TP2: Chaines de Markov

Evan Voyles

May 18

```r
library(tidyverse)
library(purrr)
library(tibble)
library(ggplot2)
library(pracma)
```

Ici j'ai redéfini les fonctions utilisé dans le TP1 pour simuler une trajectoire d'une Chaine Markov avec matrice de transition Q.

```r
sample_from <- function(p, n = 1) {

    get_x <- function() {
        p1 <- c(0, p[seq_len(length(p) - 1)])
        t <- runif(1)
        u <- (t > cumsum(p1)) & (t < cumsum(p))
        x <- which(u == 1)

        x
    }

    x <- rep(0, n)
    for (i in seq_len(n)) {
        x[i] <- get_x()
    }

    x
}

etat_suiv <- function(x, Q) {
    p <- Q[x,]
    sample_from(p)
}

sim_chain <- function(p0, Q, n = 50) {

    # get the first state
    x0 <- sample_from(p0)

    x <- rep(0, n)
    x[1] <- x0

    for (i in seq_len(n - 1)) {
        x[i + 1] <- etat_suiv(x[i], Q)
```

```
    }

    x
}
```

Ecercice 1. Le collectionneur

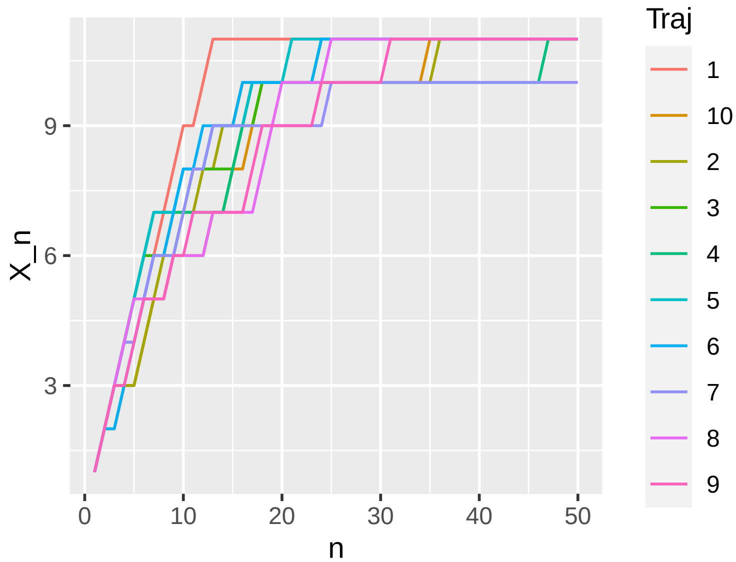
1. Ecrire dans `R` la matrice de transition P de X_n pour $N = 10$.

```
N <- 10 # Number of cards needed to fill the Panini Album

P <- matrix(rep(0, (N + 1) ** 2), nrow = (N + 1))

# Loop along the diagonals
for (i in 1:N) {
    P[i, i + 1] <- ((N + 1) - i) / N
    P[i + 1, i + 1] <- 1 - (N - i) / N
}

P
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
##  [1,]    0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0
##  [2,]    0  0.1  0.9  0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0
##  [3,]    0  0.0  0.2  0.8  0.0  0.0  0.0  0.0  0.0   0.0   0.0
##  [4,]    0  0.0  0.0  0.3  0.7  0.0  0.0  0.0  0.0   0.0   0.0
##  [5,]    0  0.0  0.0  0.0  0.4  0.6  0.0  0.0  0.0   0.0   0.0
##  [6,]    0  0.0  0.0  0.0  0.0  0.5  0.5  0.0  0.0   0.0   0.0
##  [7,]    0  0.0  0.0  0.0  0.0  0.0  0.6  0.4  0.0   0.0   0.0
##  [8,]    0  0.0  0.0  0.0  0.0  0.0  0.0  0.7  0.3   0.0   0.0
##  [9,]    0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.8   0.2   0.0
## [10,]    0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.9   0.1
## [11,]    0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0   1.0
```

2. En simulant un grand nombre de trajectoires, estimer empiriquement le temps moyen pour remplir un album avec $N = 10$ cartes.

```
# Define the initial distribution
p0 <- rep(0, 11)
p0[1] <- 1

# Simulate data
chain <- sim_chain(p0, P, 50)
df <- tibble(n = 1:50, chain)
for (i in 2:10) {
    df <- add_column(df, sim_chain(p0, P, 50), .name_repair = "unique")
}
names(df) <- c("n", as.character(1:10))

df |>
    pivot_longer(c(2:11), values_to = "val", names_to = "name") |>
    ggplot(aes(n, val, col = name)) +
    geom_line() +
    labs(col = "Traj", y = "X_n")
```



La qu'on a vérifié la simulation de notre Chaine Markov, on va procéder pour estimer le temps moyen pour remplir un album avec $N = 10$ cartes.

```
n_traj <- 1000
len_traj <- 75

traj <- list() # store trajectories
for (i in 1:n_traj) {
    traj[[i]] <- sim_chain(p0, P, len_traj)
}

find_first_11 <- function(x) {
    detect_index(x, function(x) x == 11)
}

first_11 <- map(traj, find_first_11)
x <- unlist(first_11)
x[x == 0] <- len_traj # If we never reach 11, just use len_traj

avg_cards <- mean(x)

print(avg_cards)
```

```
## [1] 30.453
```

Cette valeur conforme à ce que l'on a trouvé en TD.

3. (a) Générer un échantillon de grande taille qui donne le numéro des cartes obtenues. Ecrire une fonction qui prend en entrée cet échantillon et le nombre r de collectionneurs, et qui renvoie le temps au bout duquel les r albums sont complets.

```
n_sample <- 1e5
N <- 100

album_completion_time <- function(n_sample, n_cards, r_collectors = 2) {

    # Go ahead and generate a large sample
    s <- sample(1:n_cards, n_sample, replace = TRUE)
```

```
    # Now we need to go until every single card has >= r cards
    found_all_cards <- FALSE
    i <- 1
    counts <- rep(0, n_cards)
    missing_cards <- 1:n_cards # indices of missing cards

    # Keep counting values in the sample until the length of
    # missing cards is 0
    while (!found_all_cards) {

        counts[s[i]] <- counts[s[i]] + 1
        i <- i + 1

        # Need to actually start checking the missing cards
        missing_cards <- missing_cards[counts[missing_cards] < r_collectors]
        found_all_cards <- length(missing_cards) == 0

        # if i larger than the sample, draw another sample!
        if (i >= n_sample) {
            s <- sample(1:n_cards, n_sample, replace = TRUE)
            i <- 1
        }
    }

    i
}
```

(b) En générant un grand nombre d'echantillons, estimer le temps moyen nécessaire pour remplir 2 albums de taille $N = 10$.

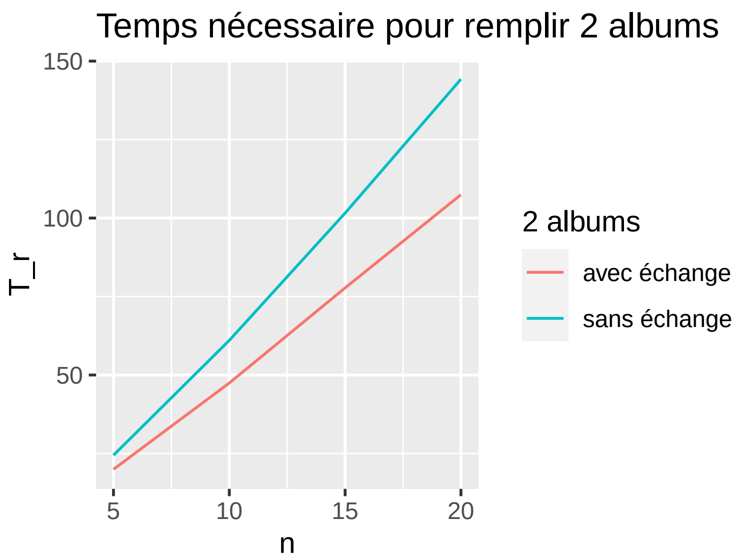
```
n_cards <- 10
r <- 2
n_sim <- 1000

completion_times <- map_dbl(1:n_sim, function(x) {
                                        album_completion_time(200, n_cards, r)
                                     })
mean(completion_times)
```

```
## [1] 46.955
```

(c) Tracer le temps nécessaire pour remplir 2 albums en fonction de N (prendre $N \in \{5, 10, 15, 20\}$). Tracer sur la même courbe le temps nécessaire pour remplir 2 albums sans les échanges.

```
N <- c(5, 10, 15, 20)

mean_completion_times <- function(n, r = 2, n_sim = 1000) {
    completion_times <- map_dbl(1:n_sim, function(x) {
        album_completion_time(200, n, r)
    })
    mean(completion_times)
}

# Fill two albums WITH exchanges
alb_exch <- map_dbl(N, mean_completion_times)
```

```r
# Fill one album, twice
two_alb <- 2 * map_dbl(N, function(n) { mean_completion_times(n, 1)})

df <- tibble(N, alb_exch, two_alb)
names(df) <- c("N", "avec échange", "sans échange")

df |>
    pivot_longer(c(2, 3), values_to = "vals", names_to = "names") |>
    ggplot(aes(N, vals, col = names)) +
    geom_line() +
    labs(col = "2 albums", y = "T_r", x = "n", title = "Temps nécessaire pour remplir 2 albums")
```



Evidamment, avec des échanges le temps nécessaire pour remplir 2 albums est plus petit que le temps qu'il faut pour remplir 2 albums sans échange. Avec échange, les collectionneurs peuvent partager les cartes entre eux.

Exercice 2. Transmission d'un message

1. Modéliser ce problème par une chaîne de Markov $(X_n)_{n\geq 1}$ où X_n désigne la réponse du n-ème intermédiare.

Ce problème peut-être étudier sous le carde d'une chaîne de Markov où $X_n \in \{\text{Oui}, \text{Non}\}$ dont la matrice de transition est:

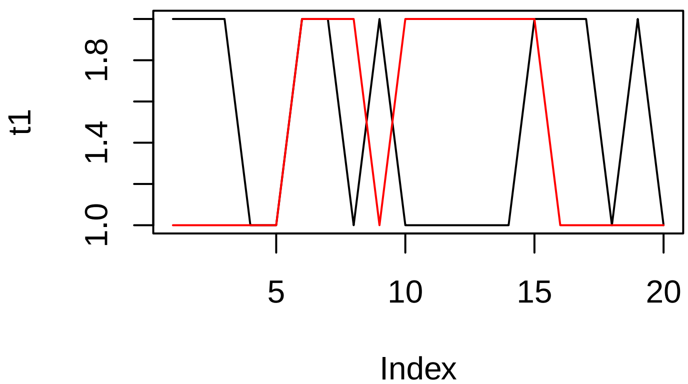
$$P = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$$

Cette chaine est irréductible et apériodique. Donc, on peut appliquer les théorèmes ergodique et convergence en loi.

2. Représenter deux trajectoires de la chaîne de Markov pour $n = 1, \ldots, 20$: $p = 0.3, p = 0.9$.

```r
get_transition_matrix <- function(p) {
    q <- 1 - p
    matrix(c(p, q, q, p), nrow = 2)
}

# We don't know if the first message should be true or false
p0 <- c(0.5, 0.5)
```

```r
t1 <- sim_chain(p0, get_transition_matrix(0.3), 20)
t2 <- sim_chain(p0, get_transition_matrix(0.9), 20)

plot(t1, type = "l")
lines(t2, type = "l", col = "red")
```



On remarque qu'en rouge on a une trajectoire avec $p = 0.9$ qui ne change pas d'état très souvent parce que la probabilité d'inversion est $1 - 0.9 = 0.1$. Par contre, pour $p = 3$ (en noir) on observe que l'état change fréquemment.

3. Déterminer la mesure invariante de cette chaîne de Markov.

Il est facile à montrer que

$$\pi = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$$

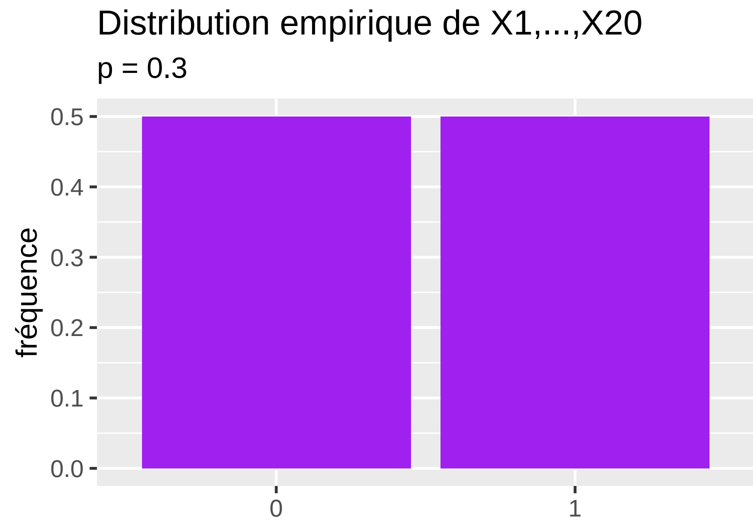
est l'unique loi stationnaire qui ne dépend pas de p. Au premier coup d'oeuil cela peut-etre confus mais après un moment de reflexion il est logique. Lorsque $n \to \infty$, on attend que la chaine passe la moitié du temps dans chaque état, comme la probabilité de passer de `Oui'' à`Non'' est la même probabilité qu'aller dans l'autre sens.

4. Que pouvez-vous dire de la convergence en loi de la chaîne $(X_n)_{n \geq 1}$?

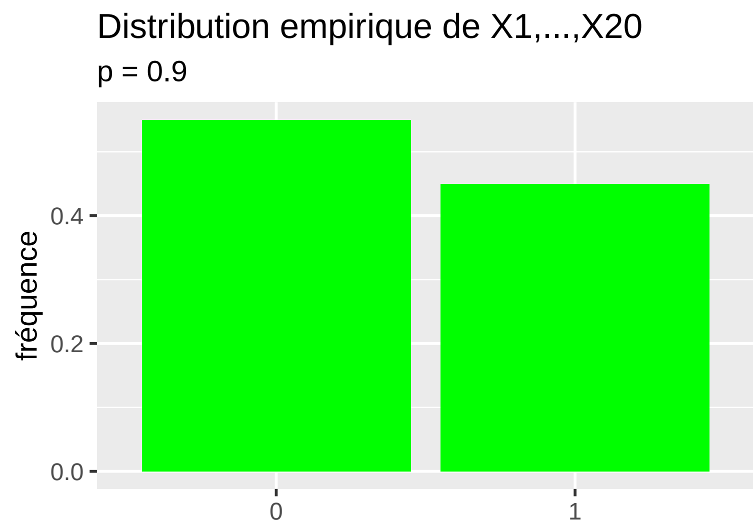
Evidamment la convergence pour les valeurs de p qui sont plus proche de 1 auront une convergence qui est très très lent par rapport aux chaînes avec p près de 0.5 et aussi n'importe quelle valeur inférieure à 0.5. Cela s'explique quand on considère que plus que p est près de 1, plus elle va rester dans le même état. Pour l'illustrer, considérons le graphique suivant.

```r
# Change the states in Exercise 2 as 1.0 -> FALSE, 2.0 -> TRUE
to_tf <- function(X) {
    tf <- X
    tf[X == 1] <- FALSE
    tf[X == 2] <- TRUE
    tf
}

t1_tf <- to_tf(t1)
t2_tf <- to_tf(t2)

tibble(t1_tf) |>
    ggplot(aes(x = factor(t1_tf))) +
    geom_bar(aes(y = ..count../sum(..count..)), fill = "purple") +
    labs(x = "",
         y = "fréquence",
```

```
        title = "Distribution empirique de X1,...,X20",
        subtitle = "p = 0.3")
```



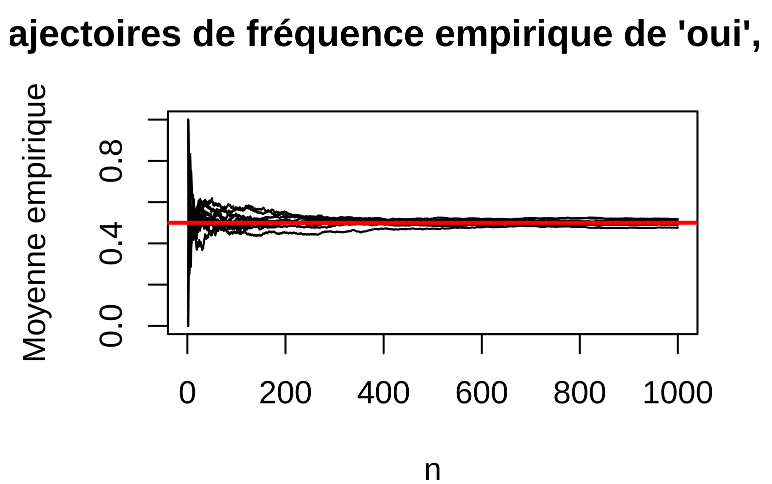

```
tibble(t2_tf) |>
    ggplot(aes(x = factor(t2_tf))) +
    geom_bar(aes(y = ..count../sum(..count..)), fill = "green") +
    labs(x = "",
         y = "fréquence",
         title = "Distribution empirique de X1,...,X20",
         subtitle = "p = 0.9")
```



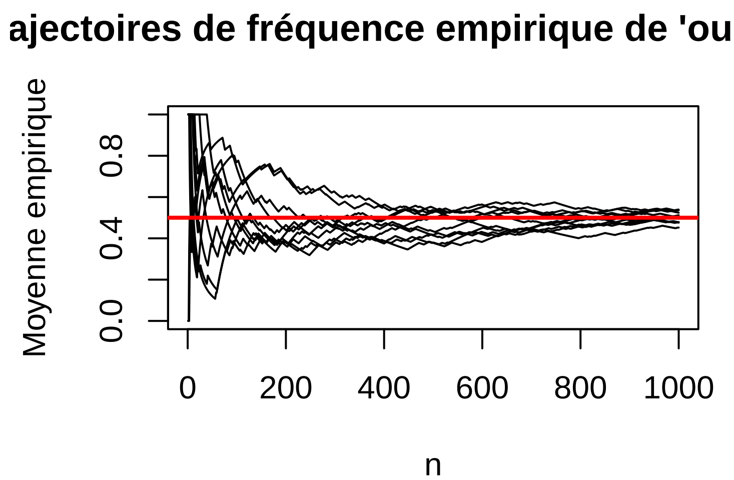
Etudions les trajectoires de plusiers chaînes avec $p = 0.3$ et $p = 0.9$.

```
# Now let's simulate 10 trajectoire's empirique distributions
# of 1000 messages
n_traj <- 10
len_traj <- 1000

emp_dist <- function(single_traj) {

    tf <- to_tf(single_traj)
```

```
    cumsum(tf) / 1:len_traj
}

draw_traj_p <- function(p = 0.5, n_traj = 10, len_traj = 1000) {

    traj <- list()
    for (i in 1:n_traj) {
        traj[[i]] <- sim_chain(c(0.5, 0.5), get_transition_matrix(p), len_traj)
    }

    traj_emp <- map(traj, emp_dist)

    plot(traj_emp[[1]],
         type = "l",
         xlab = "n",
         ylab = "Moyenne empirique",
         main = paste("Trajectoires de fréquence empirique de 'oui', p =", p),
         xlim = c(0, len_traj),
         ylim = c(0, 1))
    for (i in 2:n_traj) {
        lines(traj_emp[[i]])
    }

    abline(h = 0.5, col = "red", lwd = 2)

}

draw_traj_p(0.3)
```




```
draw_traj_p(0.9)
```



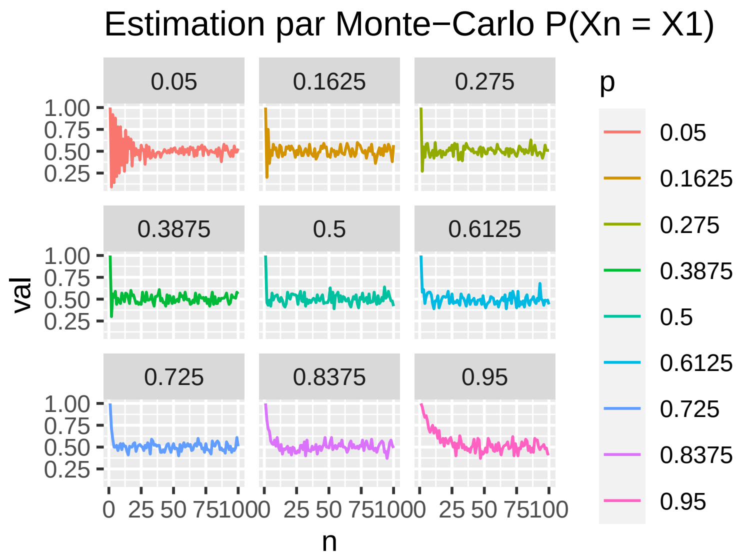
Même si le Rmarkdown a coupé le contenu de nos graphiques, il est évident que le deuxième plot correspond à $p = 0.9$ comme la convergence est beaucoup plus lente.

5. Estimer par Monte-Carlo la probabilité que la réponse transmise par le n-ième intermédiare soit conforme à la réponse initialse.

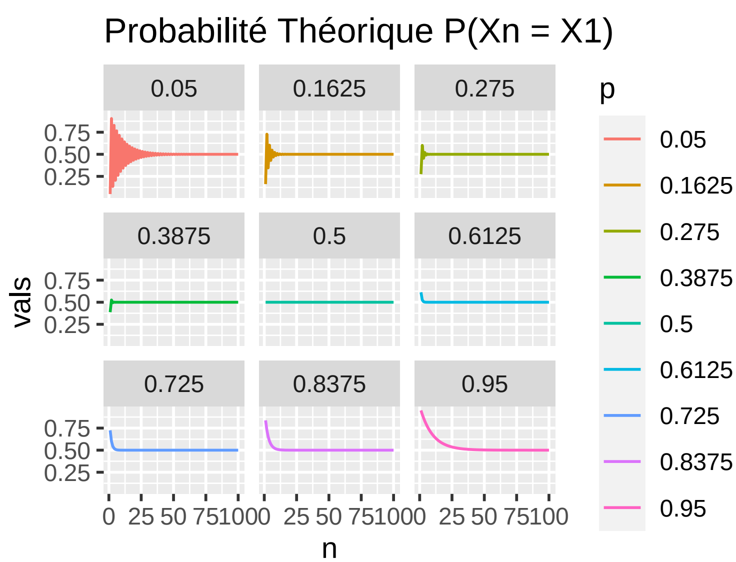
```
p_reponse_conforme <- function(n, p = 0.5) {
    (1 + (2 * p - 1) ** n) / 2
}

# Create monte carlo simulation for nieme probability
# Simulate n_traj trajectories to calculate the probability that the
# nth transmission is the same as the first!
p_reponse_conforme_mc <- function(n, p = 0.5, n_traj = 1000) {

    traj <- list()

    for (i in 1:n_traj) {
        traj[[i]] <- sim_chain(c(0.5, 0.5), get_transition_matrix(p), n)
    }

    # Now see whether or not the condition is true
    pred_fn <- function(single_traj) {
        single_traj[[1]] == single_traj[[n]]
    }

    sum(unlist(map(traj, pred_fn))) / n_traj

}

x <- seq(1, 100)

pr_mc_fn_gen <- function(p, n = 100) {
    function(x) {
        p_reponse_conforme_mc(x, p, n)
    }
}

pr_fn_gen <- function(p) {
```

```r
    function(x) {
        p_reponse_conforme(x, p)
    }
}

P <- linspace(0.05, 0.95, 9)

pr_mc_fns <- map(P, pr_mc_fn_gen)
pr_fns <- map(P, pr_fn_gen)

# I want to calculate the 9 trajectories and 9 functions needed for a data frame
N <- seq(1, 100)

n_fn <- 9

pr_mc_res <- list()
for (i in 1:n_fn) {
    pr_mc_res[[i]] <- map_dbl(N, pr_mc_fns[[i]])
}

pr_res <- list()
for (i in 1:n_fn) {
    pr_res[[i]] <- map_dbl(N, pr_fns[[i]])
}

# now rap all of these up in a data frame
df <- tibble(pr_mc_res[[1]])
for (i in 1:8) {
    df <- add_column(df, pr_mc_res[[i + 1]], .name_repair = "unique")
}

for (i in 1:9) {
    df <- add_column(df, pr_res[[i]], .name_repair = "unique")
}

df <- add_column(df, n = 1:100)

# build the first 9 column names
first_nine <- map_chr(P, function(p) {
    # paste("pr_mc", p, sep = "")
    as.character(p)
})

back_nine <- map_chr(P, function(p) {
    paste("pr", p, sep = "")
})

names(df) <- c(first_nine, back_nine, "n")
df <- df |> pivot_longer(1:9, names_to = "pr_mc", values_to = "val")
names(df) <- c(as.character(P), "n", "pr_mc", "val")
df <- df |> pivot_longer(1:9, names_to = "pr", values_to = "vals")

df |> ggplot(aes(n, val, col = pr_mc)) +
```

```r
    geom_line() +
    facet_wrap(~ pr_mc) +
    labs(col = "p", title = "Estimation par Monte-Carlo P(Xn = X1)")
```




```r
df |> ggplot(aes(n, vals, col = pr)) +
    geom_line() +
    facet_wrap(~ pr) +
    labs(col = "p", title = "Probabilité Théorique P(Xn = X1)")
```



Très jolie. Lorsque $n \to \infty$, la probabilité que $X_n = X_1$ tend vers 0.5. Quand $p < 0.5$, il y a une oscillation autour de 0.5.