Visualiser les interconnections avec R : une application aux cryptomonnaies.

Introduction

Depuis la crise de 1997, la recherche en économie a cherché à déterminer l'ampleur des phénomènes d'interdépendances et de contagion entre économies et entre marchés. Si le terme de contagion est difficile à définir, la recherche utilise largement la description faite par Forbes et Rigobon (2002) : l'interdépendance reflète l'existence d'effets de débordement d'un marché à l'autre ; la contagion reflète les changements d'interdépendance lors de certains épisodes, notamment pendant les crises.

Pour estimer les effets de *spillovers*, ou effets de débordement, plusieurs méthodes existent. Une des méthodes les plus populaires est celles des modèles GARCH multivariés (BEKK-GARCH ; CCC-GARCH ; DCC-GARCH) qui permet, entre autres, de mesurer la corrélation conditionnelle, la volatilité conditionnelle et la covariance conditionnelle entre deux ou plusieurs séries. Aussi, les coefficients alpha12 et alpha21 sont des mesures des effets de la variable 2 sur la variable 1 et inversement. Mais les modèles GARCH multivariés sont relativement compliqués à mettre en place. Une méthode bien plus simple a été mise au point par Diebold et Yilmaz, au cours d'une série de papiers entre 2009 et 2015. Cette approche se base sur la décomposition des erreurs de prévisions de la variance au sein de modèles VAR, qui permet d'obtenir une matrice de spillovers. Un *package* sur R permet de mettre en place cette méthodologie : **Spillover**.

Nous proposons ici de fournir un exemple détaillé d'application de ce package, avec l'application de l'étude des interconnections au marché des cryptomonnaies. Une multitude d'études ont déjà commencé à étudier la question, y compris avec la méthode de Diebold et Yilmaz. Il ne s'agit pas nécessairement de prolonger les études déjà existantes, mais simplement de fournir un code reproductible pour s'entraîner sur des données facilement disponibles, ainsi que quelques moyens de visualiser simplement et rapidement les résultats.

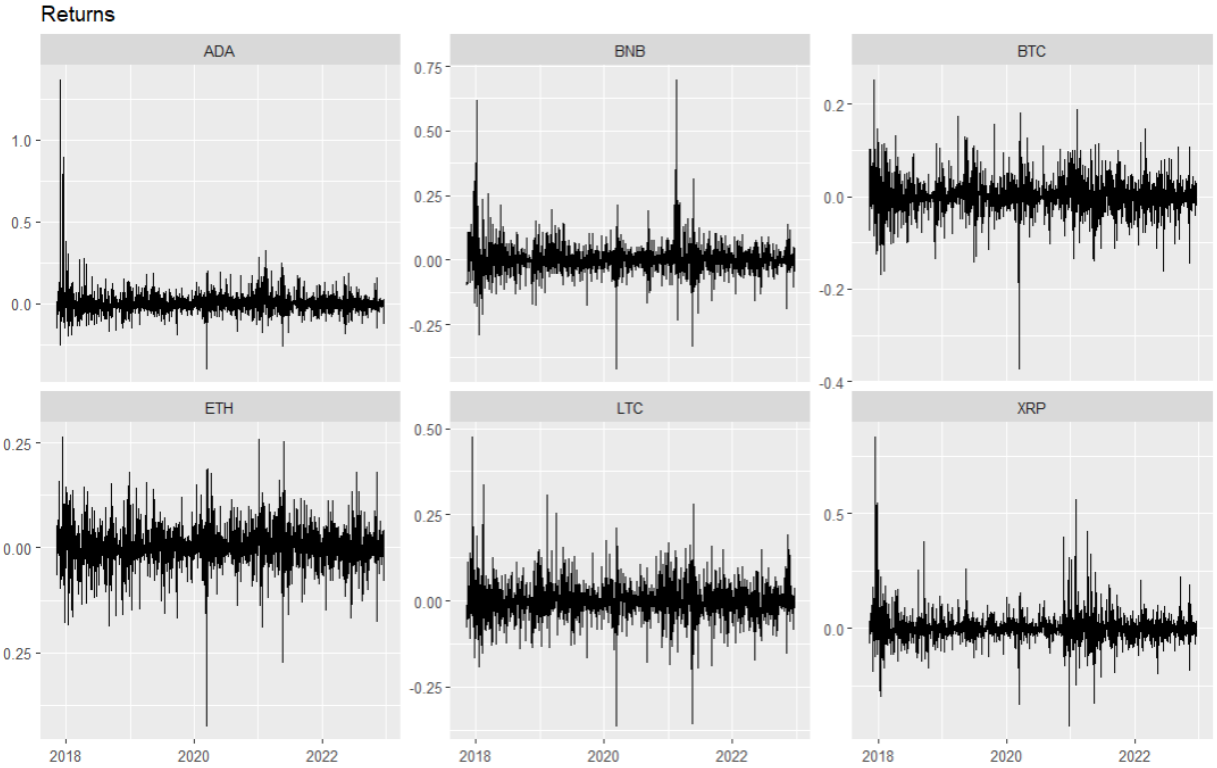
Données

Pour cela, nous utiliserons les données de six cryptomonnaies majeures (qui représentent environ 80% de la capitalisation totale des crypto-actifs), à savoir le Bitcoin (BTC), l'Ethereum (ETH), le Litecoin (LTC), le Ripple (XRP), le Binance Coin (BNB) et le Cardano (ADA). Les données sont récoltées sur le site Yahoo Finance entre le 9 novembre 2017 et le 20 décembre 2022, parce que la plupart des crypto-actifs n'ont pas de données disponibles à une fréquence journalière sur une période temporelle plus étendue sur ce site.

```
require(Spillover)
require(dplyr)
require(tidyr)
require(ggplot2)
require(vars)
cryptos<-read.csv(file.choose())
crypto<-subset(cryptos, select=-c(X))
```

Une information importante à retenir : votre base de donnée doit comporter en **première colonne** des dates, nommée 'Date', dont la structure est reconnue par R comme « character », et le reste des colonnes doivent être reconnues comme « numeric ».

```
#pour visualiser les données
crypto2 = {
  crypto %>% dplyr::mutate(Date_1 =
as.Date(as.character(crypto[,1]))) %>%
    tibble %>% dplyr::select(-1) %>%
    pivot_longer(cols = -Date_1,names_to = "variables") %>%
    ggplot(aes(x = Date_1,y = value)) +
    geom_line() +
    facet_wrap(~variables,scales = "free_y") +
    labs(title = "Returns", x= "", y = "")}
crypto2
```



On peut observer que la volatilité des cryptomonnaies choisies est relativement forte, c'est une caractéristique importante liée à ce type d'actifs déjà relevée par la littérature à de multiples reprises.

La première étape de la méthodologie est d'estimer un modèle VAR, et donc de spécifier le nombre de délais à inclure dans le modèle. La fonction VARselect() du package **vars** permet de déterminer rapidement le nombre optimal de délais.

```
VARselect(crypto[,-1], lag.max=7, type="none")
```

Il est à noter que, pour déterminer le nombre de lags, nous retirons la première colonne qui contient les dates. Les critères AIC et FPE nous indiquent que le nombre optimal de lags à inclure est 7, le critère HQ nous indique que le nombre de lags optimal est 2 et le critère SC nous indique qu'un seul lag suffit. Par simplicité, nous ne choisissons qu'un seul lag pour le reste des estimations, mais la plupart des études qui utilisent l'approche DY effectuent plusieurs estimations pour tester la robustesse du modèle en fonction du nombre de lags.

Pour obtenir la matrice de spillovers, nous pouvons effectuer la commande suivante :

```
crypto %>%
  dplyr::select(-Date) %>%
  VAR(p=1) %>%
  G.spillover(standardized = TRUE) %>%
  round(2)
```

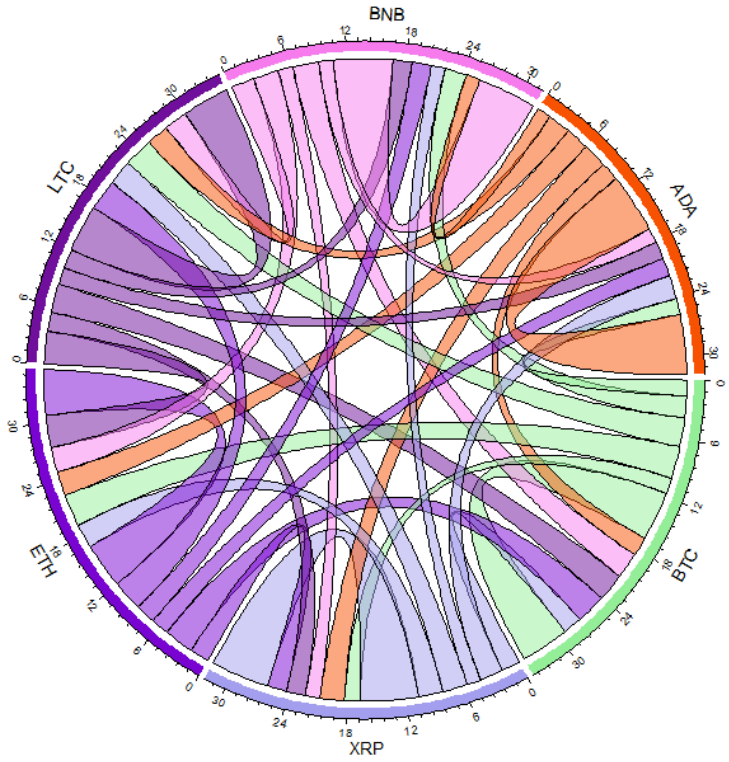
La matrice de spillovers ressemble à cela :

	BTC	XRP	ETH	LTC	BNB	ADA	C. from others
BTC	5.32	1.49	3.23	2.92	2.12	1.59	11.35
XRP	1.79	6.48	2.45	2.30	1.36	2.28	10.19
ETH	2.97	1.89	4.92	3.20	1.92	1.78	11.75
LTC	2.81	1.85	3.37	5.15	1.86	1.63	11.52
BNB	2.57	1.37	2.53	2.36	6.45	1.38	10.22
ADA	1.96	2.30	2.37	2.08	1.40	6.56	10.11
C. to others (spillover)	12.10	8.90	13.95	12.86	8.67	8.66	65.13
C. to others including own	17.41	15.38	18.86	18.01	15.12	15.22	100.00

Les résultats étant standardisés (`standardized=TRUE`), nous pouvons les interpréter comme des pourcentages. Les diagonales stockent les spillovers d'une variable sur elle-même. Autrement dit, un choc sur le Bitcoin se répercute à 5.3% sur lui-même. Par ailleurs, un choc du Bitcoin se répercute à 1.80% sur le Ripple. En cumulé, un choc du Bitcoin se répercute à 12% sur les autres cryptomonnaies.

Ce tableau contient beaucoup d'informations importantes, mais n'est pas forcément très joli. Des papiers plus récents ont fait un rapprochement entre l'approche de DY et la théorie des réseaux (networks). Sous R, le *package* **circlize** permet d'obtenir des graphiques qui permettent de faire la jonction entre les deux théories, notamment avec la fonction `chordDiagram()`. La commande est la suivante :

```
crypto_links<-as.matrix(crypto %>%
                dplyr::select(-Date) %>%
                VAR(p=1) %>%
                G.spillover(standardized = TRUE) %>%
                round(2))
circlize::chordDiagram(crypto_links[1:6,1:6], link.border=1)
```



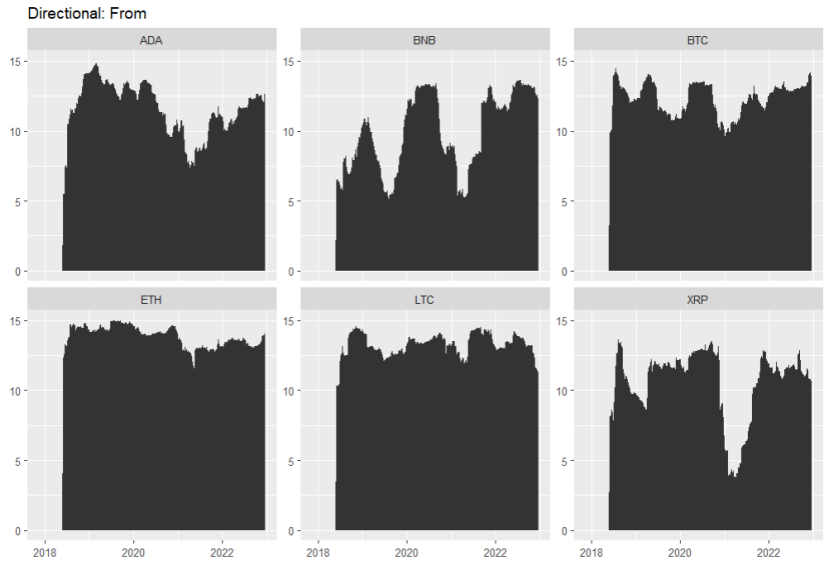
Vous noterez qu'en effectuant la même commande, vous n'obtiendrez pas forcément les mêmes couleurs que sur mon graphique ici présent. C'est parce que les couleurs sont générées aléatoirement à chaque fois que vous effectuez la commande. Vous pouvez fixer les couleurs en créant une liste de couleurs associées à chaque série en amont, et spécifier les couleurs dans la fonction `chordDiagram` avec `grid.col = list_col`.

Sur le graphique, on peut observer les spillovers qui proviennent d'une variable et qui vont vers une autre, autrement dit les interconnections. Le package fournit également un moyen d'estimer la contagion, donc l'évolution temporelle des interconnections, en effectuant la même démarche non pas sur l'ensemble des données, mais sur des fenêtres roulantes de N observations qui sont décalées d'une période à chaque estimation, pour obtenir des coefficients pour chaque période. La commande est la suivante :

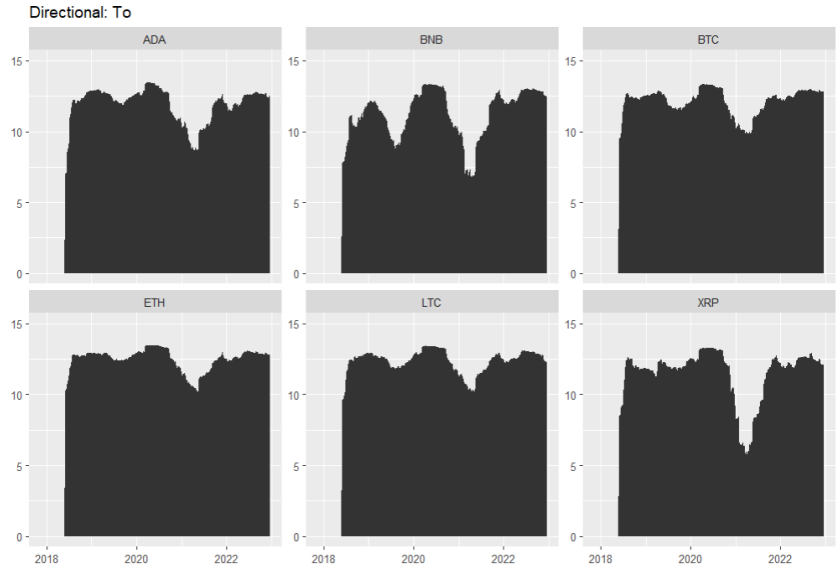
```
crypto_dyn<-dynamic.spillover(crypto, p=1, width=200,
standardized=TRUE )
```

Pour visualiser les résultats :

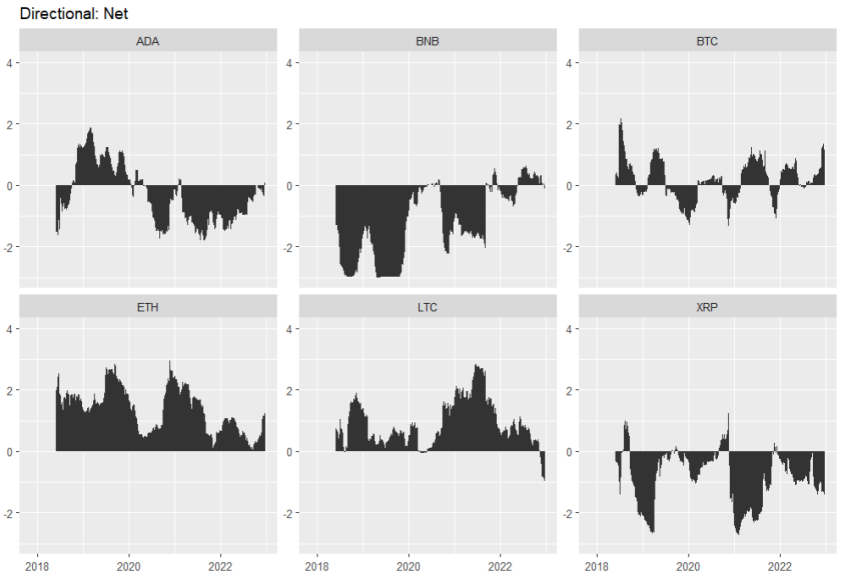
```
crypto_from<-plotdy(crypto_dyn, direction="from")
crypto_from+ylim(0,15)
```




```
crypto_to<-plotdy(crypto_dyn, direction="to")
```

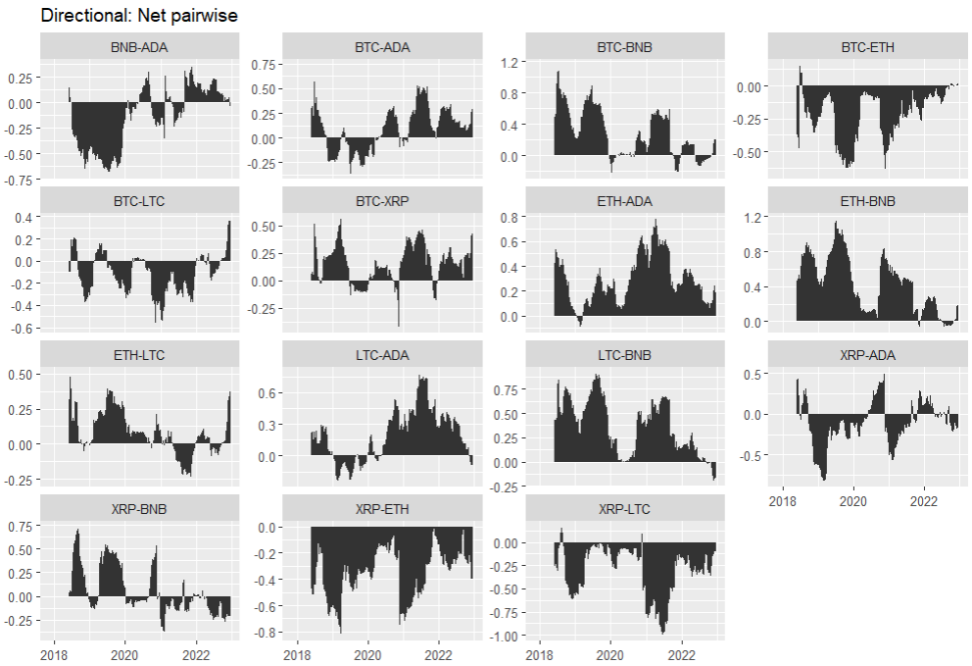



```
crypto_net<-plotdy(crypto_dyn, direction="net")
crypto_net+ylim(-3,4)
```



Il est également possible d'étudier les connections bilatérales :

```
crypto_pairwise<-plotdy(crypto_dyn, direction="net_pairwise")
```



Un coefficient significatif sur la paire (X-Y) signifie que X émet des spillovers nets vers Y.

Vous savez maintenant comment visualiser des spillovers entre plusieurs variables. L'approche Diebold-Yilmaz est très simple, et ne nécessite pas de tests statistiques étendus : vous pouvez inclure autant de variables que vous le souhaitez. En revanche, les capacités de votre ordinateur seront la principale limite, car plus vous rajouterez de données, et plus les calculs seront longs, surtout pour l'estimation des spillovers directionnels dans des fenêtres roulantes.

Il est aussi important de noter qu'ici, nous avons mesuré les spillovers de rendements entre les cryptomonnaies. Il est possible d'examiner les spillovers de volatilité, en calculant les séries de volatilité pour chaque variable, créer une base de donnée et faire la même procédure que celle qu'on

a fait ici. En revanche, mesurer la volatilité peut s'avérer compliqué car il n'y a pas de définition claire et précise de ce qu'est la volatilité. On peut calculer la volatilité de plusieurs moyens :

-écart-types mesurés dans des fenêtres roulantes, mais l'information est lissée ;

-la somme des rendements intra-journaliers élevés au carré, mais cela nécessite d'avoir accès à ces séries à haute fréquence ;

-la différence entre le prix Maximum et le prix Minimum chaque jour, éventuellement avec des pondérations avec le prix d'Ouverture et le prix de Fermeture, mais toutes les données financières ne fournissent pas autant d'informations ;

-la volatilité peut être modélisée à l'aide de modèles GARCH (univariés ou multivariés), mais de tels modèles sont souvent compliqués et lourds.