0.19292,0.19265,0.19347,0.19233,0.19224,0.19217,0.19211,0.19158]
- xlim(0,50)
- ylim(0.1,0.4)
- hlines(0.19217, 0, 50, colors='r', linestyles='dashed')
- annotate('', xy=(5, 0.19), xycoords = 'data',xytext = (5, 0.28),
- textcoords = 'data',arrowprops = {'arrowstyle':'<->'})
- annotate('', xy=(30, 0.19), xycoords = 'data',xytext = (30, 0.1),
- textcoords = 'data',arrowprops = {'arrowstyle':'<->'})
- annotate('Total portfolio risk', xy=(5,0.3),xytext=(25,0.35),
- arrowprops=dict(facecolor='black',shrink=0.02))
- figtext(0.15,0.4,"Diversiable risk")
- figtext(0.65,0.25,"Nondiversifiable risk")
- plot(n[0:17],port_sigma[0:17])

Portfolio Risk



Sharpe Ratio

Ο βασικότερος δείκτης αποδοτικότητας του χαρτοφυλακίου μας είναι ο Sharpe Ratio(William F. Sharpe-1966) ο οποίος μετρά την απόδοση ενός περιουσιακού στοιχείου όπως χρεόγραφο η χαρτοφυλάκιο σε συγκριση με ένα Risk Free περιουσιακό στοιχείο (π.χ. ομόλογα)

Ουσιαστικά αξιολογεί την απόδοση της επένδυσης για τον κίνδυνο που έχει αναλάβει ο επενδυτής

Ο υπολογισμός του είναι αρκετά απλός και τον καθιστά δημοφιλή στους επενδυτές

Τιμή μεγαλύτερη του 1 τον καθιστά αποδεκτό από τους επενδυτές. Μια τιμή μεγαλύτερη απο 2 θεωρείται πολύ καλή και μεγαλύτερη από 3 εξαιρετική

Για τιμές μικρότερες του 1 πιθανόν ο επενδυτής λαμβάνει περισσότερο κίνδυνο απ' όσο αξίζει η επένδυση του

Sharpe Ratio

Ο υπολογισμός του Sharpe Ratio ορίζεται ώς εξής:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

R_p:Απόδοση χαρτοφυλακίου

R_f:Απόδοση Risk-Free περιουσιακού στοιχείου

σ_p:Τυπική απόκλιση χαρτοφυλακίου

Sharpe Ratio

Τώρα θα υπολογίσουμε το Sharpe Ratio ενός χαρτοφυλακίου που θα αποτελείται από τις εξής 4 μετοχές : Apple , Ford, Google και Wallmart. Θα θεωρήσουμε πως η συστάση του χαρτοφυλακίου είναι ισόποση 25% .

```
import pandas_datareader as web
import numpy as np
```

- data=web.DataReader(['AAPL', 'F', 'GOOGL', 'WMT'], 'yahoo',"2020-01-01", "2021-01-01")['Adj Close’]
- portfolio = [.25, .25, .25, .25]

Ξεκινάμε με την άντληση των τιμών μέσω της web.DataReader και ορίζουμε το χαρτοφυλάκιο μας με 0.25 συντελεστή για την κάθε μετοχή.

Sharpe Ratio

- logret = np.sum(np.log(data/data.shift())*portfolio, axis=1)

#Αρχικα υπολογίζουμε τα logreturns και τα πολλαπλασίαζουμε με το εκάστοτε βάρος

- SR = (logret.mean())/logret.std()*np.sqrt(252)

Υπολογίζουμε τον Sharpe Ratio στο παράδειγμα μας θεωρούμε πως δεν έχουμε κάποιο Risk Free περιουσιακό στοιχείο.Τέλος με το np.sqrt(252) υπολογίζουμε την ετήσια τιμή του δείκτη

- SR

Sharpe Ratio

Επίσης θα μπορούσαμε να υπολογίσουμε τον Sharpe Ratio της κάθε μετοχής ξεχωριστά

- tickers=['AAPL', 'F', 'GOOGL', 'WMT']
- logret = np.log(data/data.shift())
- SR = (logret.mean()/logret.std())*np.sqrt(252)
- SR

Portfolio Diversification

Στην χρηματοοικονομική μπορούμε να αφαιρέσουμε έναν συγκεκριμένο κίνδυνο κλάδου συνδυάζοντας διαφορετικές μετοχές στο χαρτοφυλάκιο μας.Ας δουμε το ακόλουθο παράδειγμα , οπου έχουμε τις ετήσιες αποδόσεις για 2 μετοχές A & B

Year	Stock A	Stock B
2009	0.102	0.1062
2010	-0.02	0.23
2011	0.213	0.045
2012	0.12	0.234
2013	0.13	0.113

Portfolio Diversification

1. A=[0.102,-0.02, 0.213,0.12,0.13]
2. B=[0.1062,0.23, 0.045,0.234,0.113]
3. port_EW=(np.array(A)+np.array(B))/2
4. round(np.mean(A),3),round(np.mean(B),3),round(np.mean(port_EW),3)
5. round(np.std(A),3),round(np.std(B),3),round(np.std(port_EW),3)

Παρατηρούμε πως το χαρτοφυλάκιο μας μπορεί να μας παρέχει ελαφρώς μικρότερη μέση απόδοση από την μετοχή Β (12.7% έναντι 14.6%), όμως επίσης έχουμε μικρότερη τυπική απόκλιση(volatility).Συγκεκριμένα το χαρτοφυλάκιο μας παρέχει 2.7% και οι μετοχές Α και Β 7.5% και 7.4% αντίστοιχα. Τώρα ας το συγκρίνουμε οπτικά

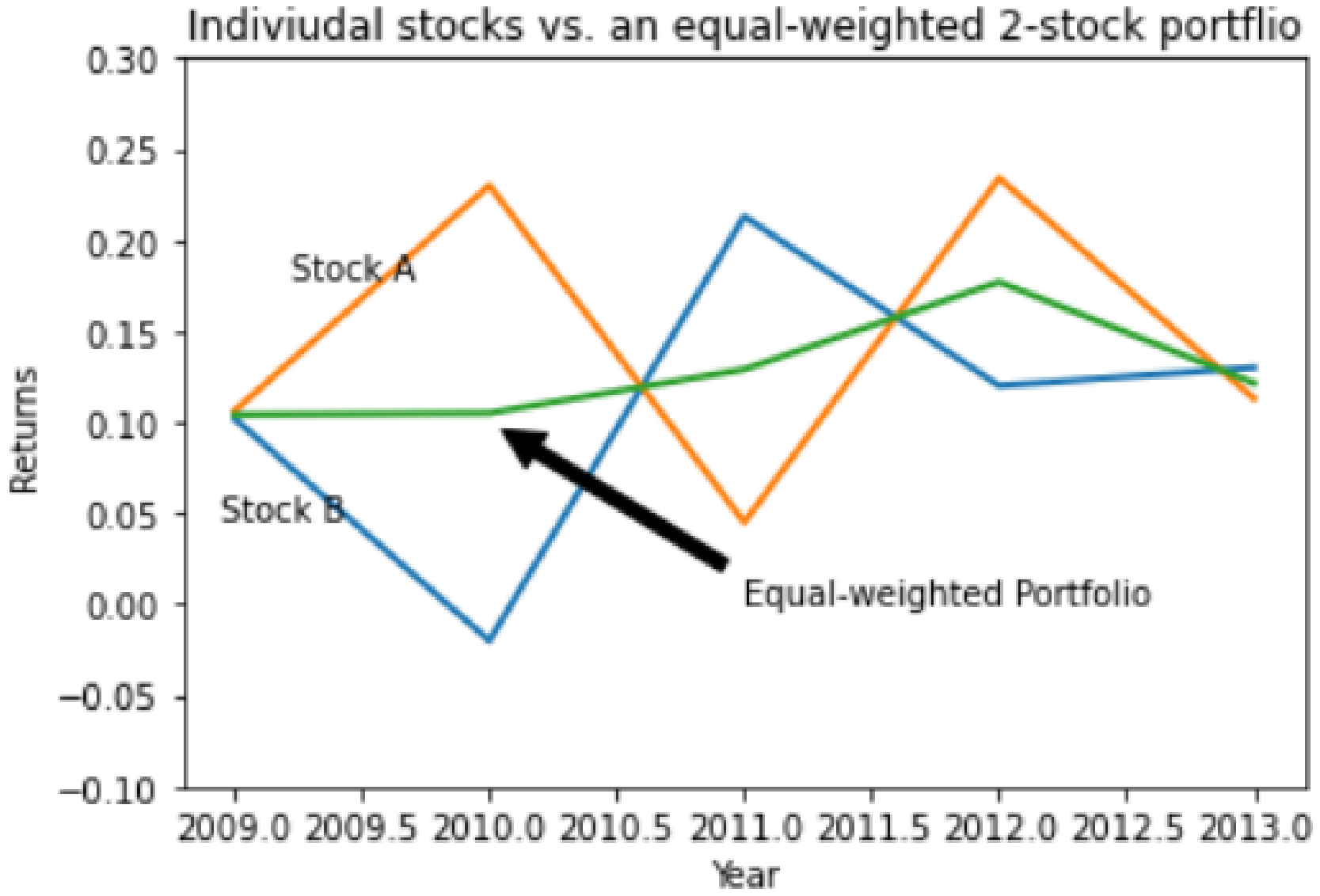
Portfolio Diversification

- import numpy as np
- import matplotlib.pyplot as plt
- year=[2009,2010,2011,2012,2013]
- ret_A=[0.102,-0.02, 0.213,0.12,0.13]
- ret_B=[0.1062,0.23, 0.045,0.234,0.113]
- port_EW=(np.array(ret_A)+np.array(ret_B))/2
- plt.figtext(0.2,0.65,"Stock A")
- plt.figtext(0.15,0.4,"Stock B")
- plt.xlabel("Year")
- plt.ylabel("Returns")

Portfolio Diversification

- plt.plot(year,ret_A,lw=2)
- plt.plot(year,ret_B,lw=2)
- plt.plot(year,port_EW,lw=2)
- plt.title("Indiviudal stocks vs. an equal-weighted 2-stock Portfolio")
- plt.annotate('Equal-weighted Portfolio', xy=(2010, 0.1), xytext=(2011.,0)
  ,arrowprops=dict(facecolor='black',shrink=0.05),)
- plt.ylim(-0.1,0.3)
- plt.show()

Portfolio Diversification



Portfolio Optimization

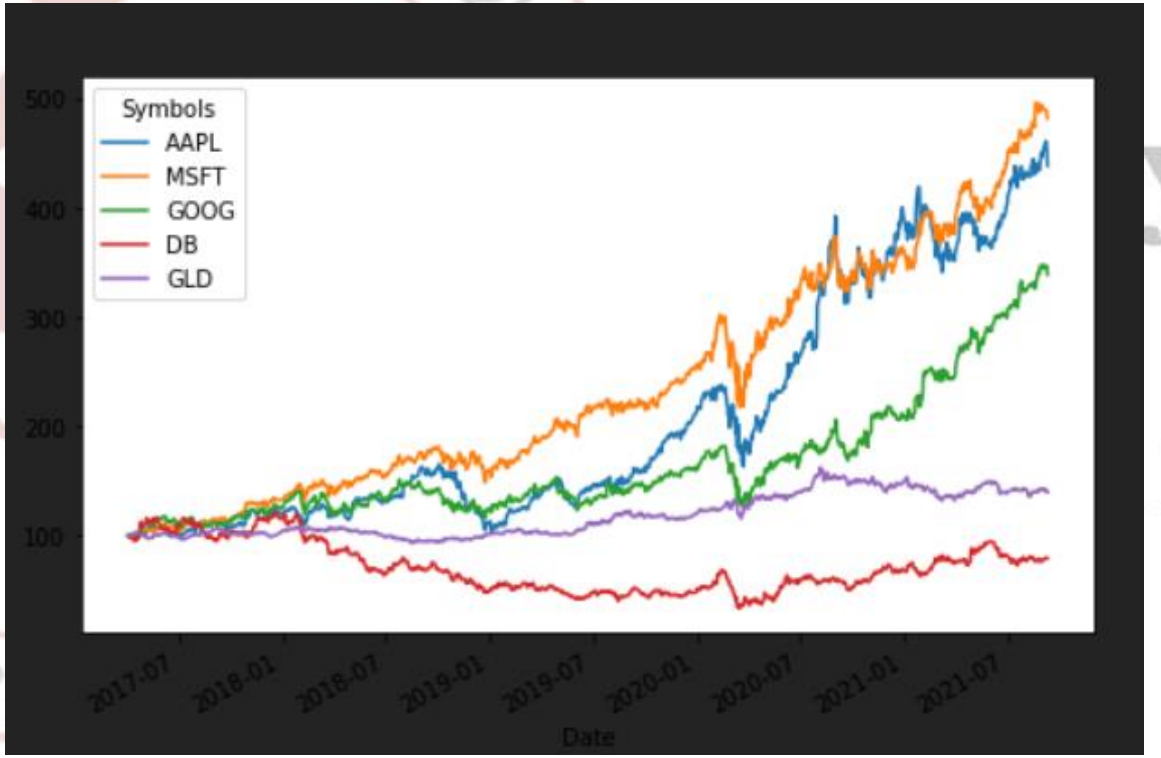
```
import numpy as np
import pandas as pd
import pandas_datareader as web
import matplotlib.pyplot as plt
%matplotlib inline
symbols= ['AAPL', 'MSFT', 'GOOG', 'DB', 'GLD']
data = web.DataReader(symbols, data_source='yahoo',start='2017-04-01',end='2021-09-12')['Adj Close’]
```

Για το παράδειγμα μας δημιουργήσαμε ένα χαρτοφυλάκιο με περιουσιακά στοιχεία από διαφορετικό τομέα , με σκοπό να επιτύχουμε ένα διαφοροποιημένο χαρτοφυλάκιο.

Portfolio Optimization

Κανονικοποιούμε τα δεδομένα μας και τα αναπαριστούμε γραφικά ως εξής :

```
(data / data.iloc[0] * 100).plot(figsize=(8, 5))
```



Portfolio Optimization

Η θεωρία Mean Variance αναφέρεται στα log returns των αξιόγραφων μας τα οποία υπολογίζονται ως εξής

```
rets = np.log(data / data.shift(1))
```

Επειτα υπολογίζουμε την ετήσια μέση τους απόδοση. Καθότι οι μέρες συναλλαγών το έτος είναι 252 θα εφαρμόσουμε την επόμενη σειρά κώδικα

```
rets.mean() * 252
```

Με τον ιδιο τρόπο θα υπολογίσουμε τον πίνακα συνδιακύμανσης των περιουσιακών μας στοιχείων

```
rets.cov() * 252
```

Portfolio Optimization

Η βασική υπόθεση της θεωρίας του MVT(Mean Variance Theory) είναι πως ο επενδυτής θα παίρνει θέση αγοράς και όχι πώλησης στα αξιόγραφα του χαρτοφυλακίου του. Οπότε το κεφάλαιο θα διαιρεθεί ανάμεσα στις 5 μετοχές που επιλέξαμε. Θα μπορούσε κάποιος να επιλέξει ισόποσα(π.χ 20% του κεφαλαίου στην κάθε μετοχή).Με την χρήση του επόμενου κώδικα παίρνουμε 5 τυχαία βάρη για την κάθε μετοχή

- weights = np.random.random(noa) ορίζουμε 5 τυχαία βάρη
- weights /= np.sum(weights) τα κανονικοποιούμε ώστε το άθροισμα τους να ισούται με 1
- weights

Portfolio Optimization

Με την επόμενη εντολή παίρνουμε την αναμενόμενη ετήσια απόδοση του χαρτοφυλακίου μας

```
np.sum(rets.mean() * weights) * 252
```

Το επόμενο βήμα είναι να βρούμε την ετήσια διακύμανση του χαρτοφυλακίου μας

```
np.dot(weights.T, np.dot(rets.cov() * 252, weights))
```

Τέλος η τυπική απόκλιση του χαρτοφυλακίου με την χρήση της np.sqrt

```
np.sqrt(np.dot(weights.T, np.dot(rets.cov() * 252, weights)))
```

Portfolio Optimization

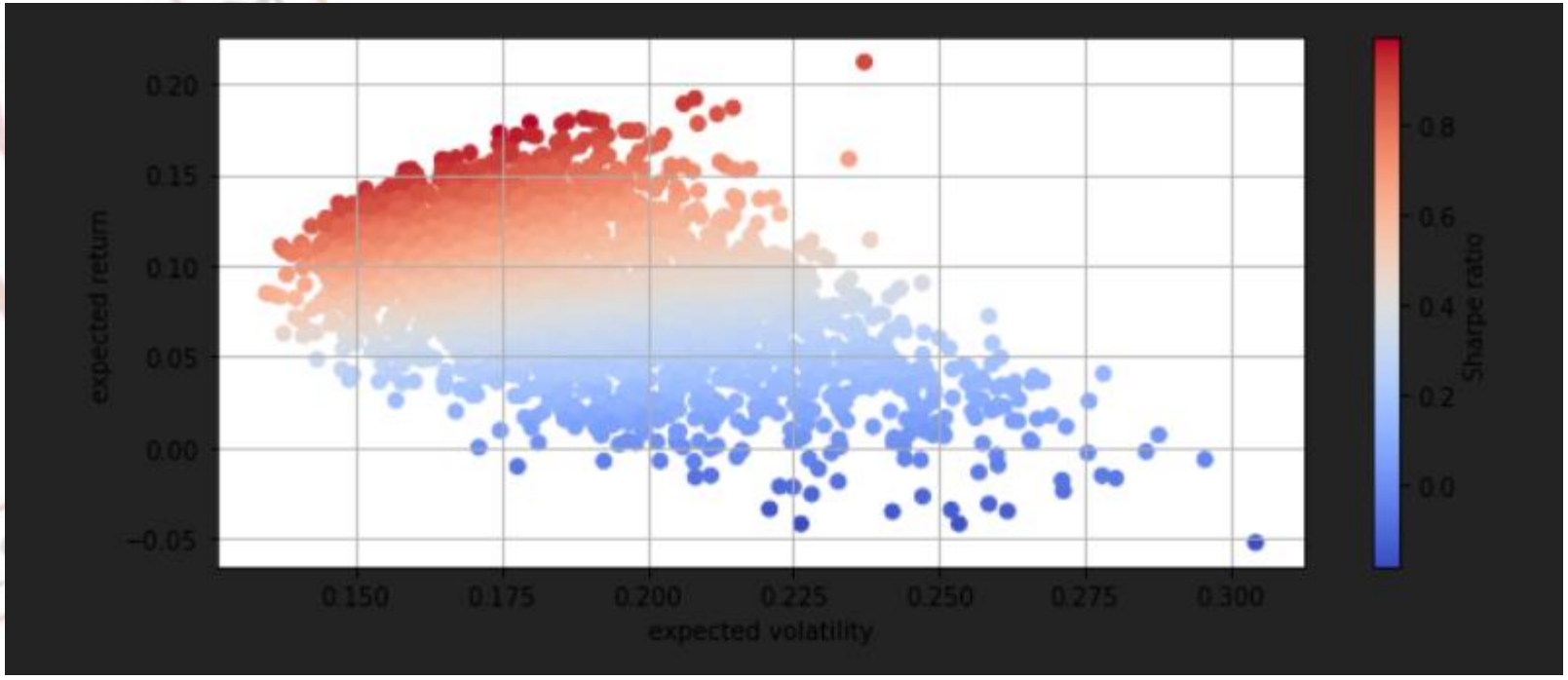
Στο επόμενο βήμα μας θα εφαρμόσουμε μια προσομοίωση Monte Carlo με σκοπό να δημιουργήσουμε πολλαπλά διαφορετικά χαρτοφυλάκια

```
prets = []                      #Ξεκινάμε με μια κενή λίστα με την απόδοση του χαρτοφυλακίου μας
pvols = []                      #Επίσης ορίζουμε κενή λίστα για τις τυπικές αποκλείσεις μας
for p in range (2500):
        weights = np.random.random(noa)
        weights /= np.sum(weights)
        prets.append(np.sum(rets.mean() * weights) * 252)
        pvols.append(np.sqrt(np.dot(weights.T,
        np.dot(rets.cov() * 252, weights))))
prets = np.array(prets)         [Πρακτικά εφαρμόσαμε αυτά που είδαμε στα προηγούμενα slides για 2500 επαναλήψεις]
pvols = np.array(pvols)
```

Portfolio Optimization

```
plt.figure(figsize=(8, 4))
plt.scatter(pvols, prets, c=prets / pvols, marker='o')  το c αποτελεί το Sharpe Ratio
plt.grid(True)
plt.xlabel('expected volatility')
plt.ylabel('expected return')
plt.colorbar(label='Sharpe ratio')
```

Στο διπλανό σχήμα, βλέπουμε όλους τους πιθανούς συνδυασμούς χαρτοφυλακίων που μπορούμε να συγκροτήσουμε με τις μετοχές που έχουμε επιλέξει.



Portfolio Optimization

Τώρα ήρθε η ώρα να εφαρμόσουμε τεχνικές βελτιστοποίησης του χαρτοφυλακίου μας. Αρχικά θα ορίσουμε την ακόλουθη συνάρτηση για δική μας ευκολία :

```
def statistics(weights):
    weights = np.array(weights)
    pret = np.sum(rets.mean() * weights) * 252
    pvol = np.sqrt(np.dot(weights.T, np.dot(rets.cov() * 252, weights)))
    return np.array([pret, pvol, pret / pvol])
```

Portfolio Optimization

Η εύρεση του βέλτιστου χαρτοφυλακίου είναι ένα πρόβλημα βελτιστοποίησης.Για αυτόν τον σκοπό θα κάνουμε χρήση της minimize από την βιβλιοθήκη scipy.optimize όπως εξής

```
import scipy.optimize as sco
```

Τωρα για το επόμενο βήμα μας θα ασχοληθούμε με την μεγιστοποίηση του Sharpe Ratio

```
def min_func_sharpe(weights):
    return -statistics(weights)[2]
```

Portfolio Optimization

Έπειτα, ορίζουμε τις κατάλληλες παραμέτρους για να εφαρμόσουμε την βελτιστοποίηση μας

```
cons = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
bnds = tuple((0, 1) for x in range(noa)) #
noa * [1. / noa,]  #Ορίζουμε καποια τυχαία αρχικά βάρη έστω ισόποσα
opts = sco.minimize(min_func_sharpe, noa * [1. / noa,],
method='SLSQP',bounds=bnds, constraints=cons)
opts['x'].round(3)
statistics(opts['x']).round(3)
```

Portfolio Optimization

Στο επόμενο βήμα μας ενδιαφέρει να βρούμε το χαρτοφυλάκιο με την μικρότερη διακύμανση

```
def min_func_variance(weights):
    return statistics(weights)[1] ** 2
optv = sco.minimize(min_func_variance, noa * [1. / noa,],method='SLSQP', bounds=bnds,
constraints=cons)
optv['x'].round(3)
statistics(optv['x']).round(3)
```

Portfolio Optimization-Efficient Frontier

Αποτελεσματικό σύνορο (Efficient Frontier) είναι ο συνδιασμός όλων εκείνων των χαρτοφυλακίων που μας προσφέρουν την μέγιστη δυνατή απόδοση για ένα συγκεκριμένο επίπεδο κινδύνου ή τον ελάχιστο κίνδυνο για μια συγκεκριμένη απόδοση.

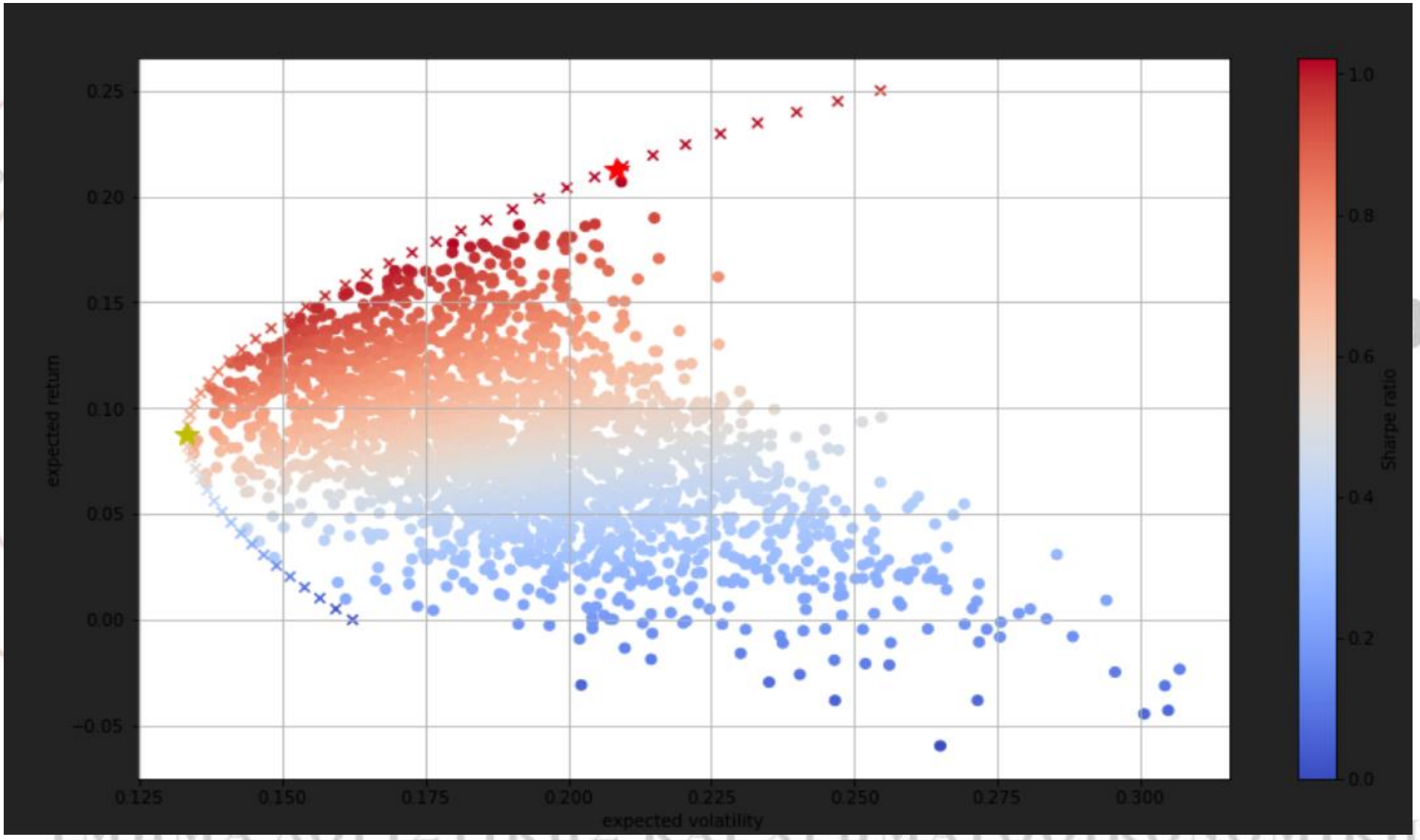
Portfolio Optimization-Efficient Frontier

```
cons = ({'type': 'eq', 'fun': lambda x: statistics(x)[0] - tret},{'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
bnds = tuple((0, 1) for x in weights)
def min_func_port(weights):
    return statistics(weight[1])
trets = np.linspace(0.0, 0.25, 50)
tvols = []
for tret in trets:
    cons = ({'type': 'eq', 'fun': lambda x: statistics(x)[0] - tret},
    {'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
    res = sco.minimize(min_func_port, noa * [1. / noa,], method='SLSQP',bounds=bnds, constraints=cons)
    tvols.append(res['fun'])
tvols = np.array(tvols)s)[1]
```

Portfolio Optimization-Efficient Frontier

```
plt.figure(figsize=(14, 8))
plt.scatter(pvols, prets,c=prets / pvols, marker='o')
# random portfolio composition
plt.scatter(tvols, trets,c=trets / tvols, marker='x')
# efficient frontier
plt.plot(statistics(opts['x'])[1], statistics(opts['x'])[0],'r*', markersize=15.0)
# portfolio with highest Sharpe ratio
plt.plot(statistics(optv['x'])[1], statistics(optv['x'])[0],'y*', markersize=15.0)
# minimum variance portfolio
plt.grid(True)
plt.xlabel('expected volatility')
plt.ylabel('expected return')
plt.colorbar(label='Sharpe ratio')
```

Portfolio Optimization-Efficient Frontier



CAPM-Capital Asset Pricing Mode

Γενικά υπάρχουν αξιόγραφα τα οποία εμπεριέχουν κάποιον επενδυτικό κίνδυνο. Τα μετρητά ή οι λογαριασμοί μετρητών σε κάποιον τραπεζικό λογαριασμό αποτελούν ένα είδος επένδυσης με σχεδόν μηδενικό κίνδυνο.

Βέβαια το αρνητικό σε μια τέτοια επένδυση είναι πως μας επιφέρει μια παρα πολύ μικρή απόδοση, κάποιες φορές σχεδόν μηδαμινή.

Ωστόσο κάποιος επενδυτής μπορεί να επιλέξει κάποιο χαρτοφυλάκιο που αποτελείται από “risky” περιουσιακά στοιχεία και έπειτα να συνδυάσει ταυτόχρονα το “non-risky” περιουσιακό του στοιχείο.

Με αυτόν το τρόπο, ο επενδυτής είναι σε θέση να αναπροσαρμόσει τα ποσοστά της επένδυσης του με τέτοιον τρόπο, ώστε να επιλέξει το βέλτιστο χαρτοφυλάκιο όπου εφάπτεται με το Efficient frontier

CAPM-Capital Asset Pricing Mode

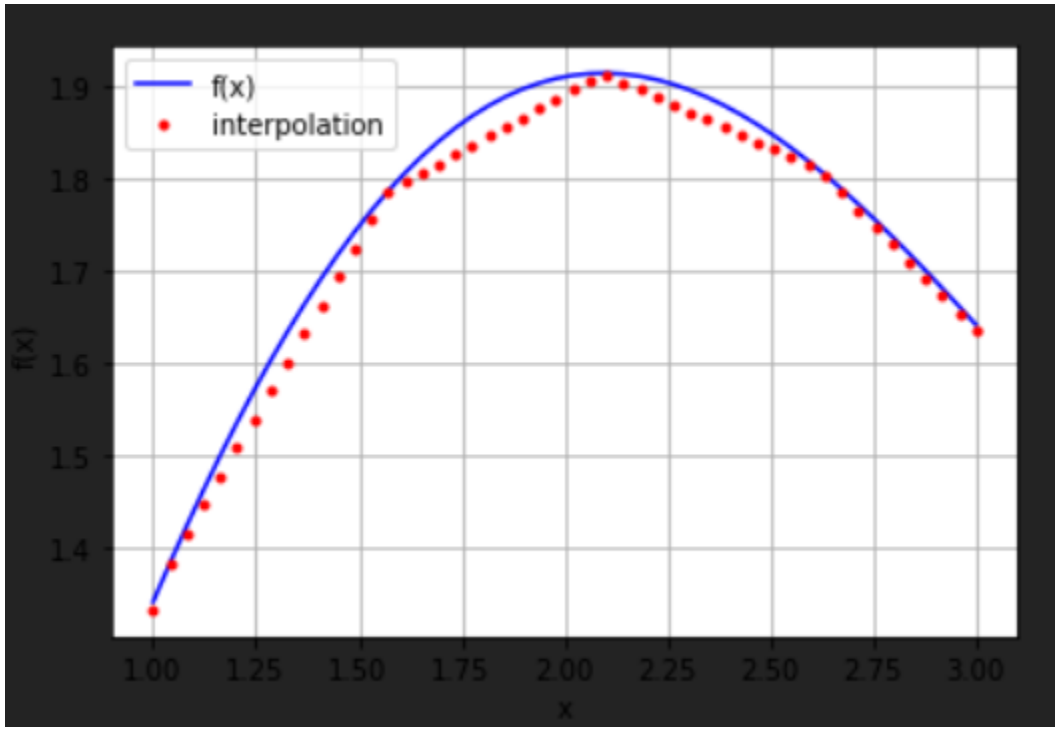
Πρώτού περάσουμε στο προγραμματιστικό κομμάτι του CAPM θεωρούμε σκόπιμο να γίνει μια μαθηματική ανάλυση σχετικά με τις τεχνικές που θα ακολουθήσουμε. Καθότι δεν είναι σκοπός του μαθήματος, θα εστιάσουμε απλά στην βασική κατανόηση των μεθόδων όπου θα επακολουθήσουν

Αρχικά θα μιλήσουμε για την μέθοδο της γραμμικής παρεμβολής(Interpolation) η οποία αποτελεί έναν τύπο εκτίμησης. Η γενική ιδέα είναι, πώς εφαρμόζουμε ένα είδος «παλινδρόμησης» σε ένα διατεταγμένο σύνολο δεδομένων με σκοπό τόσο τα δεδομένα να προσαρμόζονται στην γραμμή καθώς και επίσης η συνάρτηση να είναι συνεχώς διαφοροποιήσιμη στα σημεία των δεδομένων μας.

Στις επόμενες διαφάνειες θα περιγράψουμε ενδεικτικά τον απαραίτητο κώδικα καθώς και τα αντίστοιχα διαγράμματα ώστε να αντιληφθούμε πώς λειτουργεί αυτός ο αλγόριθμος

CAPM-Capital Asset Pricing Mode

```
import scipy.interpolate as spi
x = np.linspace(-2 * np.pi, 2 * np.pi, 25)
def f(x):
    return np.sin(x) + 0.5 * x
ipo = spi.splrep(x, f(x), k=1)
iy = spi.splev(x, ipo)
xd = np.linspace(1.0, 3.0, 50)
iyd = spi.splev(xd, ipo)
plt.plot(xd, f(xd), 'b', label='f(x)')
plt.plot(xd, iyd, 'r.', label='interpolation')
plt.legend(loc=0)
plt.grid(True)
plt.xlabel('x')
plt.ylabel('f(x)')
```



CAPM-Capital Asset Pricing Mode

Παρατηρούμε πώς η γραμμή δεν προσαρμόζεται πλήρως στα δεδομένα.Αν στην συνάρτηση spi.splrep(x, f(x), k=1) αλλάξουμε την τιμή του k=3 (cubic splines) βλέπουμε πως έχουμε μια ιδιαίτερα καλή προσέγγιση για τα δεδομένα μας.

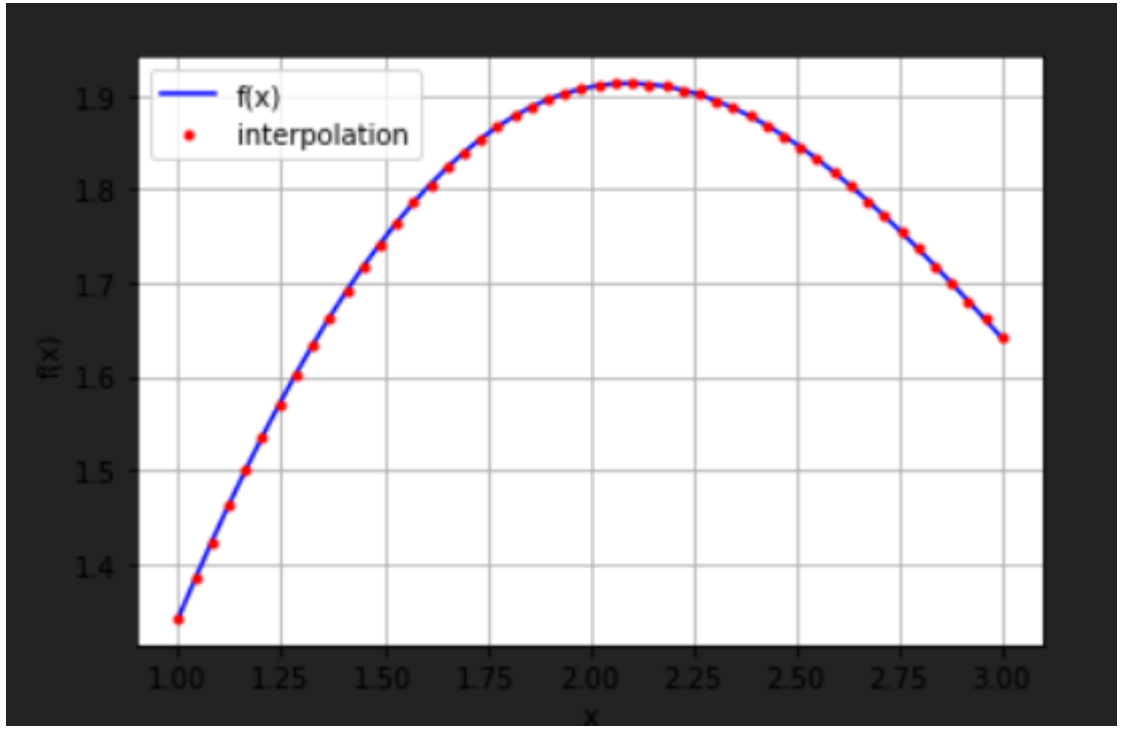
Για άλλη μια φορά σκοπός αυτής της (πάρα) πολύ σύντομής ανάλυσης του interpolation είναι η κατανόηση του αλγόριθμου που θα εφαρμόσουμε σχετικά με το efficient frontier

Για περισσότερες πληροφορίες μπορείτε να ανατρέξετε
https://docs.scipy.org/doc/scipy/reference/interpolate.html

Τέλος στην επόμενη διαφάνεια θα δούμε το νέο μας διάγραμμα για k=3

CAPM-Capital Asset Pricing Mode

- ipo = spi.splrep(x, f(x), k=3)
- iyd = spi.splev(xd, ipo)
- plt.plot(xd, f(xd), 'b', label='f(x)')
- plt.plot(xd, iyd, 'r.', label='interpolation')
- plt.legend(loc=0)
- plt.grid(True)
- plt.xlabel('x')
- plt.ylabel('f(x)')



CAPM

- import scipy.interpolate as sci
- ind = np.argmin(tvols)
- evols = tvols[ind:]
- erets = trets[ind:]
- tck = sci.splrep(evols, erets)
- def f(x):
  return sci.splev(x, tck, der=0)
- def df(x):
  return sci.splev(x, tck, der=1)

CAPM-Capital Asset Pricing Mode

Τώρα ας δούμε μια μαθηματική προσέγγιση του CAPM , πριν προχωρήσουμε στο επόμενο κομμάτι του κώδικα μας

Σκοπός μας είναι να βρούμε εκείνη την συνάρτηση που περιγράφει την ευθεία που εφάπτεται με το αποτελεσματικό σύνορο $t(x) = a + b \cdot x$ και πρέπει να ικανοποιεί τις εξής συνθήκες

$t(0) = r_f \rightarrow \alpha = r_f$

$t(x) = f(x) \rightarrow a + b \cdot x = f(x)$

$t'(x) = f'(x) \rightarrow b = f'(x)$

Καθότι δεν έχουμε κάποιον κλειστό τύπο υπολογισμού για το αποτελεσματικό σύνορο ή την πρώτη παράγωγο του, πρέπει να λύσουμε το παραπάνω σύστημα εξισώσεων

CAPM

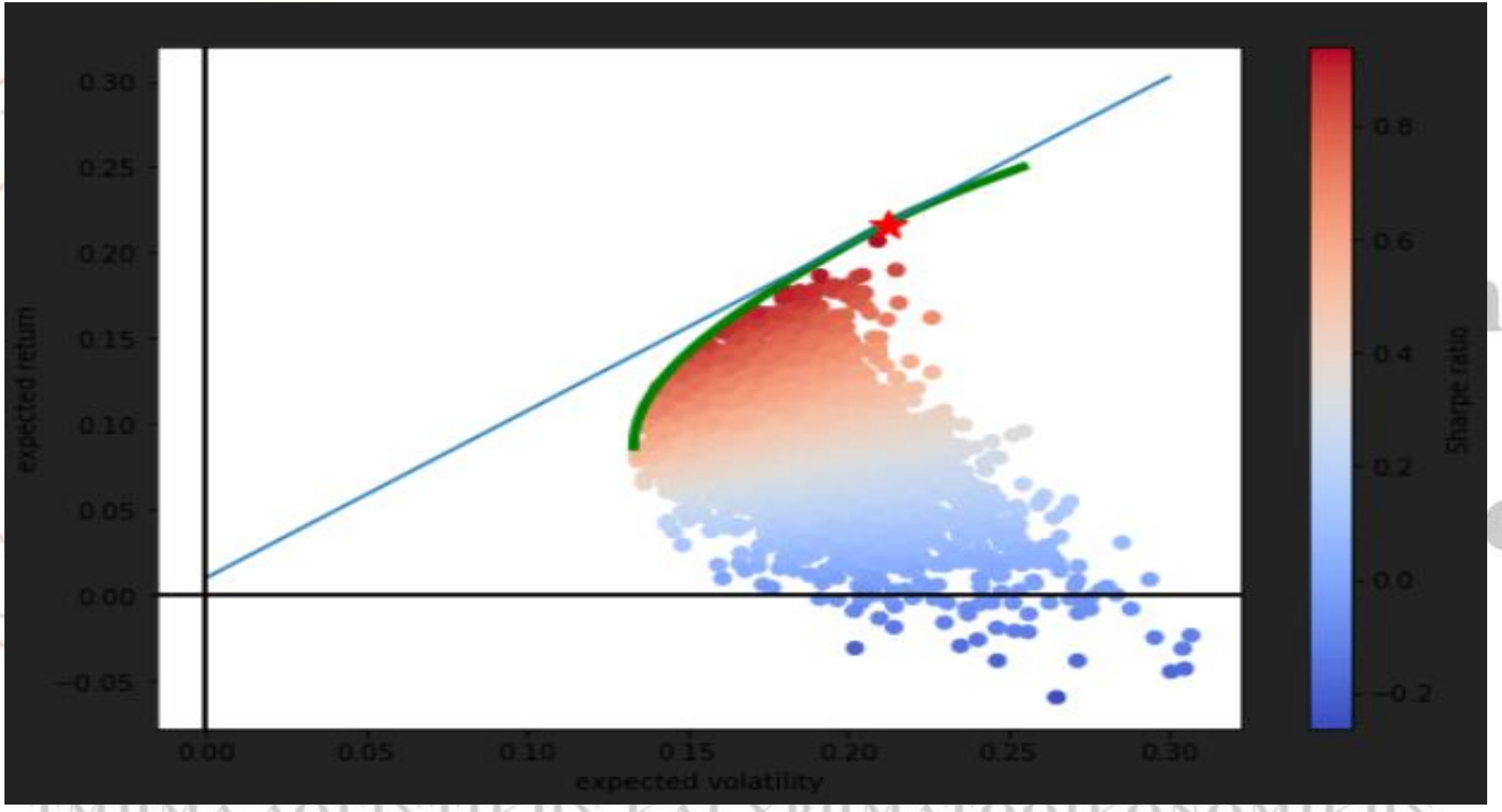
```
def equations(p, rf=0.01):
    eq1 = rf - p[0]
    eq2 = rf + p[1] * p[2] - f(p[2])
    eq3 = p[1] - df(p[2])
    return eq1, eq2, eq3
opt = sco.fsolve(equations, [0.01, 0.5, 0.15])
```

risk-free με απόδοση 1%

CAPM

```
plt.figure(figsize=(8, 4))
plt.scatter(pvols, prets,c=(prets - 0.01) / pvols, marker='o’, cmap='coolwarm')
plt.plot(evols, erets, 'g', lw=4.0)    #Σύσταση τυχαίου χαρτοφυλακίου
cx = np.linspace(0.0, 0.3)
plt.plot(cx, opt[0] + opt[1] * cx, lw=1.5) #Αποτελεσματικό Συνορο(Efficient Frontier)
plt.plot(opt[2], f(opt[2]), 'r*', markersize=15.0) #Capital Market Line
plt.axhline(0, color='k', ls='-', lw=2.0)
plt.axvline(0, color='k', ls='-', lw=2.0)
plt.xlabel('expected volatility')
plt.ylabel('expected return')
plt.colorbar(label='Sharpe ratio')
```

CAPM-Capital Asset Pricing Mode



CAPM

- cons = ({'type': 'eq', 'fun': lambda x: statistics(x)[0] - f(opt[2])},
{'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
- res = sco.minimize(min_func_port, noa * [1. / noa,], method='SLSQP',
bounds=bnds, constraints=cons)
- res['x'].round(3)

Οι σταθμίσεις για το βέλτιστο χαρτοφυλάκιο είναι οι εξής:

array([0.684, 0.059, 0.257, -0. , 0.])

Παρατηρούμε πως αποτελείται μονο από μετοχές της Apple,Microsoft και Google

Αποτίμηση παραγώγων

Με τη έννοια των χρηματοοικονομικών παραγώγων(Derivatives),ορίζουμε ένα οικονομικό αξιόγραφο, η αξία του οποίου εξαρτάται από κάποιο άλλο βασικό προϊον(Underlying Asset) όπως για παράδειγμα επιτόκια, μετοχές, εμπορεύματα κ.α.

Γενικά αποτελούν ένα ιδιαίτερα σημαντικό εργαλείο στην χρηματοικονομική επιστήμη καθώς οι επενδυτές μπορούν να εφαρμόσουν διάφορες επενδυτικές στρατηγικές όπως το Arbitrage το Hedging και το Speculation.

Αρχικά θα δούμε καποια θεωρητικά κομματια των τεχνικών καθως και μια εισαγωγή στα παράγωγα

Παραγωγα συμβόλαια

Οι βασικές κατηγορίες παραγώγων είναι οι Εξής:

- Συμβόλαια μελλοντικής εκπλήρωσης(Futures):

Τα ΣΜΕ αποτελούν μια συμφωνία μεταξύ 2 μερών για την αγορά και την πώληση ενός συμφωνημένου περιουσιακού στοιχείου σε μια συμφωνημένη μελλοντική στιγμή. Είναι τυποποιημένα συμβόλαια που πραγματοποιούνται στο χρηματιστήριο .Γενικά οι επενδυτές μπορούν να κάνουν χρήση τέτοιων συμβολαίων με σκοπό την αντιστάθμιση του κινδύνου η/και την κερδοσκοπία σχετικα με την τιμή του Underlying Asset.Τα 2 μέρη υποχρεούνται να εκπληρώσουν την συναλλαγή.

Παραγωγα συμβόλαια

Προθεσμιακά συμβόλαια(Forward Contracts):

Τα Προθεσμιακά συμβόλαια είναι μια προσαρμοσμένο συμβόλαιο ανάμεσα σε δύο μέρη για την αγορά η πώληση ενός περιουσιακού στοιχείου σε συγκεκριμένη τιμή και χρονική διάρκεια. Γενικα χρησιμοποιείται τόσο για αντιστάθμιση καθώς και για κερδοσκοπία. Η μη τυποποιημένη φύση τους τα καθιστά κατάλληλα για αντιστάθμιση.Σε αντίθεση με τα ΣΜΕ, τα προθεσμιακα συμβόλαια τροποποιούνται τόσο ως προς το εμπόρευμα καθώς και την ημερομηνία παράδοσης. Είναι εξωχρηματιστηριακα προϊόντα(OTC) το οποίο βέβαια αυξάνει τον κίνδυνο αθέτησης.

Παραγωγα συμβόλαια

- Συμφωνίες ανταλλαγής(Swaps)

Τα swaps αποτελούν μια συμφωνία μεταξύ δύο μερών για την ανταλλαγή ταμειακών ροών για μια συγκεκριμένη χρονική περίοδο. Για παράδειγμα το πρώτο SWAP είχε πραγματοποιηθεί ανάμεσα στην IBM και την World Bank το 1981.Η WB είχε δανειακές υποχρεώσεις σε αμερικάνικα δολάρια ενώ η IBM είχε δανειακές υποχρεώσεις σε γερμανικά μάρκο και ελβετικό φράγκο. Τα 2 μέρη συμφώνησαν να «ανταλλάξουν» τις υποχρεώσεις τους ως προς την αποπληρωμή των τόκων με σκοπό τη χρηση του συγκριτικού πλεονεκτήματος. Τα swaps επίσης αποτελούν εξωχρηματιστηριακά προϊόντα (OTC)

Παραγωγα συμβόλαια

- Συμβόλαια Δικαιωμάτων προαίρεσης(Option Contracts)

Τα συμβόλαια δικαιωμάτων προαίρεσης είναι συμβόλαια για μελλοντικές αγοραπωλησίες χρεογράφων παρόμοια με τα ΣΜΕ.Στην περίπτωση των Option Contracts , ο αγοραστής έχει το δικαίωμα αλλα όχι την υποχρέωση να εξασκήσει το συμβόλαιο σε μια συγκεκριμένη τιμή εξάσκησης(Strike price) πληρώνοντας την τιμή του δικαιώματος (premium) χωρίς να έχει καμία άλλη υποχρέωση.Options που αφορούν αγορά ονομάζονται Call Options και αν αυτό αφορά συμφωνία πώλησης ονομάζεται Put Option

Αποτίμηση Option στην Python

Τωρα θα προχωρήσουμε στις πρώτες μας εφαρμογές στην Python σχετικά με τα παράγωγα. Θα δούμε 2 διαφορετικές τεχνικές σχετικά με την αποτίμηση Options στην Python.Η πρώτη αφορά προσομοίωση Monte Carlo και η δεύτερη την γνωστή μεθοδο των Black-Scholes

Monte Carlo option Valuation

Στο μοντέλο των BSM η τιμή του αξιογράφου στον χρόνο λήξης S_T αποτελεί μια τυχαία μεταβλητή ,με το z να αποτελεί επίσης μια τυχαία μεταβλητή οπου ακολουθεί την τυπικη κανονική κατανομή. Περιγράφεται από την ακόλουθη εξίσωση (1)

$$S_T = S_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}z\right) \quad \textbf{(1)}$$

Monte Carlo option Valuation

- Ας δούμε αλγοριθμικά πως λειτουργεί:

1. Παράγουμε I ψευδοτυχαίους αριθμούς $z(i), i \in \{1,2, \dots, I\}$ που ακολουθούν την τυποποιημένη κανονική κατανομή
2. Υπολογίζουμε τα αντίστοιχα S_T σύμφωνα με εξίσωση **(1)**
3. Υπολογίζουμε ολες τις εσωτερικές αξίες του option στον χρόνο λήξης σύμφωνα με $h_T(i) = \max(S_T(i) - K, 0)$
4. Τέλος κάνουμε εκτίμηση της παρούσας αξίας του option συμφωνα με τον ακόλουθο εκτιμητή Monte Carlo :

$$C_0 \approx e^{-rT} \frac{1}{I} \sum_I h_T(i)$$

Monte Carlo option Valuation

```
import numpy as np
S0 = 100. # Spot price
K = 105. # Strike price
T = 1.0  #Maturity
r = 0.05 # risk-free interest rate
sigma = 0.2 # Annual volatility
I = 100000 # Number of simulations
# Valuation Algorithm
z = np.random.standard_normal(I)    #Παραγωγή των z
ST = S0 * np.exp((r - 0.5 * sigma ** 2) * T + sigma * np.sqrt(T) * z)# index values at maturity
hT = np.maximum(ST - K, 0) # inner values at maturity
C0 = np.exp(-r * T) * np.sum(hT) / I # Monte Carlo estimator
```

Black-Scholes-Merton Model

Η μέθοδος των Black-Scholes-Merton αποτελεί ίσως το πιο γνωστό μοντέλο σχετικά με την αξιολόγηση των παραγώγων:

Formula για την αποτίμηση call options.

C=$SN(d_1) - Ke^{-rt}N(d_2)$

$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + (r + \frac{\sigma^2}{2}t)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

Black-Scholes-Merton Model

Όπου ορίζουμε τα εξής :

C :Τιμή του Call Option

S :Παρούσα τιμή της μετοχής

K : Τιμή εξάσκησης

r :Ετήσιο επιτόκιο χωρίς κίνδυνο

t : Χρόνος μέχρι την ημερομηνία λήξης

σ: Μεταβλητότητα

N(·):Συνάρτηση κανονικής κατανομής

ΒΑΣΙΚΕΣ ΠΑΡΑΔΟΧΕΣ ΤΟΥ ΜΟΝΤΕΛΟΥ

- Δεν αποδίδεται μέρισμα κατά την διάρκεια του option
- Η συμπεριφορά της Αγοράς είναι τυχαια. Δεν υπάρχει δυνατότητα πρόβλεψης
- Οι αποδόσεις του υποκείμενου στοιχείου ακολουθούν την λογαριθμοκανονική κατανομή
- Το μοντέλο αφορά option Ευρωπαϊκού τύπου, οπότε κατ' επέκτασιν η εξάσκηση γίνεται κατά την διάρκεια λήξης.
- Δεν λαμβάνονται υπόψιν τα κόστη συναλλαγής κατά την αγορά του option
- Τα r & σ του υποκείμενου στοιχείου είναι γνωστά και σταθερά

Black-Scholes-Merton Formula

```
def bsm_call_value(S0, K, T, r, sigma):

    from math import log, sqrt, exp
    from scipy import stats
    S0 = float(S0)
    d1 = (log(S0 / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * sqrt(T))
    d2 = (log(S0 / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * sqrt(T))
    value = (S0 * stats.norm.cdf(d1, 0.0, 1.0)
    - K * exp(-r * T) * stats.norm.cdf(d2, 0.0, 1.0))
    # stats.norm.cdf —> cumulative distribution function
    # for normal distribution
    return value
```

Παράγωγα &VaR

Όπως είχαμε αναφέρει και νωρίτερα ο δείκτης VaR(Value at Risk) είναι ένας από τους πιο ευρέως χρησιμοποιούμενους δείκτες κινδύνου. Μας υποδεικνύει την μέγιστη δυνατή απώλεια του χαρτοφυλακίου μας.

Σκεφτείτε για παράδειγμα ότι έχετε θέση σε μια μετοχή η οποία αξίζει 1 εκατομμύριο δολάρια σήμερα έχει VaR $50.000 σε ένα διάστημα εμπιστοσύνης 99% μέσα σε μια χρονική περίοδο ενός μήνα.

Ο δείκτης VaR μας υποδεικνύει ότι με πιθανότητα 99%(99 από τις 100 περιπτώσεις) η απώλεια που πρέπει να αναμένουμε σε μια περίοδο 30 ημερών δεν θα ξεπεράσει τα $50.000.

Παράγωγα & VaR

Ας δούμε ένα μοντέλο Black-Scholes-Merton και σκεφτείτε την ακόλουθη προσομοίωση μεσα σε μια χρονική περίοδο 30 ημερών

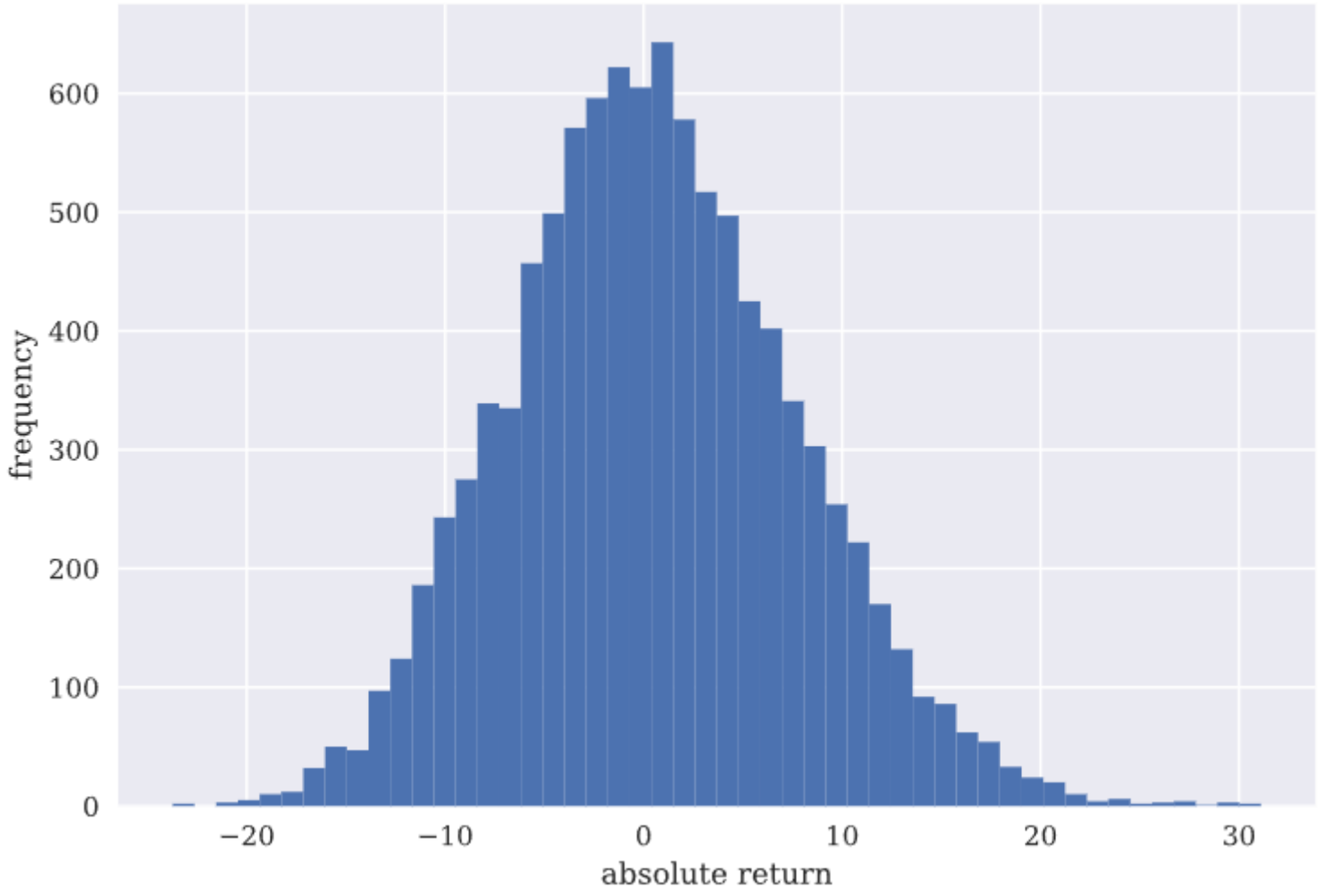
- S0 = 100
- r = 0.05
- sigma = 0.25
- T = 30 / 365
- I = 10000
- ST = S0 * np.exp((r - 0.5 * sigma ** 2) * T+ sigma * np.sqrt(T) * npr.standard_normal(I))

Επειτά θα ταξινομήσουμε τις απώλειες και τα κέρδη μας από την μεγάλυτερη απώλεια μεχρι το μέγιστο κέρδος

- R_gbm = np.sort(ST - S0)

Παράγωγα &VaR

- plt.hist(R_gbm, bins=50)
- plt.xlabel('absolute return')
- plt.ylabel('frequency')
- plt.grid(True)



Παράγωγα &VaR

Καθότι έχουμε ταξινομήσει τις τιμές μας θα κάνουμε χρήση της scoreatpercentile ώστε να υπολογίσουμε τον δείκτη VaR για πολλαπλά διαστήματα εμπιστοσύνης

Θα ορίσουμε έξι διαστήματα εμπιστοσύνης από 90% εως 99.99%.Συγκεκριμένα όπως θα δείτε στην λίστα percs, όπου θα ορίσουμε το 0.01 αντιστοιχεί στο 99.99% το 0.1 στο 99.90% κ.ο.κ

Θα παρατηρείσετε πως οσο πιο «αυστηροί» είμαστε με το διάστημα εμπιστοσύνης τόσο μεγαλύτερος είναι και η τιμή του δείκτη VaR

Παράγωγα &VaR

- percs = [0.01, 0.1, 1., 2.5, 5.0, 10.0] #Λίστα με τα διαστήματα εμπιστοσύνης που επιθυμούμε
- var = scs.scoreatpercentile(R_gbm, percs) #Υπολογισμός των VaR με χρήση της scoreatpercentile
- **print**("%16s %16s" % ('Confidence Level', 'Value-at-Risk')) #Καθορίζουμε τις στήλες[Προαιρετικός κώδικας]
- **print**(33 * "-")#Με αυτην την σειρά εκτυπώνουμε 33 παύλες ώστε να διαχωρίσουμε τους τιτλους από τις τιμες[Προαιρετικός κώδικας]
- **for** pair **in** zip(percs, var):
- **print**("%16.2f %16.3f" % (100 - pair[0], -pair[1]))

```
Confidence Level    Value-at-Risk
--------------------------------
           99.99           22.636
           99.90           19.081
           99.00           15.199
           97.50           12.839
           95.00           10.868
           90.00            8.704
```

Η μεταβλητη var επιστρέφει τις τιμές των VaR με αρνητικό πρόσημο, και για αυτόν τον λόγο στην τελευταία σειρά κώδικα βάλαμε το -pair[1].

Τέλος θα παρατηρήσετε πως οι τιμές σας διαφέρουν λίγο σε σχέση με το διπλανό πινακάκι. Καθότι μιλάμε για μια τυχαία προσομοίωση είναι λογικό να υπάρχουν μικρές αποκλίσεις.

Θα μπορούσαμε να ορίσουμε στην αρχη την npr.seed() με όρισμα ένα τυχαίο ακέραιο αριθμό και να έχουμε πάντα τα ίδια αποτελέσματα

CVaR

Άλλοι σημαντικοί δείκτες ΄κινδύνου είναι οι credit value at risk (CVaR) και η πιστωτική αξία προσαρμογής(CVA) που προέρχεται από το CVaR.

Το CVaR είναι ένα μέτρο για τον κίνδυνο που συνεπάγεται με την πιθανότητα αθέτησης της συμφωνίας από τον αντισυμβαλλόμενο(παραδείγματος χάριν χρεοκοπία του αντισυμβαλλόμενου)

Σε μια τέτοια περίπτωση υπάρχουν 2 κύριες παραδοχές που πρέπει να γίνουν :

- Πιθανότητα αθέτησης-Probability of Default (PD)
- Μέσο επίπεδο απώλειας-Average loss Level (L)

Παράγωγα &VaR

Για να γίνει αντιλυπτό το CVaR θα εφαρμόσουμε ξανα ένα μοντέλο Black-Scholes-Merton με τους εξής παραμέτρους

- S0 = 100.
- r = 0.05
- sigma = 0.2
- T = 1.
- I = 100000
- ST = S0 * np.exp((r - 0.5 * sigma ** 2) * T + sigma * np.sqrt(T) * npr.standard_normal(I))

#Στην απλουστερη περίπτωση ορίζουμε ενα σταθερό average Loss Level και σταθερο Probability gor default(per year) του αντισυμβαλλόμενου ως εξης

- L = 0.5
- p = 0.01

Παράγωγα &VaR

Τωρα θα δημιουργήσουμε διαφορα πιθανά σενάρια ως με την χρήση της κατανομής Poisson

- D = npr.poisson(p * T, I)
- D = np.where(D > 1, 1, D)

Χωρίς την πιθανότητα αθέτησης η Rik-Neutral η μελλοντική τιμη του δείκτη πρέπει να είναι ίδια την αξία του αξιόγραφου σήμερα

- np.exp(-r * T) * 1 / I * np.sum(ST)

```
99.95156192453575
```

Το CVaR υπολογίζεται ως εξής

- CVaR = np.exp(-r * T) * 1 / I * np.sum(L * D * ST)
- CVaR

```
0.4924827926628677
```

Παράγωγα &VaR

Αναλογικα η παρούσα αξία του περουσιακού στοιχείου προσαρμωσμένη για τον πιστωτικό κίνδυνο πρέπει να ισούται με

- S0_CVA = np.exp(-r * T) * 1 / I * np.sum((1 - L * D) * ST)
- S0_CVA

```
99.45907913187288
```

Αυτό το αποτέλεσμα πρέπει να είναι περίπου ίσο με την διαφορά του CVAR από την παρουσα αξία του περουσιακού στοιχείου

- S0_adj = S0 - CVaR
- S0_adj

```
99.50751720733713
```

Παράγωγα &VaR

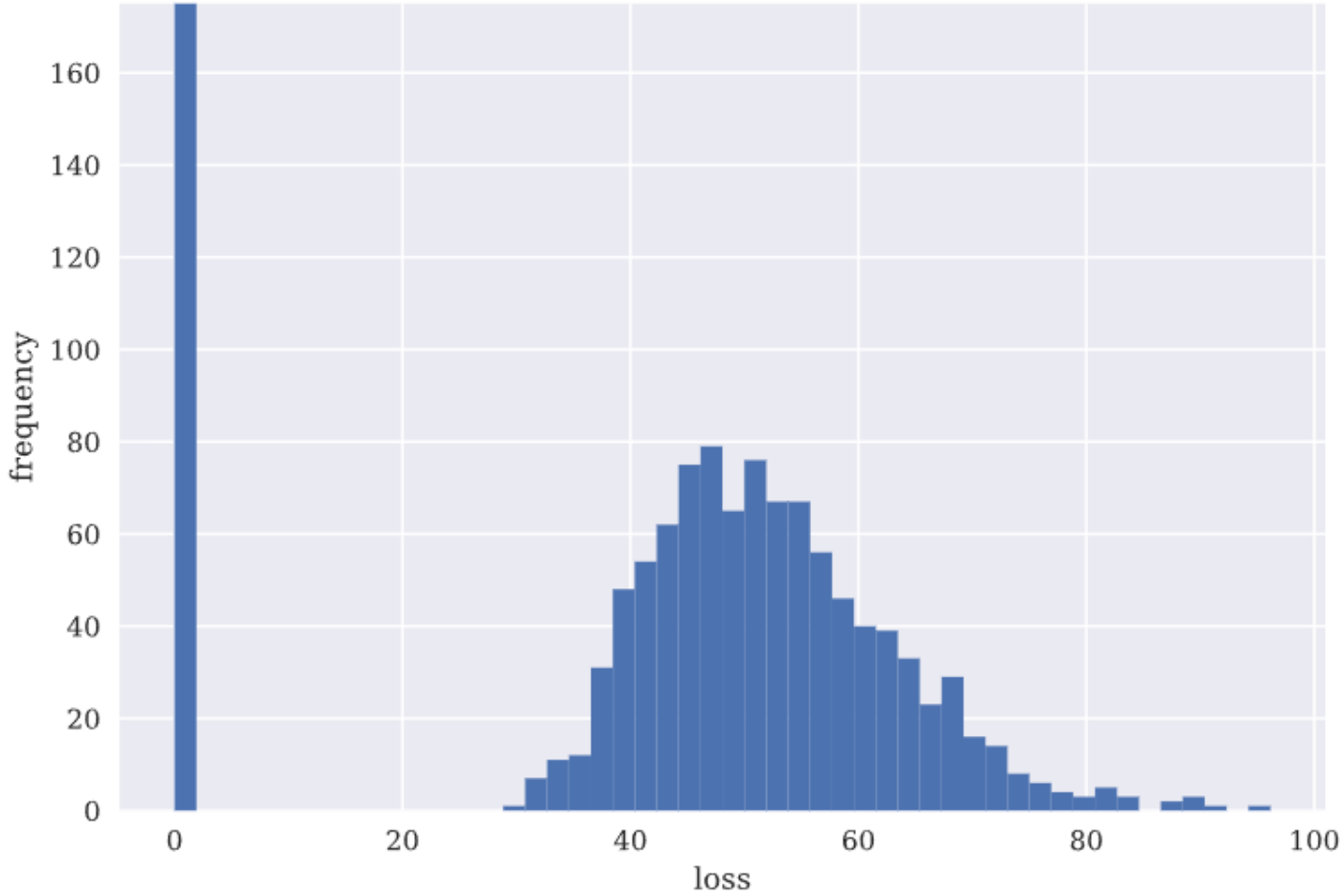
Σε αυτό παράδειγμα παρατηρήσαμε περίπου 1000 απώλειες λόγω της αθέτησης

- np.count_nonzero(L * D * ST)

Ας δούμε οπτικά την κατανομή των απωλειών λόγω αθέτησης

- plt.hist(L * D * ST, bins=50)
- plt.xlabel('loss')
- plt.ylabel('frequency') →
- plt.grid(True)
- plt.ylim(ymax=175)

Καθότι περίπου στις 99.000 απο τις 100.000 περιπτώσεις δεν είχαμε απώλεια , η τιμή 0 συγκεντρώνει τις περισσότερες παρατηρήσεις



Παράγωγα &VaR

Τώρα θα δούμε ένα παράδειγμα ενός ευρωπαϊκού call option.Αν και οι διαφορά στις τιμές είναι μικρές σε αυτό το παράδειγμα θα θέσουμε και seed(123) ώστε να έχουμε ακριβώς τα ίδια αποτελέσματα

- npr.seed(123)
- K = 100.
- hT = np.maximum(ST - K, 0)
- C0 = np.exp(-r * T) * 1 / I * np.sum(hT)
- C0

```
10.42651530071525
```

Το CVaR μας ορίζεται περίπου στα 5 cents

- CVaR = np.exp(-r * T) * 1 / I * np.sum(L * D * hT)
- CVaR

```
0.050914955176978895
```

Παράγωγα &VaR

Όπως ήταν αναμενόμενο η προσασμοσμένη τιμή του option είναι 5 cents μικρότερη

- C0_CVA = np.exp(-r * T) * 1 / I * np.sum((1 - L * D) * hT)
- C0_CVA

```
10.375600345538272
```

Το αποτέλεσμα θα ήταν ίδιο αν αφαιρούσαμε το CVaR από C0

- np.count_nonzero(L * D * hT) #Πλήθος απωλειών

```
542
```

- np.count_nonzero(D) #Πλήθος αθέτησης (defaults)

```
987
```

- I - np.count_nonzero(hT) #Zero payoff

```
44086
```

Παράγωγα &VaR

Σχετικά με ένα κλασσικό περιουσιακό στοιχείο παρατηρούμε πως η περίπτωση του δικαιώματος προαίρεσης (option), έχει κάπως διαφορετικά χαρακτηρίστηκα. Βλέπουμε πώς έχουμε 542 περιπτώσεις απώλειας εξαιτίας της αθέτησης παρόλου που έχουμε σχεδόν 1000 περιπτώσεις αθέτησης(987).

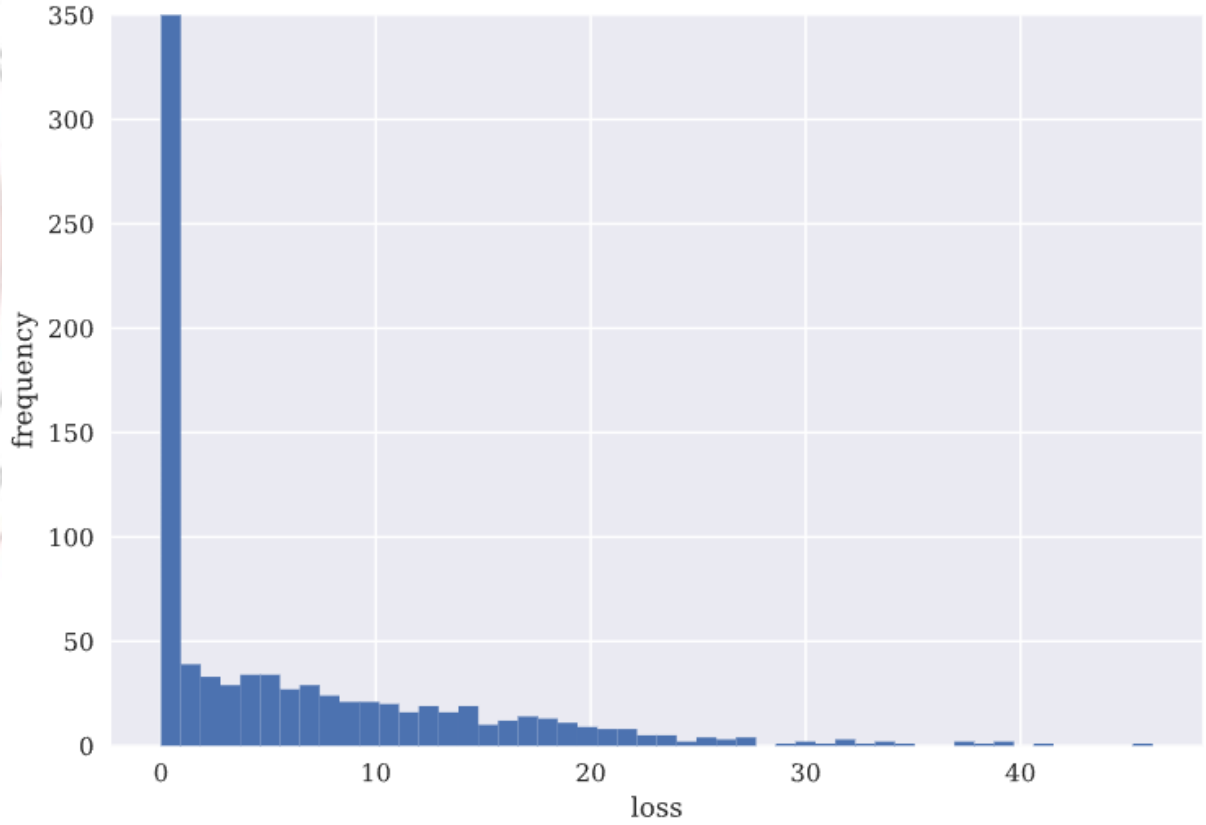
Αυτο συμβαίνει καθώς η αξία αποπληρωμής του option στην στιγμή λήξης του (maturity) έχει μεγάλες πιθανότητες να ισούται με μηδέν

$$\frac{44086}{10000} \approx 44\%$$

Παράγωγα &VaR

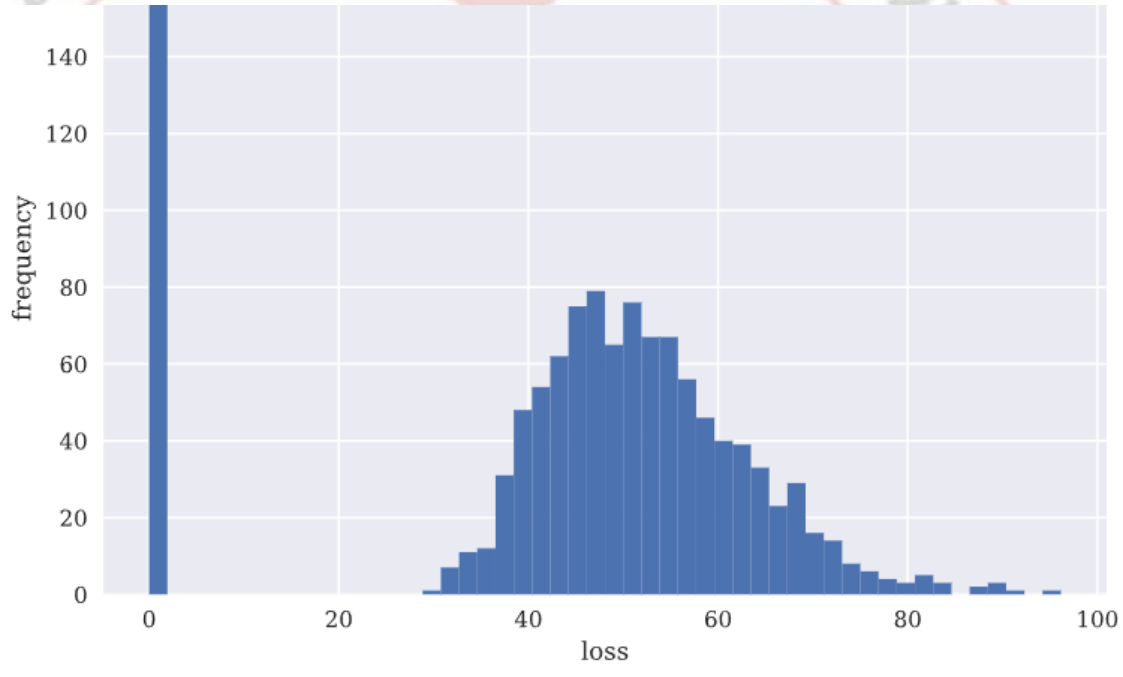
Με τις επόμενες σειρές κώδικα βλέπουμε και οπτικά, πως το CVaR του ευρωπαικού option έχει διαφορετική κατανομή

- plt.hist(L * D * hT, bins=50)
- plt.xlabel('loss')
- plt.ylabel('frequency')
- plt.grid(True)
- plt.ylim(ymax=350)

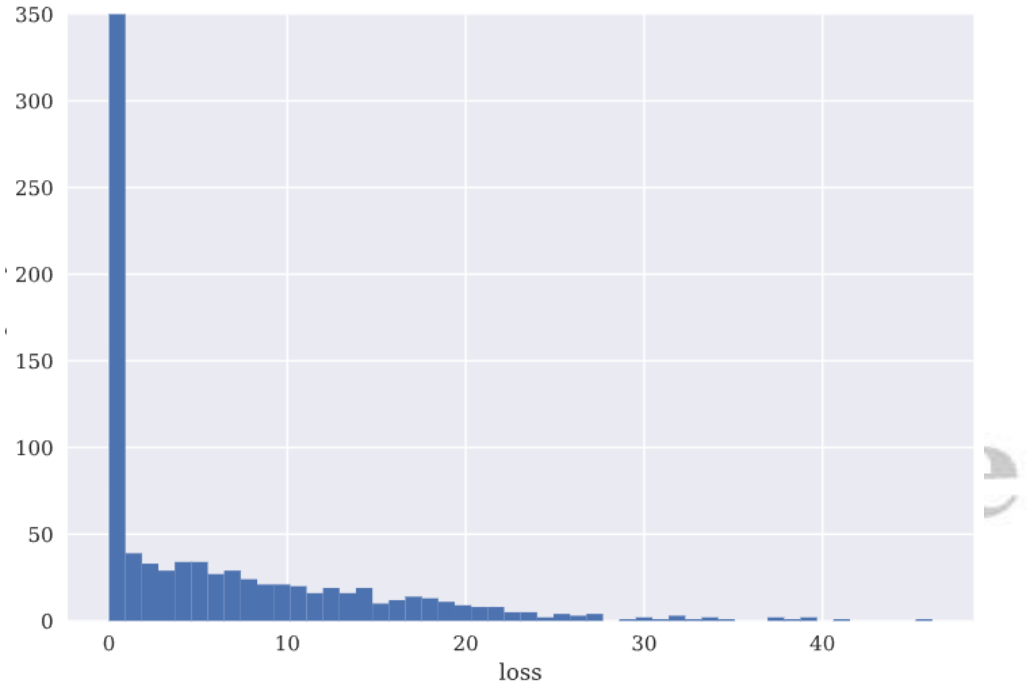


Παράγωγα &VaR

Κατανομή απωλειών



Stock



European Option

Ανάλυση χρονολογικών σειρών

Στο επόμενο κομμάτι θα εφαρμόσουμε μια ανάλυση χρονολογικών σειρών. Πιο συγκεκριμένα θα χρησιμοποιήσουμε δεδόμενα από 01/01/2010 εως και 29/06/2018 για τα ακόλουθα :

'Apple Stock', 'Microsoft Stock’, 'Intel Stock', 'Amazon Stock', 'Goldman Sachs Stock','SPDR S&P 500 ETF Trust', 'S&P 500 Index','VIX Volatility Index', 'EUR/USD Exchange Rate','Gold Price', 'VanEck Vectors Gold Miners ETF','SPDR Gold Trust’

Αρχικά θα εισάγουμε τις απαραίτητες βιβλιοθήκες και επειτα θα αντλήσουμε τα δεδομένα μας μέσω της Pandas_datareader

Ανάλυση χρονολογικών σειρών

```
import numpy as np
import pandas as pd
import pandas_datareader as web
from pylab import mpl, plt
plt.style.use('seaborn’)
mpl.rcParams['font.family'] = 'serif'
%config InlineBackend.figure_format = 'svg’
```

#Οι 3 τελευταιες εντολές προσφέρουν μια καλύτερη μορφοποίηση στα διαγράμματα μας, η χρήση τους είναι προαιρετική

Ανάλυση χρονολογικών σειρών

- data=web.DataReader(['AAPL','MSFT’,PRP','AMZN','GS','SPY','^GSPC’,
'^VIX','EURUSD=X','GC=F','GDX','GLD'], 'yahoo', start='2010/01/01', end='2018/06/29')['Close’]

Γενικά τα ticker της κάθε μετοχής[οι κωδικοί που εισάγουμε στον κώδικα μας] μπορούν να βρεθούν στο διαδίκτυο όπως για παράδειγμα στο site του Nasdaq

www.nasdaq.com/market-activity/stocks/screener

Έπειτα κατά τα γνωστά ορίζουμε ημερομηνία έναρξης και λήξης για την αναζήτηση μας και στο τέλος προσθέτουμε το ['Close’] καθότι μας ενδιαφέρει μόνο η τιμή κλεισίματος της κάθε μετοχής για την ανάλυση μας

Ανάλυση χρονολογικών σειρών

Γενικά μια καλή στρατηγική όταν κατεβάζουμε δεδομένα ,είναι να κανουμε έναν πρώτο έλεγχο σχετικα με τα χαρακτηριστικά τους. Για αυτόν τον σκοπό θα εκτελέσουμε τα ακόλουθα

- data.info()

μας αποδίδει βασικά στοιχεία για τα δεδομένα μας όπως τον τύπο τους, το πλήθος κ.α

- data.head()

μας επιστρέφει τις 5 πρώτες σειρές των δεδομένων μας.Αν επιθυμούσαμε διαφορετικό πλήθος τότε θα εισαγάγαμε n, για παράδειγμα data.head(n=10)

- data.tail()

#Αντίστοιχα η tail μας επιστρέφει τις τελευταίες σειρές των δεδομένων μας

Ανάλυση χρονολογικών σειρών

Γενικά εκτός από τα χαρακτηριστικά των δεδομένων μας, είναι συνηθισμένο στις οικονομικές χρονολογικές σειρές να θέλουμε να έχουμε μια πρώτη εικόνα. Αυτό γίνεται ευκολα με την ακόλουθη εντολή:

- data.plot(figsize=(12, 12), subplots=True);

Με το όρισμα figsize καθορίζουμε το μέγεθος των διαγραμμάτων μας. Σχετικά με το sublots το ορίζουμε True καθώς θέλουμε η κάθε μετοχή να έχει το δικό της διάγραμμα. Σε περίπτωση όπου θέλαμε όλες οι χρονοσειρές, να απεικονίζονται σε ένα ενιαίο διάγραμμα τότε θα ορίζαμε : subplots= False.

Ανάλυση χρονολογικών σειρών

Η Pandas μας παρέχει την δυνατότητα , να έχουμε πληροφορία για τα περιγραφικά στατιστικά στοιχεία των δεδομένων μας με την χρήση της describe.Για καθαρά οπτικούς και πρακτικούς λόγους, θα κάνουμε χρήση και της round(2)-στρογγυλοποίηση στα 2 δεκαδικά ψηφία .Επίσης θα μπορούσαμε να δούμε μόνο την μέση τιμή των δεδομένων μας με την χρήση της mean()

- data.describe().round(2)
- data.mean()

Τέλος έχουμε την δυνάτοτητα να επιλέξουμε εμείς τα στατιστικά στοιχεία που μας ενδιαφέρουν με χρήση της aggregate

- data.aggregate([min,np.mean,np.std,np.median,max]).round(2)

Ανάλυση χρονολογικών σειρών

Η στατιστική ανάλυση των οικονομικών σειρών ,πολλές φορές βασίζεται στην μεταβολή τους στον χρόνο και όχι στις πραγματικές τιμές. Υπάρχουν πολλές επιλογές σχετικά με τον υπολογισμό της μεταβολής μιας χρονοσειράς με την πάροδο του χρόνου. Για παράδειγμα απόλυτη διαφορά, ποσοστιαία μεταβολή, καθώς και μέσω των Log-Returns.

- data.diff().head() #Απόλυτη διαφορά
- data.diff().mean() #Μέση τιμή της Απόλυτης διαφοράς
- data.pct_change().round(3).head() #Ποσοστιαία μεταβολή
- data.pct_change().mean().plot(kind='bar', figsize=(10, 6));

Ανάλυση χρονολογικών σειρών

- rets = np.log(data / data.shift(1))
- rets.head().round(3)
- rets.cumsum().apply(np.exp).plot(figsize=(10, 6));

Ανάλυση χρονολογικών σειρών

Πιθανόν να υπάρξει περίπτωση οπου επιθυμούμε τον ανασχηματισμό των δεδομένων μας. Για παράδειγμα έχουμε τιμές σε καθημερινή βάση, αλλα μας ενδιαφέρει η ανάλυση σε εβδομαδιαία η μηνιαία περίοδο.

- data.resample('1w', label='right').last().head()
- data.resample('1m', label='right').last().head()

Με το όρισμα 1w μετατρέπουμε τα δεδομένα μας σε εβδομαδιαία. Με το 1m σε μηνιαία. Το όρισμα label μπορεί να ισούται με left ή right αναλόγως το διάστημα που επιθυμούμε.Τελος συνδιάζεται με την εντολή last()

Ανάλυση χρονολογικών σειρών

Στην οικονομική και τεχνική ανάλυση θα συναντήσουμε πολλές φορες την χρήση κυλιόμενων μέσων.Αποτελούν χρήσιμα εργαλεία ,τα οποία μας παρέχουν σημαντικές πληροφορίες για την συμπεριφορά της χρονοσειράς μας με σκοπό την ανάλυση και πρόβλεψη της.

Τώρα θα δουμε ορισμένα παραδείγματα με την χρήση της Pandas για την μετοχή της Apple

- sym = 'AAPL'
- data = pd.DataFrame(data[sym]).dropna()
- data.tail()

Ανάλυση χρονολογικών σειρών

- window = 20 #Ορίζουμε την περίοδο στις 20 ημέρες
- data['min'] = data[sym].rolling(window=window).min() #κυλιόμενη ελάχιστη τιμή
- data['mean'] = data[sym].rolling(window=window).mean() #κυλιόμενη μέση τιμή
- data['std'] = data[sym].rolling(window=window).std()#κυλιόμενη τυπικη απόκλειση
- data['median'] = data[sym].rolling(window=window).median() #κυλ.Διάμεσος
- data['max'] = data[sym].rolling(window=window).max() #κυλιόμενη μέγιστη τιμή
- data['ewma'] = data[sym].ewm(halflife=0.5, min_periods=window).mean()

#Με την ewm υπολογίζουμε τον κυλιόμενο εκθετικό μέσο

- data.dropna().head() #Η dropna αφαιρεί της ελλείπουσες τιμές

Ανάλυση χρονολογικών σειρών

Θα απεικονίσουμε οπτικά ορισμένα από τα Rolling Statistics.Θα χρησιμοποιησουμε τα 200 τελευταία στοιχεία της χρονοσειράς μας

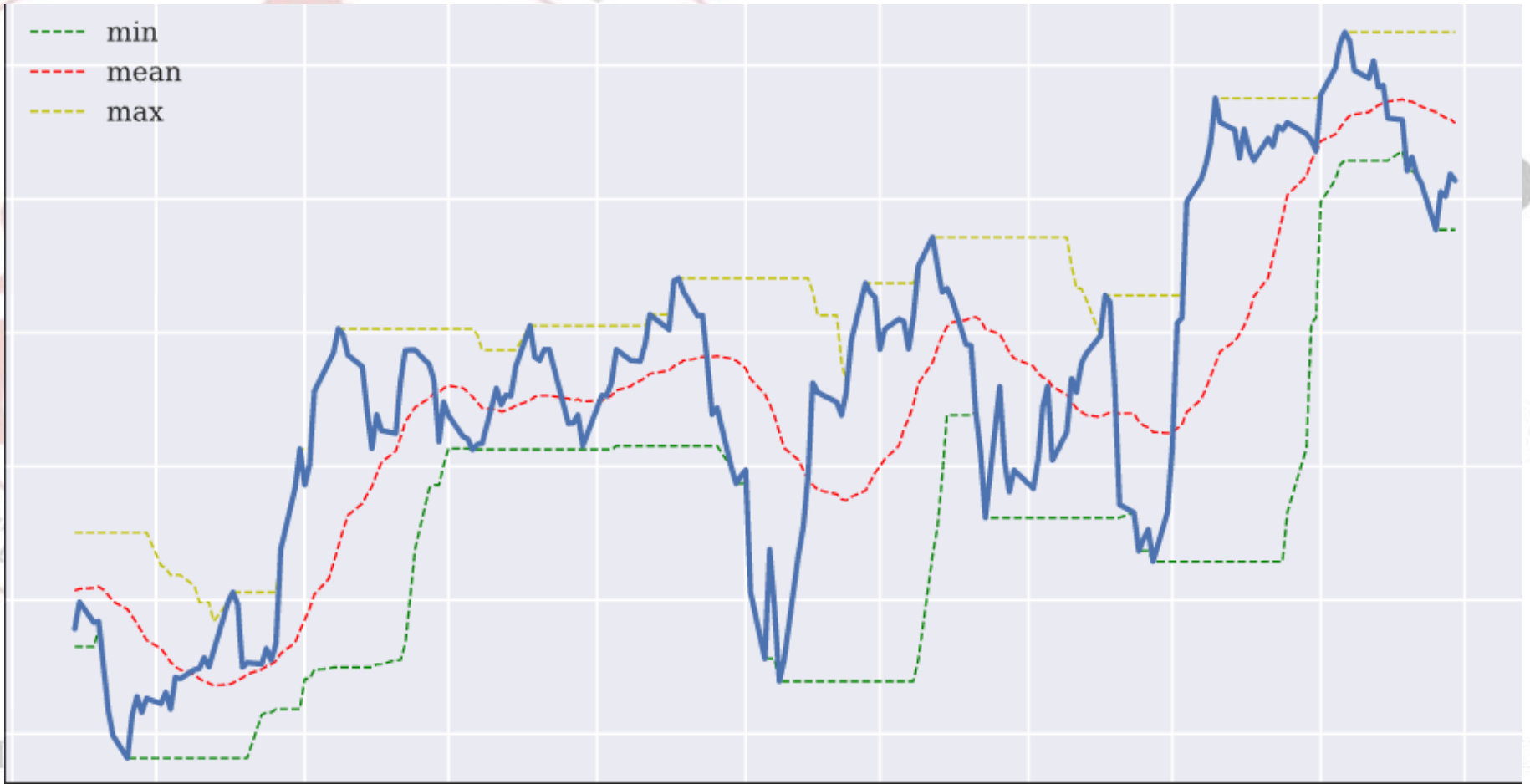
- ax = data[['min', 'mean', 'max']].iloc[-200:].plot(figsize=(10, 6), style=['g--', 'r--', 'y--'], lw=0.8)

#Επιλέγουμε κυλιόμενους μέσους για ελάχιστη, μέση και μέγιστη τιμή.Figsize για να ορίσουμε το μεγέθος.Το όρισμα style ορίζει το χρώμα και το σχέδιο της γραμμής, της κάθε τιμής και με το lw ορίζουμε το πάχος των σειρών(Line Width).Συνηθώς η χρονοσειρα των πραγματικών τιμών μας έχει μεγαλύτερο lw όπως βλέπουμε και στην επόμενη σειρά

- data[sym].iloc[-200:].plot(ax=ax, lw=2.0);

Ανάλυση χρονολογικών σειρών

- Μας επιστρέφει το ακόλουθο διάγραμμά



Ανάλυση χρονολογικών σειρών

Ας δούμε ένα παράδειγμα τεχνικής ανάλυσης:

- data['SMA1'] = data[sym].rolling(window=42).mean()

#Δημιουργούμε στήλη με τις τιμές του Κυλιόμενου μέσου 42 ημερών

- data['SMA2'] = data[sym].rolling(window=252).mean()

#Δημιουργούμε στήλη με τις τιμές του Κυλιόμενου μέσου 252 ημερών

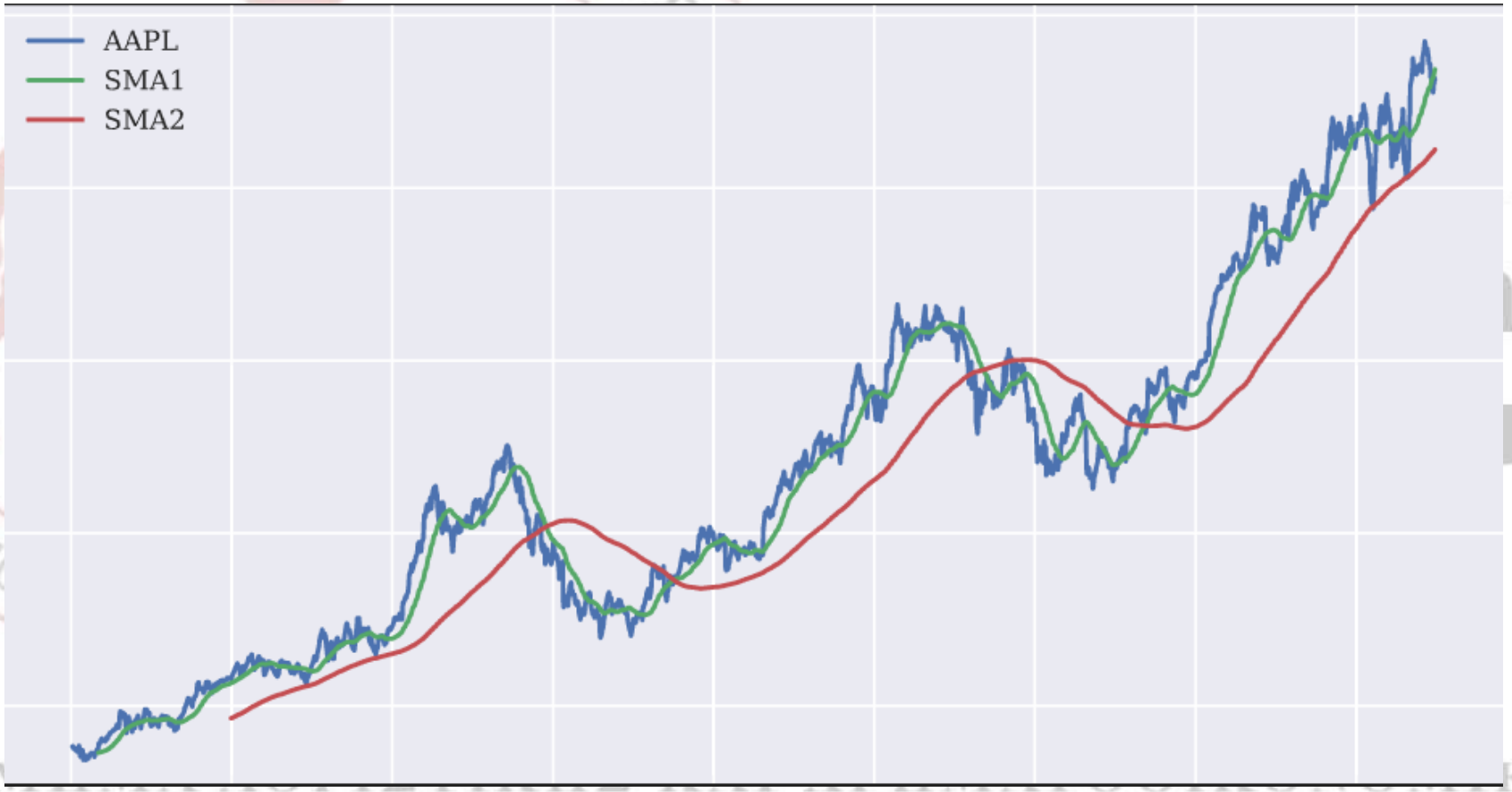
- data[[sym, 'SMA1', 'SMA2']].tail()

#Για άλλη μια φορά ελέγχουμε τα δεδομένα μας

Ανάλυση χρονολογικών σειρών

Συνεχίζουμε με την οπτική απεικόνηση των δεδομένων μας:

- data[[sym, 'SMA1', 'SMA2']].plot(figsize=(10, 6));



Ανάλυση χρονολογικών σειρών

- data.dropna(inplace=True)

```
data['positions'] = np.where(data['SMA1'] > data['SMA2'],  1,  -1)
```

#Δημιουργούμε μια νέα στήλη όπου ορίζουμε την συνθήκη αν ο SM1 είναι μεγαλύτερος του SMA2 να μας επιστρέφει την τιμής 1(Θέση αγοράς), και όταν συμβαίνει το αντίθετο θα μας επιστρέφει την τιμή -1(Θέση πώλησης).Ουσιαστικά θέλουμε να εφαρμόσουμε την τεχνική που είχαμε δει στο κομμάτι του Algorithmic Trading.Στο επόμενο slide θα ξαναθυμηθούμε την στρατηγική που εφαρμόζουμε

Algorithmic Trading: Moving Averages (SMA)

- Εισάγουμε 2 ΜΑ έναν γρήγορό , δηλαδή μικρής διάρκειας(42 ημερών)και έναν αργό μεγάλης διάρκειας(252 ημερών).

- Όταν ο γρήγορος ΜΑ περάσει πάνω από τον αργό ΜΑ είναι σήμα για είσοδο σε θέση αγοράς, καθώς μας δείχνει ότι η τάση θα έχει είναι αυξητική. Άυτο το σημείο είναι γνωστο σαν Golden cross

- Αντίστοιχα σε αντίθετη περίπτωση όταν ο αργός ΜΑ περάσει πάνω από τον γρήγορο είναι ένδειξη για είσοδο σε θέση πώλησης. Αυτό το σημείο είναι γνωστό σαν Death Cross.

Ανάλυση χρονολογικών σειρών

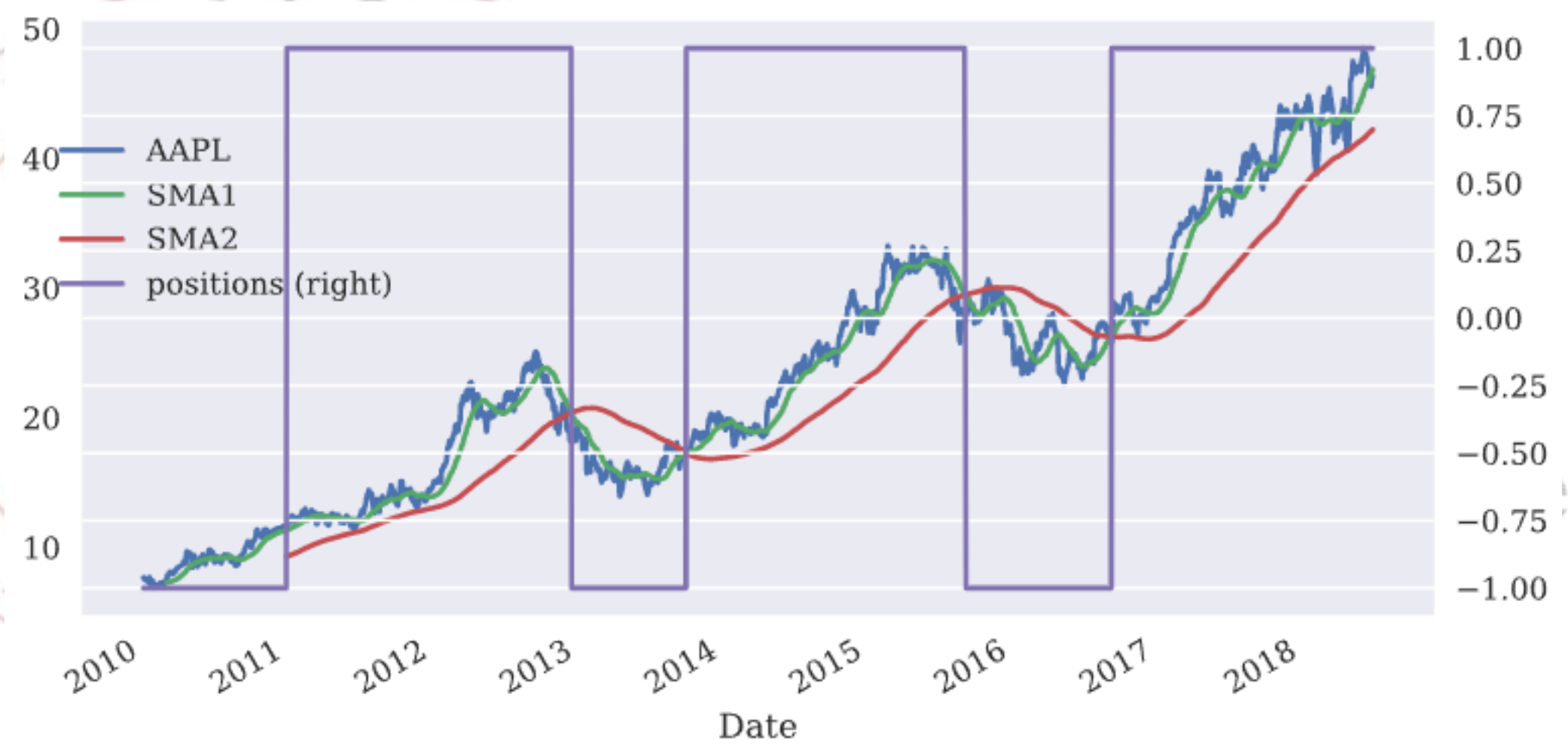
- ax = data[[sym, 'SMA1', 'SMA2', 'positions']].plot(figsize=(10, 6), secondary_y='positions')

#Σχετικα με το διάγραμμα μας εργαζόμαστε όπως συνήθως, Η διαφορά είναι πως προσθέτουμε και το όρισμα secondary_y ώστε οι τιμές της στήλης positions(-1 & 1) να απεικονίζονται σε έναν δευτερεύον αξονα y ώστε να το συγκρινουμε με τα Death & Golden Cross

- ax.get_legend().set_bbox_to_anchor((0.25, 0.85))

#Ορίζουμε την θέση για το legend box σε συγκεκριμένες συντεταγμένες

Ανάλυση χρονολογικών σειρών



Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

Γενικά ο S&P 500 stock index με τον VIX volatility index χαρακτηρίζονται από αρνητική συσχέτιση. Όταν S&P 500 βρίσκεται σε ανοδική τάση ο VIX διαγράφει την αντίθετη πορεία και το αντίστροφο.

https://www.cboe.com/insights/posts/vix-index-characteristics-why-volatility-products-may-provide-unique-hedging-and-income-strategies/

Στις επόμενες διαφάνειες θα χρησιμοποιήσουμε την python & την pandas με σκοπό να επιβεβαιώσουμε αυτήν την υπόθεση.

Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

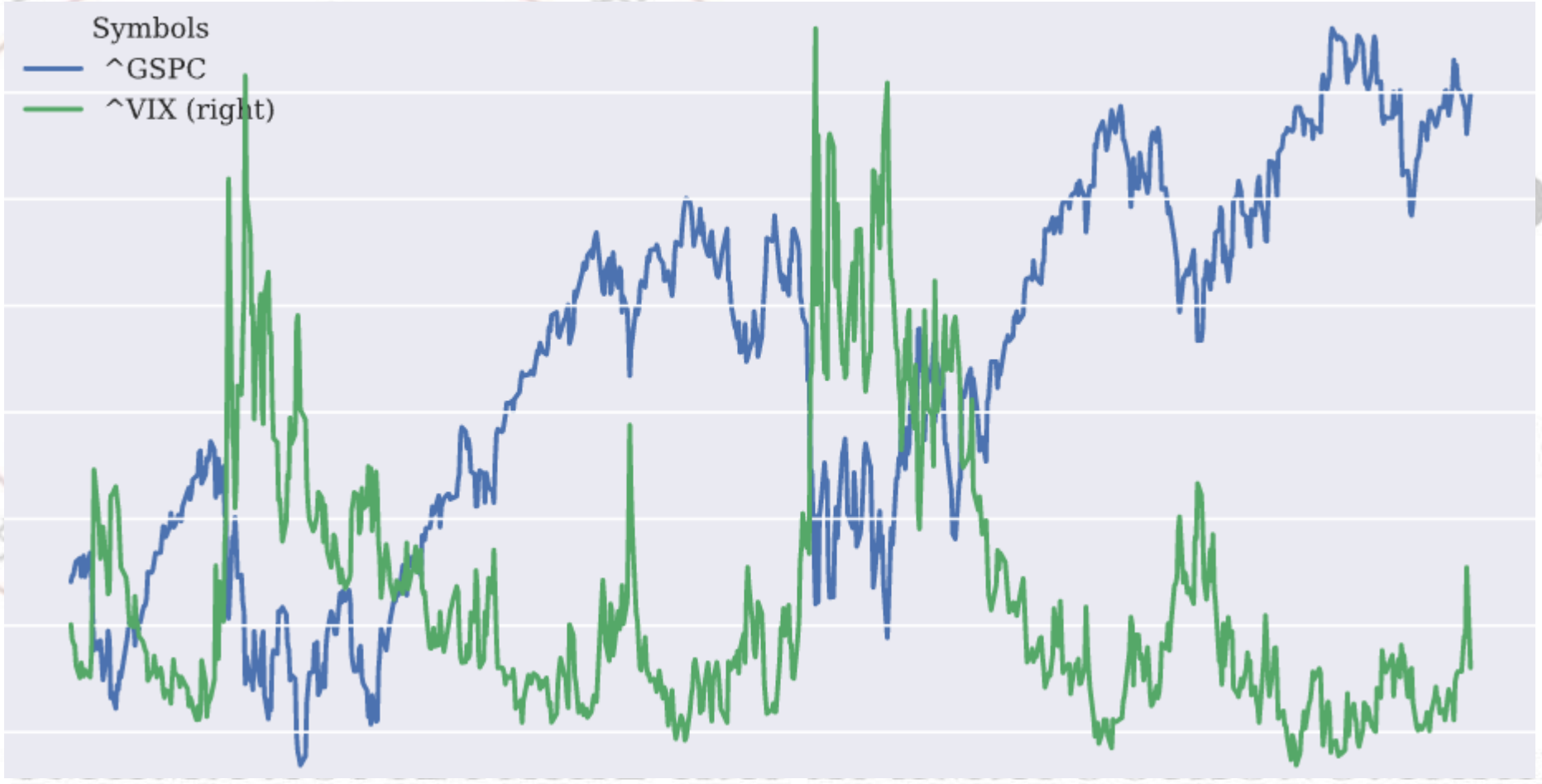
- data = web.DataReader(['^GSPC','^VIX'], 'yahoo',start='2010/01/01', end='2018/06/29')['Close']
- data.tail()
- data.plot(subplots=True, figsize=(10, 6));

Κατά τα γνωστά με την Pandas_Datareader αντλούμε τις τιμές που θέλουμε.[Σημείωση:Θα μπορούσαμε επίσης να εφαρμόσουμε slicing από το αρχικό μας Dataset].

Εφαρμόζουμε τον κλασσικό έλεγχο των δεδομένων μας, αριθμητικά και οπτικά

Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

```
data.loc[:'2012-12-31'].plot(secondary_y='^VIX', figsize=(10, 6));
```



Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

Παρατηρούμε πως, πράγματι οι 2 δείκτες ακολουθούν αντίστροφη πορεία. Σχετικά με το όρισμα secondary_y, στην συγκεκριμένη περίπτωση ήταν απαραίτητο. Θυμηθείτε όταν εκτελέσατε την εντολή

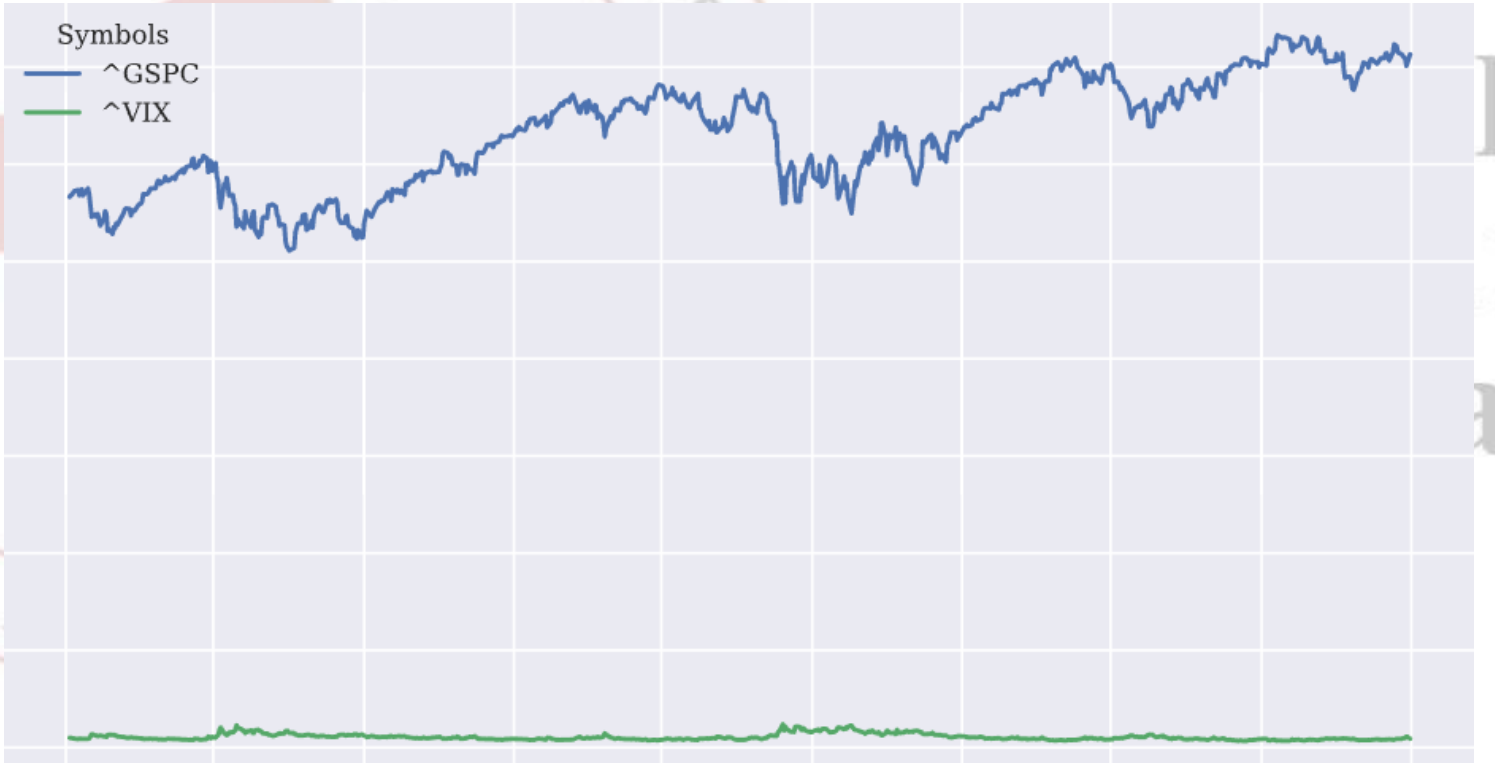
- data.tail()

Symbols	^GSPC	^VIX
Date		
2018-06-25	2717.070068	17.33
2018-06-26	2723.060059	15.92
2018-06-27	2699.629883	17.91
2018-06-28	2716.310059	16.85
2018-06-29	2718.370117	16.09

Η «απόσταση» των τιμών είναι μεγάλη. Οπότε χωρίς το επιπλέον όρισμα θα είχαμε το επόμενο διάγραμμα…

Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

Σε αυτήν την περίπτωση δεν θα ήταν εφικτό να έχουμε το κατάλληλο πόρισμα σχετικά με την συσχέτιση..



Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

Όπως αναφέραμε νωρίτερα στην οικονομική ανάλυση είθισται να γίνεται η ανάλυση σχετικά με τα Log-Returns και όχι τις απόλυτες τιμές της μετοχής μας.

Γενικά υπάρχουν πολλοί λόγοι τόσο θεωρητικοί όσο και αλγοριθμικοί.

Ένας από τους κύριους λόγους είναι πώς, μας προσφέρει μια κανονικοποίηση των δεδομένων μας. Έτσι είμαστε σε θέση να αποτιμήσουμε την σχεση 2 περιουσιακών στοιχείων , ανεξαρτήτως του μεγέθους των τιμών.

Όπως βλέπουμε και στο παράδειγμα μας υπάρχει μεγάλη απόσταση ανάμεσα στις τιμές.

Μπορείτε να ανατρέξετε στο επόμενο link για μια περεταίρω ανάλυση

https://quantivity.wordpress.com/2011/02/21/why-log-returns/

Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

- rets = np.log(data / data.shift(1))
- rets.head()

Παρατηρούμε πως οι τιμές τώρα έχουν αποκτήσει το ίδιο μέγεθος.

Συνεχίζουμε ως εξής:

- rets.dropna(inplace=True) .

Symbols	^GSPC	^VIX
Date		
2010-01-04	NaN	NaN
2010-01-05	0.003111	-0.035038
2010-01-06	0.000545	-0.009868
2010-01-07	0.003993	-0.005233
2010-01-08	0.002878	-0.050024

#Σχετικά με το inplace. Θα μπορούσαμε απλά να γράψουμε rets.dropna(). Βέβαια by default το όρισμα του inplace ισούται με False.Γενικά στην pandas υπάρχουν περιπτώσεις όπου θέλουμε να ορίσουμε μια συνάρτηση με σκοπό, να αποκτήσουμε μια γρήγορη άποψη για κάτι που ερευνάμε χωρίς να επηρεάσουμε τα δεδομένα μας. Γενικά μια σοφή επιλογή, (πιθανόν θα το παρατηρήσετε σε διάφορες εφαρμογές όπου αφορούν δεδομένα) είναι αρχικά να φτιάχνουμε ένα αντίγραφο των δεδομένων μας. Κάνοντας χρήση της .copy().Για παράδειγμα

rets1=rets.copy()

Έτσι, στην περίπτωση που πραγματοποιήσουμε αλλαγές που τελικά ήταν λανθασμένες θα έχουμε τα αρχικά μας δεδομένα ανεπηρέαστα.

Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

- rets.plot(subplots=True, figsize=(10, 6));

Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης

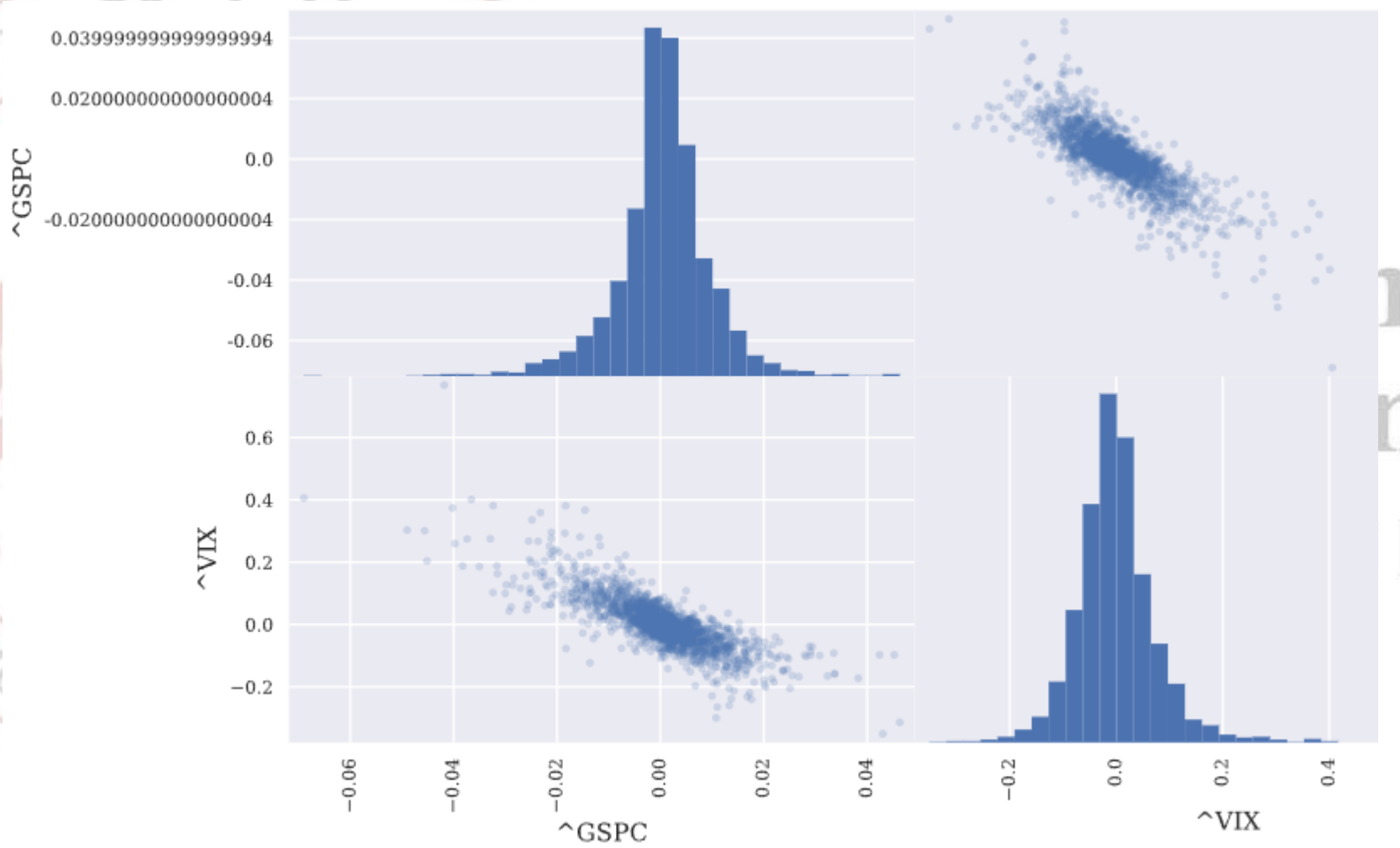
Παρατηρούμε πως υπάρχει διακύμανση στην πάροδο του χρόνου, σχετικα με τις τιμές των 2 δεικτών. Σε περιόδους έντονης αστάθειας των τιμών παρατηρούμε πώς υπάρχει κοινή συμπεριφορά.

Τέλος θα κάνουμε χρήση της scatter.matrix με σκοπό να απεικονίσουμε την αρνητική συσχέτιση των δεδομένων καθώς και την κατανομή των logreturns

```
pd.plotting.scatter_matrix(rets,alpha=0.2,diagonal='hist',  hist_kwds={'bins': 35}, figsize=(10, 6));
```

Το alpha δηλώνει το opacity των dots στο όρισμα diagonal θα μπορούσαμε να ορίσουμε και το ‘kde’

Ανάλυση χρονολογικών σειρών-Ανάλυση συσχέτισης



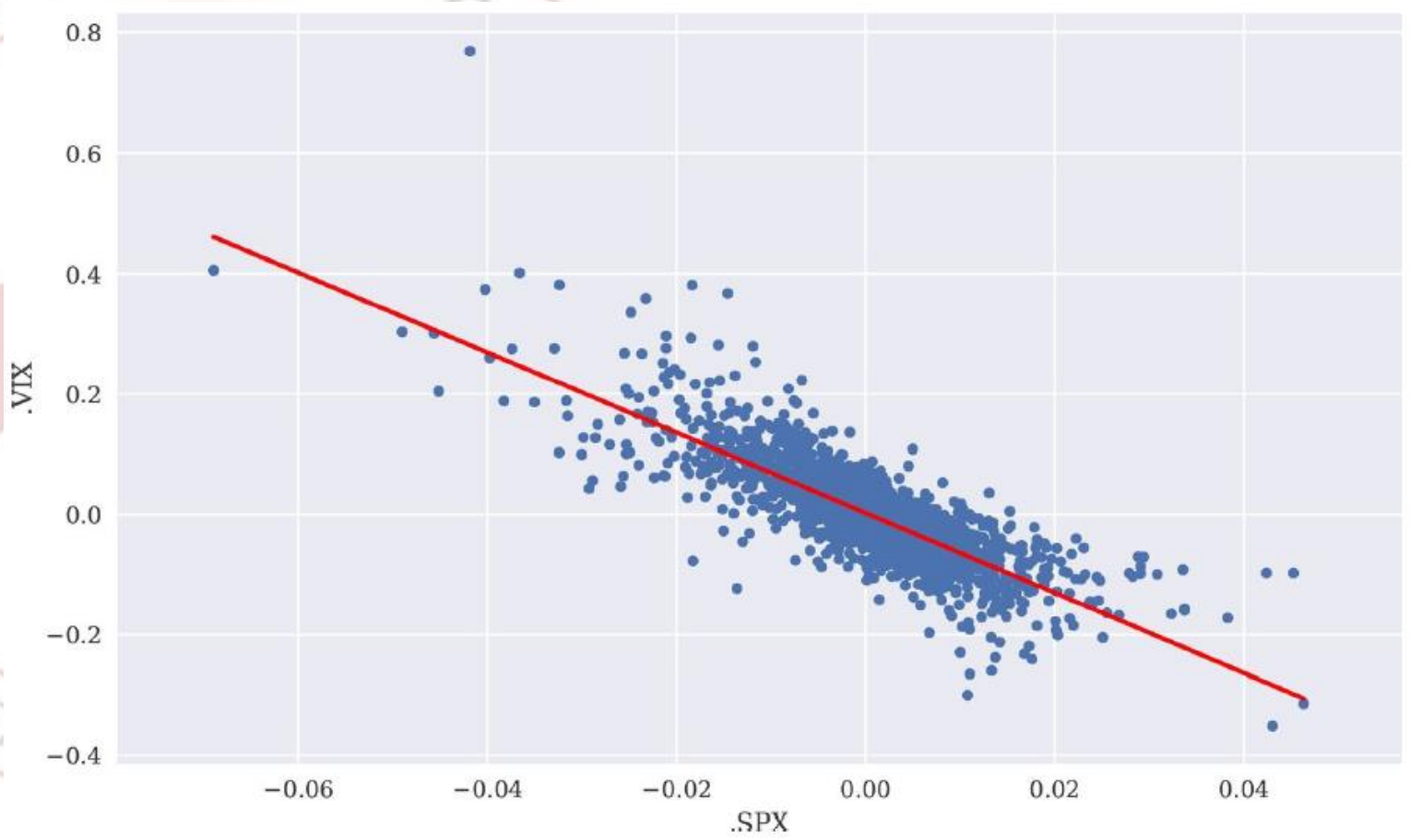
Ανάλυση χρονολογικών σειρών- Παλινδρόμηση

Τέλος θα εφαρμόσουμε ένα μοντέλο παλινδρόμησης για τα δεδομένα μας

- reg = np.polyfit(rets['^GSPC'], rets['^VIX'], deg=1)
- ax = rets.plot(kind='scatter', x='^GSPC', y='^VIX', figsize=(10, 6))
- ax.plot(rets['^GSPC'], np.polyval(reg, rets['^GSPC']), 'r', lw=2);

Παρατηρούμε πως η γραμμή παλινδρόμησης έχει αρνητική κλήση ,ισχυροποιώντας την ισχυρισμό μας.

Ανάλυση χρονολογικών σειρών- Παλινδρόμηση



Ανάλυση χρονολογικών σειρών- Αυτοσυσχέτηση

Βιβλιογραφια

- Manis, G. (2015). Εισαγωγή στον Προγραμματισμό με αρωγό τη γλώσσα Python [Undergraduate textbook]. Kallipos, Open Academic Editions. http://hdl.handle.net/11419/2745
- Magoutis, K., & Nikolaou, C. (2015). Εισαγωγή στον αντικειμενοστραφή προγραμματισμό με Python [Undergraduate textbook]. Kallipos, Open Academic Editions. http://hdl.handle.net/11419/1708
- Yuxing Y. (2014). Python For Finance. Packt Publishing limited.
- Hilpisch Y. (2018). Python for Finance: Mastering Data-Driven Finance[2nd edition]. O'Reilly Media, Inc.