Modélisation des Marchés Financiers par les Chaînes de Markov

Valentin ANGER, Damien BELHARET, Kitchi-Tawa BOURGUINAT

***Résumé*—Cet article explore l'application des chaînes de Markov à la modélisation des marchés financiers. Après avoir présenté les fondements théoriques, nous mettons en place une simulation en Python et comparons les résultats obtenus avec les données historiques du marché. Les résultats montrent que les chaînes de Markov peuvent capturer certaines tendances du marché, bien que des limitations existent en raison des hypothèses simplificatrices du modèle. Ces résultats doivent être interprétés avec précaution, car la période de simulation peut influencer significativement les observations.**

I. Introduction

L'analyse des marchés financiers repose sur une variété de modèles mathématiques et statistiques permettant de comprendre et de prévoir les évolutions des prix. Parmi les approches les plus courantes, on retrouve les modèles économétriques classiques (comme les modèles ARIMA et GARCH), les méthodes basées sur l'apprentissage automatique et les modèles inspirés de la physique statistique. Cependant, ces approches présentent certaines limites, notamment en ce qui concerne leur capacité à capturer la nature dynamique et incertaine des marchés.

Dans ce contexte, l'adaptation des outils issus de l'étude d'autres systèmes dynamiques, tels que les chaînes de Markov, offre une alternative intéressante. Ces modèles permettent de représenter les marchés financiers comme des systèmes évoluant entre différents états avec des probabilités de transition déterminées empiriquement. Cette approche simplifie l'analyse des tendances et des cycles de marché tout en conservant une modélisation probabiliste robuste.

L'intérêt de cette méthode réside dans sa capacité à fournir une représentation compacte des dynamiques de marché, tout en offrant un cadre permettant de simuler et d'anticiper les évolutions futures. Toutefois, une limite majeure de ce type de modélisation est qu'il repose uniquement sur l'historique des prix et ne prend pas en compte les facteurs externes influençant les marchés, tels que les événements géopolitiques, les politiques monétaires, les innovations technologiques ou encore les crises économiques. Cette hypothèse de stationnarité des probabilités de transition peut donc réduire la pertinence des prévisions à long terme.

Cet article propose d'étudier l'application des chaînes de Markov à la modélisation des prix du Bitcoin, en évaluant leur pertinence à travers des simulations comparées aux données historiques. Nous discuterons également des limites de ce modèle et des pistes d'amélioration pour mieux intégrer les facteurs externes influençant les marchés financiers.

II. Méthodologie

A. Définition des états du marché

Nous considérons un modèle où le marché peut être dans trois états :

- **Hausse (h)** : Lorsque le rendement est supérieur à un certain seuil τ_h.
- **Baisse (b)** : Lorsque le rendement est inférieur à un seuil τ_b.
- **Stagnation (s)** : Lorsque le rendement reste entre ces deux valeurs.

Soit X_t l'état du marché à l'instant t. Nous supposons que (X_t) suit une chaîne de Markov de premier ordre, c'est-à-dire que :

$$P(X_{t+1}|X_t, X_{t-1}, \dots) = P(X_{t+1}|X_t)$$

Cela signifie que l'évolution future dépend uniquement de l'état actuel et non du passé complet.

Le choix des seuils $\tau_h = 1\%$ et $\tau_b = -1\%$ repose sur des considérations empiriques. Un rendement quotidien de 1% est souvent perçu comme une variation significative sur des marchés financiers, tout en restant suffisamment fréquent pour permettre une estimation robuste des probabilités de transition. De plus, cette valeur permet d'éviter une granularité excessive dans la classification des états : des seuils trop faibles généreraient trop de transitions entre états, réduisant la pertinence du modèle, tandis que des seuils trop élevés conduiraient à un modèle moins réactif aux fluctuations du marché. Le choix de ces valeurs vise ainsi à équilibrer la sensibilité et la stabilité du modèle dans la simulation des trajectoires financières.

B. Construction de la matrice de transition

À partir des données historiques des marchés financiers, nous estimons la matrice de transition P définie par :

$$P = \begin{pmatrix} p_{hh} & p_{hb} & p_{hs} \\ p_{bh} & p_{bb} & p_{bs} \\ p_{sh} & p_{sb} & p_{ss} \end{pmatrix}$$

où chaque élément p_{ij} représente la probabilité de transition de l'état i vers l'état j. Ces probabilités sont estimées empiriquement en comptant les occurrences dans les données historiques.

C. Simulation des trajectoires

Nous générons N trajectoires simulées du marché sur une période de T jours en utilisant :

$$X_{t+1} \sim P(X_t)$$

avec une condition initiale choisie selon les données réelles (pour avoir continuité des courbes). Nous comparons ensuite la distribution des rendements simulés aux rendements observés pour évaluer la pertinence du modèle. [1] [2]

III. Expérience

L'objectif de cette expérience est de simuler l'évolution des prix de Bitcoin à l'aide d'une chaîne de Markov, puis de comparer les trajectoires simulées aux données réelles des prix sur la période 2013-2023. Les données pour 2024 et au-delà serviront à évaluer la précision des prévisions générées par le modèle.

Figure 1. Prix de clôture réels du BTC-USD depuis sa création

Les données utilisées dans cette expérience proviennent du site coinmarketcap.com, où les prix mensuels de Bitcoin entre 2013 et 2023 ont été récupérés dans un fichier *CSV*.

Les lignes 1 à 46 du *code Python en section VII* permettent de construire la matrice de transition à partir des données historiques :

$$P = \begin{pmatrix} 0.52307692 & 0.49152542 & 0.75 \\ 0.44615385 & 0.49152542 & 0.25 \\ 0.03076923 & 0.01694915 & 0 \end{pmatrix}$$

Cela représente les probabilités de transition entre les états du marché (hausse, baisse et stagnation).

Ensuite, à partir des lignes 49 à 78 du code, nous appliquons la matrice de transition pour simuler plusieurs trajectoires, en prenant comme condition initiale l'état du marché à la fin de 2023 (continuité des courbes). Les simulations génèrent des prévisions des prix de Bitcoin pour la période 2024-2026.

Les courbes obtenues sont tracées à l'aide du module `matplotlib`, permettant ainsi de visualiser les trajectoires simulées par rapport aux prix réels de 2013 à 2024.

IV. Résultats et Analyse

Sur l'année 2024, nous avons les données réelles pour analyser les résultats de notre modèle. Nous voyons que même si il ne reflète pas exactement les memes fluctuations de prix, notre modèle est globalement cohérent.

Si nous allongeons la date finale de prévision à 2025, la simulation reste cohérente. Les trajectoires en rouge restent proches les unes des autres ainsi que de la courbe réelle.

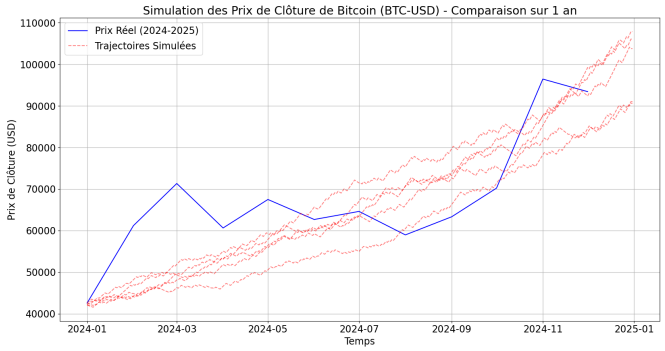

Figure 2. Comparaison sur 1 an entre notre simulation et les prix réels

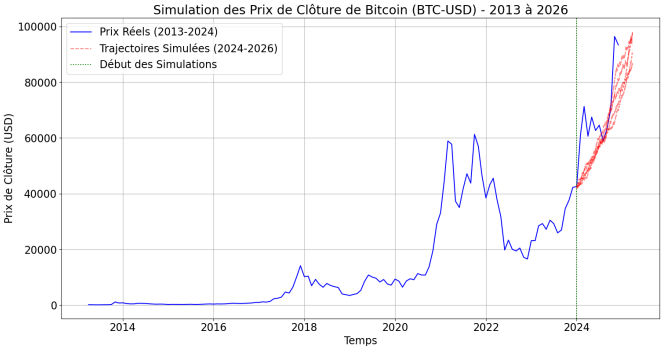

Figure 3. Simulation des prix de clôture du BTC-USD jusqu'en 2025

Cependant en allongeant encore jusqu'à 2026, la date finale de prévision, les trajectoires divergent, notre incertitude s'agrandit.

Figure 4. Simulation des prix de clôture du BTC-USD jusqu'en 2026

Les graphiques se trouvent à la fin de l'article *en section VIII* pour une meilleur visibilité. [3] [4] [5]

V. Discussion

Le modèle suppose que les transitions entre états sont stationnaires et ne dépendent que de l'état actuel (2023), sans prendre en compte les influences extérieures ou les tendances à long terme. Nous voyons qu'il devient problématique de

prévoir les prix du BTC-USD en 2026 (3 ans à partir de 2023). Alors dans plus longtemps, le problème s'accentue. On pourrait se dire qu'il suffit de mettre les données à jour (état initial et matrice de transition) en 2025 pour se rapprocher des résultats, mais le problème restera le même. Comment prédire les prix en 2030 à partir d'aujourd'hui ? Le problème que nous rencontrons est le même.

VI. Conclusion

L'utilisation des chaînes de Markov pour modéliser les marchés financiers, en particulier le cas de Bitcoin, s'avère prometteuse pour l'analyse des évolutions à court terme. En nous basant sur un modèle simple de transitions entre trois états du marché (hausse, baisse, stagnation), nous avons pu générer des trajectoires de prix qui reproduisent globalement les tendances observées dans les données historiques.

Les résultats de la simulation montrent que, bien que notre modèle soit efficace pour capturer les dynamiques à court terme (environ 1 à 2 ans), sa précision se dégrade à mesure que l'on s'éloigne de la période initiale des données. Ce phénomène est dû à l'hypothèse de stationnarité des probabilités de transition, qui ignore les évolutions à long terme ou les changements exogènes pouvant influencer le marché.

Ainsi, bien que les chaînes de Markov puissent offrir un cadre utile pour des prévisions à court terme, elles restent limitées pour des prévisions à plus long terme, où des modèles plus sophistiqués, intégrant des facteurs externes ou des comportements non linéaires, seraient nécessaires.

En conclusion, bien que notre modèle présente des limites, il constitue un premier pas vers l'utilisation des chaînes de Markov dans l'analyse des marchés financiers, et il ouvre la voie à des améliorations futures. Les recherches ultérieures pourraient se concentrer sur l'intégration de modèles hybrides combinant chaînes de Markov et autres techniques, telles que l'apprentissage automatique et l'IA, afin d'améliorer la précision des prévisions et de mieux appréhender la complexité des marchés financiers.

Références

[1] Deju Zhang. “Predicting Nifty 50 Trends Using A Markov Chain Approach”. In : *International Journal of Business and Management* (2009). Disponible ici : Accéder à l’article.

[2] FasterCapital. *Analyse de la chaîne de Markov : prévoir les états futurs avec des modèles de simulation financière*. Accéder au site. 2024.

[3] Kriti Verma. “Predicting Nifty 50 Trends Using A Markov Chain Approach”. In : *International Journal of Creative Research Thoughts* (2024). Disponible ici : Accéder à l’article.

[4] Simeyo Otieno. “Application of Markov chain to model and forecast stock market trend”. In : *International Journal of Current Research* (2015). Disponible ici : Accéder à l’article.

[5] A Fitriyanto et T E Lestari. “Application of Markov Chain to stock trend”. In : *Purpose-led Publishing* (2007). Disponible ici : Accéder à l’article.

VII. Code Python

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Charger les données
donnees = pd.read_csv('BTC_All_graph_coinmarketcap.csv', delimiter=';', header=0)
donnees['timestamp'] = pd.to_datetime(donnees['timestamp']).dt.tz_localize(None)

#Données d'entrainement
donnees_entrainement = donnees[(donnees['timestamp'] >= '2013-01-01') &
↪  (donnees['timestamp'] < '2024-01-01')]

# Seuil Tau
seuil_hausse = 0.01  # Seuil pour la hausse (1% de gain)
seuil_baisse = -0.01  # Seuil pour la baisse (-1% de perte)

# Définir les états du marché
def classifier_etat_marche(x):
    if x > seuil_hausse:
        return 0  # Hausse
    elif x < seuil_baisse:
        return 1  # Baisse
    else:
        return 2  # Stagnation

# Calcul des rendements
donnees_entrainement['rendement'] = donnees_entrainement['close'].pct_change()

# Appliquer la classification des états
donnees_entrainement['etat'] =
↪  donnees_entrainement['rendement'].apply(classifier_etat_marche)

# Construire la matrice de transition 3x3
def calculer_matrice_transition(donnees):
    n_etats = 3  # Nombre d'états (Hausse, Baisse, Stagnation)
    matrice_transition = np.zeros((n_etats, n_etats))

    for i in range(1, len(donnees)):
        etat_precedent = donnees['etat'].iloc[i-1]
        etat_courant = donnees['etat'].iloc[i]
        matrice_transition[etat_precedent, etat_courant] += 1

    # Normaliser pour obtenir des probabilités
    sommes_lignes = matrice_transition.sum(axis=1, keepdims=True)
    matrice_transition = matrice_transition / sommes_lignes  # Probabilités de transition
    return matrice_transition

matrice_transition = calculer_matrice_transition(donnees_entrainement)

# Fonction de simulation de la chaîne de Markov pour plusieurs trajectoires
def simuler_chaine_markov(matrice_transition, etat_initial, pas=100, n_trajectoires=10):
    n_etats = len(matrice_transition)
    trajectoires = []

    # Calculer la moyenne des rendements pour chaque état
    rendements_etats = {
```

```
        0: donnees_entrainement['rendement'][donnees_entrainement['etat'] == 0].mean(),  # Hausse
        1: donnees_entrainement['rendement'][donnees_entrainement['etat'] == 1].mean(),  # Baisse
        2: donnees_entrainement['rendement'][donnees_entrainement['etat'] == 2].mean()   # Stagnation
    }

    for _ in range(n_trajectoires):
        etat_courant = etat_initial
        chemin = [etat_courant]
        prix = [donnees_entrainement['close'].iloc[-1]]  # Initialiser avec le dernier prix réel

        for _ in range(pas):
            # Appliquer le rendement moyen pour l'état actuel
            etat_suivant = np.random.choice(n_etats, p=matrice_transition[etat_courant])
            chemin.append(etat_suivant)

            # Calculer le prix suivant en appliquant le rendement moyen
            nouveau_prix = prix[-1] * (1 + rendements_etats[etat_suivant]*0.2*1e-1)
            prix.append(nouveau_prix)

            etat_courant = etat_suivant

        trajectoires.append(prix[1:])  # On enlève la première valeur initiale, car elle est déjà donnée

    return trajectoires

# Initialiser la simulation à partir de l'état à la fin de 2023
etat_initial = donnees_entrainement['etat'].iloc[-1]  # Dernier état de 2023
pas = 2*365  # jours de simulation
n_trajectoires = 5  # Nombre de trajectoires à simuler

# Simuler plusieurs trajectoires pour 2 ans (2024-2026)
chemins_simules = simuler_chaine_markov(matrice_transition, etat_initial, pas=pas, n_trajectoires=n_trajectoires)

# Générer les dates simulées de 2024 à 2026 (2 ans)
dates_simulees = pd.date_range(start='2024-01-01', periods=pas+1, freq='D')

# Ajuster la longueur de chemins_simules pour qu'elle corresponde à dates_simulees
# Si nécessaire, compléter ou tronquer chemins_simules
for i in range(len(chemins_simules)):
    # Si la trajectoire est trop longue, la tronquer
    if len(chemins_simules[i]) > len(dates_simulees):
        chemins_simules[i] = chemins_simules[i][:len(dates_simulees)]
    # Si la trajectoire est trop courte, la compléter
    elif len(chemins_simules[i]) < len(dates_simulees):
        chemins_simules[i].extend([chemins_simules[i][-1]] * (len(dates_simulees) - len(chemins_simules[i])))

plt.figure(figsize=(14, 7))

# Tracer les prix réels de 2013 à 2024
plt.plot(donnees['timestamp'], donnees['close'], label='Prix Réels (2013-2024)', color='blue')

# Tracer toutes les trajectoires simulées de 2024 à 2026
for i, prix_simules in enumerate(chemins_simules):
    label = 'Trajectoires Simulées (2024-2026)' if i == 0 else ""
    plt.plot(dates_simulees, prix_simules, color='orange', linestyle='--', label=label, alpha=0.5)

plt.title('Simulation des Prix de Clôture de Bitcoin (BTC-USD) - 2013 à 2026')
plt.xlabel('Temps')
plt.ylabel('Prix de Clôture (USD)')
plt.legend()
plt.grid(True)
plt.show()
```

VIII. Graphiques

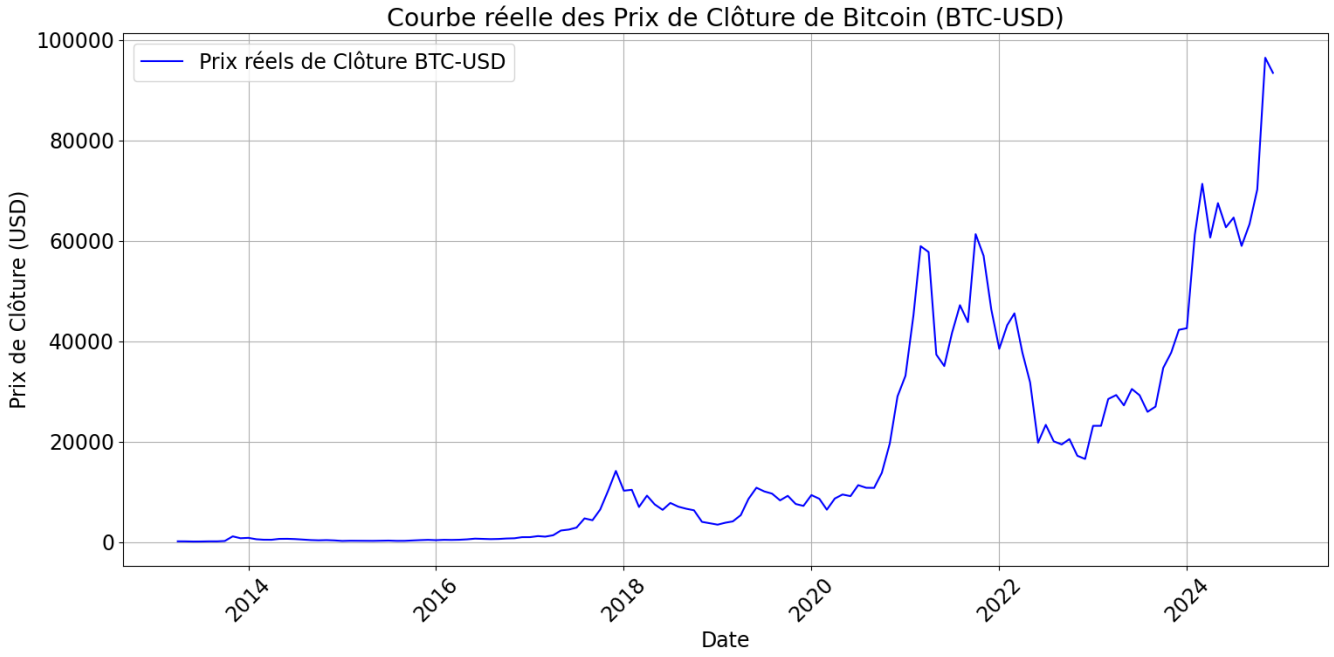

Figure 1. Prix de clôture réels du BTC-USD depuis sa création

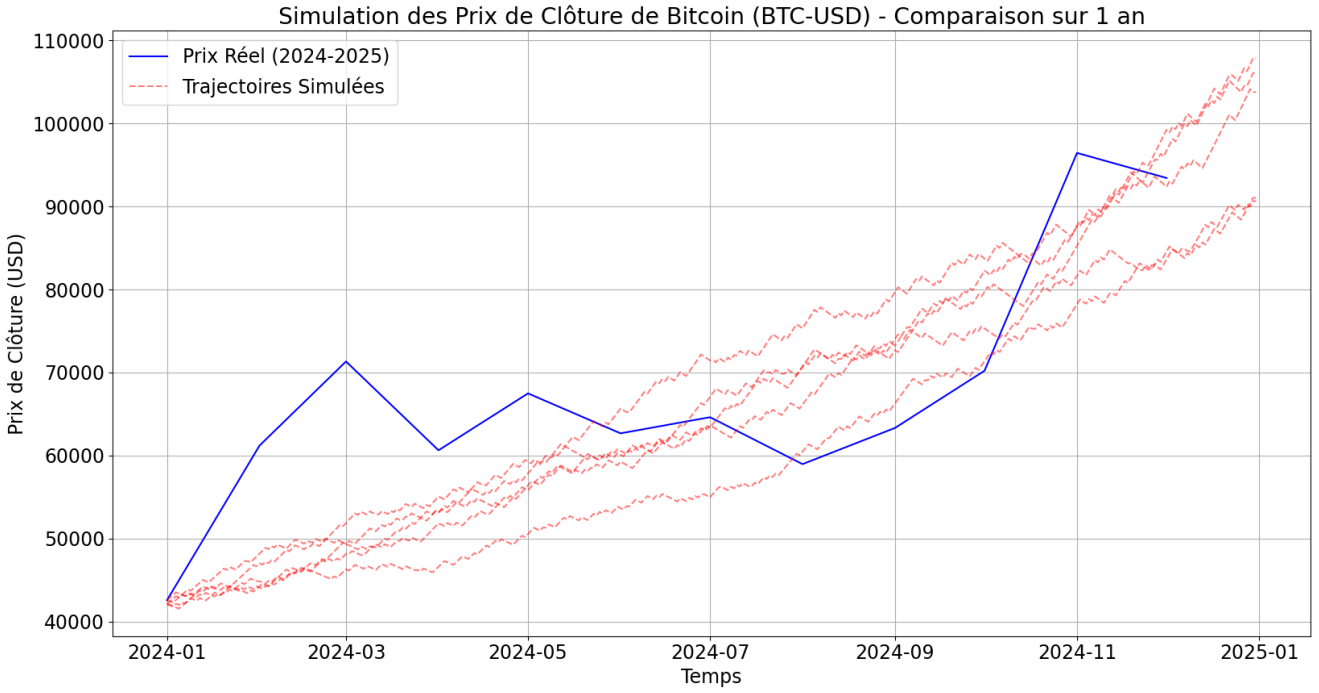

Figure 2. Comparaison sur 1 an entre notre simulation et les prix réels

FIGURE 3. Simulation des prix de clôture du BTC-USD jusqu'en 2025

FIGURE 4. Simulation des prix de clôture du BTC-USD jusqu'en 2026