Projet 3

Simulation de portefeuilles et analyse du risque : Portefeuille de 2 ou 3 titres. Comparer la vraie VaR à celle obtenue lorsque la copule reliant ces deux ou trois titres est à faible dépendance en queue gauche (dans ce dernier cas, le portefeuille est simulé à partir de cette copule et des marginales).

Dans ce projet, nous allons nous intéresser à la structure de dépendance entre deux titres. Deux plutôt que trois pour une visualisation plus facile. En tant qu'alternant à CACF, j'ai fait le choix de CA et BNP.

Récupération des données

In [1]:
```
!pip install copulae
```

```
Collecting copulae
  Downloading https://files.pythonhosted.org/packages/43/af/39c226433a708c43
d51e143f107bd9c55ad4681039105944c60c7d2b3521/copulae-0.4.1-cp36-cp36m-manyli
nux2010_x86_64.whl (1.6MB)
     |████████████████████████████████| 1.6MB 2.9MB/s
Requirement already satisfied: scipy>=1.1 in /opt/conda/lib/python3.6/site-p
ackages (from copulae) (1.3.0)
Requirement already satisfied: statsmodels>=0.9 in /opt/conda/lib/python3.6/
site-packages (from copulae) (0.9.0)
Requirement already satisfied: numpy>=1.15 in /opt/conda/lib/python3.6/site-
packages (from copulae) (1.16.4)
Requirement already satisfied: pandas>=0.23 in /opt/conda/lib/python3.6/site
-packages (from copulae) (0.23.4)
Requirement already satisfied: patsy in /opt/conda/lib/python3.6/site-packag
es (from statsmodels>=0.9->copulae) (0.5.1)
Requirement already satisfied: python-dateutil>=2.5.0 in /opt/conda/lib/pyth
on3.6/site-packages (from pandas>=0.23->copulae) (2.8.0)
Requirement already satisfied: pytz>=2011k in /opt/conda/lib/python3.6/site-
packages (from pandas>=0.23->copulae) (2019.1)
Requirement already satisfied: six in /opt/conda/lib/python3.6/site-packages
(from patsy->statsmodels>=0.9->copulae) (1.12.0)
Installing collected packages: copulae
Successfully installed copulae-0.4.1
```

In [2]:
```
!pip install scipy==1.2 --upgrade
```

```
Collecting scipy==1.2
  Downloading https://files.pythonhosted.org/packages/67/e6/6d4edaceee6a110e
cf6f318482f5229792f143e468b34a631f5a0899f56d/scipy-1.2.0-cp36-cp36m-manylinu
x1_x86_64.whl (26.6MB)
     |████████████████████████████████| 26.6MB 2.7MB/s
Requirement already satisfied, skipping upgrade: numpy>=1.8.2 in /opt/conda/
lib/python3.6/site-packages (from scipy==1.2) (1.16.4)
ERROR: allennlp 0.8.4 requires awscli>=1.11.91, which is not installed.
ERROR: allennlp 0.8.4 requires flaky, which is not installed.
ERROR: allennlp 0.8.4 requires responses>=0.7, which is not installed.
ERROR: kmeans-smote 0.1.2 has requirement numpy<1.16,>=1.13, but you'll have
numpy 1.16.4 which is incompatible.
ERROR: kmeans-smote 0.1.2 has requirement scikit-learn<0.21,>=0.19.0, but yo
u'll have scikit-learn 0.21.2 which is incompatible.
Installing collected packages: scipy
  Found existing installation: scipy 1.3.0
    Uninstalling scipy-1.3.0:
      Successfully uninstalled scipy-1.3.0
Successfully installed scipy-1.2.0
```

In [3]:
```python
import os

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import copulae
```

In [4]:
```python
os.listdir("../input")
```

Out[4]:
```
['CAC 40.xlsx']
```

In [5]:
```python
data = pd.read_excel("../input/CAC 40.xlsx")
```

In [6]:
```python
data.head()
```

Out[6]:

	ISIN	JOUR	OUVR	PHAUT	PBAS	CLOT	Volume	NOM	TICKER
0	BE0003470755	2015-10-20	94.18	95.45	92.25	95.12	375579.0	Solvay	SOLB
1	BE0003470755	2015-10-21	95.54	96.24	94.05	95.46	282786.0	Solvay	SOLB
2	BE0003470755	2015-10-22	95.50	99.42	95.10	99.35	551677.0	Solvay	SOLB
3	BE0003470755	2015-10-23	100.65	104.55	100.65	101.85	569759.0	Solvay	SOLB
4	BE0003470755	2015-10-26	101.85	102.25	100.20	100.85	283484.0	Solvay	SOLB

In [7]:
```python
data_clot = data.set_index(["JOUR","NOM"],append=False)["CLOT"]
data_clot.head()
```

Out[7]:
```
JOUR        NOM
2015-10-20  Solvay     95.12
2015-10-21  Solvay     95.46
2015-10-22  Solvay     99.35
2015-10-23  Solvay    101.85
2015-10-26  Solvay    100.85
Name: CLOT, dtype: float64
```

In [8]:
```python
data_clot = data_clot.unstack(level="NOM")
data_clot.head()
```

Out[8]:

NOM	Accor Hotels	Air Liquide	Airbus	Arcelor Mittal	Atos	Axa	Bnp Paribas	Bouygues	CAC40	Cap Gemini	Carrefou
JOUR											
2015-10-20	42.500	99.0909	55.03	15.924	69.59	22.915	54.30	33.970	4673.81	78.41	28.65
2015-10-21	42.850	99.6818	56.28	15.810	70.48	23.095	54.40	34.290	4695.10	78.45	28.71
2015-10-22	44.265	103.0455	57.52	16.170	71.22	23.820	55.01	34.690	4802.18	79.79	29.60
2015-10-23	45.090	105.3636	60.17	16.380	73.00	24.050	55.91	35.175	4923.64	82.45	30.20
2015-10-26	44.930	105.8182	59.95	16.032	72.93	24.155	55.85	35.190	4897.13	82.27	29.95

In [9]:
```python
ca_clot = data_clot["Credit Agricole"]
bnp_clot = data_clot["Bnp Paribas"]
```

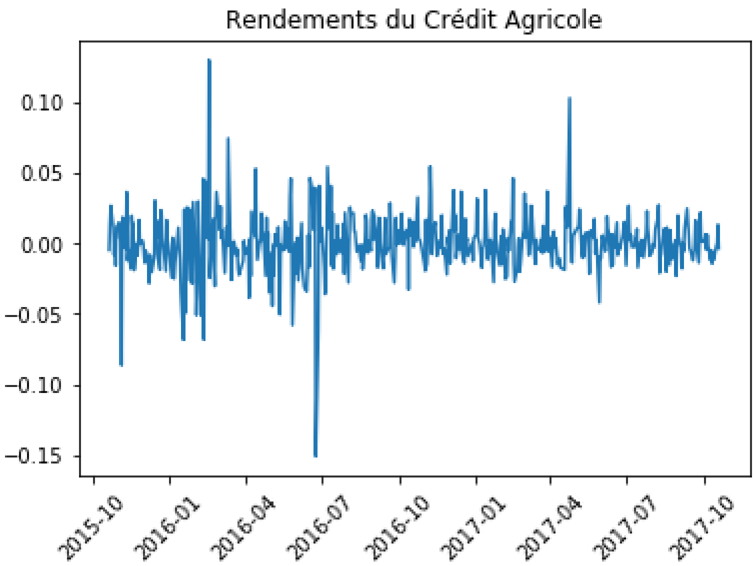
In [10]:
```
ca_returns = np.log(ca_clot/ca_clot.shift()).dropna()
bnp_returns = np.log(bnp_clot/bnp_clot.shift()).dropna()
```

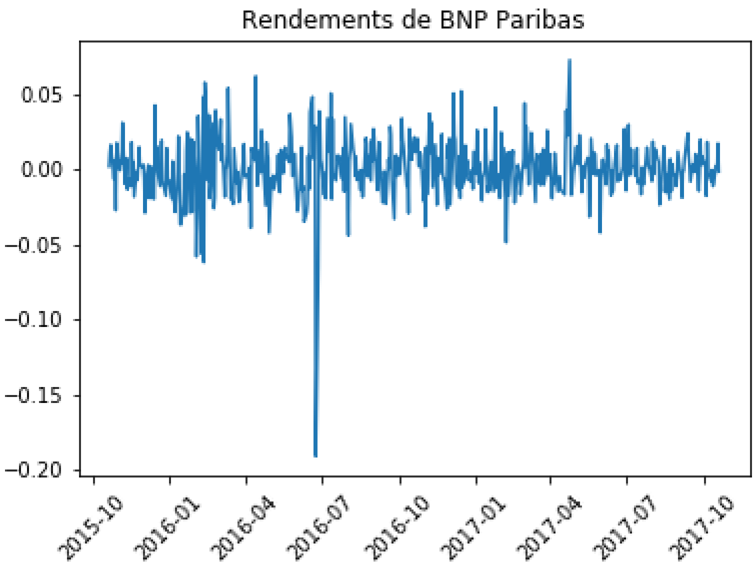
Exploration des données

Regardons à quoi ressemble les rendements de nos titres.

Tout d'abord, voici les rendements individuels

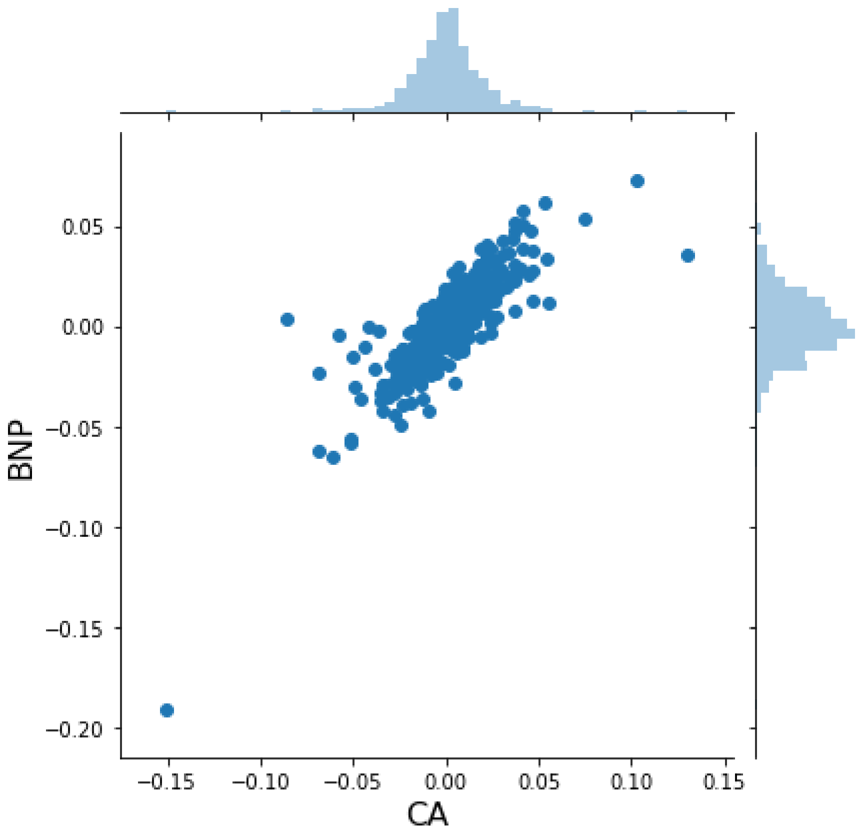
In [11]:
```
plt.plot(ca_returns)
plt.title("Rendements du Crédit Agricole")
plt.xticks(rotation=45)
plt.show()

plt.plot(bnp_returns)
plt.title("Rendements de BNP Paribas")
plt.xticks(rotation=45)
plt.show()
```

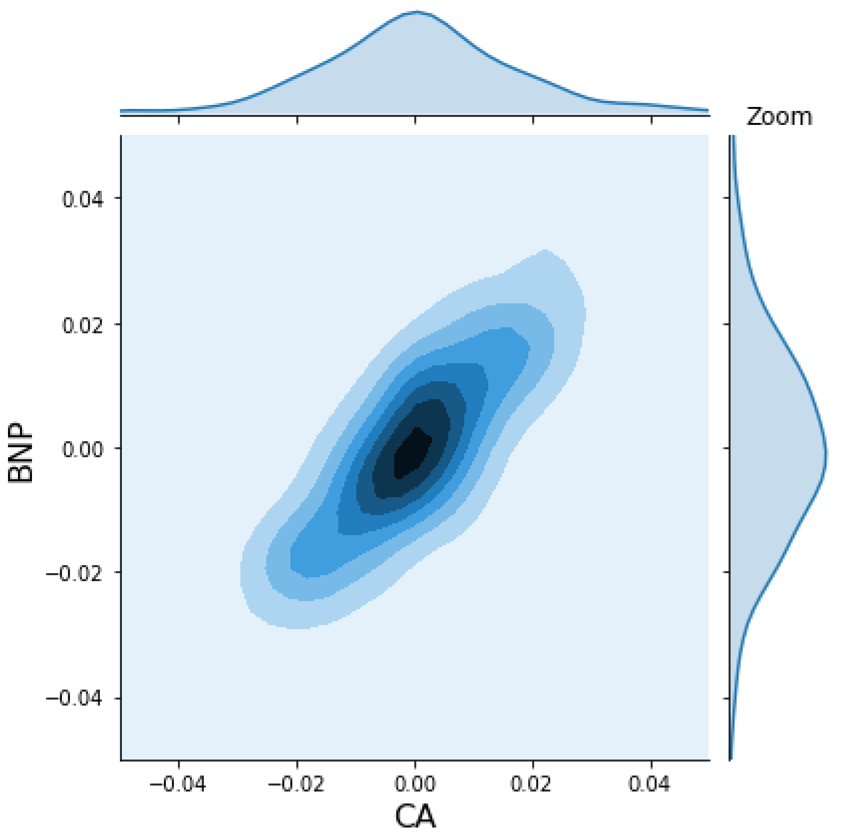




Les rendements du Crédit Agricole sont plus volatils. Les chocs sur ces rendements semblent être corrélés.

In [12]:
```
h= sns.jointplot(x= ca_returns,y=bnp_returns,kind='scatter')
h.set_axis_labels('CA', 'BNP', fontsize=16);

h= sns.jointplot(x= ca_returns,y=bnp_returns,kind='kde',xlim=[-0.05,0.05],ylim=
[-0.05,0.05])
h.set_axis_labels('CA', 'BNP', fontsize=16)
plt.title("Zoom")
plt.show()
```





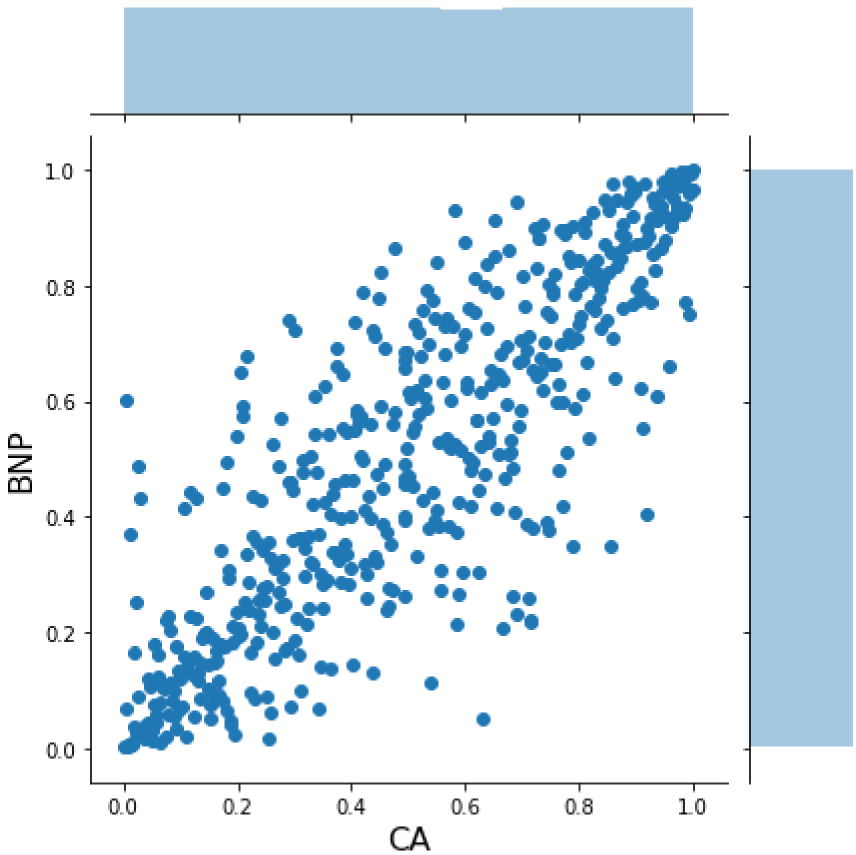
Il y a une corrélation positive des rendements.

In [13]:
```
from statsmodels.distributions.empirical_distribution import ECDF

ca_cdf = ECDF(ca_returns)
bnp_cdf = ECDF(bnp_returns)
```

Voici ci-dessous la fonction de répartition jointe, représentation usuelle des copules.

Les distributions marginales sont bien uniformes, ce qui défini une copule.

In [14]:
```
h= sns.jointplot(x= ca_cdf(ca_returns),y=bnp_cdf(bnp_returns),kind='scatter')
h.set_axis_labels('CA', 'BNP', fontsize=16)

plt.show()
```



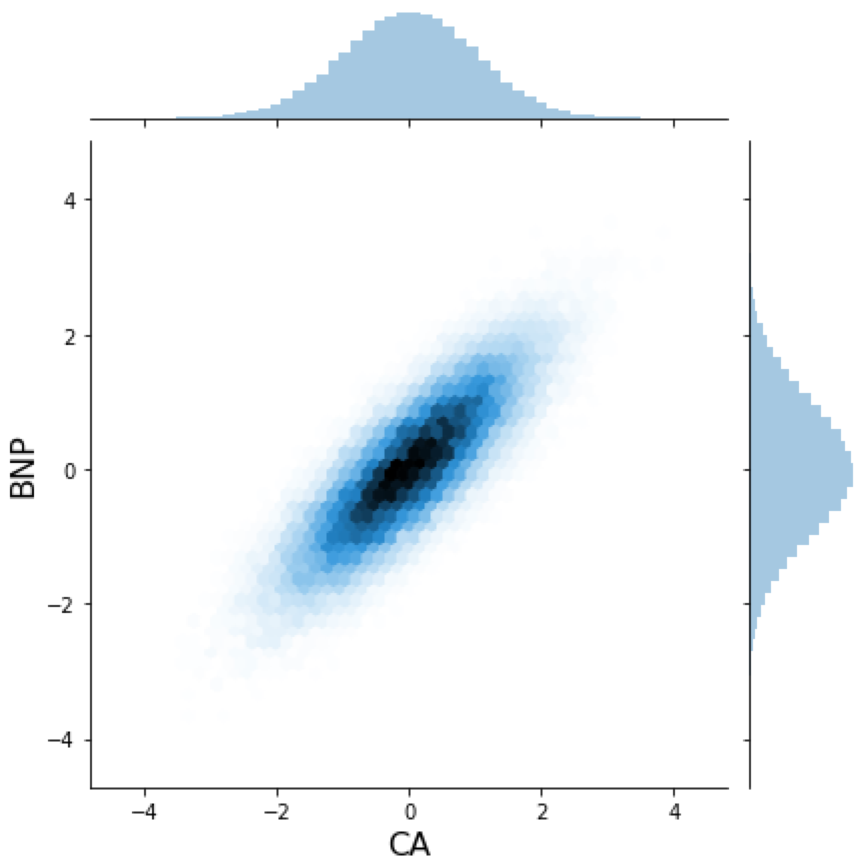
La structure de corrélation semble être symétrique selon la queue de distribution.

Copule Gaussienne

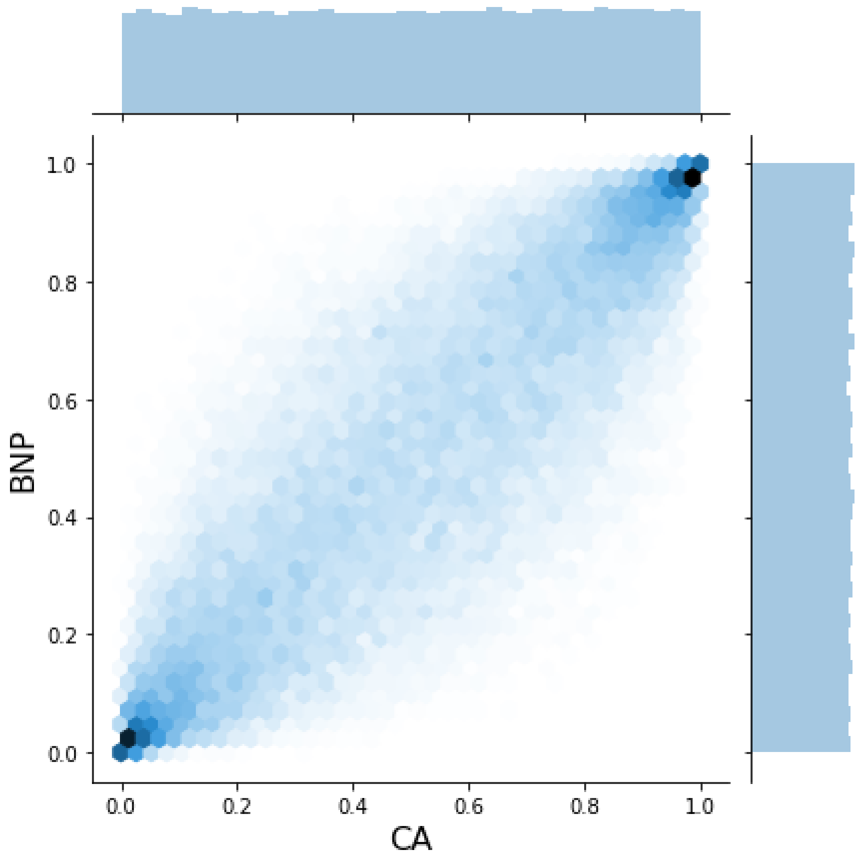
Créons une copule gaussienne depuis un tirage de gaussienne bivariée.

In [15]:
```
from scipy import stats

returns = pd.concat([ca_returns,bnp_returns],axis=1)
mv_norm = stats.multivariate_normal(mean= np.mean(returns,axis=0),cov= returns.
corr())
mv_norm_spl = mv_norm.rvs(returns.shape[0]*100)
```

In [16]:
```
h= sns.jointplot(x= mv_norm_spl[:,0],y= mv_norm_spl[:,1],kind='hex')
h.set_axis_labels('CA', 'BNP', fontsize=16);

plt.show()
```



In [17]:
```
norm = stats.norm()
mv_norm_spl_unif = norm.cdf(mv_norm_spl)

h = sns.jointplot(mv_norm_spl_unif[:, 0], mv_norm_spl_unif[:, 1], kind='hex')
h.set_axis_labels('CA', 'BNP', fontsize=16)

plt.show()
```



La structure de dépendance est symétrique dans les queues

module Copulae

On considère maintenant la copule de Gumbel, qui permet une faible corrélation de queue gauche.

In [18]:
```
c1 = copulae.archimedean.GumbelCopula(dim=2)
c1.fit(data=returns)
c1.summary()
```

Out[18]:

gumbel Copula Summary

gumbel Copula with 2 dimensions

Parameters

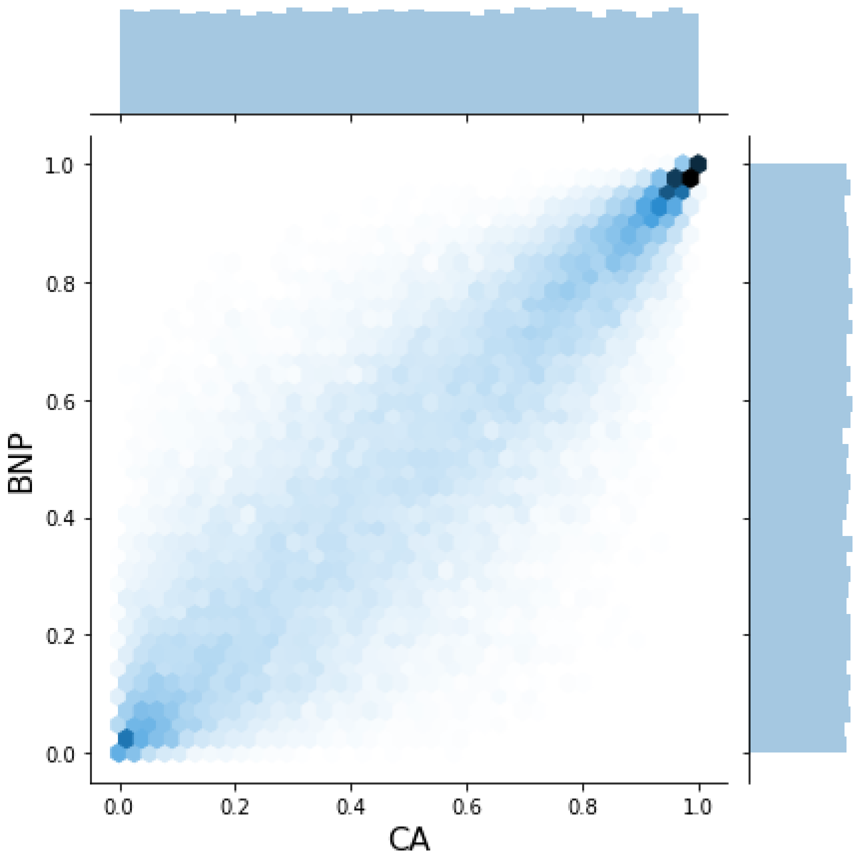
theta2.7233170771252655

Fit Summary

	Fit Summary
Log Likelihood	-318.3710778555788
Variance Estimate	Not Implemented Yet
Method	Maximum pseudo-likelihood
Data Points	514

	Optimization Setup		Results
bounds	[(1.0, inf)]	x	[2.72331708]
options	{'maxiter': 20000, 'ftol': 1e-06, 'iprint': 1, 'disp': False, 'eps': 1.5e-08}	fun	-318.3710778555788
method	SLSQP	jac	[2.27373675e-05]
None	None	nit	4
None	None	nfev	13
None	None	njev	4
None	None	status	0
None	None	message	Optimization terminated successfully.
None	None	success	True

In [19]:
```
c1_cdf = c1.random(n=returns.shape[0]*100)
h = sns.jointplot(c1_cdf[:, 0],c1_cdf[:, 1], kind='hex')
h.set_axis_labels('CA', 'BNP', fontsize=16);
```

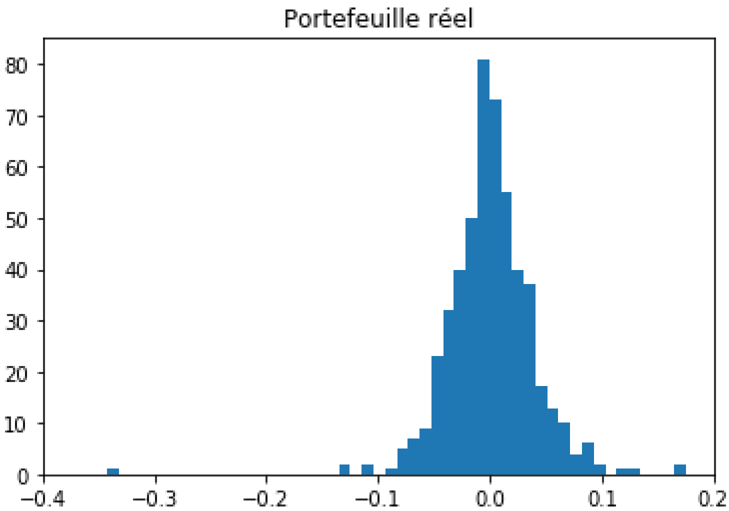


Calcul de la VaR

Les portefeuilles sont équipondérés par les prix (même nombre d'actions par entreprise)

Portefeuille Empirique

In [20]:
```
from scipy.stats import kurtosis, skew

pf_reel = returns.sum(axis=1)
plt.hist(pf_reel,bins=50)
plt.xlim([-0.4,0.2])
plt.title("Portefeuille réel")
plt.show()

print("Skewness: {} \nKurtosis: {}".format(skew(pf_reel),kurtosis(pf_reel)))
```

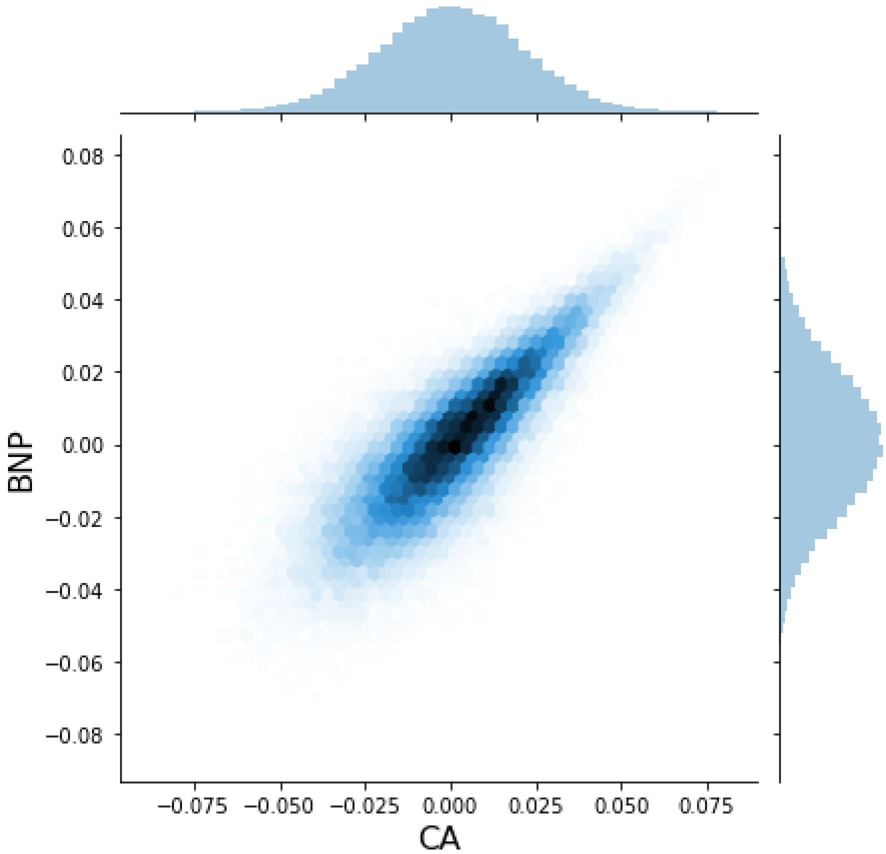



```
Skewness: -0.9747186164584692
Kurtosis: 12.227033324154064
```

Portefeuille avec copule de Gumbel

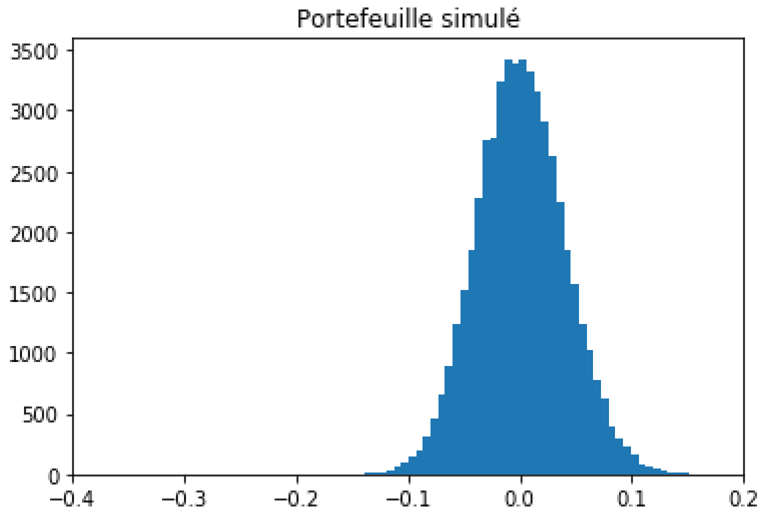
On suppose que les distributions marginales (les rendements individuels) sont normales !

In [21]:
```
m1 = stats.norm(loc=ca_returns.mean(),scale=ca_returns.std())
m2 = stats.norm(loc=bnp_returns.mean(),scale=bnp_returns.std())

ca_returns_m1 = m1.ppf(c1_cdf[:, 0])
bnp_returns_m2 = m2.ppf(c1_cdf[:, 1])

h = sns.jointplot(ca_returns_m1,bnp_returns_m2, kind='hex')
h.set_axis_labels('CA', 'BNP', fontsize=16)

plt.show()
```



La copule modélise bien une distribution jointe (de gaussiennes) à structure de corrélation assymétrique.

In [22]:
```
returns_c1 = pd.DataFrame(np.vstack([ca_returns_m1,bnp_returns_m2]).T)
pf_c1 = returns_c1.sum(axis=1)
plt.hist(pf_c1,bins=50)
plt.xlim([-0.4,0.2])
plt.title("Portefeuille simulé")
plt.show()


print("Skewness: {} \nKurtosis: {}".format(skew(pf_c1),kurtosis(pf_c1)))
```




```
Skewness: 0.10199439397559272
Kurtosis: 0.021726600851515787
```

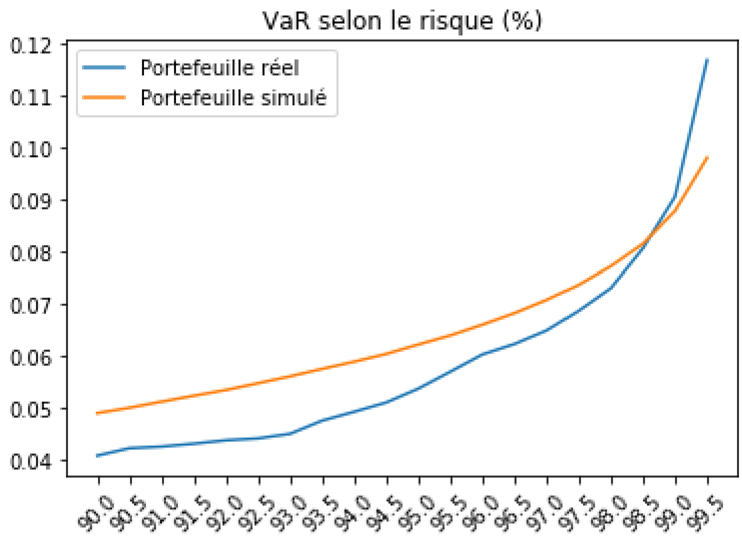
les queues sont bien moins épaisses par rapport au réel.

Les VaR

In [23]:
```
def vars(percentile,pf_c):
    var_reel = -1 * np.percentile(pf_reel, percentile)
    var_c = -1 * np.percentile(pf_c, percentile)
    return var_reel,var_c
```

```
In [24]: VaRs = pd.DataFrame([vars(i,pf_c1) for i in np.linspace(0.5,10,(10/0.5))],
                     columns=['réel','simulé'],
                     index= 100 - np.linspace(0.5,10,(10/0.5)))

         plt.plot(VaRs['réel'],label='Portefeuille réel')
         plt.plot(VaRs['simulé'],label='Portefeuille simulé')
         plt.title('VaR selon le risque (%)')
         plt.xticks(100 - np.linspace(0.5,10,(10/0.5)))
         plt.xticks(rotation=45)
         plt.legend()
         plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationW
arning: object of type <class 'float'> cannot be safely interpreted as an in
teger.
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: DeprecationW
arning: object of type <class 'float'> cannot be safely interpreted as an in
teger.
  This is separate from the ipykernel package so we can avoid doing imports
until
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:8: DeprecationW
arning: object of type <class 'float'> cannot be safely interpreted as an in
teger.
```



La copule de Gumbel donne logiquement une VaR inférieure à risque faible par rapport au réel. L'idée que les chocs négatifs sont idiosyncratiques est irréel.

Une copule de Clayton ou de Frank donnerait une VaR sûrement plus proche du portefeuille réel.

De plus, nous sommes maintenant coutûmiers du fait, modéliser les rendements individuels par des gaussiennes est incorrect. Des distributions plus épaisses dans les queues représentent mieux les séries financières.

On peut par exemple utiliser des students ou des lois stables.

Copule de Clayton et loi de student

In [25]:
```
c2 = copulae.archimedean.ClaytonCopula(dim=2)
c2.fit(data=returns)
c2.summary()
```

Out[25]:

clayton Copula Summary

clayton Copula with 2 dimensions

Parameters

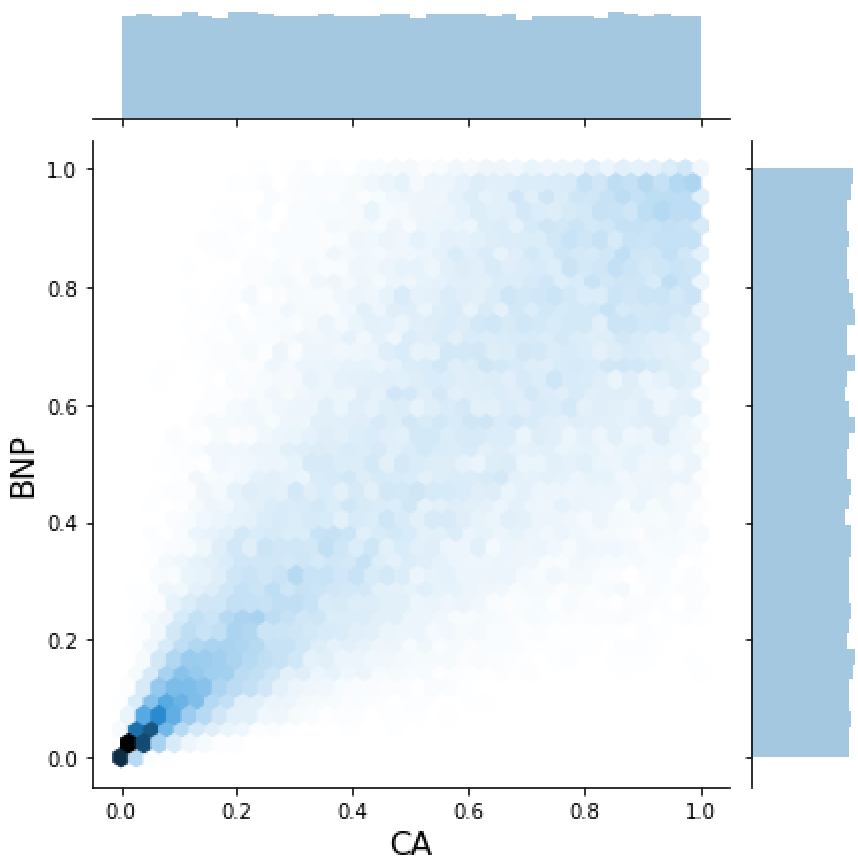
theta2.4177754076131914

Fit Summary

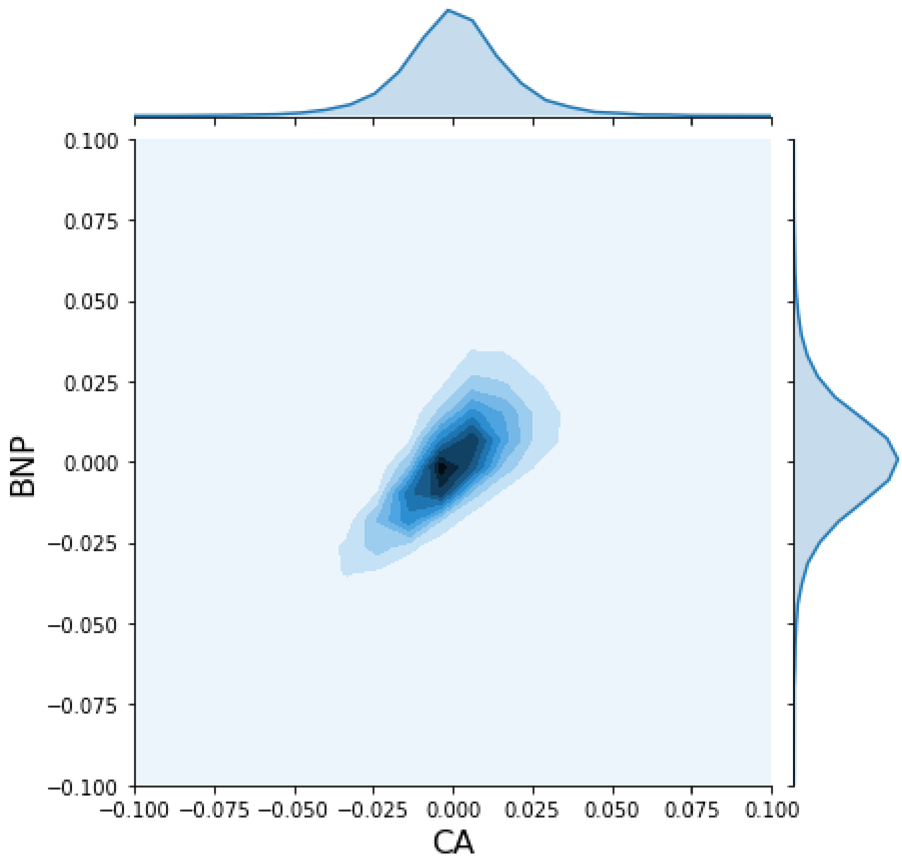
	Fit Summary
Log Likelihood	-255.79589590464178
Variance Estimate	Not Implemented Yet
Method	Maximum pseudo-likelihood
Data Points	514

	Optimization Setup		Results
bounds	[(-0.9999999999999998, inf)]	x	[2.41777541]
options	{'maxiter': 20000, 'ftol': 1e-06, 'iprint': 1, 'disp': False, 'eps': 1.5e-08}	fun	-255.79589590464178
method	SLSQP	jac	[-0.0050458]
None	None	nit	6
None	None	nfev	19
None	None	njev	6
None	None	status	0
None	None	message	Optimization terminated successfully.
None	None	success	True

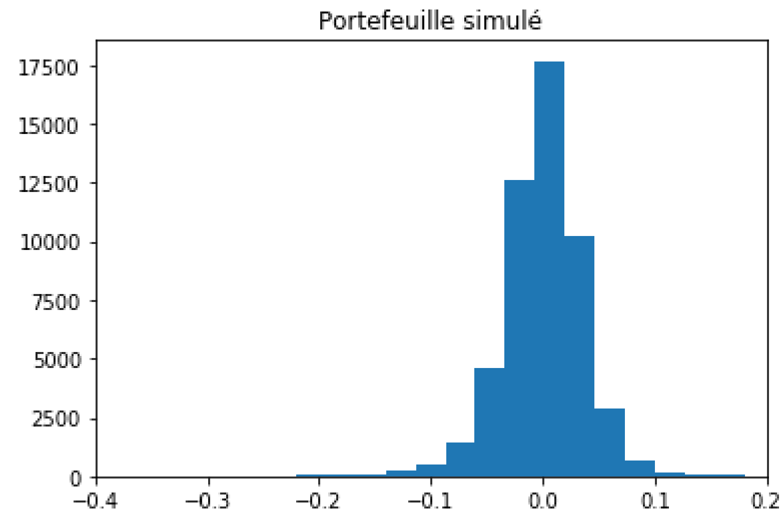
In [26]:
```
c2_cdf = c2.random(n=returns.shape[0]*100)
h = sns.jointplot(c2_cdf[:, 0],c2_cdf[:, 1], kind='hex')
h.set_axis_labels('CA', 'BNP', fontsize=16);
```



In [27]:
```
l1 = stats.t.fit(ca_returns)
l2 = stats.t.fit(bnp_returns)

l1 = stats.t(l1[0],l1[1],l1[2])
l2 = stats.t(l2[0],l2[1],l2[2])

ca_returns_l1 = l1.ppf(c2_cdf[:, 0])
bnp_returns_l2 = l2.ppf(c2_cdf[:, 1])

h = sns.jointplot(ca_returns_l1,bnp_returns_l2, kind='kde')
h.set_axis_labels('CA', 'BNP', fontsize=16)
h.ax_marg_x.set_xlim(-0.1,0.1)
h.ax_marg_y.set_ylim(-0.1,0.1)
plt.show()
```



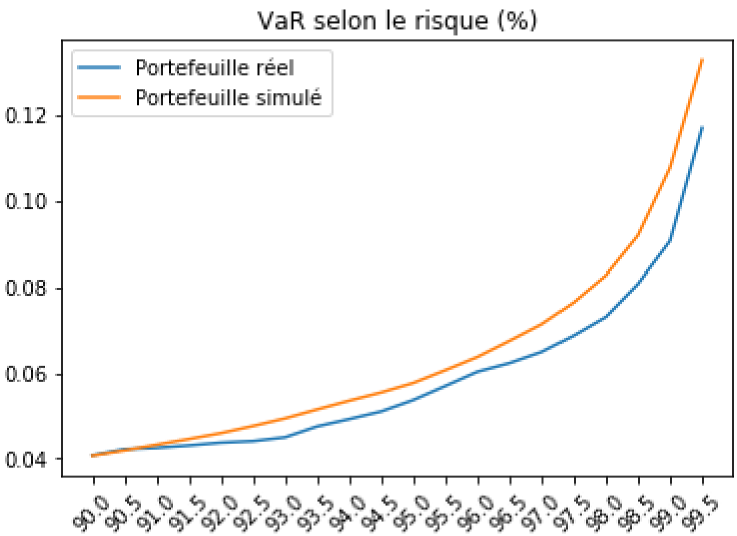
In [28]:
```python
returns_c2 = pd.DataFrame(np.vstack([ca_returns_l1,bnp_returns_l2]).T)
pf_c2 = returns_c2.sum(axis=1)
plt.hist(pf_c2,bins=50)
plt.xlim([-0.4,0.2])
plt.title("Portefeuille simulé")
plt.show()


print("Skewness: {} \nKurtosis: {}".format(skew(pf_c2),kurtosis(pf_c2)))
```




```
Skewness: -0.7025874731809285
Kurtosis: 12.666761990l6882
```

In [29]:
```
VaRs = pd.DataFrame([vars(i,pf_c2) for i in np.linspace(0.5,10,(10/0.5))],
                    columns=['réel','simulé'],
                    index= 100 - np.linspace(0.5,10,(10/0.5)))

plt.plot(VaRs['réel'],label='Portefeuille réel')
plt.plot(VaRs['simulé'],label='Portefeuille simulé')
plt.title('VaR selon le risque (%)')
plt.xticks(100 - np.linspace(0.5,10,(10/0.5)))
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationW
arning: object of type <class 'float'> cannot be safely interpreted as an in
teger.
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: DeprecationW
arning: object of type <class 'float'> cannot be safely interpreted as an in
teger.
  This is separate from the ipykernel package so we can avoid doing imports
until
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:8: DeprecationW
arning: object of type <class 'float'> cannot be safely interpreted as an in
teger.
```



Cette fois, la VaR simulée est bien supérieure à l'empirique.

L'interêt des copules pour la simulation de données multivariées à structure spécifique de dépendance est très intéressant. La modulation de la VaR en fonction de la copule et des lois marginales de rendement donnent à réfléchir quant à la modélisation de la VaR réglémentaire bâloise.