On the properties of equally-weighted risk contributions portfolios

Steven François

Mars 2023

1 Introduction

Nous nous intéressons à la problématique d'allocation d'actifs pour un **portefeuille optimisé**, il s'agit donc de définir ce que l'on entend par "optimisé". Pour cela, nous allons dans un premier temps aborder les quelques solutions connues dans la littérature afin d'en tirer quelques propriétés, avantages et incovénients, ce qui nous permettra de préciser ce que l'on souhaite comme portefeuille.
Le modèle de référence est celui formulé par **Markowitz** dans les années 50, c'est un portefeuille construit à partir d'une réflexion sur l'espérance et la variance, en maximisant la première tout en fixant le niveau de la seconde.
Ce portefeuille est parfaitement intuitif puisqu'il propose des retours maximaux pour un profil de risque donné mais présente deux inconvénients majeurs : une tendance à **concentrer les actifs choisis**, ce qui présente des risques en pratique, et un manque de **robustesse** dû à la sensibilité aux paramètres d'entrée, notamment ceux des **gains attendus**.
Des alternatives ont vu le jour au fur et à mesure mais présentent toutes leur lot de défauts et surtout davantage de **complexité** pour leur mise en place ainsi que des performances **"out-of-sample"**, c'est à dire celles que l'on subit effectivement, qui ne sont pas nécessairement meilleures que les traditionnelles.
Nous voyons donc avec ce cheminement que les deux préoccupations principales sont désormais la **complexité** de mise en place ainsi que la **robustesse**, qui sera recherchée en éliminant les paramètres de gains attendus.
Deux modèles populaires relevant ces challenges sont les portefeuilles de **variance minimale** et de **poids égaux**.
Le premier reprend le modèle espérance-variance en se plaçant sur la **frontière efficiente**, c'est à dire là où on ne peut augmenter l'espérance sans augmenter la variance, et en proposant une certaine **solution unique**, ce qui facilite la mise en place effective du portefeuille. Néanmoins, ce modèle souffre toujours de la **concentration des actifs choisis**, et à ce problème se présente le second portefeuille, qui a beau être sans doute le plus **simple** possible, se révèle être populaire.
Notre étude portera sur ces deux modèles mais aussi et surtout sur un troisième, hybride des deux, appelé portefeuille à **contributions au risque égales** (en considérant la volatilité comme mesure du risque), c'est à dire que l'on égalise les

$$\sigma_i(x) = x_i \times \partial_{x_i}\sigma(x)$$

- x : le vecteur des poids des actifs dans le portefeuille
- $\sigma_i(x)$: la **contribution au risque** de l'actif i
- x_i : le **poids** de l'actif i dans le portefeuille
- $\partial_{x_i}\sigma(x)$: la **contribution au risque marginale** de l'actif i, donné en observant le changement de risque total du portefeuille lorsque l'on augmente p_i

Cette idée est dans un premier temps relativement intuitive puisque l'on va venir **diversifier les risques**, ce qui paraît assez naturel. Mais il a été aussi montré que raisonner par contributions au risque et non plus par poids est **pertinent financièrement** car cela prédit mieux les **contributions aux pertes** et que diversifier les risques peut amener à de **meilleurs gains**. Enfin, de manière plus théorique, ces portefeuilles rejoignent la **solution optimale** du problème de Markowitz. L'étude visera donc à cerner les propriétés de ces portefeuilles et à backtester les chiffres que l'on obtiendrait, en comparaison avec les deux modèles précédemment évoqués. Nous les noterons portefeuilles ERC à l'anglaise pour ne pas compliquer la lecture.

2 Définition des portefeuilles ERC

Nous avons introduit les premières notations au paragraphe précédent mais nous devons maintenant spécifier celles qui nous permettront de développer l'égalité :

- σ_i^2 : **variance** de l'actif i
- σ_{ij} : **covariance** des actifs i et j
- Σ : **matrice de covariance** des actifs

Dès lors, le risque total du portefeuille s'écrit naturellement :

$$\sigma(x) = \sqrt{x^\top \Sigma x} = \sqrt{\sum_i x_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} x_i x_j \sigma_{ij}} \text{ : le } \textbf{risque total} \text{ du portefeuille}$$

(NB : coquille dans l'article avec oubli de la racine)

Nous pouvons ainsi développer notre définition de la **contribution au risque marginale** puisqu'il s'agissait de la dérivée partielle de σ :

$$\partial_{x_i} \sigma(x) = \frac{\partial \sigma(x)}{\partial x_i} = \frac{x_i \sigma_i^2 + \sum_{j \neq i} x_j \sigma_{ij}}{\sigma(x)}$$

Maintenant que les notations ont été mises en place et que les premières relations ont été développées, l'idée d'un portefeuille ERC s'écrit par conséquent :

$$x^\star = \left\{ x \in [0,1]^n : \sum x_i = 1, x_i \times \partial_{x_i} \sigma(x) = x_j \times \partial_{x_j} \sigma(x) \; \forall \; i, j \right\}$$

Ce qui, par un jeu de réécriture, s'exprime aussi de cette manière :

$$x^\star = \left\{ x \in [0,1]^n : \sum x_i = 1, x_i \times (\Sigma x)_i = x_j \times (\Sigma x)_j \; \forall \; i, j \right\}$$

Cela conclut donc les définitions du modèle.

3 Propriétés des portefeuilles ERC

3.1 Pour deux actifs

Nous allons d'abord prendre le cas à **deux actifs** afin de pouvoir obtenir plus facilement des résultats préliminaires et des premières conclusions. Ce cas est davantage pratique puisque l'on peut dès lors étudier **une seule corrélation** ρ et **un seul poids** w.
Les deux contributions au risque s'écrivent ainsi très simplement vu les formules précédentes et comme $\sigma_{ij} = \rho \sigma_1 \sigma_2$:

$$\frac{1}{\sigma(x)} \begin{pmatrix} w^2\sigma_1^2 + w(1-w)\rho\sigma_1\sigma_2 \\ (1-w)^2\sigma_2^2 + w(1-w)\rho\sigma_1\sigma_2 \end{pmatrix}$$

Le problème est donc de les égaliser à partir du paramètre poids w. On veut ainsi :

$$w^2\sigma_1^2 = (1-w)^2\sigma_2^2$$

Cela nous ramène à une équation du second degré : $(\sigma_1^2 - \sigma_2^2)w^2 + 2\sigma_2^2 w - \sigma_2^2 = 0$ qui nous donne comme seule solution comprise entre 0 et 1 : $\frac{\sigma_2}{\sigma_1+\sigma_2}$ ce que l'on préfère écrire avec division par $\sigma_1\sigma_2$: $\frac{\sigma_1^{-1}}{\sigma_1^{-1}+\sigma_2^{-1}}$ On obtient par conséquent notre portefeuille optimisé $x = (w, 1-w)$:

$$x^* = \left(\frac{\sigma_1^{-1}}{\sigma_1^{-1}+\sigma_2^{-1}}, \frac{\sigma_2^{-1}}{\sigma_1^{-1}+\sigma_2^{-1}}\right)$$

La solution est unique et très facile à mettre en place une fois les volatilités estimées, on note au passage une **indépendance à la corrélation** ce qui simplifie grandement le portefeuille : nous n'avons pas à estimer ce paramètre mais on perd une dimension sur laquelle jouer et donc potentiellement de la performance.
On voit par ailleurs que plus la volatilité d'un actif va augmenter, avec celle de l'autre restant fixe, plus son poids baissera, ce qui est effectivement ce qui est recherché dans le concept de ce modèle.

3.2 Pour n actifs

3.2.1 À corrélations égales

Le nombre de paramètres augmentant de l'ordre de n^2 pour les corrélations, nous allons d'abord commencer par une seule, à laquelle toutes sont égales. Nous restons donc avec le paramètre ρ mais ne pouvons plus utiliser l'unique poids w car nous perdons la symétrie que nous avions pour deux actifs. Les contributions au risque s'écrivent alors :

$$\sigma_i(x) = x_i\sigma_i \left((1-\rho)x_i\sigma_i + \rho\sum_j x_j\sigma_j\right)/\sigma(x)$$

On voit alors que le terme en somme sur j sera le même pour chaque contribution et que par conséquent optimiser le portefeuille revient à égaliser :

$$x_i\sigma_i = x_j\sigma_j \quad \forall i, j$$

Cela nous donne, couplé à $\sum_i x_i = 1$:

$$x_i^* = \frac{\sigma_i^{-1}}{\sum_{j=1}^n \sigma_j^{-1}}$$

On y voit une extension du résultat à deux actifs, en gardant l'inverse de la volatilité de l'actif au numérateur et la somme des inverses de toutes les volatilités au dénominateur. On en tire par conséquent les mêmes conclusions avec **l'indépendance à la corrélation** et la **diminution du poids lorsque la volatilité augmente**.

3.2.2 À volatilités égales

Maintenant, nous récupérons la diversité des corrélations mais nous fixons les volatilités sur une seule même constante, notée σ. Cela nous donne des contributions au risque s'écrivant (puisque $\rho_{ii} = 1$) :

$$\sigma_i(x) = x_i\sigma^2 \left(\sum_j x_j\rho_{ij}\right)/\sigma(x)$$

On voit donc qu'il s'agira cette fois d'égaliser :

$$x_i \sum_j x_j \rho_{ij} = x_k \sum_j x_j \rho_{kj} \quad \forall i,k$$

On récupère, selon le même raisonnement algébrique qu'à corrélations égales :

$$x_i^* = \frac{\left(\sum_{j=1}^n x_j \rho_{ij}\right)^{-1}}{\sum_{k=1}^n \left(\sum_{j=1}^n x_j \rho_{kj}\right)^{-1}}$$

Nous voyons que les contributions aux poids sont le symétrique du cas précédent, ce qui est naturel, avec une **indépendance au paramètre fixé, la volatilité**, et une dépendance à toutes les corrélations, ce qui crée un grand nombre de paramètres à évaluer.
Il faut aussi noter que les poids eux-mêmes interviennent dans la formule, ainsi nous n'obtenons **plus de solution explicite**.
Cela complique grandement l'interprétation du résultat, puisqu'on a dès lors du mal à véritablement dire si une augmentation des corrélations avec les autres actifs résulte réellement en une diminution du poids ou si une corrélation proche de 1 avec un autre actif résulte en une égalité des deux poids...
De plus, et c'est primordial, on perd la facilité de mise en place de la stratégie que l'on avait avec les solutions explicites des cas précédents.

3.2.3 Cas général

Dans le cas général, nous avons plus de deux actifs ainsi que des volatilités et des corrélations différentes entre elles.
Vu les résultats précédents, notamment ceux à corrélations non égales, on peut s'attendre à une solution non explicite, puisque l'on ne fait que compliquer davantage ce cas en faisant aussi varier les volatilités.
Les résultats préliminaires obtenus avec les cas simples nous ont permis de bien vérifier que les volatilités et les corrélations avaient tout leur rôle dans la répartition des poids, et qu'ainsi il était intéressant d'employer le **beta** des actifs avec le portefeuille, rendant compte de l'évolution de leurs rendements par rapport à celui du portefeuille en lui-même, ce qui engage les volatilités et les corrélations :

$$\beta_i = cov\left(r_i, \sum_j x_j r_j\right) / \sigma^2(x) \text{ avec } r_i \text{ le rendement de l'actif } i$$

Or $cov\left(r_i, \sum_j x_j r_j\right) = x_i \sigma_i^2 + \sum_{j \neq i} x_j \sigma_{ij}$ donc les contributions au risque s'écrivent :

$$\sigma_i(x) = x_i \beta_i \sigma(x)$$

Cela nous donne par égalisation des contributions à $\sigma(x)/n$:

$$x_i^* = \frac{\beta_i^{-1}}{\sum_{j=1}^n \beta_j^{-1}} = \frac{\beta_i^{-1}}{n}$$

L'étude du beta est donc effectivement pertinente puisque **le poids d'un actif dans le portefeuille ERC sera tout simplement inversement proportionnelle à son beta**.
Le beta rendant compte de la volatilité et des corrélations, on conclut donc que **plus la volatilité ou la corrélation avec les autres actifs seront grandes et plus le poids sera maigre**, ce qui est une synthèse des cas étudiés.
Nous gardons toujours cependant le **caractère non explicite de la solution**, le beta dépendant du portefeuille et donc des poids, ce qui est important à noter vis-à-vis de la grande question de la complexité de mise en place.

3.3 Résolution numérique

Afin de pallier cette absence de solution explicite, nous devons employer un **algorithme numérique** nous permettant d'obtenir de manière effective les poids de notre portefeuille.
Le premier proposé reprend la deuxième définition du portefeuille ERC :

$$x^\star = \{x \in [0,1]^n : \sum x_i = 1, x_i \times (\Sigma x)_i = x_j \times (\Sigma x)_j \ \forall \ i,j\}$$

L'algorithme prend une forme de **problème des moindres carrés** en **minimisant l'erreur quadratique** :

$$x^\star = \underset{x \in U}{argmin} \sum_{i=1}^n \sum_{j=1}^n \left(x_i(\Sigma x)_i - x_j(\Sigma x)_j\right)^2 \text{ avec } U = \{\ x \mid \mathbf{1}^\top x = 1\ ,\ \mathbf{0} \leq x \leq \mathbf{1}\}$$

On a donc existence du portefeuille ERC lorsque $f(x^\star) = 0$, solution pour laquelle toutes les contributions au risque sont égales.
Une relaxation de cet algorithme lorsqu'une solution n'est pas trouvée est de prendre la contrainte $\mathbf{1}^\top y \geq c$ et non plus $\mathbf{1}^\top x = 1$ puis d'obtenir le portefeuille ERC en normalisant $x_i^\star = y_i^* / \sum_{i=1}^n y_i^*$ pour $f\left(y^\star\right) = 0$.
Un autre algorithme employable se formule ainsi :

$$\begin{aligned} y^\star &= \arg\min \sqrt{y^\top \Sigma y} \\ \text{s.c. } &\begin{cases} \sum_{i=1}^n \ln y_i \geq c \\ y \geq \mathbf{0} \end{cases} \end{aligned}$$

Ce qui représente toujours à peu près une **minimisation de variance** mais avec comme contrainte une **diversification des poids** (la constante c en étant le paramètre fixé, choisi le plus petit possible en ayant une solution). On normalise ensuite $x_i^\star = y_i^* / \sum_{i=1}^n y_i^*$ pour le portefeuille ERC. La solution est unique lorsque Σ est définie positive.

3.4 Comparaison avec les deux autres portefeuilles

Nous avons pu définir et étudier le portefeuille ERC, avec des formules fermées pour les cas simples et une **résolution numérique** pour le cas général. Il s'agit maintenant d'évaluer cette stratégie et nous allons pour cela la **comparer** aux deux autres portefeuilles qui nous ont intéressés durant l'introduction : le portefeuille de **poids égaux** et le portefeuille **à variance minimale**.
Ces modèles égalisent chacun des contributions : celui à poids égaux les **contributions totales**, celui à variance minimale les **contributions au risque marginales** et l'ERC les **contributions au risque** :

$$\begin{array}{ll} x_i = x_j & (1/n) \\ \partial_{x_i}\sigma(x) = \partial_{x_j}\sigma(x) & (\text{vm}) \\ x_i\partial_{x_i}\sigma(x) = x_j\partial_{x_j}\sigma(x) & (\text{erc}) \end{array}$$

3.4.1 Pour deux actifs

Dans le cas à deux actifs, nous avions pour le portefeuille ERC un poids :

$$w_{erc}^* = \frac{\sigma_1^{-1}}{\sigma_1^{-1} + \sigma_2^{-1}}$$

Le poids pour le portefeuille à poids égaux vaut bien entendu :

$$w_{1/n}^* = \frac{1}{2}$$

Et celui du portefeuille à variance minimale vaut :

$$w_{\text{vm}}^* = \frac{\sigma_2^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$$

Nous avons égalité pour ces trois portefeuilles lorsque $\sigma_1 = \sigma_2$. En effet, dans ce cas :

$$\begin{aligned} w_{erc}^* &= \frac{\sigma_1^{-1}}{2\sigma_1^{-1}} = \frac{1}{2} \\ w_{1/n}^* &= \frac{1}{2} \\ w_{\text{vm}}^* &= \frac{(1-\rho)\sigma_2^2}{2(1-\rho)\sigma_2^2} = \frac{1}{2} \end{aligned}$$

3.4.2 À corrélations égales

Dans le cas où toutes les corrélations sont égales à une seule et même corrélation ρ, il y aura égalité entre le portefeuille ERC et le portefeuille à poids égaux lorsque les volatilités elles aussi seront égales, ce qui est logique car elles auront toutes les **mêmes propriétés de risque**.
L'égalité entre le portefeuille ERC et le portefeuille à variance minimale se retrouvera lorsque la corrélation est minimale, ce qui est logique par **diversification du risque**.

3.4.3 Cas général

Pour le cas général, nous reprenons les **algorithmes numériques** par manque de solution explicite et notamment le second évoqué car il nous offre un degré de liberté avec le paramètre c :

$$x^\star(c) = \arg\min \sqrt{x^\top \Sigma x}$$
$$\text{s.c.} \begin{cases} \sum_{i=1}^n \ln x_i \geq c \\ \mathbf{1}^\top x = 1 \\ x \geq \mathbf{0} \end{cases}$$

Lorsque $c = -\infty$, nous retrouvons le portefeuille à variance minimale.
Lorsque $c = -n \ln n$, nous retrouvons le portefeuille à poids égaux (et ce portefeuille donne $\sum_{i=1}^n \ln x_i$ maximale).
Par ailleurs, les volatilités des portefeuilles obtenus via cet algorithme se retrouvent dans un **ordre** qui est parfaitement naturel :

$$\sigma_{\text{vm}} \leq \sigma_{\text{erc}} \leq \sigma_{1/\text{n}}$$

On voit, grâce à cette résolution numérique, les liens clairs entre les trois portefeuilles, qui sont tous solutions d'un **algorithme de minimisation de variance** sous une **contrainte de diversification des poids**. Cette contrainte dépend d'un paramètre c, que l'on fixe pour les portefeuilles à variance minimale et à poids égaux, et qui circule entre ces deux valeurs pour le portefeuille ERC, qui est bien un **juste milieu** entre les deux autres modèles.

4 Backtests

Nous avons exploré la théorie des portefeuilles ERC, en développant les définitions et les propriétés, puis nous avons défini les algorithmes numériques nous permettant d'obtenir les poids des actifs dans ces portefeuilles, tout cela s'inscrivant dans une optique de comparaison avec deux autres portefeuilles couramment utilisés et aux définitions semblables : celui à poids égaux et celui à variance minimale.
Dès lors, la dernière étape de cette étude est de backtester ces trois portefeuilles sur les données des années passées, pour vérifier de manière effective l'étude théorique, avoir une vision plus précise des résultats et conclure.
Les données sur lesquelles nous allons travailler contiennent l'historique depuis **1992** de **l'Euro Stoxx**, indice prenant environ 300 des plus grandes actions européennes, ainsi que les **indices sectoriels Euro Stoxx**.
(NB : nous avons retiré les indices SXOT et SX86T car trop récents donc il manquait les premières années)

Tableau 1: Statistiques sur les indices sectoriels

	Gain	Volatilité
SXNT	14.96%	29.07%
SX7T	8.40%	29.16%
SX4T	14.63%	21.62%
SXQT	11.55%	20.92%
SXIT	10.21%	27.72%
SX8T	14.04%	38.19%
SXAT	12.82%	23.90%
SXDT	12.48%	21.44%
SX3T	10.95%	17.49%
SX6T	10.16%	23.11%
SXKT	14.01%	36.90%
SXET	9.70%	18.74%
SXMT	8.48%	24.78%
SXRT	10.48%	24.81%
SXTT	10.38%	22.45%
SXFT	12.68%	26.05%
SXPT	12.07%	33.79%

Tableau 2: Statistiques sur les indices sectoriels

	Matrice de corrélation (%)																
SXNT	100	25.06	95.97	97.52	70.72	60.63	96.09	95.64	94.72	70.0	61.2	86.43	65.8	80.1	94.86	90.58	69.19
SX7T		100	5.32	9.22	56.12	48.5	8.04	20.27	4.63	68.21	55.16	55.22	41.94	25.5	27.86	54.3	64.58
SX4T			100	96.7	53.82	39.74	98.02	94.96	98.57	62.73	43.52	81.33	48.3	69.78	90.17	82.76	60.46
SXQT				100	66.84	54.0	98.07	96.1	97.47	57.95	52.32	77.52	61.78	81.62	95.47	84.79	56.46
SXIT					100	89.66	61.1	71.25	58.0	55.54	76.07	61.03	91.98	85.81	78.02	75.98	45.67
SX8T						100	44.39	55.4	41.24	39.87	85.7	48.4	96.34	80.61	60.7	56.14	36.02
SXAT							100	95.14	98.55	61.18	45.86	77.04	54.33	75.82	95.02	85.72	57.98
SXDT								100	96.57	63.73	51.98	83.98	62.47	76.43	95.42	87.92	56.39
SX3T									100	59.75	42.75	77.98	50.38	72.27	93.39	83.65	54.67
SX6T										100	60.55	88.82	41.43	48.05	65.68	85.33	91.71
SXKT											100	59.8	87.34	80.65	57.29	58.64	53.88
SXET												100	49.59	59.04	78.87	89.74	85.69
SXMT													100	90.1	68.19	60.47	35.32
SXRT														100	81.98	70.37	43.65
SXTT															100	92.37	59.75
SXFT																100	81.07
SXPT																	100

Ces deux tableaux nous donnent des informations sur les indices sectoriels, avec les **gains moyens** et les **volatilités** (les deux étant pris sur les années). On remarque une tendance moyenne de gains aux alentours de 10 11% et des volatilités aux alentours des 20 30%, valeurs élevées mais nous les échelenons sur une période de temps importante.

Nous allons donc évaluer les performances de nos trois portefeuilles sur les indices sectoriels et nous en profiterons pour les comparer aussi à l'indice Euro Stoxx global pour voir s'il est finalement plus intéressant d'acheter ce dernier directement ou non.
Les backtests auront pour objectif de donner les **gains**, les **volatilités** et les **ratios de Sharpe**.

Le code sera disponible à la fin du papier dans la partie "Annexe : code" avec des commentaires expliquant les choix.

Nous noterons les portefeuilles à poids égaux "**1/n**", les portefeuilles à variance minimale "**VM**" et l'indice Euro Stoxx "**SXXT**" dans le graphique et le tableau afin de ne pas surcharger.

Tableau 3 : Résultats des backtests

	1/n	VM	ERC	SXXT
Gain	11.71%	12.84%	12.18%	10.34%
Volatilité	23.42%	21.85%	22.04%	21.53%
Sharpe	0.41	0.50	0.46	0.39

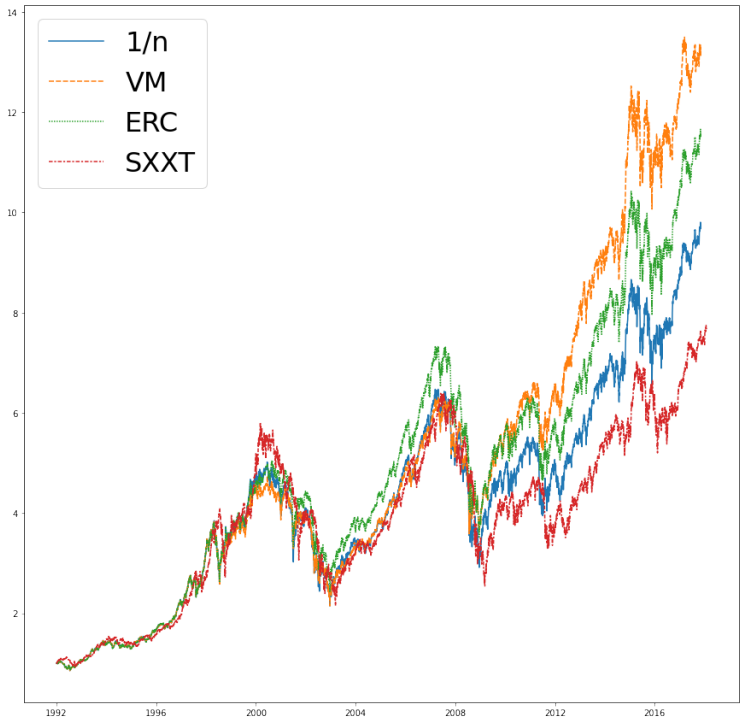

Figure 1: Gains des portefeuilles

Les valeurs du tableau sont encore une fois **échelonnées en année** et celles du graphique sont en **multiplication de la mise de départ**.
Nous voyons donc un portefeuille à variance minimale qui **domine les deux autres** aussi bien sur le gain que sur la volatilité ce qui propose un schéma où les actions les plus rentables ont globalement été les plus stables, avec un gain plutôt de l'ordre de la tendance (on peut penser par exemple à l'indice

SXNT "Industrial Goods & Services" où on aurait tout simplement une croissance tout du long; le secteur technologique (indice SX8T) aurait aussi par exemple pu faire pareil mais la bulle de l'année 2000 a fait voler la volatilité).
Le portefeuille à poids égaux propose le **plus petit gain** parmi les trois alors qu'on aurait pu espérer un peu plus de sa part étant donné sa **volatilité élevée**.
Nous retrouvons donc effectivement un portefeuille ERC à **mi-chemin** entre ses deux concurrents, les portefeuilles à poids égaux et à variance minimale.
Enfin, chacun des trois portefeuilles dépasse la performance de l'indice général Euro Stoxx sur le gain mais se fait battre sur la volatilité : le ratio de Sharpe de l'indice vient conclure en le proposant comme le **moins bon investissement**.

5 Conclusion

Une inquiétude quant à la **robustesse** des modèles et la **facilité de leur implémentation** a mené à un engouement pour des portefeuilles ne prenant plus les gains attendus comme paramètres d'entrée tels que celui à **poids égaux** et celui à **variance minimale**.
Néanmoins, chacun présente ses propres défauts et ceux-ci se révèlent complémentaires : un **manque de prise en compte du risque** pour le premier portefeuille dû à sa diversité et une **trop grande concentration des actif choisis** pour le second.
Le portefeuille ERC se présente comme une alternative à ces deux portefeuilles, se plaçant au **juste milieu** avec une **prise en compte du risque** tout en forçant une certaine **diversité des actions choisies**. Les performances elles aussi sont entre les deux et notamment lorsque l'on regarde la volatilité : nous récupérons aussi bien sur la théorie que sur la pratique un **ordre naturel avec la volatilité ERC au milieu des deux autres**.
Il faut néanmoins toujours garder en tête la grande question de la **complexité de mise en place** de ces modèles. En effet, nous avons pu voir que le portefeuille ERC impliquait des **fonctions endogènes sans formules fermées** pour les cas généraux, ce qui nécessite l'intervention **d'algorithmes numériques** pouvant être **coûteux en temps et donc en argent** là où un portefeuille à poids égaux se présente comme une solution on ne peut plus simple niveau implémentation.

6 Annexe : code

Dans cette partie , on résume le fonctionnement du code , certains choix qui ont été faits au vu des données et comment se servir du code en pratique

Le dataset initial contient deux excel sheets :
- Une première feuille contenant les poids qui ont été utilisés pour calculer l'indice Eurostoxx SXXT ainsi que les valeurs des prix associés
- Les data que nous allons traiter, c'est à dire l'ensemble des prix des différents indices entre 1996 et 2016

```
import numpy as np
import pandas as pd
from scipy.optimize import minimize


file = pd.read_excel("./Data_Euro_Stoxx.xlsx", sheet_name="Sheet2")


dates = file.iloc[:, 0]
file = file.set_index(dates)
file = file.iloc[1:, :]

file.dropna(axis=1, inplace=True)

file.index.name = None

file.columns = ["SXXT", "SXNT", "SX7T", "SX4T", "SXQT", "SXIT", "SX8T", "SXAT",
                "SXDT", "SX3T", "SX6T", "SXKT", "SXET", "SXMT", "SXRT", "SXTT", "SXFT", "SXPT"]

eurostoxx = file["SXXT"]


file.drop(['SXXT'], axis=1, inplace=True)
```

Figure 2: Nettoyage des données

Dans un premier temps nous lisons la sheet 2 du fichier excel avec la bibliothèque Pandas et nous stockons toutes les valeurs des dates/prix des indices dans une matrice ce qui nous sera utile par la suite. Nous retirons volontairement 3 indices:
- SXXT car il fera office de benchmark
- SX86T et SX0T car les data étaient trop récentes et donc difficilement exploitables
Le rendu final est un dataframe contenant le prix de chaque indice en fonction des différentes dates.

```
def calculate_returns(prices):
    return prices.pct_change().dropna()


def calculate_covariance_matrix(returns):
    return returns.cov()
```

Figure 3: Deux fonctions utiles pour les algorithmes

Nous nous servirons souvent de ces deux fonctions d'où l'implémentation en amont , nous calculons les **returns de chaque indice** et la **matrice de corrélation** des returns.

```
def equally_weighted_weights(prices):
    num_assets = prices.shape[1]
    weights = np.repeat(1 / num_assets, num_assets)
    return weights


def backtest_equally_weighted(prices, window_size=60, days_per_month=21):
    num_periods = prices.shape[0] - window_size
    portfolio_values = [1.0]
    weights = None

    for i in range(num_periods):
        sub_prices = prices.iloc[i:i+window_size, :]

        # Rééquilibrer le portefeuille si c'est le premier jour ou si la fréquence de rééquilibrage est atteinte
        if i % days_per_month == 0:
            weights = equally_weighted_weights(sub_prices)

        sub_returns = calculate_returns(sub_prices)
        period_return = np.dot(weights, sub_returns.iloc[-1, :])
        portfolio_values.append(portfolio_values[-1] * (1 + period_return))

    return pd.Series(portfolio_values, index=prices.index[:len(portfolio_values)])
```

Figure 4: Algorithme 1/n

Nous implémentons en premier la stratégie à **poids égaux** comme elle ne demande pas de résoudre un problème d'optimisation, son code est relativement court : on assigne à chaque poids la constante

$\frac{1}{nombred'assets}$

Pour implémenter les **backtests** on raisonne comme suit :
- On fixe le nombre de données historiques que l'on veut utiliser pour faire notre estimation ce qui correspond à la variable **window size**, on suppose ici qu'on veut utiliser les 3 derniers mois de jours de trading (soit environ 60 jours) pour la taille des données utilisées.
- On se donne un nombre de jours avant de rebalancer le portefeuille : en effet, **les fluctuations journalières des prix sont susceptibles au bout d'un moment d'être trop eloignées de la stratégie initiale donc nous rebalançons tous les mois avec les data des 3 derniers mois** dans cet exemple.
Enfin :
- À chaque période de rééquilibrage, on rebalance le portefeuille avec les nouveaux poids qui sont actualisés en fonction de la valeur des nouveaux indices.
- Nous stockons les poids dans un portefeuille normalisé de valeur initiale 1 et on incrémente progressivement les returns sur la même période en multipliant les poids du portefeuille par la valeur des indices correspondants.

Pour la suite nous ne détaillerons pas plus la fonction backtest car pour les deux autres stratégies, la méthode employée est exactement la même hormis le calcul des poids qui diffère évidemment.

Remarque : Evidemment pour la stratégie à **poids égaux** , le rebalancement (c'est à dire le calcul des nouveaux poids) de portefeuille est inutile car, les poids dépendent uniquement du nombre d'indices traités, mais nous avons voulu conserver une structure de code similaire pour de bonnes visibilité et interprétation.

```python
def min_variance_objective_function(weights, cov_matrix):
    return (np.dot(weights.T, np.dot(cov_matrix, weights)))


def min_variance_weights(prices, init_weights=None):
    returns = calculate_returns(prices)
    cov_matrix = calculate_covariance_matrix(returns)
    num_assets = cov_matrix.shape[0]

    if init_weights is None:
        initial_weights = np.repeat(1 / num_assets, num_assets)
    else:
        initial_weights = init_weights

    constraints = ({'type': 'eq', 'fun': lambda w: np.sum(w) - 1},
                   {'type': 'ineq', 'fun': lambda w: w})
    result = minimize(min_variance_objective_function, initial_weights, args=(
        cov_matrix,), method='SLSQP', constraints=constraints)

    return result.x


def backtest_MV(prices, window_size=60, days_per_month=21):
    num_periods = prices.shape[0] - window_size
    portfolio_values3 = [1.0]
    weights = None

    for i in range(num_periods):
        sub_prices = prices.iloc[i:i+window_size, :]

        # Rééquilibrer le portefeuille si c'est le premier jour ou si la fréquence de rééquilibrage est atteinte
        if i % days_per_month == 0:
            weights = min_variance_weights(sub_prices, init_weights=weights)

        sub_returns = calculate_returns(sub_prices)
        period_return = np.dot(weights, sub_returns.iloc[-1, :])
        portfolio_values3.append(portfolio_values3[-1] * (1 + period_return))

    return pd.Series(portfolio_values3, index=prices.index[:len(portfolio_values3)])
```

Figure 5: Algorithme VM

Pour le portefeuille à **variance minimale**, nous utlisons le problème donné dans l'article, c'est à dire la minimisation sous contrainte de la fonction :

$$\arg\min \sqrt{x^\top \Sigma x}$$
$$\text{s.c.} \quad \begin{cases} \sum_{i=1}^{n} \ln x_i \geq c \\ \mathbf{1}^\top x = 1 \\ x \geq \mathbf{0} \end{cases}$$

La fonction **min variance objective function** est donc simplement la fonction à minimiser du problème, nous notons que nous n'avons pas implémenté la fonction exacte, en effet la fonction racine étant stricte-

ment croissante, le problème d'optimisation est équivalent et donc plus rapide numériquement.

Concernant la stratégie, nous utilisons le package **Scipy.optimize** de Python qui permet de résoudre le problème d'optimisation sous contrainte numériquement, les contraintes sont listées dans l'appel de la fonction. De plus, pour un tel portefeuille, la première contrainte avec $c = -\infty$ étant trivialement vérifiée elle n'est pas prise en compte.

```python
def erc_objective_function(weights, cov_matrix):
    portfolio_variance = np.dot(weights.T, np.dot(cov_matrix, weights))
    risk_contributions = weights * \
        (np.dot(cov_matrix, weights)) / portfolio_variance
    return np.sum((risk_contributions - np.mean(risk_contributions)) ** 2)


def erc_weights(prices):
    returns = calculate_returns(prices)
    cov_matrix = calculate_covariance_matrix(returns)
    num_assets = cov_matrix.shape[0]
    initial_weights = np.repeat(1 / num_assets, num_assets)

    constraints = ({'type': 'eq', 'fun': lambda w: np.sum(w) - 1},
                   {'type': 'ineq', 'fun': lambda w: w})
    result = minimize(erc_objective_function, initial_weights, args=(
        cov_matrix,), method='SLSQP', constraints=constraints)

    return result.x


def backtest_erc(prices, window_size=60, days_per_month=21):
    num_periods = prices.shape[0] - window_size
    portfolio_values = [1.0]
    weights = None

    for i in range(num_periods):
        sub_prices = prices.iloc[i:i+window_size, :]

        # Rééquilibrer le portefeuille si c'est le premier jour ou si la fréquence de rééquilibrage est atteinte
        if i % days_per_month == 0:
            weights = erc_weights(sub_prices)

        sub_returns = calculate_returns(sub_prices)
        period_return = np.dot(weights, sub_returns.iloc[-1, :])
        portfolio_values.append(portfolio_values[-1] * (1 + period_return))

    return pd.Series(portfolio_values, index=prices.index[:len(portfolio_values)])
```

Figure 6: Algorithme ERC

Concernant la stratégie **ERC**, nous raisonnons de manière similaire : une fonction objectif à minimiser puis une optimisation sous contrainte, la seule différence étant la fonction que nous utilisons ici :

$$x^\star = \underset{x \in U}{argmin} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(x_i(\Sigma x)_i - x_j(\Sigma x)_j\right)^2 \text{ avec } U = \{\ x \mid \mathbf{1}^\top x = 1\ ,\ \mathbf{0} \leq x \leq \mathbf{1}\}$$

Le backtest quant à lui est tout à fait similaire.

```python
def closest_january_29(group):
    target_date = pd.Timestamp(year=group.name, month=1, day=29)
    closest_date = group.index.min()
    min_difference = abs((group.index - target_date).days).min()
    for date in group.index:
        difference = abs((date - target_date).days)
        if difference < min_difference:
            min_difference = difference
            closest_date = date
    return group.loc[closest_date]

if not isinstance(file.index, pd.DatetimeIndex):
    file.index = pd.to_datetime(file.index)

file = file.groupby(file.index.year).apply(closest_january_29)
```

Figure 7: Regroupement des données par année

Ce regroupement intervient lorsque l'on souhaite obtenir les statistiques moyennes par année. Nous avons pris la dernière date disponible (29 Janvier 2018) comme référence pour le jour gardé (29 Janvier) mais cette date n'était pas forcément disponible pour chaque année. Cet algorithme sert donc à garder la date la plus proche du 29 Janvier pour chaque année afin d'obtenir les résultats recherchés.

```python
def estimate_volatility(prices):
    returns = calculate_returns(prices)
    volatility = returns.std()
    return volatility

def calculate_sharpe_ratio(prices, risk_free_rate=0.02):
    excess_returns = calculate_returns(prices) - risk_free_rate
    sharpe_ratio = excess_returns.mean() / excess_returns.std()
    return sharpe_ratio
```

Figure 8: Volatilité et ratio de Sharpe

Afin de pouvoir **évaluer** nos stratégies et obtenir quelques **métriques** utiles, nous avons choisi de conserver la volatilité et le ratio de Sharpe, que nous considérons comme les deux indices les plus pertinents et populaires pour une stratégie d'investissement :

- Concernant la **volatilité**, il y a plusieurs estimateurs qui existent; nous avons choisi d'aller au plus simple et de prendre simplement la **standard deviation** sur la moyenne des returns.
- Le **ratio de Sharpe** $= \frac{R_p - R_f}{\sigma_p}$ où R_p est le rendement moyen de notre stratégie, R_f est le taux sans risque que nous avons fixé à 0.02 pour coller aux rendements imposés par la FED et σ est la volatilité calculée précédemment.

```python
# Calculer les rendements moyens, les volatilités et les corrélations entre les rendements des actifs
mean_returns = returns.pct_change().dropna().mean()
volatilities = returns.std()

def calculate_correlation_matrix(returns):
    return round(returns.corr()*100, 2)

correlation_matrix = calculate_correlation_matrix(returns)

# Créer un tableau à trois colonnes pour chaque actif
result = pd.DataFrame(columns=["Mean Return", "Volatility", "Correlations"])

for asset in returns.columns:
    result.loc[asset, "Mean Return"] = mean_returns[asset]
    result.loc[asset, "Volatility"] = estimate_volatility(returns[asset])
    result.loc[asset, "Correlations"] = [correlation_matrix[asset][other_asset]
                                         if other_asset != asset else 100 for other_asset in returns.columns]
```

Figure 9: Statistiques sur les indices

Pour finir, nous regroupons dans un dataframe les informations relatives aux indices directement, c'est à dire la volatilité de chaque indice, son rendement moyen ainsi que les corrélations entre chaque indice. Pour plus de lisibilité, il est possible de convertir ce dataframe en fichier excel.