

Analyse de séries temporelles avec R

Alexis Gabadinho
alexis.gabadinho@protonmail.com

3 novembre 2024



Table des matières I

- 1 Introduction
- 2 Environnement de travail (rappels)
- 3 Librairies spécialisées et structures de séries temporelles dans R
- 4 Définitions
- 5 Analyse descriptive et représentations graphiques
- 6 Régression Linéaire
- 7 Décomposition d'une série temporelle
- 8 Transformation des données et stabilisation de la variance

Table des matières II

- 9 Séries temporelles non-stationnaires
- 10 Modélisation de séries temporelles stationnaires
- 11 Modélisation de séries non-stationnaires
- 12 Modèles multivariés pour l'analyse de séries temporelles stationnaires
- 13 Cointégration et modèle à correction d'erreur
- 14 Modèles à correction d'erreur

Section 1

Introduction

Support de cours et exercices

- Ce support est écrit avec LaTeX et `knitr` (document incluant du code R exécuté dynamiquement)
- L'environnement RStudio sera utilisé lors de la formation
- Le fichier source du support sera fourni aux participant-e-s, et leur permettra d'exécuter le code contenu dans les diapositives sur leur ordinateur
- Plusieurs jeux de données contenant des séries macroéconomiques sont utilisées pour les exemples. Ces données sont disponibles sous la forme de fichiers 'csv' ou fournies par certaines des librairies R utilisées
- De nombreuses formules mathématiques sont présentes dans les diapositives. Il n'est pas nécessaire de les comprendre, elles seront expliquées en détail lors de la formation

Librairies R requises

■ Les librairies suivantes doivent être installées :

- `insee`
- `tidyverse` (il s'agit en fait d'une collection de librairies dont : `tibble`, `tidyr`, `ggplot2`)
- `GGally` (ajout de fonctions à `ggplot2`)
- `ggfortify` (ajout de fonctions à `ggplot2`)
- `tsibble` (objets de type 'tidy' pour le stockage de données temporelles)
- `feasts` (description, décomposition, représentations graphiques de séries temporelles)
- `fable`
- `structchange` (tests de changement structurel)
- `urca` (test de racine unitaire)
- `vars` (modèles VAR et VEC)

Section 2

Environnement de travail (rappels)

Le tidyverse et ggplot

- Pour importer et manipuler les données, on utilisera principalement le **tidyverse** une collection de bibliothèques pour la science des données (data science)
- Les bibliothèques partagent des structures de données, une philosophie

```
library(tidyverse)

## - Attaching core tidyverse packages ----- tidyverse 2.0.0 -
## v dplyr      1.1.4      v readr      2.1.5
## v forcats    1.0.0      v stringr   1.5.1
## v ggplot2     3.5.1      v tibble    3.2.1
## v lubridate   1.9.3      v tidyr     1.3.1
## v purrr       1.0.2
## - Conflicts ----- tidyverse_conflicts() -
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

- Pour les graphiques nous utiliserons principalement la bibliothèque **ggplot2**, avec le thème 'minimal'

```
library(ggplot2)
theme_set(theme_minimal())
```


Données Nelson-Plosser

Introduction

- Les données 'Nelson-Plosser' contiennent 14 séries temporelles macroéconomiques
- L'article original [1], [Trends and Random Walks in Macroeconomic Time Series](#) est paru dans *Journal of Monetary Economics*)
- Dans leur article, Nelson et Plosser font l'hypothèse que la plupart des séries temporelles macroéconomiques sont mieux décrites par une nonstationarité de type stochastique (unit-root stationarity) que par une tendance déterministe
- La longueur des séries est variable, mais elles se terminent toutes en 1988
- Le jeu de données est décrit [ici](#)

Données Nelson-Plosser

Liste des séries

■ Les 14 séries temporelles :

- cpi = consumer price index
- ip = industrial production
- gnp.nom = nominal GNP (millions de dollars 1988)
- vel = velocity
- emp = employment
- int.rate = interest rate
- nom.wages = nominal wages
- gnp.def = GNP deflator
- money.stock = money stock
- gnp.real = real GNP (milliards de dollars 1958)
- stock.prices = stock prices (SP500)
- gnp.capita = GNP per capita (dollars 1958)
- real.wages = real wages
- unemp = unemployment

Données Nelson-Plosser

Importation des données au format csv

- La fonction `read.csv2` permet d'importer les données à partir du fichier csv
- A noter : dans le fichier csv, le séparateur de champ est une virgule et le séparateur décimal est un point (format européen)

```
Nelson_Plosser <- read.csv2("/home/alex/Devel/Cours-R-TS/data/Nelson_Plosser.csv",
                             header=TRUE, sep=",", dec = ".")
head(Nelson_Plosser)
```

```
##   year      cpi      ip gnp.nom vel emp int.rate nom.wages gnp.def
## 1 1860 3.295837 -0.1053605    NA  NA  NA      NA      NA      NA
## 2 1861 3.295837 -0.1053605    NA  NA  NA      NA      NA      NA
## 3 1862 3.401197 -0.1053605    NA  NA  NA      NA      NA      NA
## 4 1863 3.610918  0.0000000    NA  NA  NA      NA      NA      NA
## 5 1864 3.871201  0.0000000    NA  NA  NA      NA      NA      NA
## 6 1865 3.850148  0.0000000    NA  NA  NA      NA      NA      NA
##  money.stock gnp.real stock.prices gnp.capita real.wages unemp
## 1          NA      NA          NA      NA      NA      NA
## 2          NA      NA          NA      NA      NA      NA
## 3          NA      NA          NA      NA      NA      NA
## 4          NA      NA          NA      NA      NA      NA
## 5          NA      NA          NA      NA      NA      NA
## 6          NA      NA          NA      NA      NA      NA
```

Données Nelson-Plosser

Statistiques descriptives (1)

- Pour les statistiques descriptives de chaque variable on peut utiliser la fonction `descr` de la librairie `summarytools`

```
library(summarytools)
descr(Nelson_Plosser %>% select(1:6))
```

Descriptive Statistics
Nelson_Plosser
N: 129
##

	cpi	emp	gnp.nom	ip	vel	year
Mean	3.99	10.85	12.59	2.73	0.78	1924.00
Std.Dev	0.71	0.45	1.47	1.56	0.38	37.38
Min	3.22	9.96	10.42	-0.11	0.15	1860.00
Q1	3.39	10.53	11.35	1.48	0.53	1892.00
Median	3.78	10.78	12.46	2.77	0.68	1924.00
Q3	4.40	11.19	13.71	4.07	0.92	1956.00
Max	5.87	11.67	15.40	5.23	1.72	1988.00
MAD	0.67	0.49	1.67	1.92	0.24	47.44
IQR	1.01	0.63	2.34	2.59	0.38	64.00
CV	0.18	0.04	0.12	0.57	0.48	0.02
Skewness	1.06	-0.09	0.34	-0.12	0.97	0.00
SE.Skewness	0.21	0.24	0.27	0.21	0.22	0.21
Kurtosis	0.24	-0.94	-1.13	-1.13	0.13	-1.23
N.Valid	129.00	99.00	80.00	129.00	120.00	129.00
Pct.Valid	100.00	76.74	62.02	100.00	93.02	100.00

Données Nelson-Plosser

Matrice de corrélation

- La matrice de corrélation contient les coefficients de corrélation linéaire entre les variables prises 2 à 2

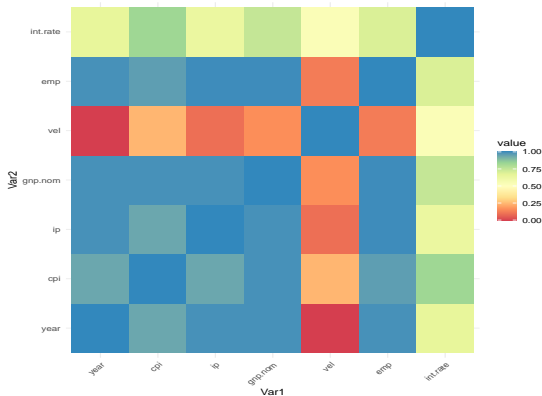
```
datanum <- Nelson_Plosser %>%
  filter(year > 1909) %>%
  select(1:7)
cormat <- round(cor(datanum), 2)
cormat
```

	year	cpi	ip	gnp.nom	vel	emp	int.rate
year	1.00	0.93	0.98	0.98	-0.01	0.98	0.65
cpi	0.93	1.00	0.93	0.98	0.24	0.95	0.82
ip	0.98	0.93	1.00	0.98	0.09	0.99	0.63
gnp.nom	0.98	0.98	0.98	1.00	0.16	0.99	0.74
vel	-0.01	0.24	0.09	0.16	1.00	0.12	0.53
emp	0.98	0.95	0.99	0.99	0.12	1.00	0.69
int.rate	0.65	0.82	0.63	0.74	0.53	0.69	1.00

Données Nelson-Plosser

Matrice de corrélation : Heatmap

```
library(reshape2)
cormat %>% melt() %>% ggplot(aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)) +
  scale_fill_distiller(palette = "Spectral", direction = 1)
```



Données Nelson-Plosser

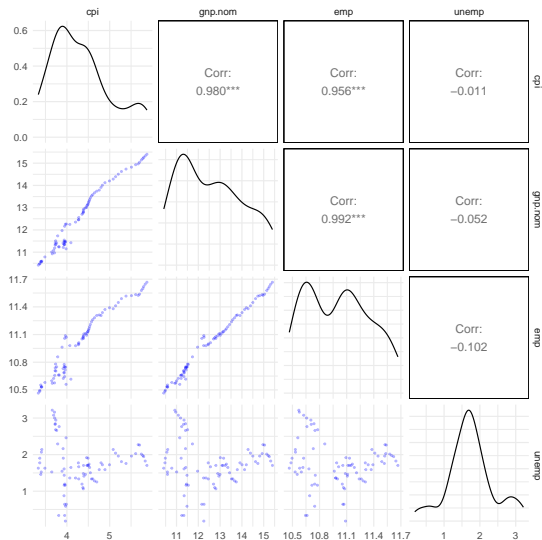
Pairplot (1)

- On peut visualiser la distribution et la corrélation entre variables avec la fonction `ggpairs` de la librairie `GGally`
- On sélectionne ici un sous-ensemble de variables pour limiter la taille du graphique

```
Nelson_Plosser %>%  
  filter(year>=1909) %>%  
  select(cpi, gnp.nom, emp, unemp) %>%  
  ggpairs(upper = list(continuous = wrap("cor", size = 4)),  
         lower = list(continuous = wrap("points", colour="blue", alpha=0.3, size=0.5)))
```

Données Nelson-Plosser

Pairplot (1)



Données retail

Importation des données au format csv

- Série des ventes du commerce de détail et nourriture (Advance retail sales)
- Données mensuelles, en millions de dollars, non corrigées des variations saisonnières
- Source : FRED (Federal Reserve Bank Economic Data)
- On importe les données à partir du fichier csv
- A noter : dans le fichier csv, le séparateur décimal est un point

```
retail <- read.csv2("/home/alex/Devel/Cours-R-TS/data/RXFSN.csv", header=TRUE, sep=";", dec = ".")  
head(retail)
```

```
##          DATE RXFSN  
## 1 1992-01-01 130683  
## 2 1992-02-01 131244  
## 3 1992-03-01 142488  
## 4 1992-04-01 147175  
## 5 1992-05-01 152420  
## 6 1992-06-01 151849
```

Données retail

Conversion de la date

- La date est au format `character`, il faut la convertir

```
summary(retail)

##      DATE              RSXFSN
## Length:381          Min.   :130683
## Class :character    1st Qu.:236830
## Mode  :character    Median :315540
##                      Mean   :326550
##                      3rd Qu.:394764
##                      Max.   :654825
```

- On utilise la fonction `as.Date`

```
retail$DATE <- as.Date(retail$DATE,format="%Y-%m-%d")
summary(retail)

##      DATE              RSXFSN
## Min.   :1992-01-01    Min.   :130683
## 1st Qu.:1999-12-01    1st Qu.:236830
## Median :2007-11-01    Median :315540
## Mean   :2007-10-31    Mean   :326550
## 3rd Qu.:2015-10-01    3rd Qu.:394764
## Max.   :2023-09-01    Max.   :654825
```

La librairie `insee`

Taux de chômage (1)

- La librairie `insee` permet d'accéder directement à de nombreuses séries temporelles macro-économiques
- Ici on récupère la série des taux de chômage

```
library(insee)
df_idbank_list_selected =
  get_idbank_list("CHOMAGE-TRIM-NATIONAL") %>% #Unemployment dataset
  add_insee_title() %>%
  filter(INDICATEUR == "CTTXC") %>% #unemployment rate based on ILO standards
  filter(REF_AREA == "FM") %>% # all France excluding overseas departements
  filter(SEXE == 0) # men and women

list_idbank = df_idbank_list_selected %>% pull(idbank)

fr_unemp = get_insee_idbank(list_idbank, startPeriod = "1950-01") %>% split_title()
```

La librairie insee

Taux de chômage (2)

- La série est sauvegardée dans le fichier `fr_unemp.RData`, elle peut être importée avec la fonction `load`

```
load("/home/alex/Devel/Cours-R-TS/data/fr_unemp.RData")
head(fr_unemp)
```

```
## # A tibble: 6 x 24
```

	DATE	TIME_PERIOD	OBS_VALUE	OBS_STATUS	OBS_QUAL	OBS_TYPE	IDBANK	FREQ
	<date>	<chr>	<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>
## 1	2024-04-01	2024-Q2	7.1	A	DEF	A	001688526	T
## 2	2024-01-01	2024-Q1	7.2	A	DEF	A	001688526	T
## 3	2023-10-01	2023-Q4	7.3	A	DEF	A	001688526	T
## 4	2023-07-01	2023-Q3	7.2	A	DEF	A	001688526	T
## 5	2023-04-01	2023-Q2	7	A	DEF	A	001688526	T
## 6	2023-01-01	2023-Q1	6.9	A	DEF	A	001688526	T

```
## # i 16 more variables: TITLE_FR <chr>, TITLE_FR1 <chr>, TITLE_FR2 <chr>,
## #   TITLE_FR3 <chr>, TITLE_FR4 <chr>, TITLE_EN <chr>, TITLE_EN1 <chr>,
## #   TITLE_EN2 <chr>, TITLE_EN3 <chr>, TITLE_EN4 <chr>, LAST_UPDATE <chr>,
## #   UNIT_MEASURE <chr>, UNIT_MULT <chr>, REF_AREA <chr>, DECIMALS <chr>,
## #   OBS_REV <chr>
```

La librairie `insee`

Taux de chômage (3)

- Les colonnes `TITLE_xxx` contiennent l'intitulé des séries

```
fr_unemp %>% distinct(TITLE_FR)

## # A tibble: 4 x 1
##   TITLE_FR
##   <chr>
## 1 Taux de chômage au sens du BIT - Ensemble - France métropolitaine - Données C-
## 2 Taux de chômage au sens du BIT - Ensemble des 25 à 49 ans - France métropolit-
## 3 Taux de chômage au sens du BIT - Ensemble des 50 ans ou plus - France métropo-
## 4 Taux de chômage au sens du BIT - Ensemble des moins de 25 ans - France métrop-
```

- Début et fin de la série

```
fr_unemp %>% summarise(début=min(DATE), fin=max(DATE))

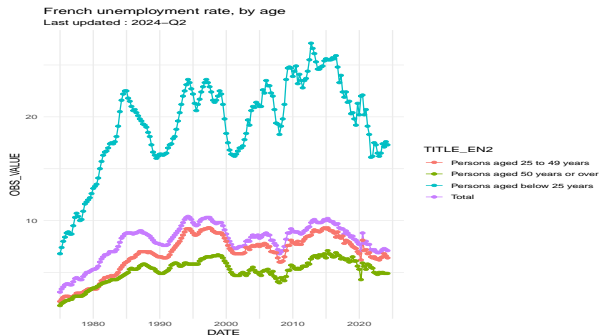
## # A tibble: 1 x 2
##   début      fin
##   <date>    <date>
## 1 1975-01-01 2024-04-01
```

La librairie `insee`

Taux de chômage (4)

■ On visualise les séries avec `ggplot`

```
ggplot(fr_unemp, aes(x = DATE, y = OBS_VALUE, colour = TITLE_EN2)) +
  geom_line() +
  geom_point() +
  ggtitle("French unemployment rate, by age") +
  labs(subtitle = sprintf("Last updated : %s", fr_unemp$TIME_PERIOD[1]))
```



La librairie insee

PIB (1)

- Téléchargement de la série trimestrielle du PIB, corrigé des variations saisonnières (CVS)

```
df_idbank_list_selected =  
    get_idbank_list("CNT-2014-PIB-EQB-RF") %>% # Gross domestic product balance  
    filter(FREQ == "T") %>% #quarter  
    add_insee_title() %>% #add titles  
    filter(OPERATION == "PIB") %>% #GDP  
    filter(NATURE == "VALEUR_ABSOLUE") %>% #rate  
    filter(CORRECTION == "CVS-CJO") #SA-WDA, seasonally adjusted, working day adjusted  
  
idbank = df_idbank_list_selected %>% pull(idbank)  
  
fr_pib = get_insee_idbank(idbank)
```

La librairie insee

PIB (2)

- La série est sauvegardée dans le fichier `fr_pib.RData`, elle peut être importée avec la fonction `load`

```
load("/home/alex/Devel/Cours-R-TS/data/fr_pib.RData")
head(fr_pib)

## # A tibble: 6 x 16
##   DATE      TIME_PERIOD OBS_VALUE OBS_STATUS OBS_QUAL OBS_TYPE IDBANK  FREQ
##   <date>    <chr>         <dbl> <chr>      <chr>    <chr>   <chr>   <chr>
## 1 2024-01-01 2024-Q1      595694 A        DEF      A       010565708 T
## 2 2023-10-01 2023-Q4      594336 A        DEF      A       010565708 T
## 3 2023-07-01 2023-Q3      593602 A        DEF      A       010565708 T
## 4 2023-04-01 2023-Q2      593156 A        DEF      A       010565708 T
## 5 2023-01-01 2023-Q1      589375 A        DEF      A       010565708 T
## 6 2022-10-01 2022-Q4      589499 A        DEF      A       010565708 T
## # i 8 more variables: TITLE_FR <chr>, TITLE_EN <chr>, LAST_UPDATE <chr>,
## #   UNIT_MEASURE <chr>, UNIT_MULT <chr>, REF_AREA <chr>, DECIMALS <chr>,
## #   OBS_REV <chr>
```


La librairie insee

PIB (3)

■ Date de début de la série

```
min(fr_pib$DATE)
## [1] "1949-01-01"
```

■ Les colonnes TITLE_xxx contiennent l'intitulé des séries

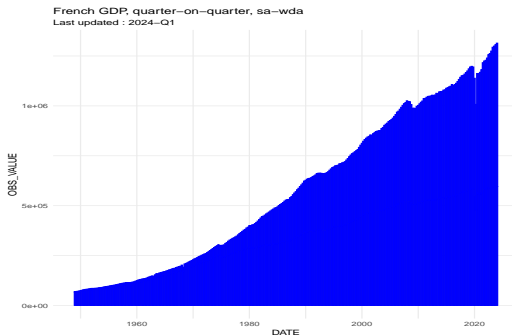
```
fr_pib %>% distinct(TITLE_EN)
## # A tibble: 2 x 1
##   TITLE_EN
##   <chr>
## 1 Total gross domestic product - Volumes chained at previous year prices - SA-W-
## 2 Total gross domestic product - Values at current prices - SA-WDA series - Sto-
```

La librairie `insee`

PIB (4)

■ Visualisation avec `ggplot`

```
ggplot(fr_pib, aes(x = DATE, y = OBS_VALUE)) +  
  geom_col(color="blue") +  
  ggtitle("French GDP, quarter-on-quarter, sa-wda") +  
  labs(subtitle = sprintf("Last updated : %s", fr_pib$TIME_PERIOD[1]))
```



La librairie `insee`

Indice des prix à la consommation (1)

- Téléchargement de la série mensuelle de l'indice des prix à la consommation (attention, il y a énormément de séries, et il faut repérer l'identifiant de la série (`idbank`))

```
df_idbank_list_selected =  
  get_idbank_list("IPC-2015") %>% # Gross domestic product balance  
  filter(FREQ == "M" & idbank=="001759970") %>%  
  add_insee_title()  
  
idbank = df_idbank_list_selected %>% pull(idbank)  
  
fr_cpi = get_insee_idbank(idbank)
```

La librairie `insee`

Indice des prix à la consommation (2)

- La série est sauvegardée dans le fichier `fr_cpi.RData`
- Pour cette série les données sont **mensuelles**

```
load("/home/alex/Devel/Cours-R-TS/data/fr_cpi.RData")
head(fr_cpi)
```

```
## # A tibble: 6 x 17
```

##	DATE	TIME_PERIOD	OBS_VALUE	OBS_STATUS	OBS_QUAL	OBS_TYPE	IDBANK	FREQ
##	<date>	<chr>	<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>
## 1	2024-10-01	2024-10	120.	P	P	A	001759970	M
## 2	2024-09-01	2024-09	120.	P	P	A	001759970	M
## 3	2024-08-01	2024-08	121.	A	DEF	A	001759970	M
## 4	2024-07-01	2024-07	120.	A	DEF	A	001759970	M
## 5	2024-06-01	2024-06	120.	A	DEF	A	001759970	M
## 6	2024-05-01	2024-05	120.	A	DEF	A	001759970	M

```
## # i 9 more variables: TITLE_FR <chr>, TITLE_EN <chr>, LAST_UPDATE <chr>,
## #   UNIT_MEASURE <chr>, UNIT_MULT <chr>, REF_AREA <chr>, DECIMALS <chr>,
## #   DATE_JO <chr>, OBS_REV <chr>
```

La librairie insee

Indice des prix à la consommation (3)

■ Les colonnes TITLE_xxx contiennent l'intitulé des séries

```
fr_cpi %>% distinct(TITLE_FR)
## # A tibble: 1 x 1
##   TITLE_FR
##   <chr>
## 1 Indice des prix à la consommation - Base 2015 - Ensemble des ménages - France-
```

■ Début et fin de la série

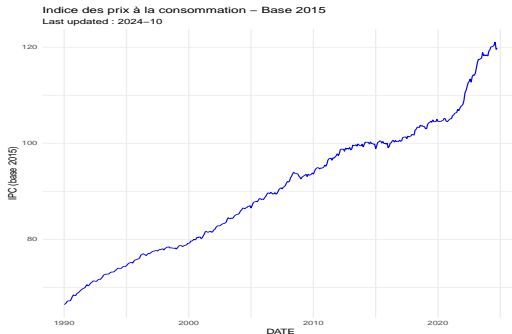
```
fr_cpi %>% summarise(début=min(fr_cpi$DATE), fin=max(fr_cpi$DATE))
## # A tibble: 1 x 2
##   début      fin
##   <date>    <date>
## 1 1990-01-01 2024-10-01
```

La librairie `insee`

Indice des prix à la consommation (4)

■ Visualisation avec `ggplot`

```
ggplot(fr_cpi, aes(x = DATE, y = OBS_VALUE)) +  
  geom_line(color="blue") +  
  ggtitle("Indice des prix à la consommation - Base 2015") +  
  labs(subtitle = sprintf("Last updated : %s", fr_cpi$TIME_PERIOD[1])) +  
  ylab("IPC (base 2015)")
```



Environnement de travail

Exercices

- 1 Le fichier `pib_fr.csv` contient la série temporelle du PIB et de ses composants depuis 1949 (valeurs aux prix courants - Source : INSEE) :
 - Importez les données à partir du fichier `csv`
 - Explorez les données avec des statistiques descriptives et des graphiques
- 2 La série de l'indice des prix à la consommation est contenue dans le fichier `fr_cpi.RData`
 - Importez les données
 - La fréquence de cette série est mensuelle. Transformez-là en série trimestrielle (moyenne sur les trois mois d'un trimestre). Les fonctions `quarter` et `year` vous seront utiles.
- 3 Fusionnez les séries `fr_pbi`, `fr_unemp` et `fr_cpi` dans un jeu de données unique `fr_macro` (utilisez les fonctions `'_join'` de `dlpyr`)

Environnement de travail

Exercices : Solutions (1)

■ Importation du fichier csv

```
pib <- read.csv("/home/alex/Devel/Cours-R-TS/data/pib_fr.csv", sep=";", dec=',')
head(pib)
```

```
##  PERIODE PIB  P7 P3M P3P P31G P32G  P3 P51S P51B P51G P51M P51P P51 P54  P6
## 1  1949T1 3.2 0.4 1.9 0.1  0.2  0.2 2.4  0.4    0  0.1  0.1    0 0.6 0.1 0.4
## 2  1949T2 3.2 0.4 2.0 0.1  0.3  0.2 2.5  0.4    0  0.1  0.1    0 0.6 0.1 0.5
## 3  1949T3 3.3 0.4 2.1 0.1  0.3  0.2 2.6  0.4    0  0.1  0.1    0 0.6 0.1 0.5
## 4  1949T4 3.4 0.4 2.1 0.1  0.3  0.3 2.6  0.4    0  0.1  0.1    0 0.7 0.1 0.5
## 5  1950T1 3.6 0.5 2.1 0.1  0.3  0.3 2.7  0.5    0  0.1  0.1    0 0.7 0.2 0.5
## 6  1950T2 3.8 0.5 2.2 0.1  0.3  0.3 2.8  0.5    0  0.1  0.1    0 0.7 0.2 0.6
```


Environnement de travail

Exercices : Solutions (2)

■ Importation et transformation de la série fr_cpi

```
load("/home/alex/Devel/Cours-R-TS/data/fr_cpi.RData")
fr_cpi <- fr_cpi %>% mutate(Année=year(DATE), Trimestre=quarter(DATE)) %>%
  group_by(Année, Trimestre) %>%
  rename(cpi=OBS_VALUE) %>%
  summarise(cpi=mean(cpi))

## 'summarise()' has grouped output by 'Année'. You can override using the
## '.groups' argument.

head(fr_cpi)

## # A tibble: 6 x 3
## # Groups:   Année [2]
##   Année Trimestre   cpi
##   <dbl>     <int> <dbl>
## 1  1990         1  66.6
## 2  1990         2  67.2
## 3  1990         3  67.7
## 4  1990         4  68.3
## 5  1991         1  68.8
## 6  1991         2  69.4
```

Environnement de travail

Exercices : Solutions (3)

■ Fusion des jeux de données

```
load("/home/alex/Devel/Cours-R-TS/data/fr_pib.RData")
fr_pib <- fr_pib %>% mutate(Année=year(Date), Trimestre=quarter(Date)) %>%
  rename(gnp=OBS_VALUE) %>%
  select(Année, Trimestre, gnp)

load("/home/alex/Devel/Cours-R-TS/data/fr_unemp.RData")
fr_unemp <- fr_unemp %>% mutate(Année=year(Date), Trimestre=quarter(Date)) %>%
  rename(unemp=OBS_VALUE) %>%
  filter(TITLE_FR2=='Ensemble') %>%
  select(Année, Trimestre, unemp)

fr_macro <- fr_pib %>%
  left_join(fr_unemp, by=c("Année", "Trimestre")) %>%
  left_join(fr_cpi, by=c("Année", "Trimestre"))

head(fr_macro)

## # A tibble: 6 x 5
##   Année Trimestre   gnp unemp   cpi
##   <dbl>   <int>   <dbl> <dbl> <dbl>
## 1  2024         1 595694   7.2  119.
## 2  2023         4 594336   7.3  118.
## 3  2023         3 593602   7.2  118.
## 4  2023         2 593156    7   118.
## 5  2023         1 589375   6.9  116.
## 6  2022         4 589499   6.9  114.
```

Section 3

Librairies spécialisées et structures de séries temporelles dans R

Liste des librairies

- En plus de la librairie standard `stats`, il existe plusieurs librairies R pour la manipulation et l'analyse des séries temporelles
 - `tsibble`
 - `forecasts`
 - `feasts`
 - `fable`
 - `ggfortify`
 - `tseries`
 - `zoo`
- Dans cette partie on se focalise sur les classes permettant de stocker les séries temporelles fournies par les librairies `stats` et `tsibble`

Les objets `ts` et `mts`

- La classe de base fournie par R pour représenter des séries temporelles s'appelle `ts` (`ts` = time series, série univariée) ou `mts` (`mts` = multiple time series, série multivariée)
- Cette classe est définie dans le package `stats`
- Elle concerne des séries temporelles qui sont échantillonnées à des périodes **équidistantes** dans le temps

Les objets `ts` et `mts`

Paramètres

- Les objets de classe `ts` ou `mts` possèdent trois paramètres caractéristiques :
 - `frequency` : nombre d'observations par unité de temps. Si l'unité de temps de la série est l'année, la valeur 4 correspond à des trimestres et la valeur 12 à des mois
 - `start` : date de début de la série temporelle (nombre unique ou vecteur de deux entiers représentant respectivement une unité temporelle (e.g. une année) et une subdivision de cette unité (mois ou trimestre selon la valeur du paramètre `frequency`))
 - `end` : date de fin de la série temporelle, exprimée comme pour le paramètre `start`

Les objets `ts` et `mts`

Données `NelPlo` (1)

- Les données Nelson-Plosser sont également présentes dans la librairie `tseries`
- Les données annuelles (`frequency=1`) sont contenues dans un objet de type `mts`

```
library(tseries)

## Registered S3 method overwritten by 'quantmod':
##   method      from
##   as.zoo.data.frame zoo

data(NelPlo)
class(NelPlo)

## [1] "mts" "ts"
```

Les objets `ts` et `mts`

Données `NelPlo` (2)

- La fonction `window` permet d'extraire une portion d'une série temporelle `ts`
- L'ensemble des séries est complet à partir de 1909 (il y a des valeurs manquantes sur certaines séries avant)

```
window(NelPlo, start=1905, end=1910)

## Time Series:
## Start = 1905
## End = 1910
## Frequency = 1
##      cpi      ip  gnp.nom      vel      emp int.rate nom.wages  gnp.def
## 1905 3.295837 2.219203      NA 0.7561220 10.37274      3.50 6.329721 3.277145
## 1906 3.332205 2.282382      NA 0.8241754 10.42671      3.55 6.357842 3.303217
## 1907 3.367296 2.302585      NA 0.8241754 10.44497      3.80 6.393591 3.342862
## 1908 3.332205 2.140066      NA 0.7227060 10.41169      3.95 6.306275 3.335770
## 1909 3.332205 2.302585 10.41631 0.7839015 10.46516      3.77 6.395262 3.370738
## 1910 3.367296 2.360854 10.47164 0.7747272 10.48464      3.80 6.478510 3.397858
##      money.stock gnp.real stock.prices gnp.capita real.wages      unemp
## 1905      2.326302      NA      2.196113      NA      3.033991 1.4586150
## 1906      2.405142      NA      2.265921      NA      3.025776 0.5306283
## 1907      2.451005      NA      2.059239      NA      3.026261 1.0296194
## 1908      2.437116      NA      2.051556      NA      2.973998 2.0794415
## 1909      2.540026 4.760463      2.273156      7.163172 3.062924 1.6292405
## 1910      2.590767 4.788325      2.235376      7.170120 3.111291 1.7749524
```


Les objets `ts` et `mts`

Données `growthofmoney` (Growth of Money Supply)

- Deux séries temporelles concernant le contrôle de la monnaie par la réserve fédérale aux USA
- Article original : R.L. Hetzel (1981), *The Federal Reserve System and Control of the Money Supply in the 1970's*. *Journal of Money, Credit and Banking*
- Série temporelle multivariée, données trimestrielles

```
library(lmtest)
data(growthofmoney)
window(growthofmoney, end=c(1971,4))

##          TG1.TGO AGO.TGO
## 1970 Q2         0.0   -0.4
## 1970 Q3         1.0   -1.0
## 1970 Q4         1.0    1.1
## 1971 Q1         2.5    5.8
## 1971 Q2        -6.0   -4.4
## 1971 Q3         4.5   -1.6
## 1971 Q4        -0.5    1.6
```

Les objets `ts` et `mts`

Données `USIncExp` (US Income and Expenditure)

- Deux séries macro-économiques (USA) de janvier 1959 à février 2001, corrigées des variations saisonnières :
 - Revenus personnels mensuels agrégés (millions de dollars)
 - Dépenses de consommation agrégées (millions de dollars)

```
library(strucchange)
data("USIncExp")
window(USIncExp, start = c(1959, 6), end = c(1960, 6))
```

##		income	expenditure
##	Jun 1959	396.3	319.2
##	Jul 1959	396.5	318.8
##	Aug 1959	395.0	321.2
##	Sep 1959	396.2	325.2
##	Oct 1959	397.8	323.8
##	Nov 1959	401.2	323.9
##	Dec 1959	405.7	323.9
##	Jan 1960	407.0	324.6
##	Feb 1960	407.7	326.4
##	Mar 1960	408.6	331.2
##	Apr 1960	411.3	337.6
##	May 1960	412.8	331.1
##	Jun 1960	413.1	331.2

Les objets `ts` et `mts`

Conversion des données `retail`

- Pour les données `retail`, `frequency=12` (mois)

```
retail_ts <- ts(retail['RSXFSN'], frequency=12, start=c(1997,3))
window(retail_ts, start=2020)
```

```
##          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct
## 2020 391483 451076 350291 341828 393094 387358 409453 397809 406550 404923
## 2021 396067 464014 351787 363543 405012 394764 413059 410100 406013 416538
## 2022 413933 482011 369102 365610 423016 407256 434511 423071 416908 431797
## 2023 441923 496745 389703 380562 444873 420099 464101 443603 441604 455946
## 2024 459172 493167 398783 384243 446272 441878 474963 447578 461010 472517
## 2025 467189 518979 417123 413923 426258 376867 461785 479784 492689 487325
## 2026 490941 557696 461308 437031 560218 554269 565177 557446 552179 550169
## 2027 574978 627113 516923 507901 598541 596690 616626 609743 599929 613508
## 2028 605205 654825 547156 529374 604084 588220 631496 612243 605403 628892
##          Nov      Dec
## 2020 381996 392602
## 2021 394237 397425
## 2022 412522 417753
## 2023 417072 441746
## 2024 428241 454941
## 2025 474038 493601
## 2026 529133 553588
## 2027 577966 597170
## 2028 592660
```

Les objets `ts` et `mts`

Extraction de l'index

- La fonction `tsp` permet d'extraire les propriétés d'un objet `ts` ou `mts`

```
tsp(retail_ts)
## [1] 1997.167 2028.833 12.000
```

- Extraction de l'index

```
retail2022 <- window(retail_ts, start=c(2022,1), end=c(2022,12))
time(retail2022)

##           Jan           Feb           Mar           Apr           May           Jun           Jul           Aug
## 2022 2022.000 2022.083 2022.167 2022.250 2022.333 2022.417 2022.500 2022.583
##           Sep           Oct           Nov           Dec
## 2022 2022.667 2022.750 2022.833 2022.917
```

- Conversion de l'index au format numérique

```
library(lubridate)
as.numeric(time(retail2022))

## [1] 2022.000 2022.083 2022.167 2022.250 2022.333 2022.417 2022.500 2022.583
## [9] 2022.667 2022.750 2022.833 2022.917
```

Les objets `tsibble`

- La librairie `tsibble` fournit une structure de type 'tidy' pour les données temporelles ainsi que des outils pour le traitement de ces données
- Les objets de la classe `tsibble` possèdent deux attributs principaux :
 - `index` est la variable qui représente le temps (qui permet d'ordonner du passé au présent)
 - `key` identifie (éventuellement) la série (unité d'observation)
- Chaque observation est identifiée de manière unique par la combinaison `index` et `key`

Les objets `tsibble`

Conversion des données `growthofmoney`

- On utilise la fonction `as_tsibble`

```
library(tsibble)
gmoney_tsbl <- growthofmoney %>% as_tsibble()
gmoney_tsbl

## # A tsibble: 38 x 3 [1Q]
## # Key:           key [2]
##   index key      value
##   <qtr> <chr>   <dbl>
## 1 1970 Q2 TG1.TGO    0
## 2 1970 Q3 TG1.TGO    1
## 3 1970 Q4 TG1.TGO    1
## 4 1971 Q1 TG1.TGO   2.5
## 5 1971 Q2 TG1.TGO  -6
## 6 1971 Q3 TG1.TGO   4.5
## 7 1971 Q4 TG1.TGO  -0.5
## 8 1972 Q1 TG1.TGO  -1
## 9 1972 Q2 TG1.TGO   0.5
## 10 1972 Q3 TG1.TGO  -1.5
## # i 28 more rows
```

Les objets `tsibble`

Conversion des données `NelPlo` (1)

- La transformation d'un objet de type `mts` en objet `tsibble` produit un jeu de données au format long

```
nelplo_tsbl <- NelPlo %>% as_tsibble()
nelplo_tsbl %>% filter(index>1980 & key=="gnp.capita")

## # A tsibble: 8 x 3 [1Y]
## # Key:      key [1]
##   index key      value
##   <dbl> <chr>    <dbl>
## 1  1981 gnp.capita  8.34
## 2  1982 gnp.capita  8.31
## 3  1983 gnp.capita  8.33
## 4  1984 gnp.capita  8.39
## 5  1985 gnp.capita  8.41
## 6  1986 gnp.capita  8.43
## 7  1987 gnp.capita  8.46
## 8  1988 gnp.capita  8.49
```

Les objets `tsibble`

Conversion des données `NelPlo` (2)

- Les 14 séries sont identifiées par la variable `key`

```
nelplo_tsbl %>% distinct(key)
```

```
## # A tibble: 14 x 1
##   key
##   <chr>
## 1 cpi
## 2 ip
## 3 gnp.nom
## 4 vel
## 5 emp
## 6 int.rate
## 7 nom.wages
## 8 gnp.def
## 9 money.stock
## 10 gnp.real
## 11 stock.prices
## 12 gnp.capita
## 13 real.wages
## 14 unemp
```


Les objets `tsibble`

Conversion des données `retail`

- Les données `retail` ne contiennent qu'une seule série, il n'y a pas de variable `key`

```
retail_tsbl <- retail_ts %>% as_tsibble()
head(retail_tsbl)

## # A tsibble: 6 x 2 [1M]
##       index  value
##   <mth> <dbl>
## 1 1997 mars 130683
## 2 1997 avril 131244
## 3 1997 mai 142488
## 4 1997 juin 147175
## 5 1997 juil. 152420
## 6 1997 août 151849
```

Librairies spécialisées et structure de séries temporelles

Exercice

- Convertissez les données PIB en objet `mts`
- Convertissez les données PIB en objet `tsibble`

Section 4

Définitions

Série temporelle et processus stochastique

- Série temporelle univariée à temps discret = ensemble d'observations dans \mathbb{R} enregistrées à un temps spécifique $t \in \mathbb{Z}$
- En statistique l'observation x est considérée comme la réalisation d'une **variable aléatoire** X
- Une série temporelle $(x_t)_{t \in \mathbb{Z}}$ sera considérée comme la réalisation d'un **processus stochastique** $(X_t)_{t \in \mathbb{Z}}$
- Processus stochastique \Rightarrow pour tout $t \in \mathbb{Z}$ fixé, X_t est une variable aléatoire réelle
- L'objectif est d'étudier les caractéristiques principales de ce processus (tendance, variation saisonnière), de le modéliser et de faire des prévisions

Stationnarité

- Dans de très nombreux cas, on ne peut pas renouveler la suite de mesures dans des conditions identiques
- Pour que le modèle déduit à partir d'une suite d'observations ait un sens, il faut que toute portion de la trajectoire observée fournisse des informations sur la loi du processus et que des portions différentes, mais de même longueur, fournissent les mêmes indications. D'où la notion de stationnarité.
- Un processus $(X_t)_{t \in \mathbb{Z}}$ est (faiblement) stationnaire si son espérance et ses autocovariances sont invariantes par translation dans le temps :
 - $\forall t \in \mathbb{Z} : \mathbb{E}(X_t) = \mu$
 - $\forall t \in \mathbb{Z}, \forall h \in \mathbb{Z} : \text{Cov}(X_t, X_{t-h})$ dépend de l'intervalle h , mais pas de t

Fonction d'autocovariance et d'autocorrélation

- La fonction d'autocorrélation (ACF) est la corrélation entre x_t et x_{t-1} , x_{t-2} , x_{t-3} , etc ... :

$$\rho_j = \frac{\text{Cov}(y_t, y_{t-j})}{\sqrt{\text{Var}(y_t) \cdot \text{Var}(y_{t-j})}}$$

- Elle permet d'identifier une structure dans la série temporelle

Fonction d'autocorrélation avec `acf`

- Pour les objets de la classe `ts` ou `mts` on peut utiliser la fonction `acf` (cette fonction produit un graphique par défaut)

```
acf(NelPlo[, 'cpi'], plot=FALSE)

##
## Autocorrelations of series 'NelPlo[, "cpi"]', by lag
##
##      0      1      2      3      4      5      6      7      8      9      10      11      12
## 1.000 0.966 0.928 0.889 0.853 0.818 0.784 0.748 0.712 0.677 0.645 0.615 0.585
##      13      14      15      16      17      18      19      20      21
## 0.558 0.533 0.509 0.489 0.469 0.450 0.431 0.412 0.396
```

- La corrélation entre le cpi à l'instant t et le cpi à l'instant $t - 1$ (une année avant) est de 0.966

Fonction d'autocorrélation avec ACF

- Pour les objets de la classe `tsibble` on peut utiliser la fonction `ACF` de la librairie `feasts`

```
library(feasts)

## Le chargement a nécessité le package : fabletools

nelplo_tsbl %>%
  filter(key=="cpi") %>%
  ACF(value)

## # A tsibble: 21 x 3 [1Y]
## # Key:          key [1]
##   key          lag  acf
##   <chr> <cf_lag> <dbl>
## 1 cpi          1Y 0.966
## 2 cpi          2Y 0.928
## 3 cpi          3Y 0.889
## 4 cpi          4Y 0.853
## 5 cpi          5Y 0.818
## 6 cpi          6Y 0.784
## 7 cpi          7Y 0.748
## 8 cpi          8Y 0.712
## 9 cpi          9Y 0.677
## 10 cpi         10Y 0.645
## # i 11 more rows
```


Bruit blanc

Définition

- Un bruit blanc (white noise) ϵ_t est une série de variables aléatoires **non corrélées** de moyenne nulle et de variance constante

- $(\epsilon_t)_{t \in \mathbb{Z}}$ est un bruit blanc faible si :

- Son espérance est égale à 0 :

$$\forall t \in \mathbb{Z} : \mathbb{E}(\epsilon_t) = 0$$

- Sa variance est constante :

$$\mathbb{E}(\sigma_t^2) = \sigma^2$$

- La covariance entre (ϵ_t) et (ϵ_{t-h}) est nulle :

$$\forall t \in \mathbb{Z}, \forall h \in \mathbb{Z} : \text{Cov}(\epsilon_t, \epsilon_{t-h}) = 0$$

- Un bruit blanc gaussien ϵ_t est une série de variables aléatoires indépendantes et identiquement distribuées (i.i.d), suivant une loi normale $N(0, \sigma_z^2)$ de moyenne nulle et de variance σ_z^2

- Un bruit blanc gaussien est une série **strictement stationnaire**

Bruit blanc gaussien

Simulation (1)

- On simule un bruit blanc gaussien en générant 100 nombres aléatoires issus d'une loi normale $N(0, 1)$

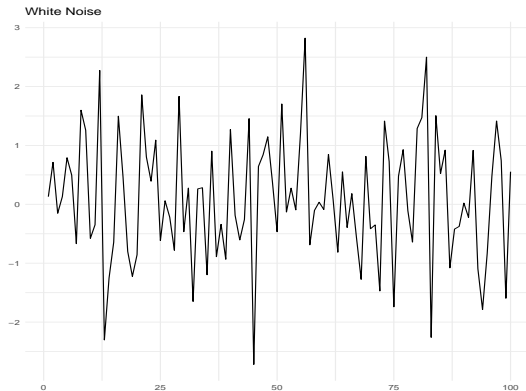
```
bbg <- ts(rnorm(100))  
window(bbg, start=1, end=10)  
  
## Time Series:  
## Start = 1  
## End = 10  
## Frequency = 1  
## [1] 0.1300670 0.7136762 -0.1519150 0.1307468 0.7910630 0.4960524  
## [7] -0.6695178 1.5989119 1.2586362 -0.5778682
```

- Pour la représentation graphique, on utilise la fonction `autoplot` de la librairie `ggfortify`

```
library(ggfortify)  
autoplot(bbg) +  
  ggtitle("White Noise")
```

Bruit blanc gaussien

Simulation (2)



Bruit blanc gaussien

Simulation (3)

- On peut vérifier que la série ne présente pas d'autocorrélation avec la fonction `acf`

```
acf(bbg, plot=FALSE)
##
## Autocorrelations of series 'bbg', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
## 1.000 -0.146 -0.143 -0.075  0.188  0.073 -0.105  0.060  0.086 -0.161 -0.098
##     11     12     13     14     15     16     17     18     19     20
## -0.096  0.035  0.015 -0.217  0.099 -0.108 -0.031  0.059 -0.081  0.065
```

Bruit blanc

Test de Portmanteau

- A l'issue d'une modélisation, il nous faudrait idéalement obtenir un signal résiduel qui ne contient plus d'information temporelle
- Dans le cadre des modèles ARMA, on souhaite que le résidu soit un bruit blanc (faible), c'est-à-dire sans dépendance temporelle linéaire
- On peut tester la **blancheur** d'une série $y_t, t = 1, \dots, T$ en utilisant le test de Portmanteau
- La statistique de Portmanteau est calculée à partir des k premiers coefficients d'autocorrélation $Q_k = T \sum_{h=1}^k \hat{\rho}^2(h)$ où k est un décalage choisi par l'utilisateur et ρ_j l'estimateur du coefficient d'autocorrélation d'ordre h de la série y_t

Bruit blanc

Test de Portmanteau : Exemple

- On peut utiliser les fonctions `box_pierce` et `ljung_box` (pour les petits échantillons) de la librairie `feasts`
- Par défaut $k = 1$ (argument `lag=1`)
- Hypothèse H_0 : pas d'autocorrélation

```
box_pierce(rnorm(100))  
##      bp_stat bp_pvalue  
## 0.6371817 0.4247332
```

- La p-valeur est supérieure à 0.05, on accepte H_0
- A noter : quand le test est appliqué non sur des v.a. indépendantes, mais sur les résidus d'un ajustement estimant m paramètres, on utilise le paramètre 'dof' (degrees of freedom, degrés de liberté)

Marche aléatoire

Définition

- La série $y_t = y_{t-1} + \epsilon_t$ est une **marche aléatoire** (random walk)
- Une série suivant une marche aléatoire prend, à deux dates consécutives, des valeurs proches et la variation par rapport à la date précédente est **indépendante du passé** (c'est un bruit blanc)
- En exprimant y_{t-1} en fonction de y_{t-2}, \dots et d'une valeur initiale y_0 on obtient :

$$y_t = y_0 + \epsilon_1 + \epsilon_2 + \dots + \epsilon_t$$

- Espérance :

$$E(y_t) = y_0$$

- Variance :

$$\text{var}(y_t) = t \cdot \sigma_\epsilon^2$$

- Covariance :

$$\text{cov}(y_t, y_{t+k}) = t \cdot \sigma_\epsilon^2, (k > 0)$$

- Il s'agit d'une série **non-stationnaire** car ni la variance ni l'autocorrélation ne sont constantes

Marche aléatoire

Simulation (1)

- On simule une marche aléatoire avec un bruit blanc gaussien de moyenne 0 et d'écart type 1

```
tmax <- 100
wnoise <- rnorm(99, mean=0, sd=1)
y <- rep(0,tmax)

for (t in 2:tmax) {
  y[t] = y[t-1] + wnoise[t-1]
}
rw <- tibble(time=1:tmax, y=y)
head(rw)

## # A tibble: 6 x 2
##   time     y
##   <int> <dbl>
## 1     1  0
## 2     2  1.00
## 3     3  1.49
## 4     4  0.599
## 5     5  0.0623
## 6     6  0.377
```


Marche aléatoire

Simulation (2)

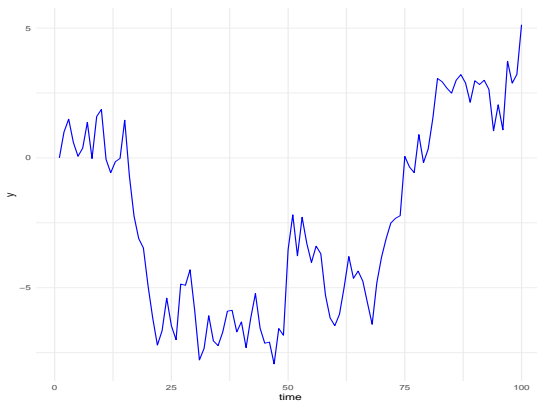
■ L'auto-corrélation est significative

```
acf(rw[, "y"], plot=FALSE)
##
## Autocorrelations of series 'rw[, "y"]', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10     11     12
## 1.000 0.933 0.880 0.835 0.780 0.749 0.704 0.661 0.622 0.577 0.527 0.473 0.434
##     13     14     15     16     17     18     19     20
## 0.389 0.329 0.264 0.202 0.157 0.118 0.079 0.047
```

Marche aléatoire (random walk)

Simulation (3)

```
ggplot(rw) +  
  geom_line(aes(x=time, y=y), colour="blue")
```



Marche aléatoire avec dérive

Définition

- La série $y_t = \alpha + y_{t-1} + \epsilon_t$ est une **marche aléatoire** avec dérive (random walk with drift)
- En exprimant y_{t-1} en fonction de y_{t-2}, \dots et d'une valeur initiale y_0 on obtient :

$$y_t = \alpha \cdot t + y_0 + \epsilon_1 + \epsilon_2 + \dots + \epsilon_t$$

- Espérance :

$$E(y_t) = \alpha \cdot t + y_0$$

- Variance :

$$\text{var}(y_t) = t \cdot \sigma_z^2$$

- Covariance :

$$\text{cov}(y_t, y_{t+k}) = t \cdot \sigma_z^2, (k > 0)$$

Marche aléatoire avec dérive

Simulation (1)

- On simule une marche aléatoire avec dérive ($\alpha = 0.8$), on réutilise bruit blanc gaussien `wnoise` généré pour l'exemple précédent

```
tmax <- 100
alpha <- 0.8

y <- rep(0,tmax)

for (t in 2:tmax) {
  y[t] = alpha + y[t-1] + wnoise[t-1]
}
rwd <- tibble(time=1:tmax, y=y)
head(rwd)

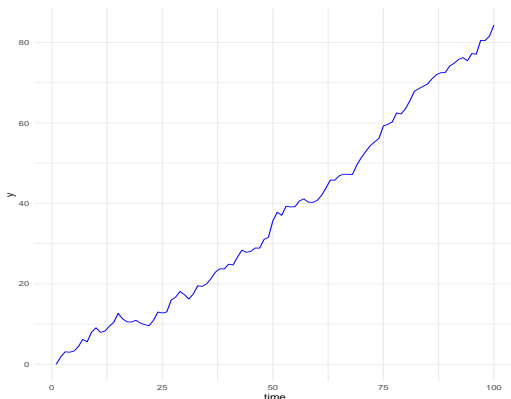
## # A tibble: 6 x 2
##   time     y
##   <int> <dbl>
## 1     1  0
## 2     2  1.80
## 3     3  3.09
## 4     4  3.00
## 5     5  3.26
## 6     6  4.38
```

Marche aléatoire avec dérive

Simulation (2)

- Le graphique de y_t en fonction du temps est donc celui d'une droite à laquelle est superposée une marche aléatoire

```
ggplot(rwd) +  
  geom_line(aes(x=time, y=y), colour="blue")
```



Modèles théoriques

Exercices

- Simulez une série temporelle $y_t, t = 1, \dots, 1000$ représentant bruit blanc gaussien suivant une loi normale $N(0, 3)$ de moyenne 0 et d'écart type 3
- Calculez la moyenne, l'écart-type et les coefficients d'autocorrélation de la série
- Construisez une série $y_t, t = 1, \dots, 1000$ représentant une marche aléatoire avec $y_0 = 1.39$ et $\sigma_\epsilon^2 = 0.41$
- Calculez la moyenne et l'écart-type pour les observations $y_0, \dots, y_{99}, y_{100}, \dots, y_{199}$, etc ... et représentez les graphiquement
- Calculez les coefficients d'autocorrélation de la série
- Réalisez un test de Portmanteau sur la série, que concluez vous ?

Section 5

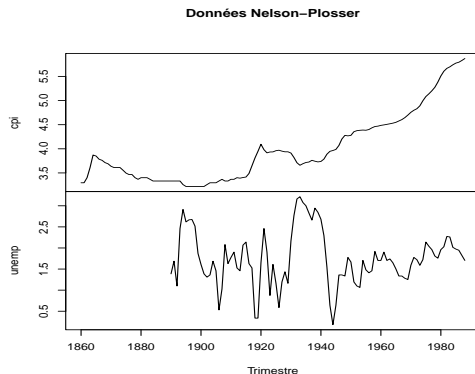
Analyse descriptive et représentations graphiques

Représentations graphiques de séries temporelles

Le chronogramme

- Chronogramme : diagramme des points (x =date, y =valeur de l'observation)
- Pour un objet de la classe `ts` on peut utiliser la méthode générique `plot`

```
plot(NelPlo[,c("cpi", "unemp")], xlab="Trimestre", main="Données Nelson-Plosser")
```

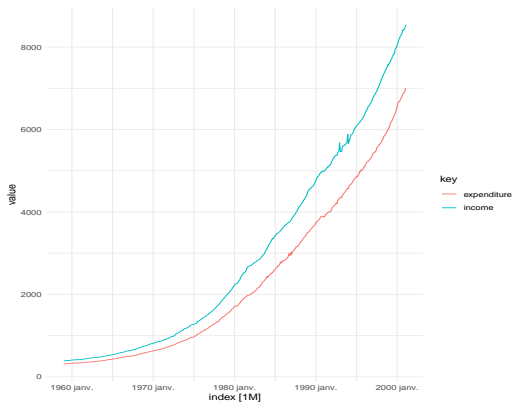


Représentations graphiques de séries temporelles

Chronogramme : Données USIncExp

- Pour un objet de la classe `tsibble`, on peut utiliser la fonction `autoplot`

```
USIncExp %>% as_tsibble() %>% autoplot()
```

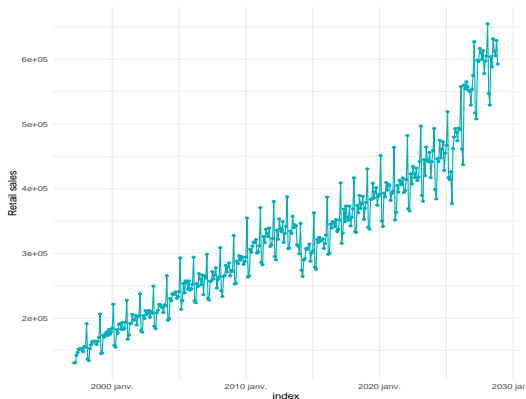


Représentations graphiques de séries temporelles

Chronogramme : Données retail

- Pour un objet de la classe `tsibble`, on peut également utiliser `ggplot` pour plus de contrôle

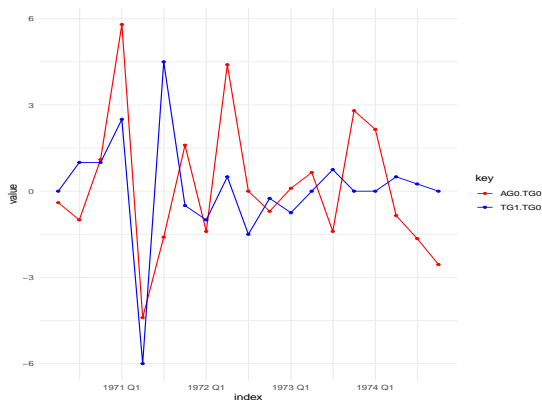
```
ggplot(retail_tsbl, aes(x = index, y = value)) +  
  geom_line(color = "#00AFBB", size = 0.5) +  
  geom_point(color = "#00AFBB", size = 1) + ylab("Retail sales")
```



Représentations graphiques de séries temporelles

Chronogramme : Données growthofmoney

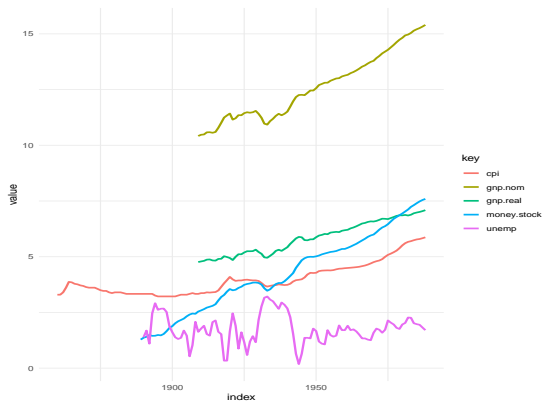
```
ggplot(gmoney_tsbl, aes(x = index, y = value)) +  
  geom_line(aes(color = key), size = 0.5) +  
  geom_point(aes(color = key), size = 1) +  
  scale_color_manual(values = c("red", "blue"))
```



Représentations graphiques de séries temporelles

Chronogramme : Données nelplo (Nelson-Plosser)

```
nelplo_tsbl %>% filter(key %in% c("gnp.nom", "gnp.real", "unemp", "cpi", "money.stock")) %>%
  ggplot(aes(x = index, y = value)) +
  geom_line(aes(color = key), size = 1)
```

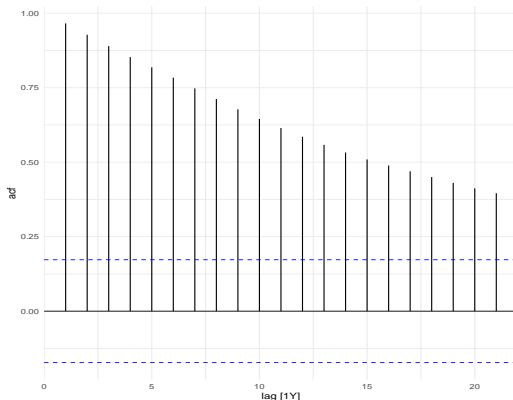


Représentations graphiques de séries temporelles

Fonction d'autocorrélation

- On peut représenter simplement la fonction d'autocorrélation obtenue avec la fonction **ACF**, ici la série **cpi** des données Nelson-Plosser

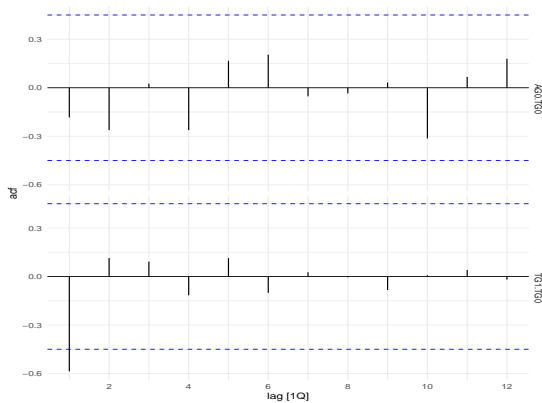
```
nelplo_tsbl %>% filter(key=="cpi") %>%  
  ACF(value) %>% autoplot()
```



Représentations graphiques de séries temporelles

Fonction d'autocorrélation : Données `growthofmoney`

```
gmoney_tsbl %>% ACF() %>% autoplot()
```

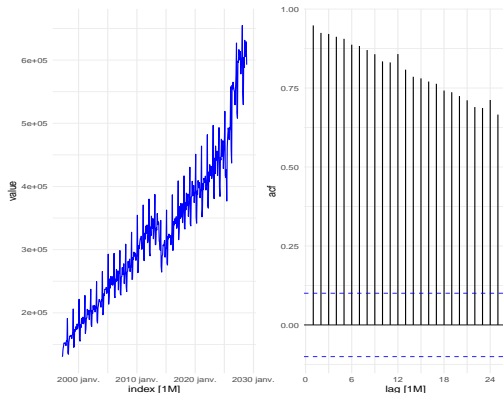


Représentations graphiques de séries temporelles

Fonction d'autocorrélation : Données retail

- Ici on utilise la librairie **patchwork** pour combiner deux graphiques

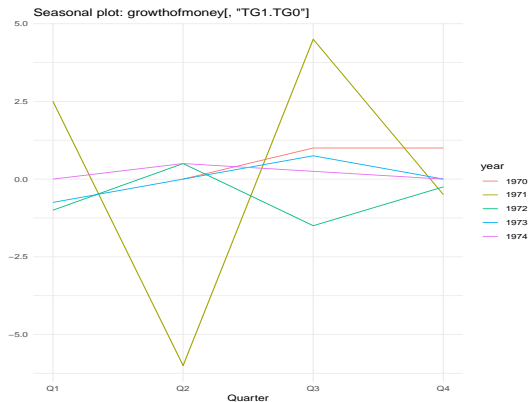
```
library(patchwork)
g1 <- retail_tsbl %>% autoplot(color='blue')
g2 <- ACF(retail_tsbl) %>% autoplot()
g1+g2
```



Représentations graphiques de séries temporelles

Season plot : Données growthofmoney

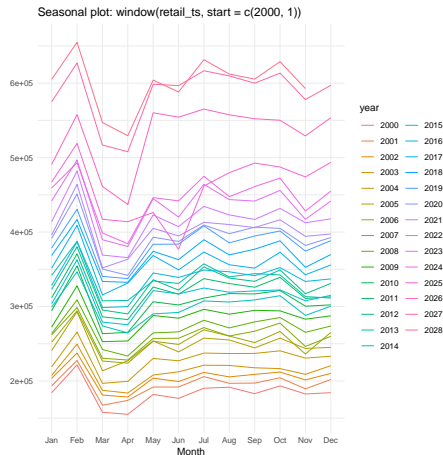
```
library(forecast)
ggseasonplot(growthofmoney[, "TG1.TG0"])
```



Représentations graphiques de séries temporelles

Season plot : Données retail

```
ggseasonplot(window(retail_ts, start=c(2000,1)))
```

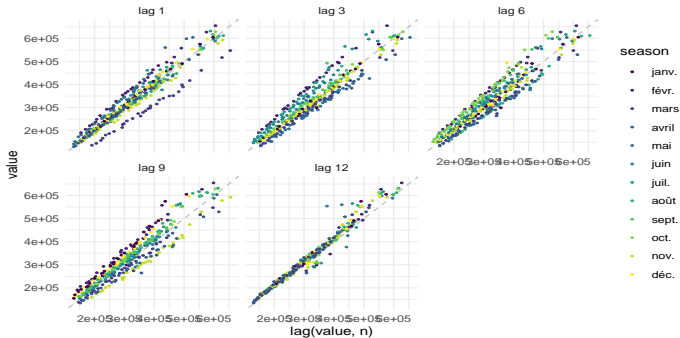


Représentations graphiques de séries temporelles

Lag plot : Données retail

- Le **lag plot** représente la série temporelle et ses valeurs précédentes

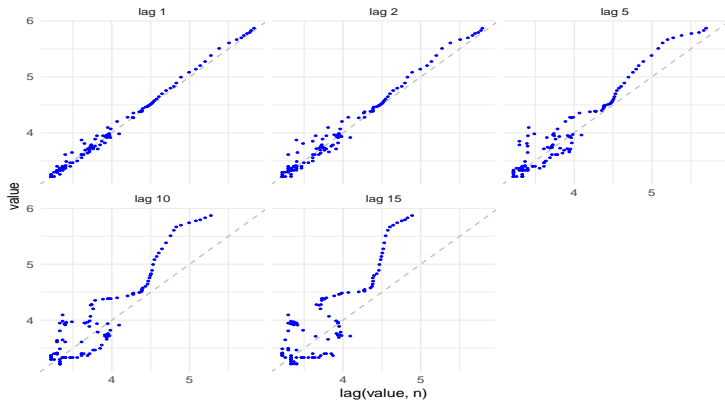
```
retail_tsbl %>% gg_lag(y=value, geom="point", size=0.5, lags=c(1,3,6,9,12))
```



Représentations graphiques de séries temporelles

Lag plot : Données Nelson-Plosser (série cpi)

```
nelplo_tsbl %>% filter(key=='cpi') %>%  
  gg_lag(y=value, geom="point", size=0.5, colour="blue", lags=c(1,2,5,10,15))
```



Exercice

- Représentez les séries des données PIB (chronogramme, fonction d'autocorrélation, lag plot)
- Représentez la série du taux de chômage (chronogramme, fonction d'autocorrélation, lag plot)

Section 6

Régression Linéaire

Le modèle de régression linéaire simple

- Modèle de régression linéaire simple (une seule variable indépendante) :

$$y_i = \beta_1 + \beta_2 \cdot x_i + \epsilon_i$$

- Les observations sont indicées par i , ($i = 1, \dots, N$)
 - y_i est la variable **expliquée** (dépendante)
 - x_i est la variable **explicative** (indépendante)
 - β_1 et β_2 sont les **paramètres** (à estimer)
 - ϵ_i est le **résidu** (écart aléatoire, erreur)
- L'équation de la droite de régression est déterminée par la pente (slope) (β_0) et l'intercept (β_1) :

$$E(y|x) = \beta_1 + \beta_2 \cdot x$$

Hypothèses de base du modèle linéaire

- On parle de Moindres Carrés Ordinaires (MCO) ou Ordinary Least Square model (OLS) car l'objectif lors de l'estimation est de minimiser la somme des erreurs au carré $\sum_i \epsilon_i^2$
- Les hypothèses de base concernent la distribution de probabilité des résidus ϵ_i :
 - Hypothèse 1 : ϵ_i suit une distribution normale $N(\mu, \sigma^2)$
 - Hypothèse 2 : l'espérance de ϵ_i est nulle : $\forall i, E(\epsilon_i) = 0$
 - Hypothèse 3 : la variance de ϵ_i est constante (homoscédasticité) : $\forall i, V(\epsilon_i) = \sigma^2$
 - Hypothèse 4 : la covariance entre deux observations est nulle : $\forall i \neq j, Cov(\epsilon_i, \epsilon_j) = 0$, il n'y a pas d'autocorrélation des résidus, ils sont sériellement indépendants

Régression linéaire simple

Données Nelson-Plosser

- On considère les données Nelson-Plosser à partir de l'année 1909 (toutes les séries complètes)
- On utilise la fonction `spread` pour passer du format "long" au format "large"

```
nelplo1909 <- nelplo_tsbl %>%
  filter(index>=1909) %>%
  spread(key = key, value=value)
head(nelplo1909)
```

```
## # A tsibble: 6 x 15 [1Y]
##   index  cpi  emp gnp.capita gnp.def gnp.nom gnp.real int.rate  ip
##   <dbl> <dbl> <dbl>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1 1909  3.33 10.5      7.16   3.37   10.4   4.76   3.77  2.30
## 2 1910  3.37 10.5      7.17   3.40   10.5   4.79   3.8   2.36
## 3 1911  3.37 10.5      7.18   3.39   10.5   4.81   3.9   2.32
## 4 1912  3.40 10.5      7.22   3.43   10.6   4.87   3.9   2.46
## 5 1913  3.39 10.6      7.21   3.44   10.6   4.88   4     2.53
## 6 1914  3.40 10.5      7.14   3.45   10.6   4.83   4.1   2.46
## # i 6 more variables: money.stock <dbl>, nom.wages <dbl>, real.wages <dbl>,
## #   stock.prices <dbl>, unemp <dbl>, vel <dbl>
```


Régression linéaire simple

Données Nelson-Plosser : Modèle

- Modèle bivarié sur les données Nelson-Plosser :
 - variable expliquée y = gnp.nom (PNB, millions de dollars)
 - variable explicative x = index (année)

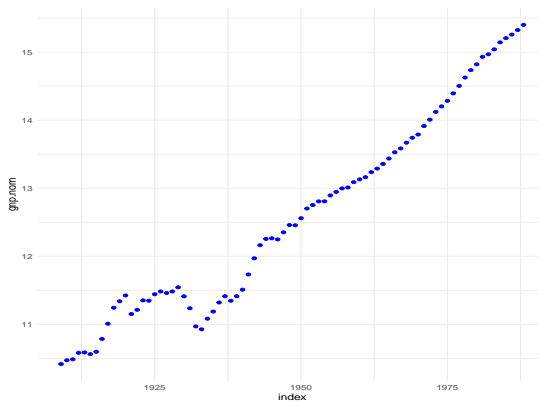
```
nelplo1909 %>% select(index, gnp.nom, emp)
```

```
## # A tsibble: 80 x 3 [1Y]
##   index gnp.nom  emp
##   <dbl> <dbl> <dbl>
## 1 1909    10.4  10.5
## 2 1910    10.5  10.5
## 3 1911    10.5  10.5
## 4 1912    10.6  10.5
## 5 1913    10.6  10.6
## 6 1914    10.6  10.5
## 7 1915    10.6  10.5
## 8 1916    10.8  10.6
## 9 1917    11.0  10.6
## 10 1918    11.2  10.7
## # i 70 more rows
```

Régression linéaire simple

Données Nelson-Plosser : Visualisation

```
nelplo1909 %>%  
  ggplot(aes(index, gnp.nom)) +  
    geom_point(colour="blue")
```



Régression linéaire simple

Données Nelson-Plosser : Estimation

- L'estimation des paramètres se fait avec la fonction `lm` (Linear Models)

```
mod1 <- lm(gnp.nom ~ index, data=nelplo1909)
summary(mod1)

##
## Call:
## lm(formula = gnp.nom ~ index, data = nelplo1909)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70453 -0.16422 -0.04624  0.26225  0.59689
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.078e+02  2.840e+00  -37.96  <2e-16 ***
## index        6.179e-02  1.458e-03   42.39  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.301 on 78 degrees of freedom
## Multiple R-squared:  0.9584, Adjusted R-squared:  0.9579
## F-statistic: 1797 on 1 and 78 DF,  p-value: < 2.2e-16
```

Régression linéaire simple

Données Nelson-Plosser : Résultats

- Un test avec $H_0 : \beta = 0$ est réalisé pour chacun des coefficients

```
summary(mod1)$coefficients
##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) -107.81083108  2.840221275  -37.95860  4.766536e-52
## index        0.06179065  0.001457543   42.39371  1.276677e-55
```

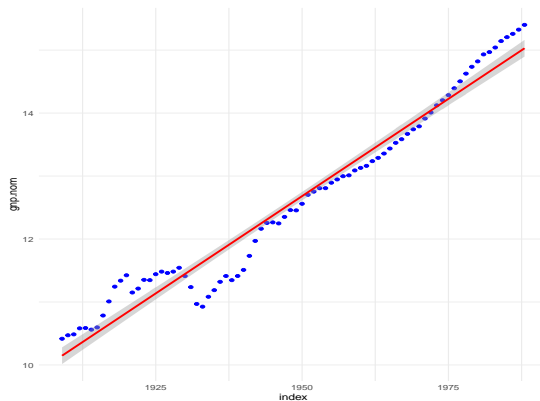
- Le coefficient associé à la variable explicative `index` (année) est statistiquement significatif (p-valeur < 0.001)
- La valeur de R^2 (variance expliquée) est élevée

```
summary(mod1)$adj.r.squared
## [1] 0.9578718
```

Régression linéaire simple

Données Nelson-Plosser : Droite de régression

```
ggplot(nelplo1909, aes(x=index, y=gnp.nom)) +  
  geom_point(colour="blue") +  
  geom_smooth(method='lm', color="red")
```



Régression linéaire simple

Données Nelson-Plosser : Valeurs observées, valeurs prédites, résidus

- Les valeurs prédites par le modèle se trouvent dans l'attribut `fitted.values`
- Les résidus représentent la différence valeur observée-valeur prédite, ils se trouvent dans l'attribut `residuals`

```
mod1_diag <- tibble(gnp=nelplo1909$gnp.nom, predicted=mod1$fitted.values,
                    residual=mod1$residuals)
```

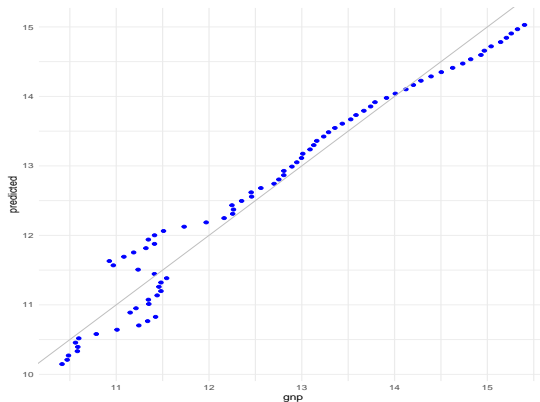
```
mod1_diag
```

```
## # A tibble: 80 x 3
##       gnp predicted residual
##   <dbl>     <dbl>    <dbl>
## 1  10.4      10.1    0.269
## 2  10.5      10.2    0.262
## 3  10.5      10.3    0.215
## 4  10.6      10.3    0.249
## 5  10.6      10.4    0.192
## 6  10.6      10.5    0.105
## 7  10.6      10.5    0.0784
## 8  10.8      10.6    0.205
## 9  11.0      10.6    0.367
## 10 11.2      10.7    0.540
## # i 70 more rows
```

Régression linéaire simple

Données Nelson-Plosser : Valeurs prédites x observées

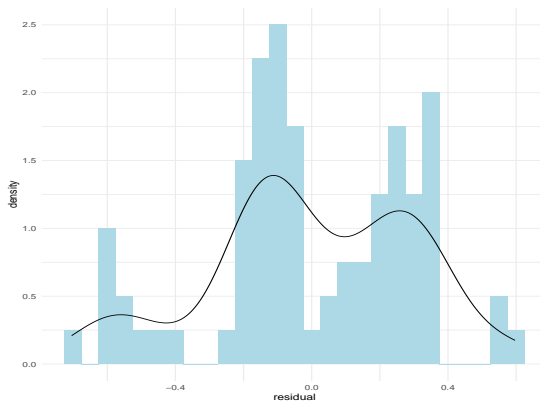
```
ggplot(mod1_diag) +  
  geom_point(aes(x=gnp, y=predicted), colour="blue") +  
  geom_abline(colour="grey")
```



Régression linéaire simple

Données Nelson-Plosser : Distribution des résidus

```
ggplot(mod1_diag, aes(x=residual)) +  
  geom_histogram(aes(y = after_stat(density)), fill="lightblue", binwidth = 0.05) +  
  geom_density(aes(x=residual))
```



Régression linéaire simple

Données Nelson-Plosser : Normalité des résidus

- Pour tester la normalité des résidus on utilise le test de **Shapiro-Wilk**
- Hypothèse H_0 : les résidus suivent une distribution normale
- La p-valeur est inférieure à 0.05, on rejette H_0 : les résidus ne sont pas distribués normalement

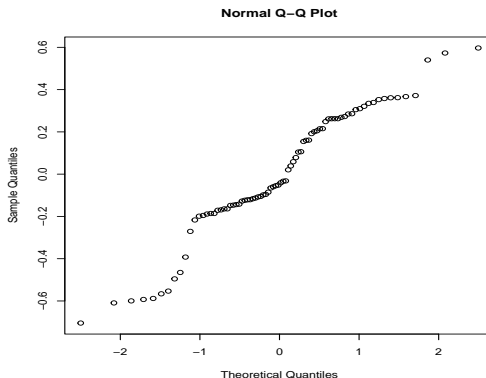
```
shapiro.test(mod1_diag$residual)
##
##  Shapiro-Wilk normality test
##
## data:  mod1_diag$residual
## W = 0.95726, p-value = 0.009177
```

Régression linéaire simple

Normalité des résidus - Quantile-Quantile plot

- Le quantile-quantile plot permet de comparer les quantiles des résidus à ceux d'une loi normale

```
qqnorm(mod1_diag$residual)
```



Régression linéaire simple

Homoscédasticité des résidus

- L'homoscédasticité des résidus est une des hypothèses fondamentales du modèle OLS/MCO
- **Homoscédasticité** = la variance des résidus est constante (sur la plage des valeurs prédites par le modèle) :

$$\text{Var}(\epsilon_i) = \sigma^2$$

- **Hétéroscédasticité** = la variance des résidus n'est pas constante :

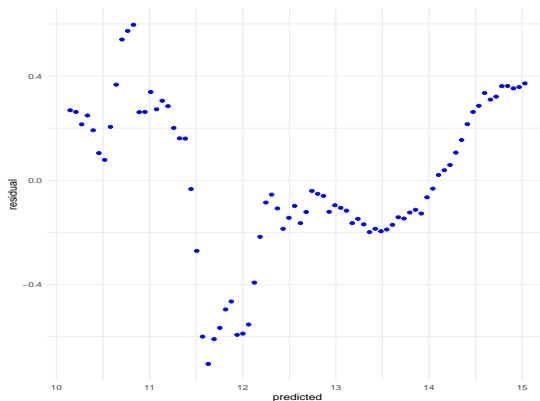
$$\text{Var}(\epsilon_i) = \sigma_i^2$$

- Si l'hypothèse d'homoscédasticité des résidus est violée, les tests d'hypothèse sur les coefficients β du modèle OLS ne sont plus valides

Régression linéaire simple

Données Nelson-Plosser : Résidus x valeurs prédites

```
ggplot(mod1_diag) +  
  geom_point(aes(x=predicted, y=residual), colour="blue")
```



Régression linéaire simple

Homoscédasticité des résidus : Le test de Breusch-Pagan-Godfrey

- Le test de **Breusch-Pagan-Godfrey** permet de tester l'homoscédasticité des résidus
- Hypothèse nulle (H_0) : homoscédasticité (les résidus ont une variance constante)
- On utilise la fonction `bptest` de la librairie R `lmtest`
- La p-valeur du test est supérieure à 0.05, on accepte H_0

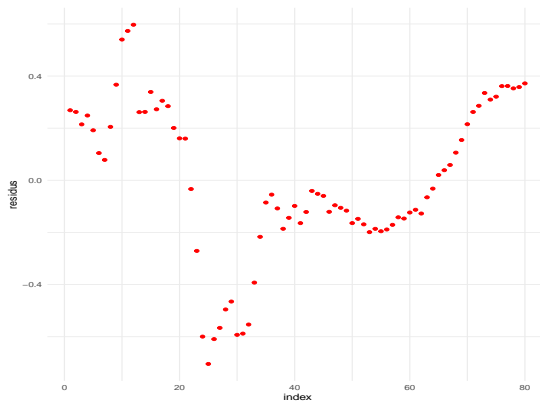
```
library(lmtest)
bptest(mod1)

##
## studentized Breusch-Pagan test
##
## data:  mod1
## BP = 5.3717, df = 1, p-value = 0.02047
```

Régression linéaire simple

Données Nelson-Plosser : Autocorrélation des résidus (1)

```
gdata <- tibble(index=1:nrow(mod1_diag), residus=mod1_diag$residual)
gdata %>% ggplot() + geom_point(aes(x=index, y=residus), color="red")
```

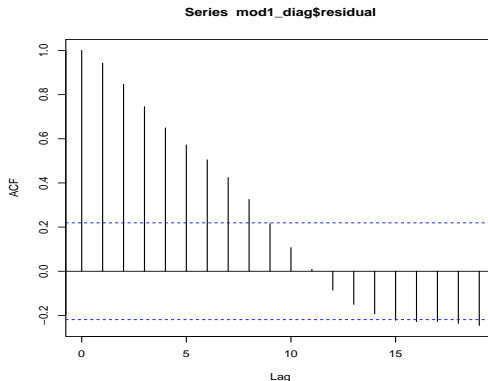


Régression linéaire simple

Données Nelson-Plosser : Autocorrélation des résidus (2)

- Un graphique **acf** permet de visualiser la fonction d'autocorrélation des résidus

```
acf(mod1_diag$residual)
```



Régression linéaire simple

Autocorrélation des résidus : Le test de Durbin-Watson

- Le test de Durbin-Watson est un test d'absence d'autocorrélation d'ordre 1 sur les résidus d'une régression linéaire
- On utilise la fonction `dwtest` de la librairie `lmtest`
- Hypothèse nulle (H_0) : pas d'autocorrélation des résidus

```
dwtest(mod1)
##
## Durbin-Watson test
##
## data: mod1
## DW = 0.085604, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

- Ici on rejette l'hypothèse H_0 , il y a autocorrélation des résidus

Régression linéaire simple

Données growthofmoney (1)

- Le modèle de régression utilisé par [2] sur les données growthofmoney est décrit par [3] :

Hetzel argues as follows : If the Federal Open Market Committee had indeed viewed the money supply as its primary intermediate target of policy, it would have provided for operating procedures that would have ensured control of the money supply. This implies a negative value for the coefficient β_2 in the regression

$$(TG_1 - TG_0) = \beta_1 + \beta_2(AG_0 - TG_0)$$

where TG_1 is the current target for the growth rate of the money supply, TG_0 is the target of the preceding period, and AG_0 is the actual growth rate of the preceding period.

Régression linéaire simple

Données growthofmoney (2)

- Les deux variables utilisées dans le modèle sont :
 - variable dépendante : TG1.TG0, difference of current and preceding target for the growth rate of the money supply
 - variable indépendante : AGO.TG0, difference of actual growth rate and target growth rate for the preceding period

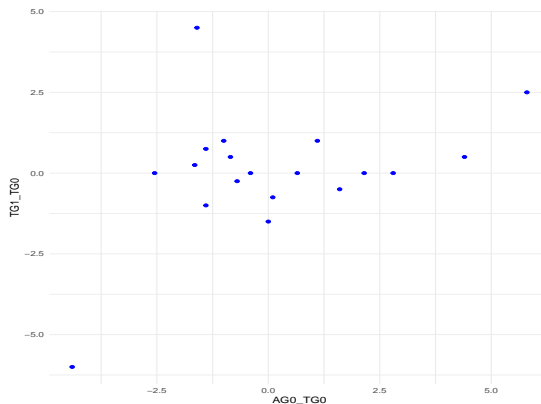
```
data("growthofmoney")
head(growthofmoney)

##           TG1.TG0 AGO.TG0
## 1970 Q2         0.0   -0.4
## 1970 Q3         1.0   -1.0
## 1970 Q4         1.0    1.1
## 1971 Q1         2.5    5.8
## 1971 Q2        -6.0   -4.4
## 1971 Q3         4.5   -1.6
```

Régression linéaire simple

Données growthofmoney (3)

```
gdata <- tibble(AGO_TGO=growthofmoney[, "AGO.TGO"], TG1_TGO=growthofmoney[, "TG1.TGO"])  
ggplot(gdata) + geom_point(aes(x=AGO_TGO, y=TG1_TGO), color="blue")
```



Régression linéaire simple

Données `growthofmoney` : Estimation

■ Estimation du modèle utilisé par Hetzel dans son article

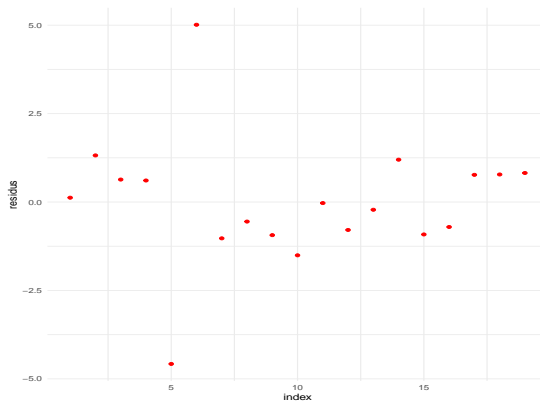
```
modelHetzel <- TG1.TGO ~ AGO.TGO
gom.mod1 <- lm(modelHetzel, data=growthofmoney)
summary(gom.mod1)

##
## Call:
## lm(formula = modelHetzel, data = growthofmoney)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.5779 -0.8534 -0.0299  0.7737  5.0125
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.007322   0.426070   0.017   0.986
## AGO.TGO      0.324858   0.179456   1.810   0.088 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.854 on 17 degrees of freedom
## Multiple R-squared:  0.1616, Adjusted R-squared:  0.1123
## F-statistic: 3.277 on 1 and 17 DF, p-value: 0.08797
```

Régression linéaire simple

Données growthofmoney : Série des résidus

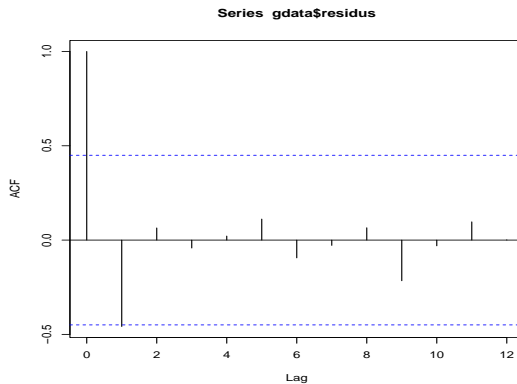
```
gdata <- tibble(index=1:nrow(growthofmoney), residus=gom.mod1$residual)  
gdata %>% ggplot() + geom_point(aes(x=index, y=residus), color="red")
```



Régression linéaire simple

Données `growthofmoney` : Autocorrélation des résidus (1)

```
acf(gdata$residus)
```



Régression linéaire simple

Données growthofmoney : Autocorrélation des résidus (2)

- La p-valeur du test de **Durbin-Watson** est largement supérieure à 0.05, on accepte H_0 , il n'y a pas d'autocorrélation des résidus

```
dwtest(modelHetzel, data=growthofmoney)

##
## Durbin-Watson test
##
## data: modelHetzel
## DW = 2.9046, p-value = 0.9839
## alternative hypothesis: true autocorrelation is greater than 0
```

Régression linéaire simple

Données `growthofmoney` : Diagnostic du modèle

- La distance de Cook (Cook's D) est utilisée pour estimer l'influence d'une observation dans un modèle de régression linéaire
- La distance de Cook peut être utilisée :
 - pour indiquer les observations pour lesquelles une vérification de la validité est nécessaire
 - pour indiquer les zones où il serait utile d'avoir plus d'e données'observations

Régression linéaire simple

Données `growthofmoney` : Distance de Cook (1)

■ Calcul des distances de Cook avec la fonction `cooks.distance`

```
dcook <- gdata <- data.frame(obs=1:nrow(growthofmoney), dcook=cooks.distance(gom.mod1))
head(dcook)

##   obs      dcook
## 1    1 0.0001356853
## 2    2 0.0187077141
## 3    3 0.0040829364
## 4    4 0.0453754681
## 5    5 1.3164884600
## 6    6 0.3504193583
```

■ On peut utiliser un seuil pour identifier les observations influentes :

- $D > 1$
- $D > 4/N$ ou $D > 4/(Nk1)$, ou N est le nombre d'observations et k le nombre de variables indépendantes

```
tcook <- 4/nrow(growthofmoney)
tcook

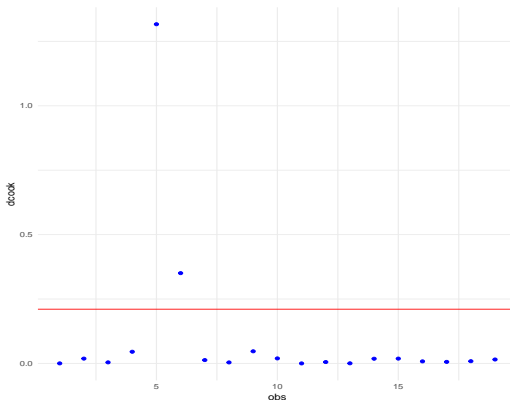
## [1] 0.2105263
```

Régression linéaire simple

Données *growthofmoney* : Distance de Cook (2)

- Les observations particulièrement influentes sont celles dont la distance de Cook est au dessus du seuil matérialisé par la ligne rouge

```
ggplot(dcook, aes(x=obs, y=dcook)) +  
  geom_point(color="blue") +  
  geom_hline(aes(yintercept=tcook), color="red")
```



Régression linéaire simple

Données `growthofmoney` : Résidus studentisés

- Les résidus studentisés sont calculés en divisant les résidus par leur écart type

```
stres <- data.frame(obs=1:nrow(growthofmoney), stres=rstudent(gom.mod1))  
head(stres)
```

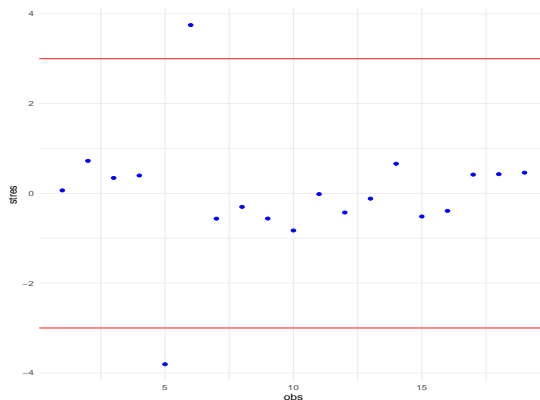
```
##   obs   stres  
## 1    1 0.0660265  
## 2    2 0.7245157  
## 3    3 0.3444020  
## 4    4 0.3977586  
## 5    5 -3.8084498  
## 6    6 3.7506138
```

- Un résidu studentisé supérieur à 3 (2) est considéré comme une valeur aberrante

Régression linéaire simple

Données growthofmoney : Résidus studentisés

```
ggplot(stres, aes(x=obs, y=stres)) +  
  geom_point(color="blue") +  
  geom_hline(aes(yintercept=3), color="red") +  
  geom_hline(aes(yintercept=-3), color="red")
```



Modèle de régression multivarié

- Modèle de régression multivarié = plusieurs variables indépendantes :

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_k \cdot x_k + \epsilon$$

Modèle de régression multivarié

Données Nelson-Plosser

- On souhaite prédire le PNB (`gnp.nom`, millions de dollars) par le taux de chômage, l'indice des prix et l'année
- On utilise la fonction `spread` pour passer du format "long" au format "large"

```
nelplo1909 <- nelplo_tsbl %>%  
  filter(index>=1909) %>%  
  spread(key = key, value=value)  
regdata <- nelplo1909 %>%  
  select(index, gnp.capita, unemp, cpi)
```

Modèle de régression multivarié

Données Nelson-Plosser : Estimation

```
mod2 <- lm(gnp.capita ~ index+unemp+cpi, data=regdata)
summary(mod2)
```

```
##
## Call:
## lm(formula = gnp.capita ~ index + unemp + cpi, data = regdata)
##
## Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.178391	-0.034936	-0.001147	0.033163	0.181541

```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-2.679e+01	1.719e+00	-15.583	<2e-16 ***
index	1.775e-02	9.505e-04	18.680	<2e-16 ***
unemp	-1.489e-01	1.231e-02	-12.095	<2e-16 ***
cpi	4.099e-02	3.198e-02	1.282	0.204

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06824 on 76 degrees of freedom
## Multiple R-squared:  0.9779, Adjusted R-squared:  0.977
## F-statistic: 1119 on 3 and 76 DF, p-value: < 2.2e-16
```

Modèle de régression multivarié

Données Nelson-Plosser : Sélection du modèle (1)

- On élimine les variables dont le coefficient n'est pas significativement différent de 0 (p-valeur > 0.05)

```
mod3 <- lm(gnp.capita ~ index+unemp, data=regdata)
summary(mod3)

##
## Call:
## lm(formula = gnp.capita ~ index + unemp, data = regdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.184686 -0.035709 -0.007246  0.032664  0.184953
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.883e+01  6.466e-01  -44.59  <2e-16 ***
## index        1.890e-02  3.322e-04   56.88  <2e-16 ***
## unemp       -1.516e-01  1.219e-02  -12.44  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06852 on 77 degrees of freedom
## Multiple R-squared:  0.9774, Adjusted R-squared:  0.9768
## F-statistic: 1663 on 2 and 77 DF,  p-value: < 2.2e-16
```


Modèle de régression multivarié

Données Nelson-Plosser : Sélection du modèle (2)

- Pour comparer des modèles imbriqués, on peut utiliser un critère d'information (AIC ou BIC)

```
AIC(mod1, mod2, mod3)
```

```
##      df      AIC  
## mod1  3  38.92556  
## mod2  5 -196.63615  
## mod3  4 -196.92493
```

- On retient le modèle ayant le plus faible AIC, ici le modèle 3

Modèle de régression linéaire

Test de changement structurel

- Soit le modèle de régression linéaire standard :

$$y_i = x_i^T \beta_i + u_i \quad (i = 1, \dots, N)$$

où y_i est l'observation de la variable dépendante et le vecteur $x_i = (1, x_{i2}, \dots, x_{ik})$ l'observation des variables indépendantes au temps i

- Les tests de changement structurel proposent de tester l'hypothèse

$$H_0 : \beta_i = \beta_0 \quad (i = 1, \dots, N)$$

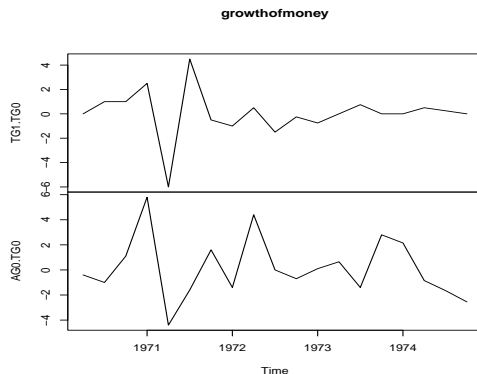
- Le test de **Chow** permet d'identifier un éventuel changement structurel dans les données au point fourni **a priori** par l'utilisateur
- Le rejet de l'hypothèse H_0 signifie qu'un meilleur ajustement peut être obtenu avec deux droites de régression (i.e. les paramètres du modèle ne sont pas stables)

Modèle de régression linéaire

Test de changement structurel : Données `growthofmoney` (1)

- On utilise la librairie `strucchange` (voir [article](#))
- On reproduit ici l'exemple de [3] : test de changement structurel au premier trimestre 1974
- Voici pour rappel le chronogramme des deux séries

```
plot(growthofmoney)
```



Modèle de régression linéaire

Test de changement structurel : Données `growthofmoney` (2)

- La formule de la régression est contenue dans l'objet `modelHetzel`

```
modelHetzel
## TG1.TGO ~ AGO.TGO
```

- Le test est réalisé avec la fonction `sctest`

```
sctest(modelHetzel, point=c(1973,4), data=growthofmoney, type="Chow")
##
## Chow test
##
## data: modelHetzel
## F = 0.37876, p-value = 0.6911
```

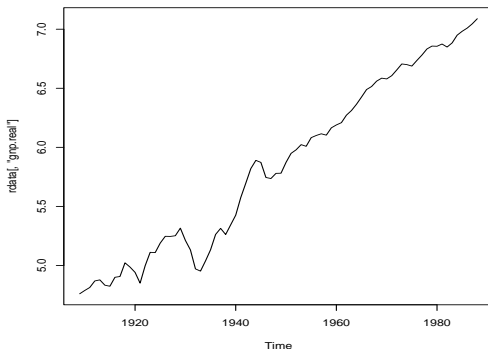
- La p-valeur est supérieure à 0.05, on accepte H_0 , il n'y a pas de changement structurel au premier trimestre 1974

Modèle de régression linéaire

Test de changement structurel : Données Nelson-Plosser (1)

- Pour la série du PNB des USA (log) on teste un changement structurel en 1933

```
rdata <- Nelson_Plosser[,c("year", "gnp.real")]  
rdata <- rdata[rdata$year>=1909,]  
rdata <- ts(rdata, start=1909)  
plot(rdata[, "gnp.real"])
```



Modèle de régression linéaire

Test de changement structurel : Données Nelson-Plosser (2)

- La p-valeur du test est inférieure à 0.01, on rejette H_0 : il y a un changement structurel en 1933

```
library(strucchange)
model <- gnp.real ~ year
sctest(model, point=c(1933,1), data=rdata, type="Chow")

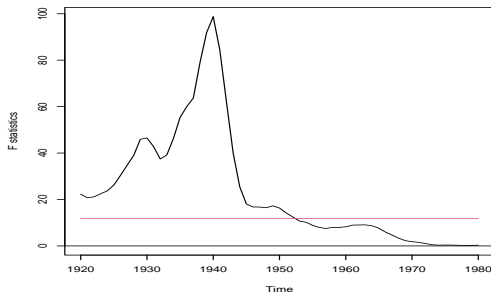
##
## Chow test
##
## data: model
## F = 19.598, p-value = 1.369e-07
```

Modèle de régression linéaire

Test de changement structurel - Extension

- La librairie **strucchange** propose une extension du test de Chow : le changement structurel est recherché sur un intervalle
- La statistique F est calculée pour chacun des points de l'intervalle
- On rejette H_0 pour des valeurs élevées de F (supérieure au seuil représenté par la ligne rouge)

```
fs <- Fstats(model, from = c(1920,1), to = c(1980,1), data = rdata)
plot(fs)
```



Modèle de régression linéaire

Exercice

- Analysez les résidus de la régression

```
mod3 <- lm(gnp.capita ~ index+unemp, data=regdata)
```

- Les résidus sont-ils normalement distribués ?
- Sont-ils autocorrélés ?
- Leur variance est-elle constante ?

Section 7

Décomposition d'une série temporelle

Composants d'une série temporelle

- Une série temporelle peut être décomposée en trois éléments :
 - Tendence
 - Saisonnalité
 - Résidus
- La méthode classique de décomposition (voir la fonction `decompose`) utilise un filtre basé sur les moyennes mobiles

Décomposition avec la librairie `feasts` (1)

- La librairie `feast` propose deux méthodes de décomposition (classique et STL)
- La décomposition classique utilise les moyennes mobiles, la saisonnalité peut être additive ou multiplicative

```
dcmp <- retail_tsbl %>%
  model(classical_decomposition(value))
components(dcmp)
```

```
## # A tibble: 381 x 7 [1M]
## # Key:   .model [1]
## # :     value = trend + seasonal + random
```

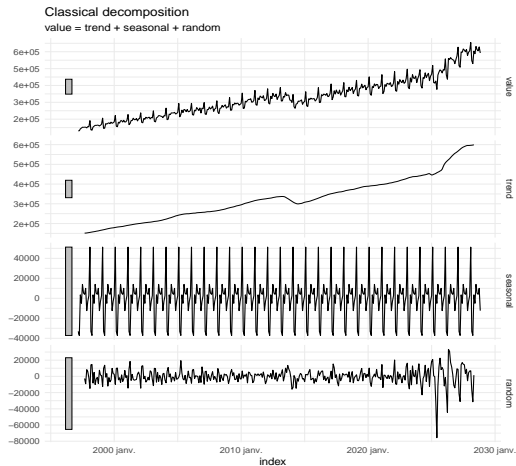
##	.model	index	value	trend	seasonal	random	season_adjust
##	<chr>	<mth>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	classical_decomposit~	1997 mars	130683	NA	-33112.	NA	163795.
## 2	classical_decomposit~	1997 avril	131244	NA	-37134.	NA	168378.
## 3	classical_decomposit~	1997 mai	142488	NA	3459.	NA	139029.
## 4	classical_decomposit~	1997 juin	147175	NA	-4965.	NA	152140.
## 5	classical_decomposit~	1997 juil.	152420	NA	13966.	NA	138454.
## 6	classical_decomposit~	1997 août	151849	NA	5222.	NA	146627.
## 7	classical_decomposit~	1997 sept.	152586	151200.	4004.	-2619.	148582.
## 8	classical_decomposit~	1997 oct.	152476	151599.	10017.	-9140.	142459.
## 9	classical_decomposit~	1997 nov.	148158	152172.	-12342.	8329.	160500.
## 10	classical_decomposit~	1997 déc.	155987	153087.	-2997.	5897.	158984.

```
## # i 371 more rows
```

Décomposition avec la librairie `feasts` (2)

- On obtient une représentation graphique de la décomposition avec la méthode générique `autoplot`

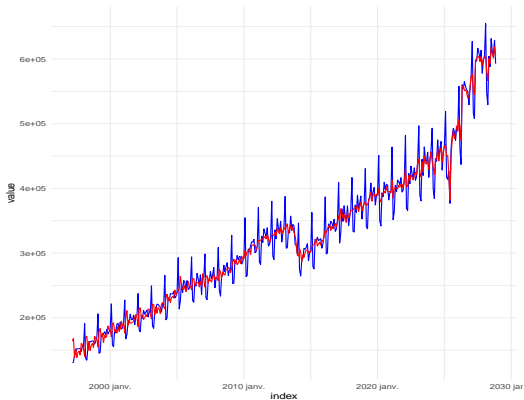
```
components(dcmp) %>% autoplot()
```



Décomposition avec la librairie `feasts` (3)

- La colonne `season_adjust` de l'objet renvoyé par `components` contient les données CVS

```
components(dcmp) %>% ggplot() +  
  geom_line(aes(x=index,y=value), colour="blue") +  
  geom_line(aes(x=index,y=season_adjust), colour="red")
```



Section 8

Transformation des données et stabilisation de la variance

Tendance linéaire

- Une série temporelle dont l'évolution est une fonction **déterministe** du temps est non-stationnaire
- Une série dont l'évolution autour d'une fonction déterministe du temps est stationnaire est dite stationnaire à une tendance près (trend stationary)
- On peut décrire une tendance linéaire par le modèle

$$y_t = \beta_0 + \beta_1 \cdot t + \epsilon_t$$

- y_t est non-stationnaire si $\beta_1 \neq 0$, la moyenne de y_t est

$$\mu_t = \beta_0 + \beta_1 \cdot t$$

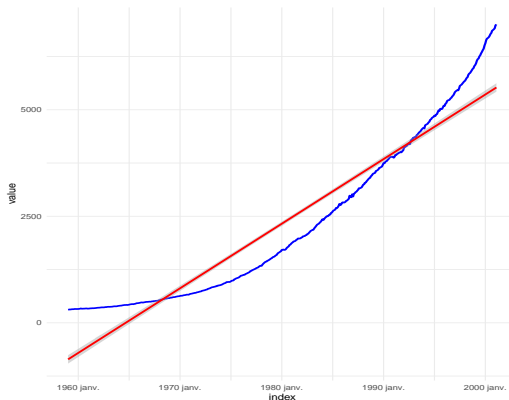
- On peut tester l'hypothèse $\beta_1 \neq 0$ (existence d'une tendance) en ajustant un modèle MCO sur les données (cette approche est valide si les erreurs ϵ_t sont un bruit blanc non corrélé)
- On peut ajuster un modèle pour estimer β_0 et β_1 puis analyser les résidus comme un processus stationnaire $\epsilon_t = y_t - \beta_0 - \beta_1 \cdot t$

Tendance linéaire

Données USIncExp

■ Données USIncExp, dépenses agrégées de consommation en millions de dollars

```
USIncExp %>% as_tsibble() %>% filter(key=='expenditure') %>%
  ggplot(aes(x = index, y = value)) +
  geom_line(color = "blue", size = 1) +
  geom_smooth(method='lm', color="red")
```



Tendance linéaire

Données USIncExp : Estimation du modèle

■ Estimation du modèle par MCO

```
regdata <- USIncExp %>% as_tsibble() %>% filter(key=='expenditure')
summary(lm(value ~ index, data=regdata))

##
## Call:
## lm(formula = value ~ index, data = regdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -688.01 -507.37  -97.57   411.41 1480.36
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.101e+02  3.277e+01  24.72  <2e-16 ***
## index        4.151e-01  5.686e-03   73.00  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 568.7 on 504 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.9134
## F-statistic: 5329 on 1 and 504 DF,  p-value: < 2.2e-16
```

- Le coefficient associé au temps (index) est significativement différent de 0 (p-valeur < 0.05)

Transformation logarithmique (1)

- On utilise souvent le logarithme (naturel) d'une série temporelle pour l'analyse
- Par exemple dans l'article original de Nelson et Plosser, toutes les séries à l'exception de la série `stock.price` (prix des actions) sont transformées en log :
The tendency of economic time series to exhibit variation that increases in mean and dispersion in proportion to absolute level motivates the transformation to natural logs and the assumption that trends are linear in the transformed data.
- L'utilisation du log d'une série :
 - diminue l'hétéroscédasticité (stabilisation de la variance) (cf. par ex. [4], p. 15)
 - transforme une tendance exponentielle en tendance linéaire

Transformation logarithmique (2)

- *Many economic time series (such as consumption, national income and expenditure, or the price level) do grow over time, but the amount by which they grow in each period also tends to rise. [5]*
- However, $\Delta x_t = x_t - x_{t-1}$ will be stationary only if the absolute amount of growth is stationary, in which case for $\mu > 0$, σ/x_t will tend to zero.
- Percentage growth, by contrast, often displays no obvious tendency to rise or fall, making it a more likely candidate for stationarity.
- Percentage growth, by contrast, often displays no obvious tendency to rise or fall, making it a more likely candidate for stationarity.
- Since the levels of many economic variables are initially positive, and recalling that

$$\Delta \log(x_t) = \log(x_t) - \log(x_{t-1}) = \log(x_t/x_{t-1})$$

we see that stationarity of the rate of growth implies stationarity of $\Delta \log(x_t)$

- Changes in the logarithms of economic data series such as those just mentioned, therefore, seem more likely to be stationary than changes in the levels.

Tendance exponentielle

- Si la tendance est exponentielle, on peut ramener à une tendance linéaire en utilisant le log :

$$\log(y_t) = \beta_0 + \beta_1 \cdot t + \epsilon_t$$

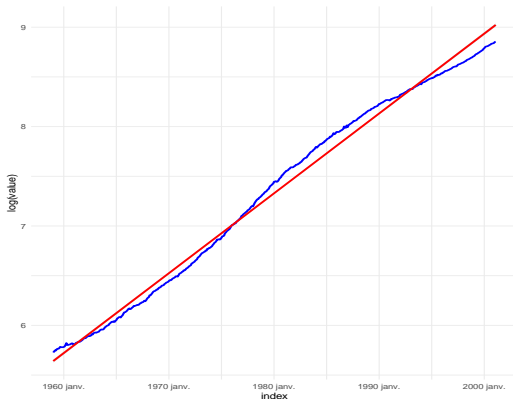
- Note : β_1 dans le modèle à tendance exponentielle est le taux de croissance annuel moyen (si t est exprimé en années)

Tendance exponentielle

Transformation logarithmique

- L'utilisation du logarithme de la série USIncExp transforme la tendance

```
USIncExp %>% as_tsibble() %>% filter(key=='expenditure') %>%
  ggplot(aes(x = index, y = log(value))) +
  geom_line(color = "blue", size = 1) +
  geom_smooth(method='lm', color="red")
```



Tendance exponentielle - Modèle linéaire

- La valeur de R^2 est nettement plus élevée que pour le modèle précédent sur la série brute

```
regdata <- USIncExp %>% as_tsibble() %>%
  filter(key=='expenditure') %>%
  mutate(value=log(value))
summary(lm(value ~ index, data=regdata))

##
## Call:
## lm(formula = value ~ index, data = regdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.16858 -0.07288 -0.01324  0.09148  0.15056
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.525e+00  5.134e-03   1271  <2e-16 ***
## index        2.200e-04  8.908e-07    247  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08909 on 504 degrees of freedom
## Multiple R-squared:  0.9918, Adjusted R-squared:  0.9918
## F-statistic: 6.099e+04 on 1 and 504 DF,  p-value: < 2.2e-16
```

Désaisonnalisation

- De nombreuses séries économiques présentent des comportements périodiques, rendant difficile la comparaison de deux instants successifs
- Cela peut être le cas particulièrement lorsque la série est trimestrielle ou mensuelle (par exemple données retail)
- Le recours à une désaisonnalisation permet d'obtenir des séries dites corrigées des variations saisonnières (CVS)

Désaisonnalisation

Régression linéaire

- On inclut la saisonnalité dans un modèle linéaire tendenciel avec des variables 'dummy' (0 ou 1) représentant les mois, trimestres, etc ...
- Par exemple pour des trimestres on ajoute $4 - 1 = 3$ variables 'dummy' (modèle avec constante) :

$$y_t = \beta_0 + \delta_1 \cdot Q1_t + \delta_2 \cdot Q2_t + \delta_3 \cdot Q3_t + \beta_1 \cdot t + \epsilon_t$$

Désaisonnalisation

Données retail (1)

- Données retail sur le commerce de détail
- On crée une variable catégorielle (factor) pour le mois
- Le temps est un index t de 1 à 381 (nombre de mois dans la série)

```
retail$mois <- factor(month(retail$DATE), labels=month(1:12, label=TRUE))
retail$t <- c(1:nrow(retail))
head(retail)
```

```
##      DATE RSXFSN mois t
## 1 1992-01-01 130683 janv 1
## 2 1992-02-01 131244 févr 2
## 3 1992-03-01 142488 mars 3
## 4 1992-04-01 147175 avril 4
## 5 1992-05-01 152420 mai 5
## 6 1992-06-01 151849 juin 6
```

- Dans la formule, on ajoute -1 pour un modèle sans constante (intercept)

```
modst <- lm(RSXFSN ~ t+ mois-1, data=retail)
```

Désaisonnalisation

Données retail (2)

```
summary(modst)
```

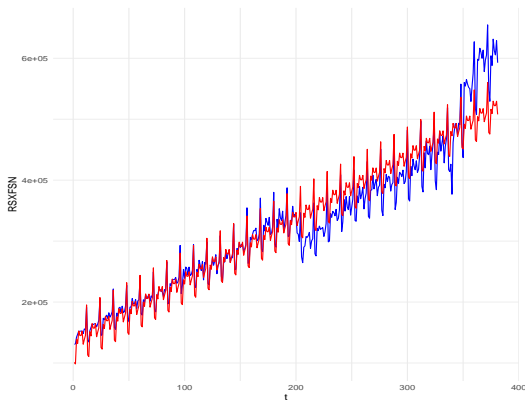
```
##
## Call:
## lm(formula = RSXFSN ~ t + mois - 1, data = retail)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -95922 -22769   753  10775 101781
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## t              1013.92      14.43   70.25 <2e-16 ***
## moisjanv      99954.45     6105.34   16.37 <2e-16 ***
## moisfévr      96272.09     6111.74   15.75 <2e-16 ***
## moismars     136132.45     6118.16   22.25 <2e-16 ***
## moisavril    128055.40     6124.61   20.91 <2e-16 ***
## moismai      147466.57     6131.09   24.05 <2e-16 ***
## moisjuin     138825.03     6137.59   22.62 <2e-16 ***
## moisjuil     137696.67     6144.12   22.41 <2e-16 ***
## moisaoût     144302.31     6150.68   23.46 <2e-16 ***
## moissept     121614.92     6157.26   19.75 <2e-16 ***
## moisoct      128425.61     6203.09   20.70 <2e-16 ***
## moisnov      134270.91     6209.48   21.62 <2e-16 ***
## moisdéc      183151.15     6215.90   29.46 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 30980 on 368 degrees of freedom
## Multiple R-squared:  0.9923, Adjusted R-squared:  0.992
## F-statistic: 3652 on 13 and 368 DF, p-value: < 2.2e-16
```

Désaisonnalisation

Données retail (3)

- Données originales (bleu) et valeurs prédites par le modèle (rouge)

```
retail$prediction <- predict.lm(modst)
ggplot(retail)+
  geom_line(mapping=aes(x=t,y=RSXFSN),color="blue")+
  geom_line(mapping=aes(x=t,y=prediction), color="red")
```



Désaisonnalisation

Données retail : Série CVS (1)

■ Création des variables indicatrices pour le mois ('dummies')

```
retail_1992_2022 <- retail %>% filter(year(DATE)<2023)
annees = nrow(retail_1992_2022)/12
t=1:annees

for (i in 1:12)
{
  su=rep(0,times=12)
  su[i]=1
  s=rep(su,times=annees)
  assign(paste("s",i,sep=""),s)
}
cbind(retail_1992_2022[, "RSXFSN"], s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12)[1:12,]
```

##		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
##	[1,]	130683	1	0	0	0	0	0	0	0	0	0	0
##	[2,]	131244	0	1	0	0	0	0	0	0	0	0	0
##	[3,]	142488	0	0	1	0	0	0	0	0	0	0	0
##	[4,]	147175	0	0	0	1	0	0	0	0	0	0	0
##	[5,]	152420	0	0	0	0	1	0	0	0	0	0	0
##	[6,]	151849	0	0	0	0	0	1	0	0	0	0	0
##	[7,]	152586	0	0	0	0	0	0	1	0	0	0	0
##	[8,]	152476	0	0	0	0	0	0	0	1	0	0	0
##	[9,]	148158	0	0	0	0	0	0	0	0	1	0	0
##	[10,]	155987	0	0	0	0	0	0	0	0	0	1	0
##	[11,]	154824	0	0	0	0	0	0	0	0	0	0	1
##	[12,]	191347	0	0	0	0	0	0	0	0	0	0	1

Désaisonnalisation

Données retail : Série CVS (1)

- Pour obtenir les données CVS on extrait les coefficients

```
coefst <- modst$coefficients
coefst
```

	t	moisjanv	moisfévr	moismars	moisavril	moismai	moisjuin
##	1013.922	99954.447	96272.087	136132.447	128055.400	147466.572	138825.026
##	moisjuil	moisaoût	moissept	moisoct	moisnov	moisdéc	
##	137696.666	144302.307	121614.917	128425.611	134270.915	183151.155	

- On soustrait ensuite les coefficients saisonniers à la série originale

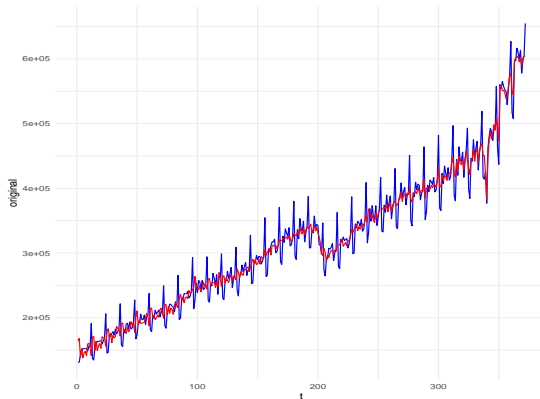
```
a <- mean(coefst[2:13])
b <- coefst[1]
c <- coefst[2:13]-mean(coefst[2:13])
y_cvs <- retail_1992_2022$RSXFSN-(c[1]*s1+c[2]*s2+c[3]*s3+c[4]*s4+c[5]*s5+c[6]*s6+
c[7]*s7+c[8]*s8+c[9]*s9+c[10]*s10+c[11]*s11+c[12]*s12)
```

Désaisonnalisation

Données retail : Série CVS (2)

```
gdata <- tibble(t=retail_1992_2022$t,  
                original=retail_1992_2022$RSXFSN, cvs = y_cvs)
```

```
ggplot(gdata)+  
  geom_line(mapping=aes(x=t,y=original),color="blue")+  
  geom_line(mapping=aes(x=t,y=cvs), color="red")
```



Désaisonnalisation

La fonction `tslm` (1)

- On peut également utiliser la fonction `tslm` de la librairie `forecast`
- Cette fonction ajuste un modèle linéaire incluant la saisonnalité et la tendance (et éventuellement la tendance au carré)
- Modèle linéaire avec saisonnalité et tendance - données 'retail'

```
library(forecast)
bhat = tslm(retail_ts~trend+I(trend^2)+season)
```

Désaisonnalisation

La fonction `tslm` (2)

```
summary(bhat)

##
## Call:
## tslm(formula = retail_ts ~ trend + I(trend^2) + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -111154  -19147   -5324   17916   72473
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.726e+05  6.052e+03  28.517 < 2e-16 ***
## trend        4.276e+02  4.859e+01   8.799 < 2e-16 ***
## I(trend^2)    1.535e+00  1.232e-01  12.459 < 2e-16 ***
## season2       4.888e+04  6.605e+03   7.400 9.38e-13 ***
## season3      -3.550e+04  6.554e+03  -5.416 1.10e-07 ***
## season4      -3.917e+04  6.554e+03  -5.977 5.39e-09 ***
## season5       6.950e+02  6.554e+03   0.106  0.9156
## season6      -7.377e+03  6.554e+03  -1.126  0.2611
## season7       1.204e+04  6.554e+03   1.836  0.0671 .
## season8       3.392e+03  6.554e+03   0.518  0.6051
## season9       2.259e+03  6.554e+03   0.345  0.7305
## season10      8.857e+03  6.554e+03   1.351  0.1774
## season11     -1.384e+04  6.554e+03  -2.112  0.0354 *
## season12     -5.847e+03  6.605e+03  -0.885  0.3766
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26010 on 367 degrees of freedom
## Multiple R-squared:  0.953, Adjusted R-squared:  0.9514
## F-statistic: 572.9 on 13 and 367 DF,  p-value: < 2.2e-16
```


Section 9

Séries temporelles non-stationnaires

Stationarité des séries macroéconomiques

- De nombreuses séries temporelles macroéconomiques ne sont pas stationnaires
- Il existe différents types de non-stationnarité
- C'est l'objet notamment de l'article de [1], qui arguent que les séries temporelles macroéconomiques sont plus souvent caractérisées par une non-stationnarité de type stochastique (unit-root nonstationarity) que par une tendance déterministe [6, p. 164]

Tendance déterministe et tendance stochastique

- Différence entre un processus stationnaire à une tendance près (trend stationary) et un processus stationnaire par différence :

$$y_t = y_{t-1} + \mu = y_0 + \mu t \quad (1)$$

$$y_t = y_{t-1} + \epsilon_t = y_0 + \sum_{s=1}^t \epsilon_s \quad (2)$$

où μ est une constante et ϵ_t un bruit blanc

- Dans l'équation (1), y_t est représenté par une tendance **déterministe**
- Dans l'équation (2) la série est expliquée par ses chocs passés, i.e. une tendance **stochastique**

Tendance déterministe et tendance stochastique

Tendance déterministe : Simulation (1)

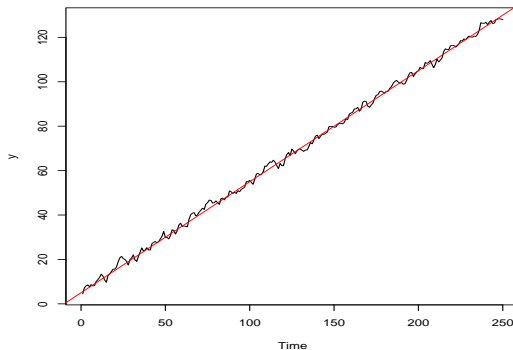
- Simulation d'une série présentant une **tendance déterministe** (trend stationary time series, exemple tiré de [4])
- La série est la combinaison d'une fonction déterministe du temps et d'un processus stationnaire (auto-régressif)

```
set.seed(12345)
y.tsar2 <- 5 + 0.5 * seq(250) +
  arima.sim(list(ar = c(0.8, -0.2)), n = 250)
```

Tendance déterministe et tendance stochastique

Tendance déterministe : Simulation (2)

```
plot(y.tsar2, ylab="y", xlab = "Time")  
abline(a=5, b=0.5, col = "red")
```



Tendance déterministe et tendance stochastique

Tendance stochastique : Simulation (1)

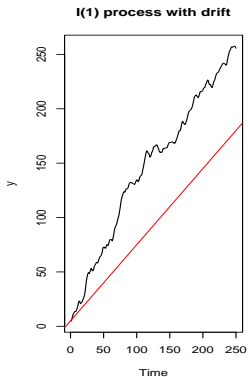
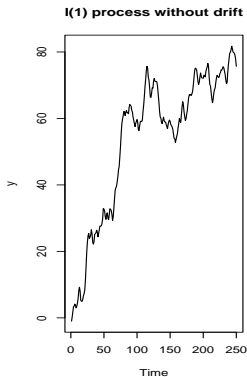
- Simulation d'une série présentant une **tendance stochastique** (difference stationary time series, exemple tiré [4])
- La série `y1` est l'accumulation d'erreurs passées
- La série `y1.d` inclu une **dérive**

```
set.seed(12345)
u.ar2 <- arima.sim(list(ar = c(0.8, -0.2)), n = 250)
y1 <- cumsum(u.ar2)
TD <- 5.0 + 0.7 * seq(250)
y1.d <- y1 + TD
```

Tendance déterministe et tendance stochastique

Tendance stochastique : Simulation (2)

```
layout(matrix(1:2, nrow = 1, ncol = 2))
plot.ts(y1, main = "I(1) process without drift", ylab="y", xlab = "Time")
plot.ts(y1.d, main = "I(1) process with drift", ylab="y", xlab = "Time")
abline(a=5, b=0.7, col = "red")
```



Stationnarisation

- Une série présentant une tendance et/ou une saisonnalité ne pourra pas être modélisée par un processus stationnaire
- Soit le processus $(X_t)_{t \in \mathbb{Z}}$ vérifiant :

$$X_t = at + b + \epsilon_t$$

avec $a \neq 0$

- On a :

$$\mathbf{E}(X_t) = \mathbf{E}(at + b + \epsilon_t) = at + b$$

et $\text{Cov}(X_t, X_{t-h}) = \sigma^2$ si $h = 0$ et $\text{Cov}(X_t, X_{t-h}) = 0$ si $h \neq 0$

- Le processus $(X_t)_{t \in \mathbb{Z}}$ n'est pas stationnaire car $E(X_t)$ dépend de t

Stationarisation par différenciation (1)

- Si on considère maintenant le processus $Y_t = X_t - X_{t-1}$, on a :

$$\begin{aligned} Y_t &= X_t - X_{t-1} \\ Y_t &= at + b + \epsilon_t - a(t-1) - b + \epsilon_{t-1} \\ &= a + \epsilon_t - \epsilon_{t-1} \end{aligned}$$

et $\mathbf{E}(Y_t) = a$

- La covariance $\text{Cov}(Y_t, Y_{t-h})$ est également constante dans le temps
- On obtient un processus stationnaire Y_t par **différenciation** de X_t
- Une série est dite **intégrée** d'ordre d , notée $I(d)$, s'il faut la différencier d fois pour obtenir une série stationnaire

Stationarisation par différenciation

Données Nelson-Plosser, indice des prix à la consommation (1)

- La fonction `diff` permet d'obtenir la série intégrée, ici la série `cpi` des données Nelson-Plosser
- La série intégrée débute au temps $t + 1$

```
cpi_diff <- diff(NelPlo[, "cpi"], 1)
window(cpi_diff, start=1861, end=1900)

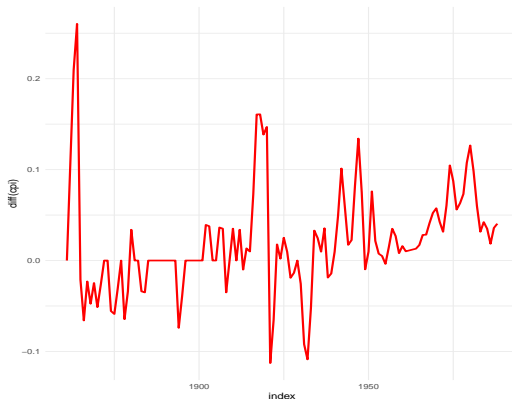
## Time Series:
## Start = 1861
## End = 1900
## Frequency = 1
## [1] 0.0000000 0.1053605 0.2097205 0.2602831 -0.0210534 -0.0659580
## [7] -0.0229895 -0.0476280 -0.0246926 -0.0512933 -0.0266683 0.0000000
## [13] 0.0000000 -0.0555698 -0.0588405 -0.0307717 0.0000000 -0.0645385
## [19] -0.0339016 0.0339016 0.0000000 0.0000000 -0.0339016 -0.0350913
## [25] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## [31] 0.0000000 0.0000000 0.0000000 -0.0741080 -0.0392207 0.0000000
## [37] 0.0000000 0.0000000 0.0000000 0.0000000
```

Stationnarisation par différenciation

Données Nelson-Plosser, indice des prix à la consommation (2)

- Pour représenter la série on la transforme en objet `tsibble`

```
api_diff %>% as_tsibble() %>%  
  ggplot(aes(x = index, y = value)) +  
  geom_line(colour="red", size = 1) + ylab("diff(cpi)")
```



Test de non-stationarité

Deux cas de non-stationarité

- Avant de réaliser un test de non-stationnarité, il faut examiner le **chronogramme** de la série pour voir si la série présente une **tendance**
 - soit stochastique (marche aléatoire avec dérive),
 - soit déterministe (série stationnaire à un trend déterministe près)
- La série y_t est stationnaire autour d'une tendance déterministe (pas de racine unitaire) : on peut éliminer la tendance de la série originale y_t et ajuster un modèle $ARMA(p, q)$ sur les résidus
- La série y_t est intégrée (d'ordre d) : la série est stationnaire par différence

Test de non-stationarité

Le test ADF (Augmented Dickey-Fuller)

- Les tests de racine unitaire (unit-root tests) permet de tester la non-stationarité d'une série
- Le test ADF (Augmented Dickey-Fuller) est le plus couramment utilisé
- Il est basé sur la régression (cf. [4])

$$y_t = \beta' D_t + \phi y_{t-1} + \sum_{j=1}^p \psi \Delta y_{t-j} + u_t \quad (1)$$

$$\Delta y_t = \beta' D_t + \pi y_{t-1} + \sum_{j=1}^p \psi \Delta y_{t-j} + u_t \quad (2)$$

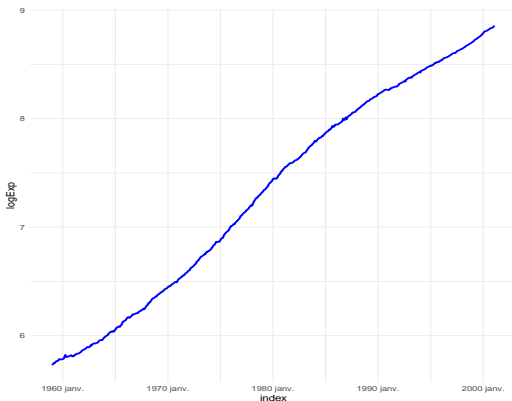
avec $\pi = \phi - 1$

Test de non-stationarité

Dépenses de consommation aux USA

- La série du log des dépenses de consommation aux USA présente une tendance

```
USExp <- USIncExp %>% as_tsibble() %>% filter(key=='expenditure') %>%
  mutate(logExp=log(value))
ggplot(USExp, aes(x = index, y = logExp)) +
  geom_line(color = "blue", size = 1)
```



Test de non-stationarité

Test ADF

- On utilise la fonction `ur.df` (bibliothèque `urca`)
- Nous allons tester les hypothèses :
 - Hypothèse nulle H_0 : la série est **non stationnaire avec dérive**
 - Hypothèse alternative H_1 : la série est **stationnaire avec trend déterministe**
- Le test considère un modèle autorégressif d'ordre p (p représenté par le lag est inconnu)
- On choisit pour commencer une valeur de p élevée et on retient la première valeur du lag fortement significative

Test de non-stationarité

Test ADF : Série USExp (1)

- On choisit dans un premier temps $p=12$
- La fonction `attr` permet d'accéder aux éléments de l'objet `exp.df` contenant le résultat du test (`attributes(exp.df)` renvoie la liste des éléments)

```
library(urca)
exp.df <- ur.df(y=as.data.frame(USExp)[,"logExp"], lags=12, type='trend')
attr(exp.df, "testreg")

##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.0208491 -0.0030259 -0.0000296  0.0034857  0.0233391
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.130e-02  1.613e-02   0.701  0.483735
## z.lag.1      -1.339e-03  2.901e-03  -0.461  0.644675
## tt           7.074e-06  1.965e-05   0.360  0.719044
## z.diff.lag1  -2.085e-01  4.567e-02 -4.565  6.37e-06 ***
## z.diff.lag2  -4.538e-02  4.627e-02  -0.981  0.327235
## z.diff.lag3  -1.030e-02  4.614e-02  -0.223  0.823428
## z.diff.lag4  -2.322e-03  4.589e-02  -0.051  0.959664
## z.diff.lag5   6.135e-02  4.562e-02   1.345  0.179309
## z.diff.lag6   1.505e-01  4.517e-02   3.332  0.000931 ***
## z.diff.lag7   1.621e-01  4.515e-02   3.590  0.000365 ***
## z.diff.lag8   1.202e-01  4.569e-02   2.631  0.008776 **
## z.diff.lag9   4.947e-02  4.593e-02   1.077  0.281906
## z.diff.lag10  3.670e-02  4.598e-02   0.798  0.425159
## z.diff.lag11  1.181e-01  4.592e-02   2.573  0.010389 *
## z.diff.lag12  4.532e-02  4.521e-02   1.002  0.316631
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


Test de non-stationarité

Test ADF : Série USExp (2)

- La première valeur p fortement significative est 8, on refait le test avec $p=8$

```
library(urca)
exp.df <- ur.df(y=as.data.frame(USExp)[,"logExp"], lags=8, type='trend')
```

- Valeur de la statistique du test

```
attr(exp.df, "teststat")

##              tau3      phi2      phi3
## statistic -0.4287925  7.896003  0.3583672
```

- Valeurs critiques

```
attr(exp.df, "cval")

##      1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

- La valeur de tau3 est supérieure au seuil de 10%, on retient l'hypothèse H_0 , la série ne présente pas de tendance déterministe et peut être rendue stationnaire par différenciation

Test de non-stationarité

Test ADF : Série USExp (2)

- Le modèle de régression utilisé pour le test est contenu dans l'attribut `testreg`

```
attr(exp.df, "testreg")

##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.0205542 -0.0030498 -0.0001604  0.0036908  0.0222845
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.117e-02  1.609e-02   0.694 0.487809
## z.lag.1      -1.236e-03  2.883e-03  -0.429 0.668264
## tt           7.040e-06  1.949e-05   0.361 0.718134
## z.diff.lag1  -1.886e-01  4.512e-02  -4.180 3.46e-05 ***
## z.diff.lag2  -2.393e-02  4.538e-02  -0.527 0.598191
## z.diff.lag3   2.050e-02  4.467e-02   0.459 0.646565
## z.diff.lag4   2.866e-02  4.455e-02   0.643 0.520301
## z.diff.lag5   8.980e-02  4.455e-02   2.016 0.044395 *
## z.diff.lag6   1.715e-01  4.466e-02   3.841 0.000139 ***
## z.diff.lag7   1.632e-01  4.530e-02   3.604 0.000346 ***
## z.diff.lag8   1.080e-01  4.500e-02   2.399 0.016807 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005581 on 486 degrees of freedom
## Multiple R-squared:  0.08049, Adjusted R-squared:  0.06157
## F-statistic: 4.254 on 10 and 486 DF, p-value: 1.076e-05
```

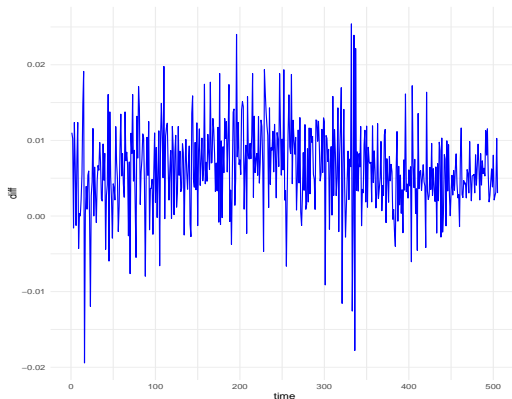
- On peut obtenir plus de détails avec la méthode générique `summary`

Série différenciée

Dépenses de consommation aux USA (série USExp)

- On utilise la fonction `diff` pour obtenir la série intégrée

```
USExp.diff <- tibble(time=1:(nrow(USExp)-1), diff= diff(as.data.frame(USExp)[, "logExp"]))  
ggplot(USExp.diff, aes(x = time, y = diff)) +  
  geom_line(color = "blue", size = 0.5)
```



Test de non-stationarité

Dépenses de consommation aux USA (série USExp) : Série différenciée

- On réalise un test ADF sur la série différenciée
 - Hypothèse H_0 : la série présente une tendance stochastique (racine unitaire)
 - Hypothèse H_1 : la série est stationnaire

```
USExp.diff.df <- ur.df(y=as.data.frame(USExp.diff)[, "diff"], lags=1, type='drift')
```

- Valeur de la statistique du test

```
attr(USExp.diff.df, "teststat")
##                tau2      phi1
## statistic -16.8418 141.8248
```

- Valeurs critiques

```
attr(USExp.diff.df, "cval")
##      1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

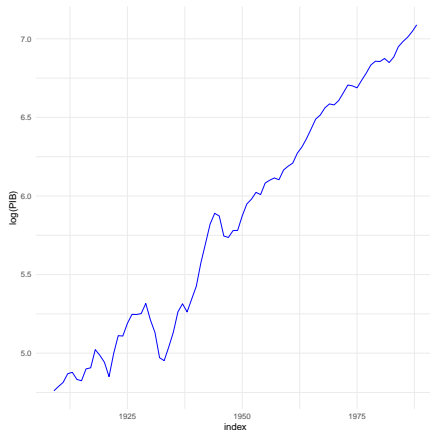
- La valeur de tau2 est inférieure à la valeur critique à 1% donc on rejete H_0 et on conclut à la stationarité de la série

Test de non-stationarité

Test ADF : Données `nelplo`, série `gnp.real` (1)

- La série du log du PIB des USA en milliards de dollars 1958 présente une tendance

```
nelplo_tsbl %>% filter(key=="gnp.real" & index>=1909) %>%  
  ggplot(aes(x = index, y = value)) +  
    geom_line(color='blue') + ylab("log(PIB)")
```



Test de non-stationarité

Test ADF : Données `nelplo`, série `gnp.real` (2)

- On réalise un test ADF (on choisit dans un premier temps $p=12$) :
 - Hypothèse nulle H_0 : la série est **non stationnaire avec dérive**
 - Hypothèse alternative H_1 : la série est **stationnaire avec trend déterministe**
- La première valeur p fortement significative est 1, on refait le test avec $p=1$

```
gnpreal.df <- ur.df(y=as.data.frame(nelplo1909)[,"gnp.real"], lags=1, type='trend')
```

- Valeur de la statistique du test

```
attr(gnpreal.df, "teststat")
##          tau3      phi2    phi3
## statistic -3.454521 7.017575 6.0513
```

- Valeurs critiques

```
attr(gnpreal.df, "cval")
##      1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

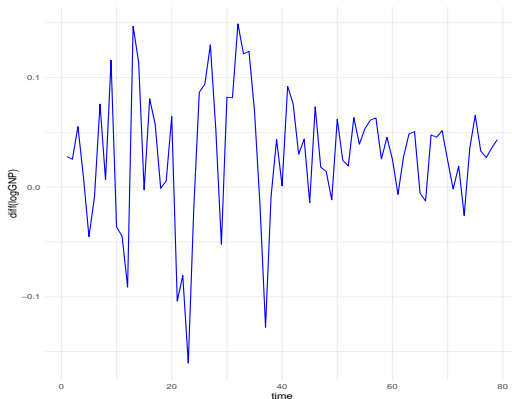
- La valeur de `tau3` est supérieure au seuil de 5%, on retient l'hypothèse H_0 , la série ne présente pas de tendance déterministe et peut être rendue stationnaire par différenciation

Série différenciée

Données `nelplo`, série `gnp.real`

- On utilise la fonction `diff` pour obtenir la série intégrée

```
gnpreal.diff <- tibble(time=1:(nrow(nelplo1909)-1), diff= diff(as.data.frame(nelplo1909)$gnp.real))
ggplot(gnpreal.diff, aes(x = time, y = diff)) +
  geom_line(color = "blue", size = 0.5) + ylab("diff(logGNP)")
```



Test de non-stationarité

Données `nelplo`, série différenciée `gnp.real`

- On réalise un test ADF sur la série différenciée
 - Hypothèse H_0 : la série présente une tendance stochastique (racine unitaire)
 - Hypothèse H_1 : la série est stationnaire

```
gnpreal.diff.df <- ur.df(y=as.data.frame(gnpreal.diff)[, "diff"], lags=1, type='drift')
```

- Valeur de la statistique du test

```
attr(gnpreal.diff.df, "teststat")
##          tau2      phi1
## statistic -5.471812 14.9712
```

- Valeurs critiques

```
attr(gnpreal.diff.df, "cval")
##      1pct  5pct 10pct
## tau2 -3.51 -2.89 -2.58
## phi1  6.70  4.71  3.86
```

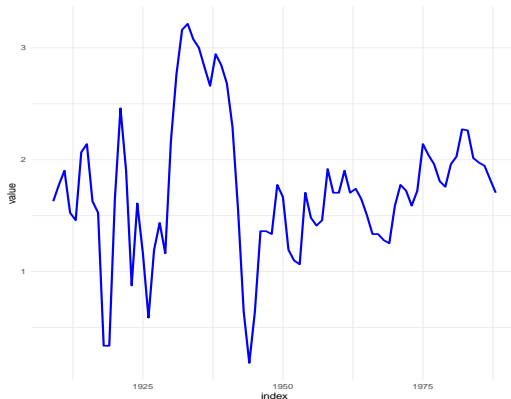
- La valeur de `tau2` est inférieure à la valeur critique à 1% donc on rejete H_0 et on conclut à la stationnarité de la série

Test de non-stationarité

Taux de chômage aux USA

- La série du taux de chômage aux USA ne présente pas de tendance

```
Unemp <- nelplo_tsbl %>% filter(key=='unemp' & index >= 1909)
ggplot(Unemp, aes(x = index, y = value)) +
  geom_line(color = "blue", size = 1)
```



Test de non-stationarité

Taux de chômage aux USA : Test ADF (1)

- Après un premier test avec `lags=6`, le premier lag significatif est 1

```
unemp.df <- ur.df(y=as.data.frame(Unemp)[,"value"], lags=1, type='drift')
```

- Valeur de la statistique du test

```
attr(unemp.df, "teststat")
##              tau2      phi1
## statistic -3.909361 7.642016
```

- Valeurs critiques

```
attr(unemp.df, "cval")
##      1pct  5pct 10pct
## tau2 -3.51 -2.89 -2.58
## phi1  6.70  4.71  3.86
```

- La statistique tau2 est inférieure à la valeur critique à 1% donc on rejete H_0 et on conclut à la stationarité de la série

Test de non-stationarité

Taux de chômage aux USA : Test ADF (2)

- On peut également utiliser la fonction `adf.test` de la librairie `tseries`
- H_0 : la série a une racine unitaire (unit root)
- La p-valeur est inférieure au seuil critique à 5%, on rejette H_0 : la série est stationnaire

```
adf.test(as.data.frame(Unemp)[,"value"], k=1)

##
## Augmented Dickey-Fuller Test
##
## data: as.data.frame(Unemp)[, "value"]
## Dickey-Fuller = -3.8903, Lag order = 1, p-value = 0.01899
## alternative hypothesis: stationary
```

Test de stationarité

Le test KPSS

- Le test KPSS proposé par Kwiatkowski est disponible dans les librairies **urca** et **tseries**
- Hypothèse nulle H_0 : la série est stationnaire (soit à une tendance près, soit à une moyenne non nulle près)
- Le test suppose que la série est la somme d'une marche aléatoire R_t , d'un trend déterministe et d'une erreur stationnaire U_t :

$$y_t = R_t + \beta_1 + \beta_2 t + U_t$$

où $R_t = R_{t-1} + z_t$

- Pour tester que la série y_t est stationnaire à une tendance près, l'hypothèse nulle est $\sigma_z^2 = 0$

Test de stationarité

Le test KPSS : Série USExp (1)

- On teste ici l'hypothèse H_0 : la série des dépenses de consommation aux USA est stationnaire à une tendance près (option `type="tau"`)

```
ktest <- ur.kpss(USExp$logExp, type="tau", lags="long")
summary(ktest)

##
## #####
## # KPSS Unit Root Test #
## #####
##
## Test is of type: tau with 17 lags.
##
## Value of test-statistic is: 0.488
##
## Critical value for a significance level of:
##          10pct  5pct 2.5pct  1pct
## critical values 0.119 0.146  0.176 0.216
```

- La valeur de la statistique est nettement supérieure à la valeur critique au seuil de 1%, on rejette H_0 , la série ne présente pas de tendance linéaire

Test de stationarité

Test KPSS : Série USExp (2)

- On teste maintenant l'hypothèse H_0 : la série des dépenses de consommation aux USA est stationnaire de moyenne constante (option `type="mu"`)

```
ktest <- ur.kpss(USExp$logExp, type="mu", lags="long")
summary(ktest)

##
## #####
## # KPSS Unit Root Test #
## #####
##
## Test is of type: mu with 17 lags.
##
## Value of test-statistic is: 2.9218
##
## Critical value for a significance level of:
##          10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

- La valeur de la statistique est nettement supérieure à la valeur critique au seuil de 1%, on rejette H_0 , la série n'est pas stationnaire

Test de stationarité

Test KPSS : Série USExp intégrée

- On teste maintenant la stationarité de la série intégrée

```
ktest <- ur.kpss(USExp.diff$diff, type="mu", lags="long")
summary(ktest)

##
## #####
## # KPSS Unit Root Test #
## #####
##
## Test is of type: mu with 17 lags.
##
## Value of test-statistic is: 0.5706
##
## Critical value for a significance level of:
##          10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463 0.574 0.739
```

- La valeur de la statistique se situe entre les valeurs critiques à 2.5% et 5%, on ne peut pas conclure à la stationarité de la série différenciée

Séries temporelles non-stationnaires

Exercices

- 1 Les séries du PIB français présente-t-elle une tendance ?
 - Cette tendance est-elle déterministe ou stochastique ?
 - Appliquez le test de stationarité approprié
 - Le cas échéant, différenciez la série pour la rendre stationnaire
- 2 Même question pour la série de l'index des prix à la consommation
- 3 Même question pour la série du taux de chômage

Section 10

Modélisation de séries temporelles stationnaires

Introduction

- Les modèles AR (AutoRegressive), MA (Moving Average) et ARMA (AutoRegressive Moving Average) sont des modèles fondamentaux pour étudier et décrire le comportement des séries temporelles
- Ces modèles ont été popularisés par George Box et Gwilym Jenkins
- Leur principale limitation est qu'ils ne peuvent modéliser que des séries stationnaires, cependant, on peut transformer des séries non-stationnaires par différenciation pour les étudier avec ce cadre (modèles ARIMA)

Processus autorégressif (AR)

Processus autorégressif AR(1) stationnaire

- Dans un modèle auto régressif, les variables explicatives sont des valeurs passées (lags) de la variable expliquée
- Dans un modèle AR(1) (autorégressif d'ordre 1), y est retardé d'une période

$$y_t = \alpha + \beta_1 \cdot y_{t-1} + \epsilon_t$$

avec $\epsilon_t \sim N(0, \sigma^2)$

- La valeur de $|\beta_1|$ détermine l'évolution du processus
- Si $|\beta_1| < 1$, le processus AR(1) est **stationnaire**
- Si $|\beta_1| \geq 1$, le processus est **non-stationnaire** (les chocs s'accumulent dans le temps)
 - Si $|\beta_1| > 1$ le processus croît sans limite
 - Si $|\beta_1| = 1$ le processus a une **racine unitaire** (unit root), c'est une marche aléatoire

Processus autorégressif (AR)

Processus autorégressif AR(1) stationnaire : Propriétés

■ Propriétés d'un processus AR(1) :

■ Moyenne de y_t

$$\mu = \frac{\alpha}{1 - \beta_1}$$

■ Variance :

$$\text{Var}(x_t) = \frac{\sigma_w^2}{1 - \beta_1^2}$$

■ Corrélation : $\rho_h = \beta_1^h$

- Si $(X_t)_{t \in \mathbb{Z}}$ est un processus $AR(p)$ ses **autocorrélations partielles** s'annulent à partir du rang $p + 1$
- Les **autocorrélations simples** décroissent rapidement vers 0 (de manière exponentielle ou sinusoïdale amortie)

Processus autorégressif AR(1)

Simulation (1)

- On utilise la fonction `arima.sim` pour simuler un processus AR(1) avec $\beta_1 = 0.9$ (cf. [4])
- La fonction `set.seed` permet d'initialiser le générateur de nombres aléatoires (la série de 100 nombres aléatoires sera toujours la même)

```
set.seed(123456)
wn <- rnorm(100)
AR1sim <- arima.sim(n = 100, list(ar = 0.9), innov=wn)
```

- Les corrélations partielles s'annulent bien à partir du rang 2

```
pacf(AR1sim, plot=FALSE)

##
## Partial autocorrelations of series 'AR1sim', by lag
##
##      1      2      3      4      5      6      7      8      9     10     11
## 0.937 -0.169 0.109 0.034 0.062 -0.016 -0.069 -0.038 0.096 0.030 -0.181
##      12     13     14     15     16     17     18     19     20
## 0.019 0.098 -0.041 -0.014 0.092 -0.041 -0.138 -0.042 0.005
```

Processus autorégressif AR(1)

Simulation (2)

- Le code suivant permet de visualiser la série ainsi que ses autocorrélations (partielles)

```
op <- par(no.readonly=TRUE)

layout(matrix(c(1, 1, 2, 3), 2, 2, byrow=TRUE))

plot.ts(AR1sim, ylab="", main="Processus AR(1) avec B1=0.9")

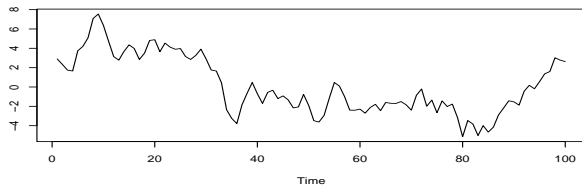
acf(AR1sim, main="Autocorrelations", ylab="",
    ylim=c(-1, 1), ci.col = "black")

pacf(AR1sim, main="Partial Autocorrelations", ylab="",
     ylim=c(-1, 1), ci.col = "black")

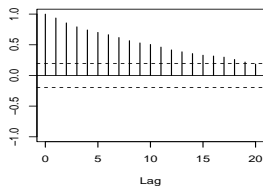
par(op)
```

Processus autorégressif AR(1)

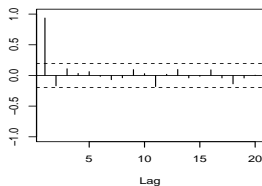
Simulation (3)



Autocorrelations



Partial Autocorrelations

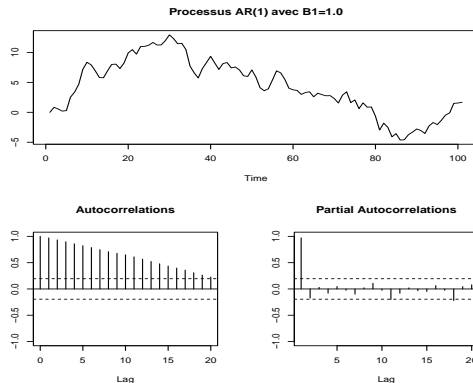


Processus autorégressif AR(1)

Simulation (3)

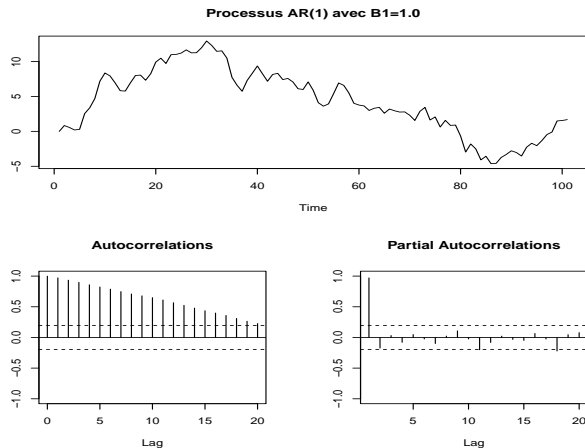
- Lorsque $\beta_1 = 1.0$ il s'agit d'un processus avec racine unitaire, i.e. une marche aléatoire (cf [5])
- La fonction `arima.sim` produit une erreur si le paramètre `ar` est ≥ 1

```
rusim <- 0
for (i in 1:length(wn)) {rusim[i+1] <- rusim[i] + wn[i]}
```



Processus autorégressif AR(1)

Simulation (3)



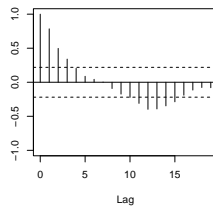
Modèle autorégressif

Données Nelson-Plosser : ACF et PACF

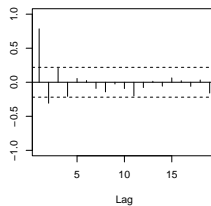
- On modélise le taux de chômage aux USA (données Nelson-Plosser)



Autocorrelations



Partial Autocorrelations

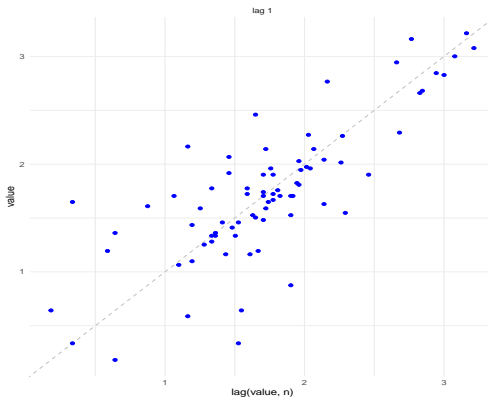


Modèle autorégressif

Données Nelson-Plosser

- On visualise $y \times y_{t-1}$ à l'aide de la fonction `gg_lag`

```
unemp1909p <- nelplo_tsbl %>% filter(index>1909 & key=='unemp')
unemp1909p %>% gg_lag(y=value, lags=1, geom="point", colour="blue")
```



Modèle autorégressif

Données Nelson-Plosser : Estimation (1)

- On peut estimer le modèle $AR(1)$ avec les fonctions `lm` (linear model) et `lag` :

```
summary(lm(value ~ lag(value), data=unemp1909p))

##
## Call:
## lm(formula = value ~ lag(value), data = unemp1909p)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.23467 -0.16288 -0.01882  0.18815  1.01130
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.37324    0.13168   2.834  0.00588 **
## lag(value)   0.78497    0.07107  11.046 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4001 on 76 degrees of freedom
## (1 observation effacée parce que manquante)
## Multiple R-squared:  0.6162, Adjusted R-squared:  0.6111
## F-statistic: 122 on 1 and 76 DF, p-value: < 2.2e-16
```

- Le coefficient associé à y_{t-1} est significatif

Modèle autorégressif

Données Nelson-Plosser : Estimation (2)

- On peut également estimer le modèle $AR(1)$ avec la fonction `arima` :

```
summary(arima(unemp1909p[, "value"], c(1,0,0)))
##
## Call:
## arima(x = unemp1909p[, "value"], order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##      0.7756      1.7394
## s.e.  0.0680      0.1887
##
## sigma^2 estimated as 0.154:  log likelihood = -38.66,  aic = 83.32
##
## Training set error measures:
##              ME          RMSE          MAE          MPE          MAPE          MASE
## Training set -0.0004783172  0.3924347  0.2859196 -14.03967  28.36622  0.9722669
##              ACF1
## Training set 0.2482824
```

- Le coefficient associé à y_{t-1} est significatif

Modèle moyenne mobile

Définition

- y_t est un processus moyenne mobile (Moving Average, MA) d'ordre q noté $MA(q)$ si

$$y_t = \mu + \epsilon_t + \theta_1 \cdot \epsilon_{t-1} + \dots + \theta_q \cdot \epsilon_{t-q}$$

- On considère que le processus est la résultante d'une combinaison linéaire de perturbations decorrélées (un bruit blanc et son passé)
- Un $MA(q)$ est **toujours stationnaire** quelles que soient les valeurs des θ
- Les propriétés d'un modèle $MA(1)$

$$y_t = \mu + \epsilon_t + \theta_1 \cdot \epsilon_{t-1}$$

- $E[y_t] = \mu$
- $Var(y_t) = \sigma_\epsilon^2(1 + \theta_1^2)$
- ACF : $\rho_1 = \theta_1/(1 + \theta_1^2)$ et $\rho_h = 0$ pour $h \geq 2$

Modèle moyenne mobile

Fonction d'autocorrélation

- Si $(X_t)_{t \in \mathbb{Z}}$ est un processus $MA(q)$, sa fonction d'autocorrélation est égale à 0 pour les rangs supérieurs à q
- Les autocorrélations observées sont un bon indicateur de l'ordre du processus
- Les autocorrélations partielles tendent vers 0 exponentiellement lorsque le rang augmente (comme les corrélations simples d'un processus AR) (cf par ex. [7])
- Plus généralement, on peut montrer que les autocorrélations partielles d'un modèle $MA(q)$ se comportent comme les autocorrélations d'un modèle $AR(q)$

Modèle moyenne mobile

Simulation (1)

- En utilisant la même série w_n de nombre aléatoires, on simule un processus $MA(1)$

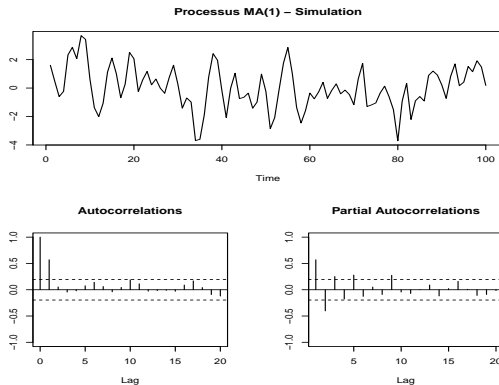
```
MA1sim <- arima.sim(n = 100, list(ma = 0.9), innov=wn)
```

- Le code suivant permet de visualiser la série ainsi que ses autocorrélations (partielles)

```
op <- par(no.readonly=TRUE)
layout(matrix(c(1, 1, 2, 3), 2, 2, byrow=TRUE))
plot.ts(MA1sim, ylab="") acf(MA1sim, main="Autocorrelations", ylab="", main="Processus MA(1) - Simulation",
ylim=c(-1, 1), ci.col = "black") pacf(MA1sim, main="Partial Autocorrelations", ylab="", ylim=c(-1, 1), ci.col = "black")
par(op)
```


Modèle moyenne mobile

Simulation (2)



Modèle ARMA

- Un processus ARMA (Auto Regressive Moving Average) est une synthèse des processus AR et MA
- y_t obéit à un modèle ARMA(p, q) s'il est stationnaire et vérifie :

$$y_t = c + \phi_1 \cdot y_{t-1} + \dots + \phi_p \cdot y_{t-p} + \epsilon_t + \theta_1 \cdot \epsilon_{t-1} + \dots + \theta_q \cdot \epsilon_{t-q}$$

Identification d'un modèle ARMA

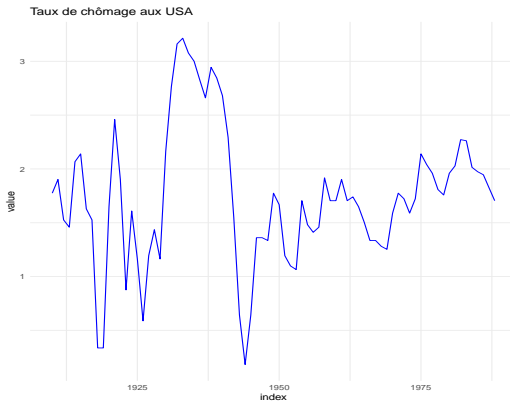
- Soit la trajectoire observée y_1, \dots, y_t d'une série y_t , éventuellement transformée par passage en log
- Si cette trajectoire peut-être considérée comme stationnaire, on peut lui ajuster un modèle ARMA(p, q) (on ne traite pas ici des séries présentant une saisonnalité)
- La première étape consiste à choisir les ordres p et q
- Le choix de p et q n'est pas unique, il faut comparer plusieurs modèles
- Le premier critère pour juger de la qualité d'un modèle est la blancheur du résidu obtenu (voir la section définitions)

Modèle ARMA

Taux de chômage aux USA

- Nous avons vu que la série des taux de chômage aux USA peut-être considérée comme stationnaire

```
ggplot(data=unemp1909p) + geom_line(aes(x=index, y=value), color="blue") +  
  ggtitle("Taux de chômage aux USA")
```

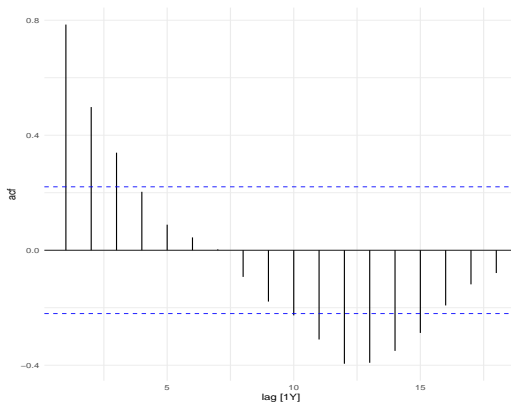


Modèle ARMA

Taux de chômage aux USA : Autocorrélation

- On commence par examiner sa fonction d'autocorrélation

```
unemp1909p %>% ACF(value) %>% autoplot()
```

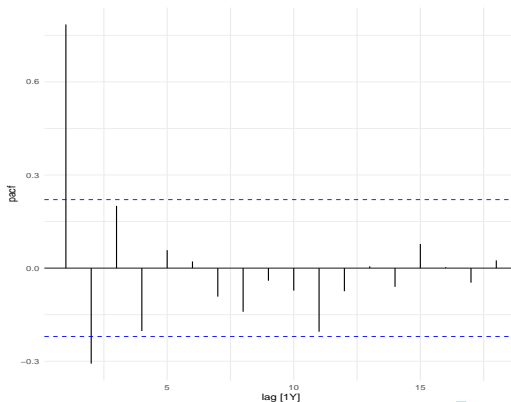


Modèle ARMA

Taux de chômage aux USA : Autocorrélation partielle

- Le coefficient d'autocorrélation partielle ϕ_k , k représente l'apport d'explication de y_{t-k} à y_t , **toutes choses égales par ailleurs**
- La fonction d'autocorrélation partielle permet d'identifier l'ordre p d'un processus $AR(p)$

```
unemp1909p %>% PACF(value) %>% autoplot()
```



Modèle ARMA

Taux de chômage aux USA : Estimation du modèle (1)

- On commence par un modèle ARMA(2,2)
- L'erreur standard est élevée par rapport à la valeur du coefficient

```
arma.mod1 <- arima(unemp1909p[, "value"], c(2,0,2))
summary(arma.mod1)

##
## Call:
## arima(x = unemp1909p[, "value"], order = c(2, 0, 2))
##
## Coefficients:
##          ar1          ar2          ma1          ma2  intercept
##      0.7228  -0.0376  0.4009  -0.1253   1.7357
## s.e.  0.9293   0.5663  0.9222   0.5102   0.1598
##
## sigma^2 estimated as 0.1296:  log likelihood = -32.05,  aic = 76.11
##
## Training set error measures:
##              ME          RMSE          MAE          MPE          MAPE          MASE
## Training set 0.0004553661 0.3599465 0.2635755 -10.65611 24.60626 0.896286
##              ACF1
## Training set 0.01310071
```

Modèle ARMA

Taux de chômage aux USA : Modèle $ARMA(2, 2)$ - significativité des paramètres

- Les p-valeurs ne sont pas fournies directement par les fonctions `arima` et `auto.arima`
- On peut les calculer en utilisant les erreurs standard des coefficients

```
# Extract standard errors
se <- sqrt(diag(vcov(arma.mod1)))

# Calculate the t-values
t_values <- coef(arma.mod1) / se

# Calculate the p-values
p_values <- 2 * (1 - pnorm(abs(t_values)))

# Create a data frame with coefficients, standard errors, t-values, and p-values
results <- data.frame(
  Coefficient = coef(arma.mod1),
  Std_Error = se,
  T_Value = t_values,
  P_Value = p_values
)
print(results)
```

	Coefficient	Std_Error	T_Value	P_Value
## ar1	0.72276639	0.9292858	0.77776541	0.4367073
## ar2	-0.03759621	0.5663260	-0.06638616	0.9470704
## ma1	0.40092547	0.9222413	0.43472947	0.6637588
## ma2	-0.12526964	0.5102430	-0.24550974	0.8060618
## intercept	1.73571612	0.1597567	10.86474781	0.0000000

Modèle ARMA

Taux de chômage aux USA : Estimation du modèle (2)

- On ajuste un modèle ARMA(1,1)
- L'erreur standard des coefficients a diminué fortement

```
arma.mod2 <- arima(unemp1909p[, "value"], c(1,0,1))
summary(arma.mod2)

##
## Call:
## arima(x = unemp1909p[, "value"], order = c(1, 0, 1))
##
## Coefficients:
##          ar1          ma1  intercept
##      0.5895   0.5555      1.7356
## s.e.  0.1032   0.1103      0.1503
##
## sigma^2 estimated as 0.13:  log likelihood = -32.19,  aic = 72.38
##
## Training set error measures:
##              ME          RMSE          MAE          MPE          MAPE          MASE
## Training set 0.0005107137 0.3605539 0.2665423 -10.46991 24.69032 0.9063747
##              ACF1
## Training set -0.006430895
```

Modèle ARMA

Taux de chômage aux USA : Sélection du modèle

- On utilise l'AIC (Akaike Information Criterion) pour sélectionner le modèle (la plus petite valeur de l'AIC)
- On retient le second modèle avec moins de paramètres ($p = 1$ et $q = 1$)

```
AIC(arma.mod1, arma.mod2)
```

```
##           df      AIC  
## arma.mod1  6 76.10996  
## arma.mod2  4 72.37670
```

- NOTE : le calcul de l'AIC requiert la log-vraisemblance (log-likelihood) du modèle, celle ci n'est pas disponible lorsque le modèle est estimé par MCO

Modèle ARMA

Taux de chômage aux USA : Utilisation de la fonction `auto.arima`

- La fonction `auto.arima` recherche le meilleur modèle en utilisant un critère d'information
- Le résultat confirme le choix du modèle ARMA(1,1)

```
nelplo.arma <- auto.arima(unemp1909p$value)
nelplo.arma

## Series: unemp1909p$value
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ma1      mean
##       0.5895  0.5555  1.7356
## s.e.  0.1032  0.1103  0.1503
##
## sigma^2 = 0.1351: log likelihood = -32.19
## AIC=72.38   AICc=72.92   BIC=81.85
```

Modèle ARMA

Taux de chômage aux USA : Significativité des paramètres

■ Calcul des p-valeurs en utilisant les erreurs standard des coefficients

```
# Extract standard errors
se <- sqrt(diag(vcov(nelplo.arma)))

# Calculate the t-values
t_values <- coef(nelplo.arma) / se

# Calculate the p-values
p_values <- 2 * (1 - pnorm(abs(t_values)))

# Create a data frame with coefficients, standard errors, t-values, and p-values
results <- data.frame(
  Coefficient = coef(nelplo.arma),
  Std_Error = se,
  T_Value = t_values,
  P_Value = p_values
)
print(results)
```

	Coefficient	Std_Error	T_Value	P_Value
## ar1	0.5894997	0.1032109	5.711603	1.119166e-08
## ma1	0.5554587	0.1103101	5.035431	4.767737e-07
## intercept	1.7355500	0.1503400	11.544166	0.000000e+00

Modèle ARMA

Taux de chômage aux USA : Diagnostic du modèle - Préparation

- Afin de vérifier la validité des modèles estimés, on doit vérifier au minimum la significativité des paramètres et la blancheur du résidu
- On prépare un data frame avec les valeurs observées, les valeurs prédites par le modèle, et les résidus

```
nelplo.arma.diag <- data.frame(index=unemp1909p$index, obs=unemp1909p$value,
                                prediction=as.data.frame(nelplo.arma$fitted)$x,
                                residus=as.data.frame(nelplo.arma$residuals)$x)
```

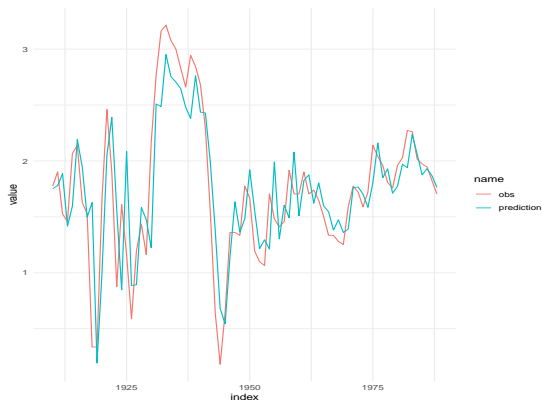
```
head(nelplo.arma.diag)
```

	index	obs	prediction	residus
## 1	1910	1.774952	1.752238	0.02271448
## 2	1911	1.902108	1.778215	0.12389255
## 3	1912	1.526056	1.887011	-0.36095467
## 4	1913	1.458615	1.416962	0.04165325
## 5	1914	2.066863	1.596360	0.47050309
## 6	1915	2.140066	2.191556	-0.05148945

Modèle ARMA

Taux de chômage aux USA : Valeurs observées x prédites

```
nelplo.arma.diag %>%
  pivot_longer(cols=2:4) %>%
  filter(name %in% c("prediction", "obs")) %>%
  ggplot() + geom_line(aes(x=index, y=value, color=name))
```



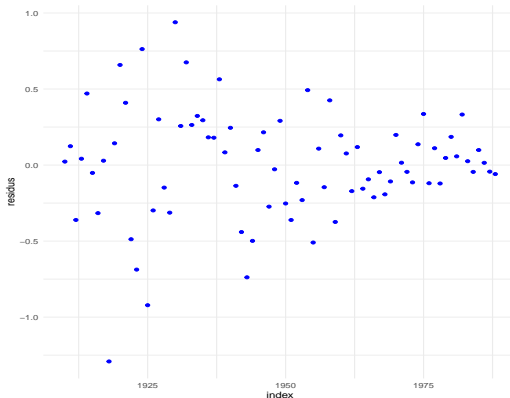
Modèle ARMA

Taux de chômage aux USA : Résidus (1)

- Si notre modèle est adéquat, les résidus devraient fluctuer autour de 0, sans tendance

```
nelplo.arma.diag %>% ggplot() +  
  geom_point(aes(x=index, y=residus), color="blue")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting  
## to continuous.
```

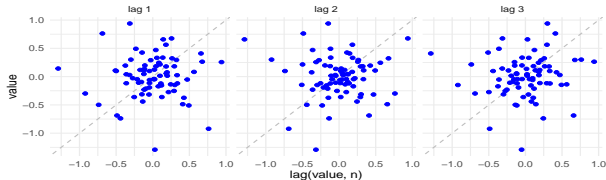


Modèle ARMA

Taux de chômage aux USA : Résidus (2)

- On visualise l'autocorrélation des résidus à l'aide de la fonction `gg_lag`

```
nelplo.arma$residuals %>% as_tsibble() %>%  
  gg_lag(y=value, lags=1:3, geom="point", colour="blue")
```



Modèle ARMA

Taux de chômage aux USA : Résidus (3)

- On teste la blancheur des résidus aux retards 1 à 3
- Toutes les p-valeur sont largement supérieure à 0.05, on peut conclure à la blancheur des résidus

```
lags <- 1:3
pval <- NULL
for (l in lags) {
  pval <- c(pval, box_pierce(nelplo.arma.diag$residus, lag=l)["bp_pvalue"])
}
res <- data.frame(lags, pval)
res
```

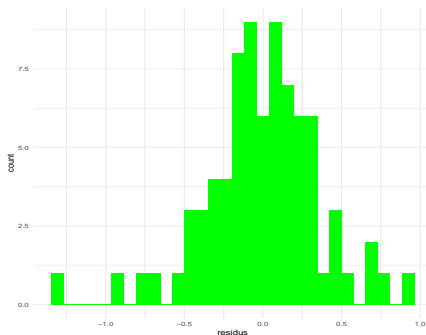
##	lags	pval
## 1	1	0.9544185
## 2	2	0.9906342
## 3	3	0.8891629

Modèle ARMA

Taux de chômage aux USA : Résidus (4)

- L'hypothèse de normalité des résidus (test de Jarque-Bera) est rejetée

```
ggplot(nelplo.arma.diag) + geom_histogram(aes(x=residus), fill='green')
```



```
jarque.bera.test(nelplo.arma.diag$residus)
##
##  Jarque Bera Test
##
## data:  nelplo.arma.diag$residus
## X-squared = 11.985, df = 2, p-value = 0.002498
```

Modèle ARMA

Taux de chômage aux USA : Projection (1)

- On peut utiliser la fonction générique `forecast` pour la projection du taux de chômage
- On obtient la prédiction et les intervalles de confiance à 80% et 95%

```
forecast(nelplo.arma, h=10)
```

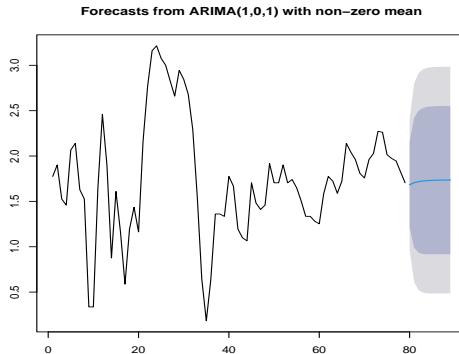
##	Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 80		1.684490	1.2133901	2.155590	0.9640049	2.404975
## 81		1.705450	0.9892962	2.421604	0.6101872	2.800713
## 82		1.717806	0.9342365	2.501376	0.5194397	2.916173
## 83		1.725090	0.9194124	2.530768	0.4929124	2.957268
## 84		1.729384	0.9161642	2.542603	0.4856717	2.973096
## 85		1.731915	0.9160908	2.547739	0.4842196	2.979611
## 86		1.733407	0.9166798	2.550135	0.4843304	2.982484
## 87		1.734287	0.9172458	2.551328	0.4847304	2.983843
## 88		1.734805	0.9176554	2.551955	0.4850823	2.984528
## 89		1.735111	0.9179232	2.552299	0.4853301	2.984892

Modèle ARMA

Taux de chômage aux USA : Projection (2)

- La méthode générique `plot` permet de visualiser la projection

```
plot(forecast(nelplo.arma, h=10))
```



Section 11

Modélisation de séries non-stationnaires

Modèle ARIMA

Identification d'un modèle

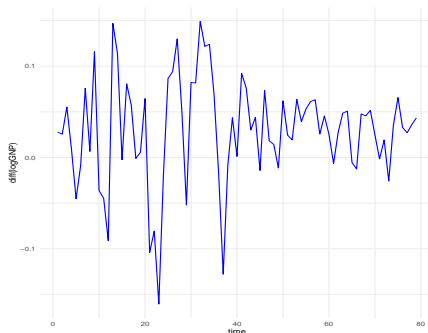
- Un modèle $\text{ARIMA}(p,d,q)$ est un modèle ARMA sur la série différenciée, d est l'ordre de différenciation
- Soit la trajectoire observée y_1, \dots, y_t d'une série y_t , éventuellement obtenue après transformation d'une série initiale par passage en log
- La série n'est pas stationnaire et on veut lui ajuster un modèle $\text{ARIMA}(p, d, q)$ (on ne traite pas ici des séries présentant une saisonnalité)
- Une fois d choisis, on est ramené à l'identification d'un ARMA sur la série différenciée
- Pour choisir d , on peut tester l'hypothèse $d = 1$ contre $d = 0$ avec un test ADF
- On peut également comparer les modèles avec et sans différenciation à l'aide d'un critère d'information ou de la valeur prédictive

Modèle ARIMA

PIB des USA

- Nous avons vu que cette série (après **transformation logarithmique**) est non stationnaire et qu'elle possède un trend stochastique
- On peut donc la différencier puis ajuster un modèle ARMA
- La série différenciée `gnpreal.diff` a été calculée précédemment

```
ggplot(gnpreal.diff) + geom_line(aes(x=time, y=diff), color="blue") + ylab("diff(logGNP)")
```

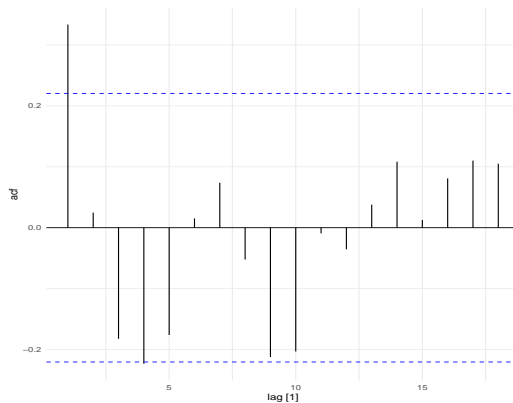


Modèle ARIMA

PIB des USA : Autocorrélation (série différenciée)

- L'autocorrélation est significative uniquement pour le lag 1

```
gnpreal.diff %>% as_tsibble(index=time) %>% ACF(diff) %>% autoplot()
```

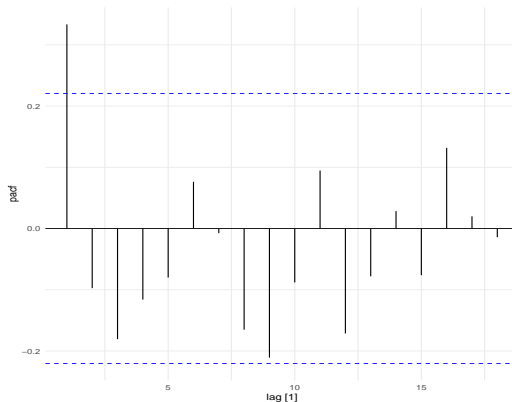


Modèle ARIMA

PIB des USA : Autocorrélation partielle (série différenciée)

- La fonction d'autocorrélation partielle suggère un modèle $AR(1)$

```
gnpreal.diff %>% as_tsibble(index=time) %>% PACF(diff) %>% autoplot()
```



Modèle ARIMA

PIB des USA : Modélisation

- On utilise la fonction `auto.arima`
- Le modèle retenu est $ARIMA(1, 1, 0)$: un modèle $ARMA(1, 0)$ sur la série intégrée $I(1)$

```
regdata <- nelplo1909$gnp.real
gnpreal.arima <- auto.arima(regdata)
gnpreal.arima

## Series: regdata
## ARIMA(1,1,0) with drift
##
## Coefficients:
##          ar1    drift
##          0.3296  0.0295
## s.e.      0.1052  0.0090
##
## sigma^2 = 0.003015:  log likelihood = 118.12
## AIC=-230.24   AICc=-229.92   BIC=-223.14
```

Modèle ARIMA

PIB des USA : Diagnostics (1)

■ Calcul des p-valeurs en utilisant les erreurs standard des coefficients

```
# Extract standard errors
se <- sqrt(diag(vcov(gnpreal.arma)))

# Calculate the t-values
t_values <- coef(gnpreal.arma) / se

# Calculate the p-values
p_values <- 2 * (1 - pnorm(abs(t_values)))

# Create a data frame with coefficients, standard errors, t-values, and p-values
results <- data.frame(
  Coefficient = coef(gnpreal.arma),
  Std_Error = se,
  T_Value = t_values,
  P_Value = p_values
)
print(results)
```

	Coefficient	Std_Error	T_Value	P_Value
## ar1	0.32961164	0.105246210	3.131815	0.001737294
## drift	0.02954975	0.009042602	3.267837	0.001083726

Modèle ARIMA

PIB des USA : Diagnostics (2)

- On prépare un data frame avec les valeurs prédites par le modèle, et les résidus

```
gnpreal.arma.diag <- data.frame(index=nelpl01909$index,
                                prediction=as.data.frame(gnpreal.arma$fitted)$x,
                                residus=as.data.frame(gnpreal.arma$residuals)$x)

head(gnpreal.arma.diag)
```

##	index	prediction	residus
## 1	1909	4.755732	0.004730911
## 2	1910	4.789920	-0.001595465
## 3	1911	4.817318	-0.003508918
## 4	1912	4.842019	0.027052834
## 5	1913	4.907097	-0.028850607
## 6	1914	4.901080	-0.067977600

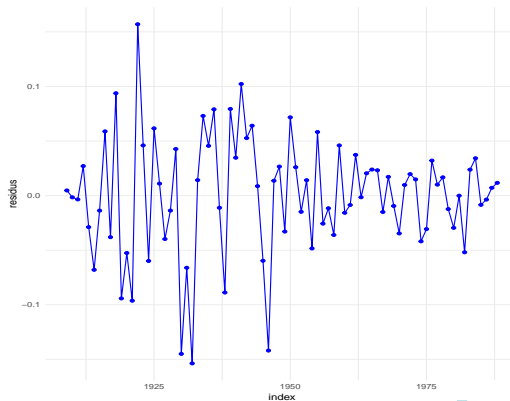
Modèle ARIMA

PIB des USA : Résidus (1)

- Si notre modèle est adéquat, les résidus devraient fluctuer autour de 0, sans tendance

```
gnpreal.arima.diag %>% ggplot(aes(x=index, y=residus)) +  
  geom_point(color="blue") +  
  geom_line(color="blue")
```

*## Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.*

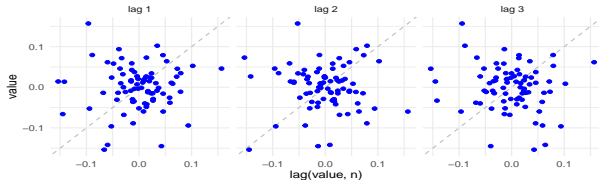


Modèle ARIMA

PIB des USA : Résidus (2)

- On visualise l'autocorrélation des résidus à l'aide de la fonction `gg_lag`

```
gnpreal.arima$residuals %>% as_tsibble() %>%  
  gg_lag(y=value, lags=1:3, geom="point", colour="blue")
```



Modèle ARIMA

PIB des USA : Résidus (3)

- On teste la blancheur des résidus aux retards 1 à 3
- Toutes les p-valeur sont largement supérieure à 0.05, on peut conclure à la blancheur des résidus

```
lags <- 1:3
pval <- NULL
for (l in lags) {
  pval <-c(pval, box_pierce(gnpreal.arima.diag$residus, lag=l)["bp_pvalue"])
}
res <- data.frame(lags, pval)
res
```

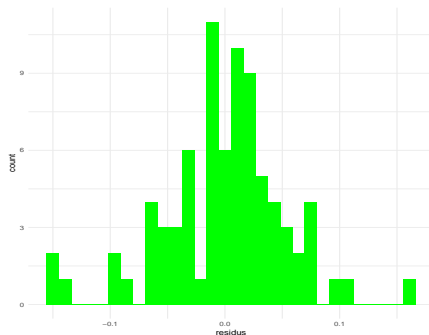
##	lags	pval
## 1	1	0.7460147
## 2	2	0.9252396
## 3	3	0.5645515

Modèle ARIMA

PIB des USA : Résidus (4)

- L'hypothèse de normalité des résidus (test de Jarque-Bera) est acceptée

```
ggplot(gnpreal.arima.diag) + geom_histogram(aes(x=residus), fill='green')
```



```
jarque.bera.test(gnpreal.arima.diag$residus)
##
##  Jarque Bera Test
##
## data:  gnpreal.arima.diag$residus
## X-squared = 5.7526, df = 2, p-value = 0.05634
```


Modèle ARIMA

PIB des USA : Prédiction (1)

- La fonction `forecast` produit les prédictions à partir du modèle sélectionné, avec un intervalle de confiance
- Les données sont annuelles, on prédit à un horizon de 10 ans

```
forecast(gnpreal.arima, h=10)
```

##	Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 81	7.122980	7.052606	7.193353	7.015353	7.230606	
## 82	7.154028	7.036948	7.271107	6.974970	7.333085	
## 83	7.184071	7.029307	7.338836	6.947380	7.420763	
## 84	7.213784	7.027470	7.400098	6.928841	7.498727	
## 85	7.243387	7.029736	7.457039	6.916636	7.570139	
## 86	7.272955	7.034967	7.510943	6.908984	7.636926	
## 87	7.302510	7.042418	7.562603	6.904733	7.700288	
## 88	7.332062	7.051590	7.612535	6.903117	7.761008	
## 89	7.361612	7.062141	7.661084	6.903610	7.819615	
## 90	7.391162	7.073826	7.708499	6.905838	7.876487	

Modèle ARIMA

Log du PIB des USA : Prédiction (2)

La méthode générique `plot` produit une représentation graphique de la prédiction

```
plot(forecast(arimod, h=10))
```

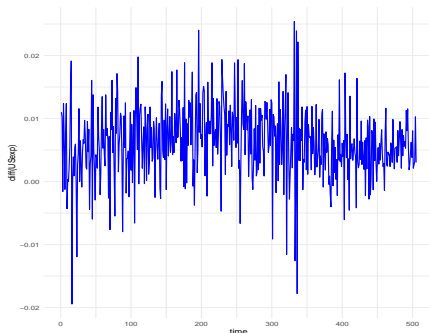
```
## Error in h(simpleError(msg, call)): erreur d'évaluation de l'argument 'x' lors de la sélection d'une  
méthode pour la fonction 'plot' : objet 'arimod' introuvable
```

Modèle ARIMA

Dépenses agrégées de consommation aux USA

- Nous avons vu que cette série (après transformation logarithmique) est non stationnaire et qu'elle possède un trend stochastique
- On peut donc la différencier puis ajuster un modèle ARMA
- La série différenciée USExp_diff a été calculée précédemment

```
ggplot(USExp.diff) + geom_line(aes(x=time, y=diff), color="blue") + ylab("diff(USExp)")
```

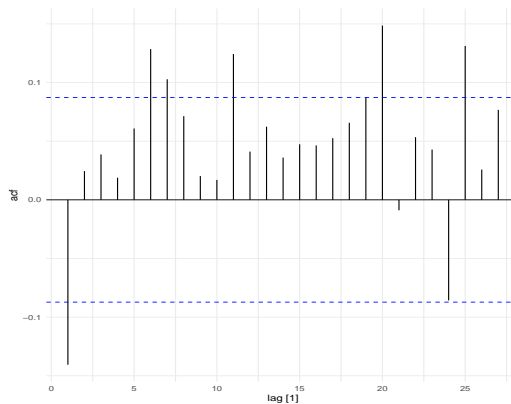


Modèle ARIMA

Dépenses agréées de consommation aux USA : Fonction d'autocorrélation (série différenciée)

■ La fonction d'autocorrélation ne montre pas de tendance

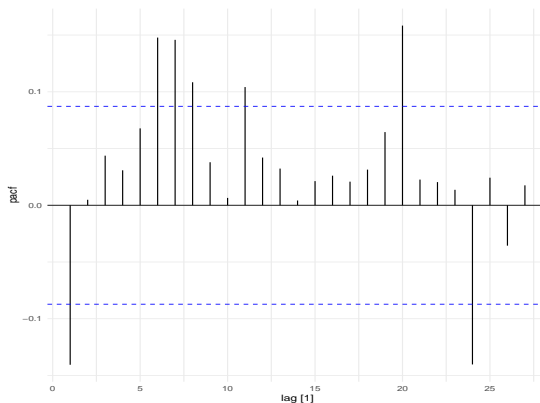
```
USExp.diff %>% as_tsibble(index=time) %>% ACF(diff) %>% autoplot()
```



Modèle ARIMA

Dépenses agrégées de consommation aux USA : Fonction d'autocorrélation partielle

```
USExp.diff %>% as_tsibble(index=time) %>% PACF(diff) %>% autoplot()
```



Modèle ARIMA

Dépenses agrégées de consommation aux USA : Modélisation

■ On utilise la fonction `auto.arima`

```
regdata <- USExp[, "logExp"]
arimod <- auto.arima(regdata)
arimod

## Series: regdata
## ARIMA(2,2,2)
##
## Coefficients:
##          ar1      ar2      ma1      ma2
##          0.3384  0.0158 -1.5509  0.5725
## s.e.      0.1704  0.0679   0.1644  0.1578
##
## sigma^2 = 3.053e-05: log likelihood = 1906.35
## AIC=-3802.69   AICc=-3802.57   BIC=-3781.58
```

Modèle ARIMA

Dépenses agrégées de consommation aux USA : Prédiction (1)

- La fonction `forecast()` produit les prédictions à partir du modèle sélectionné, avec un intervalle de confiance

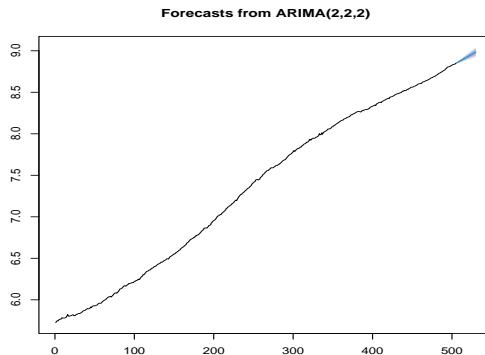
```
forecast(arimod, h=24)
```

##	Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 507		8.859892	8.852811	8.866972	8.849063	8.870721
## 508		8.865359	8.856346	8.874371	8.851575	8.879142
## 509		8.870780	8.860308	8.881252	8.854765	8.886796
## 510		8.876182	8.864410	8.887955	8.858178	8.894187
## 511		8.881577	8.868565	8.894590	8.861676	8.901478
## 512		8.886969	8.872742	8.901197	8.865210	8.908729
## 513		8.892361	8.876928	8.907794	8.868758	8.915963
## 514		8.897751	8.881115	8.914388	8.872308	8.923195
## 515		8.903142	8.885300	8.920984	8.875854	8.930429
## 516		8.908532	8.889479	8.927586	8.879393	8.937672
## 517		8.913923	8.893652	8.934194	8.882921	8.944925
## 518		8.919313	8.897816	8.940811	8.886435	8.952191
## 519		8.924704	8.901970	8.947438	8.889935	8.959473
## 520		8.930094	8.906114	8.954075	8.893419	8.966770
## 521		8.935485	8.910246	8.960723	8.896886	8.974084
## 522		8.940875	8.914367	8.967383	8.900335	8.981416
## 523		8.946266	8.918476	8.974055	8.903765	8.988766
## 524		8.951656	8.922573	8.980740	8.907177	8.996136
## 525		8.957047	8.926656	8.987437	8.910569	9.003525
## 526		8.962437	8.930727	8.994147	8.913941	9.010934
## 527		8.967828	8.934785	9.000870	8.917293	9.018362
## 528		8.973218	8.938829	9.007607	8.920625	9.025811
## 529		8.978609	8.942861	9.014356	8.923937	9.033280
## 530		8.983999	8.946879	9.021119	8.927229	9.040769

Modèle ARIMA

Dépenses agrégées de consommation aux USA : Prédiction (2)

```
plot(forecast(arimod, h=24))
```



Exercice

- Ajustez un modèle ARMA ou ARIMA sur la série du PIB français
- Réalisez le diagnostic du modèle
- Prédisez le PIB pour les trimestres à venir

Section 12

Modèles multivariés pour l'analyse de séries temporelles stationnaires

Série temporelle multivariée

Introduction

- L'analyse de séries temporelles multivariées permet notamment
 - 1 D'étudier les relations **dynamiques** entre variables
 - 2 D'améliorer la précision des prédictions
- Nous allons nous focaliser ici sur l'analyse de séries temporelles multivariées **stationnaires**
- Les modèles VAR (Vector Autoregressive Models) représentent l'instrument principal pour l'études de séries temporelles multivariées stationnaires

Série temporelle multivariée

PIB et taux de chômage aux USA (1)

- La série multivariée $y_t = (y_{1t}, \dots, y_{kt}, \dots, y_{Kt})$, $k = 1, \dots, K$ contient K variables
- Par exemple, soit y_{1t} le PIB et y_{2t} le taux de chômage aux USA (exemple tiré de [8] (voir ici))
- Ici, $k = 2$
- Les données sont disponibles dans le fichier `q-gdpunemp.txt`

```
gdpunemp <- read.csv('/home/alex/Devel/Cours-R-TS/data/q-gdpunemp.txt', sep=' ')
head(gdpunemp)
```

```
##   year mon      gdp      rate
## 1 1948   1 1821.809 3.733333
## 2 1948   4 1855.345 3.666667
## 3 1948   7 1865.320 3.766667
## 4 1948  10 1868.184 3.833333
## 5 1949   1 1842.240 4.666667
## 6 1949   4 1835.512 5.866667
```

- En étudiant conjointement les deux séries, on peut estimer les relations temporelles et simultanées entre le PIB et le taux de chômage

Série temporelle multivariée

PIB et taux de chômage aux USA (2)

- Séries trimestrielles CVS obtenues à partir de données mensuelles (moyenne), de 1948 à 2011
- On considère le logarithme du PIB (en millions de dollars 2005)
- Création d'un index et conversion en objet `mts`

```
gdpunemp$loggdp <- log(gdpunemp$gdp)
gdpunemp <- ts(gdpunemp[, c("loggdp", "rate")], start=1948, frequency = 4)
head(gdpunemp)
```

```
##           loggdp      rate
## 1948 Q1  7.507585  3.733333
## 1948 Q2  7.525826  3.666667
## 1948 Q3  7.531188  3.766667
## 1948 Q4  7.532722  3.833333
## 1949 Q1  7.518738  4.666667
## 1949 Q2  7.515079  5.866667
```

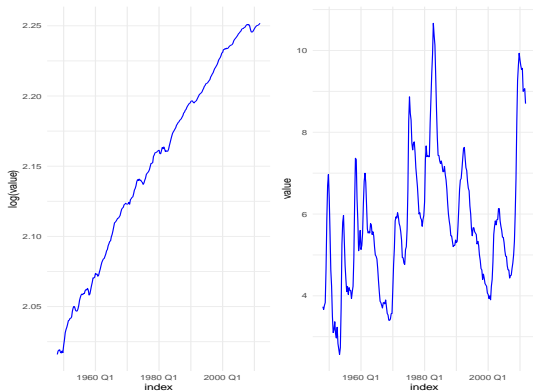
- Création d'un objet `tsibble`

```
gdpunemp_tsbl <- as_tsibble(gdpunemp)
```

Série temporelle multivariée

PIB et taux de chômage aux USA : Chronogramme

```
g1 <- gdpunemp_tsbl %>% filter(key=="loggdp") %>%
  ggplot() + geom_line(aes(x=index, y=log(value)), color="blue")
g2 <- gdpunemp_tsbl %>% filter(key=="rate") %>%
  ggplot() + geom_line(aes(x=index, y=value), color="blue")
g1 + g2
```



Série temporelle multivariée

PIB et taux de chômage aux USA : Variations (1)

- On calcule le taux de croissance du PIB et la variation du taux de chômage en différenciant les séries

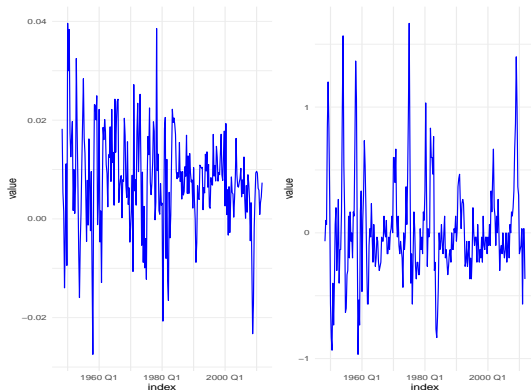
```
gdpunemp_i <- diff(gdpunemp) %>% as_tsibble()
head(gdpunemp_i)

## # A tsibble: 6 x 3 [1Q]
## # Key:           key [1]
##   index key      value
##   <qtr> <chr>    <dbl>
## 1 1948 Q2 loggdp  0.0182
## 2 1948 Q3 loggdp  0.00536
## 3 1948 Q4 loggdp  0.00153
## 4 1949 Q1 loggdp -0.0140
## 5 1949 Q2 loggdp -0.00366
## 6 1949 Q3 loggdp  0.0112
```

Série temporelle multivariée

PIB et taux de chômage aux USA : Variations (2)

```
g1 <- gdpunemp_i %>% filter(key=="loggdp") %>%
  ggplot() + geom_line(aes(x=index, y=value), color="blue")
g2 <- gdpunemp_i %>% filter(key=="rate") %>%
  ggplot() + geom_line(aes(x=index, y=value), color="blue")
g1 + g2
```

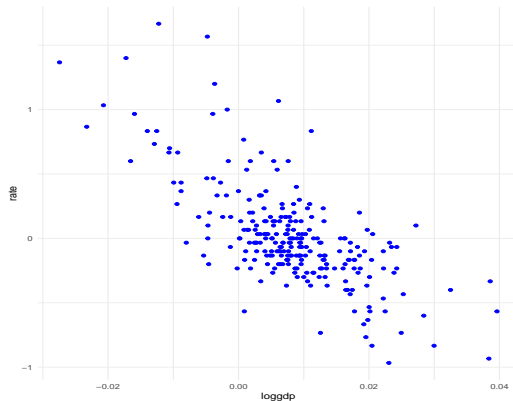


Série temporelle multivariée

PIB et taux de chômage aux USA : Corrélation

- Les deux séries présentent une corrélation instantanée négative

```
diff(gdpunemp) %>% as.data.frame() %>%  
  ggplot() + geom_point(aes(x=loggdp, y=rate), color="blue")
```



Série temporelle multivariée

Processus VAR

- Les modèles VAR (Vector Autoregression) sont l'extension des modèles AR à des séries multivariées
- L'évolution d'une série est modélisée par ses valeurs passées et celles des autres séries
- La série multivariée $y_t = (y_{1t}, \dots, y_{kt}, \dots, y_{Kt})$, $k = 1, \dots, K$ contient K variables
- Un processus $VAR(p)$ est défini par :

$$y_t = A_1 y_{t-1} + \dots + A_p y_{t-p} + \mu_t$$

avec $E(\mu_t) = 0$

- A_i est une matrice de $(K \times K)$ coefficients, $i = 1, \dots, p$

Série temporelle multivariée

Stationarité

- La série multivariée y_t est stationnaire (covariance stationary) si :
 - Son espérance est constante : $E(y_t) = \mu$, $\mu = (\mu_1, \mu_2, \dots, \mu_K)^T$
 - La covariance $cov(X_{t_1 i}, X_{t_2 j})$ est fonction uniquement de $t_2 - t_1$ pour chaque i et j

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) (1)

- Exemple tiré de [8] : taux de croissances trimestriels du PIB, en pourcent, au Royaume-Uni, Canada, et USA du 2ème trimestre 1980 au 2ème trimestre 2011
- Données CVS (données Federal Reserve Bank, St. Louis)
- Le PIB est exprimé en millions (monnaie nationale)

```
varpib <- read.table("/home/alex/Devel/Cours-R-TS/data/q-gdp-ukcaus.txt",header=T)
varpib <- log(varpib[,3:5])
head(varpib)

##          uk          ca          us
## 1 12.05778 13.34518 15.59190
## 2 12.03978 13.34300 15.57119
## 3 12.03788 13.34265 15.56933
## 4 12.02683 13.35382 15.58766
## 5 12.02006 13.37849 15.60822
## 6 12.02211 13.38773 15.60021
```

- Le taux de croissance représente les séries intégrées du log(PIB)

```
varpib_ts <- ts(varpib, start=c(1980,1), frequency=4)
pibgr <- diff(varpib_ts)*100
head(pibgr)

##          uk          ca          us
## 1980 Q2 -1.8005431 -0.21806940 -2.0708851
## 1980 Q3 -0.1897181 -0.03497379 -0.1867866
## 1980 Q4 -1.1052465  1.11694674  1.8336549
## 1981 Q1 -0.6770084  2.46719071  2.0557195
## 1981 Q2  0.2051468  0.92410692 -0.8007733
## 1981 Q3  1.3479031 -0.70853560  1.2078892
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) (2)

- Les séries intégrées du log(PIB) représentent les taux de croissance

```
varpib_ts <- ts(varpib, start=c(1980,1), frequency=4)
pibgr <- diff(varpib_ts)*100
head(pibgr)
```

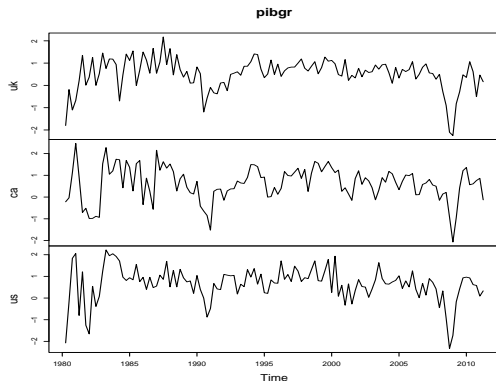
		uk	ca	us
##	1980 Q2	-1.8005431	-0.21806940	-2.0708851
##	1980 Q3	-0.1897181	-0.03497379	-0.1867866
##	1980 Q4	-1.1052465	1.11694674	1.8336549
##	1981 Q1	-0.6770084	2.46719071	2.0557195
##	1981 Q2	0.2051468	0.92410692	-0.8007733
##	1981 Q3	1.3479031	-0.70853560	1.2078892

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) (3)

■ Chronogramme des trois séries intégrées (taux de croissance)

```
plot(pibgr)
```



Modèle VAR

Test de stationarité sur les séries intégrée

■ On vérifie que la série est stationnaire après différenciation

```
summary(ur.df(pibgr[, "uk"], type = "drift", lags = 1))

##
## #####
## # Augmented Dickey-Fuller Test Unit Root Test #
## #####
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.84751 -0.28775  0.07458  0.32909  1.38476
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.21044    0.06686   3.148  0.00208 **
## z.lag.1       -0.37640    0.08203  -4.588  1.11e-05 ***
## z.diff.lag    -0.15066    0.08589  -1.754  0.08195 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5584 on 120 degrees of freedom
## Multiple R-squared:  0.2402, Adjusted R-squared:  0.2275
## F-statistic: 18.97 on 2 and 120 DF,  p-value: 6.964e-08
##
##
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Estimation (1)

- On utilise la librairie **vars** (voir l'article [ici](#))

```
library("vars")

## Le chargement a nécessité le package : MASS
##
## Attachement du package : 'MASS'
## L'objet suivant est masqué depuis 'package:patchwork':
##
##      area
## L'objet suivant est masqué depuis 'package:dplyr':
##
##      select

pibgr.VAR <- VAR(pibgr, p=2)
pibgr.VAR

##
## VAR Estimation Results:
## =====
##
## Estimated coefficients for equation uk:
## =====
## Call:
## uk = uk.l1 + ca.l1 + us.l1 + uk.l2 + ca.l2 + us.l2 + const
##
##      uk.l1      ca.l1      us.l1      uk.l2      ca.l2      us.l2      const
## 0.39306691 0.10310572 0.05213660 0.05660120 0.10552241 0.01889462 0.12581630
##
##
## Estimated coefficients for equation ca:
## =====
## Call:
## ca = ca.l1 + uk.l1 + us.l1 + ca.l2 + uk.l2 + us.l2 + const
##
##      ca.l1      uk.l1      us.l1      ca.l2      uk.l2      us.l2      const
## 0.14111111 0.14111111 0.14111111 0.14111111 0.14111111 0.14111111 0.14111111
```


Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Estimation (2)

- La méthode `summary` permet d'obtenir les estimations pour chaque série
- Pour la série `uk`, seul le paramètre associé à `uk.l1` (lag 1) est significatif

```
summary(pibgr.VAR)$varresult$uk
##
## Call:
## lm(formula = y ~ -1 + ., data = datamat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85491 -0.23752  0.05079  0.33566  1.31252
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.39307    0.09342   4.208 5.11e-05 ***
## ca.l1  0.10311    0.09838   1.048   0.297
## us.l1  0.05214    0.09113   0.572   0.568
## uk.l2  0.05660    0.09237   0.613   0.541
## ca.l2  0.10552    0.08756   1.205   0.231
## us.l2  0.01889    0.09382   0.201   0.841
## const  0.12582    0.07266   1.731   0.086 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5473 on 116 degrees of freedom
## Multiple R-squared:  0.3829, Adjusted R-squared:  0.3509
## F-statistic: 11.99 on 6 and 116 DF, p-value: 1.907e-10
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Estimation (3)

■ Estimations pour l'équation de la série ca

```
summary(pibgr.VAR)$varresult$ca
##
## Call:
## lm(formula = y ~ -1 + ., data = datamat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.51554 -0.31867  0.04956  0.34149  1.57007
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.351314   0.094917   3.701 0.000330 ***
## ca.l1  0.338142   0.099963   3.383 0.000979 ***
## us.l1  0.469094   0.092589   5.066 1.55e-06 ***
## uk.l2 -0.191350   0.093856  -2.039 0.043747 *
## ca.l2 -0.174833   0.088964  -1.965 0.051780 .
## us.l2 -0.008678   0.095326  -0.091 0.927624
## const  0.123158   0.073829   1.668 0.097984 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.556 on 116 degrees of freedom
## Multiple R-squared:  0.5238, Adjusted R-squared:  0.4992
## F-statistic: 21.27 on 6 and 116 DF, p-value: < 2.2e-16
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Estimation (4)

■ Estimations pour l'équation de la série us

```
summary(pibgr.VAR)$varresult$us
##
## Call:
## lm(formula = y ~ -1 + ., data = datamat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.18937 -0.27457  0.03623  0.32495  1.56051
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.49070     0.10502   4.672 8.08e-06 ***
## ca.l1  0.24000     0.11060   2.170 0.032053 *
## us.l1  0.23564     0.10245   2.300 0.023226 *
## uk.l2 -0.31196     0.10385  -3.004 0.003265 **
## ca.l2 -0.13118     0.09843  -1.333 0.185259
## us.l2  0.08531     0.10547   0.809 0.420253
## const  0.28956     0.08169   3.545 0.000568 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6152 on 116 degrees of freedom
## Multiple R-squared:  0.3606, Adjusted R-squared:  0.3276
## F-statistic: 10.9 on 6 and 116 DF, p-value: 1.325e-09
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Selection du modèle (1)

- La fonction **VARselect** propose différents critères pour la sélection du modèle

```
VARselect(pibgr, lag.max = 10)

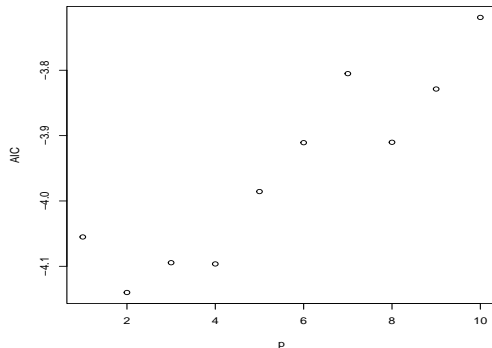
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      1      1      2
##
## $criteria
##           1           2           3           4           5           6
## AIC(n) -4.05509087 -4.1400600 -4.09449192 -4.09635510 -3.98562143 -3.91082343
## HQ(n)  -3.93883130 -3.9366058 -3.80384298 -3.71851148 -3.52058313 -3.35859045
## SC(n)  -3.76866317 -3.6388115 -3.37842267 -3.16546507 -2.83991063 -2.55029185
## FPE(n)  0.01733536 0.0159291 0.01668624 0.01668169 0.01868243 0.02020848
##           7           8           9          10
## AIC(n) -3.80517918 -3.91020870 -3.82872219 -3.71917784
## HQ(n)  -3.16575152 -3.18358636 -3.01490516 -2.81816614
## SC(n)  -2.22982683 -2.12003557 -1.82372828 -1.49936317
## FPE(n)  0.02257616 0.02046461 0.02239803 0.02526843
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Selection du modèle (2)

- Selon l'AIC, la valeur optimale est $p = 2$

```
plot(VARselect(pibgr, lag.max = 10)$criteria[1,], xlab='p', ylab='AIC')
```



Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Diagnostics (1)

- La fonction `serial.test` de la librairie `vars` calcul un test de Portmanteau multivarié

```
pibgr.VAR.serial <- serial.test(pibgr.VAR, type="PT.asymptotic")
pibgr.VAR.serial

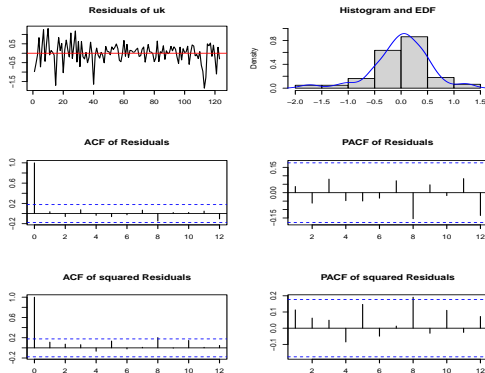
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object pibgr.VAR
## Chi-squared = 124.51, df = 126, p-value = 0.5207
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Diagnostics (2)

- La méthode générique `plot` pour les objets renvoyés par la fonction `serial.test` permet de visualiser les autocorrélations des résidus

```
plot(pibgr.VAR.serial, names="uk")
```

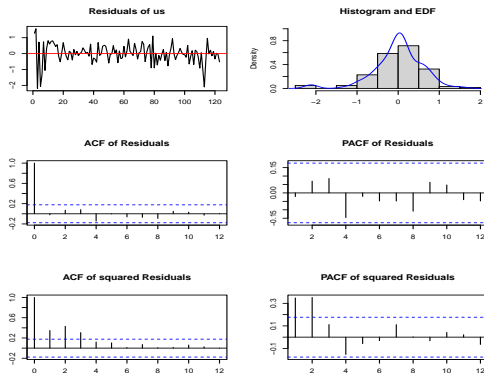


Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Diagnostics (3)

■ Résidus pour la série us

```
plot(pibgr.VAR.serial, names="us")
```



Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Diagnostics (4)

■ Test de la normalité des résidus

```
pibgr.VAR.norm <- normality.test(pibgr.VAR, multivariate.only = TRUE)
pibgr.VAR.norm

## $JB
##
## JB-Test (multivariate)
##
## data: Residuals of VAR object pibgr.VAR
## Chi-squared = 44.251, df = 6, p-value = 6.592e-08
##
##
## $Skewness
##
## Skewness only (multivariate)
##
## data: Residuals of VAR object pibgr.VAR
## Chi-squared = 17.229, df = 3, p-value = 0.000634
##
##
## $Kurtosis
##
## Kurtosis only (multivariate)
##
## data: Residuals of VAR object pibgr.VAR
## Chi-squared = 27.021, df = 3, p-value = 5.827e-06
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Diagnostics (5)

- Test de l'homoscédasticité des résidus (test ARCH, HAMILTON [9] et LÜTKEPOHL [10])

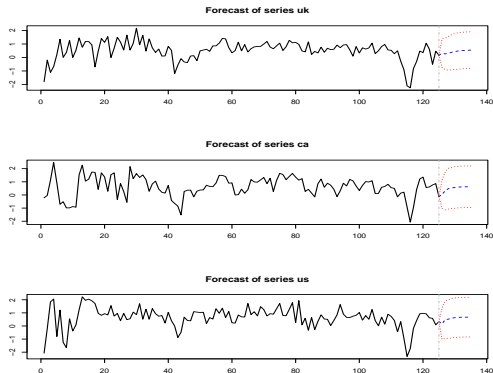
```
pibgr.VAR.arch <- arch.test(pibgr.VAR, lags.multi = 5, multivariate.only = TRUE)
pibgr.VAR.arch
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object pibgr.VAR
## Chi-squared = 275.77, df = 180, p-value = 5.661e-06
```

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Prédiction

- On utilise la méthode générique `predict` pour la projection

```
plot(predict(pibgr.VAR))
```

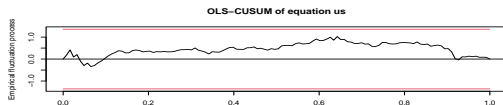
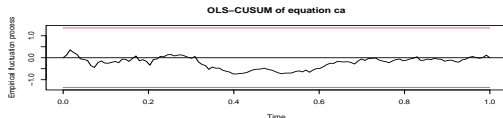
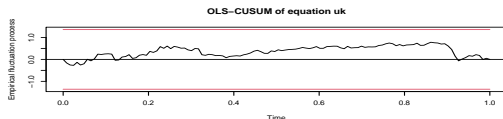


Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Test structurel (1)

- La librairie `vars` propose également un test de stabilité structurelle basé sur la fonction `efp` (empirical fluctuation process), librairie `strucchange`, voir ZEILEIS et al. [11])
- La fonction `stability` permet d'appliquer les différents types de test à chacune des équations du modèle VAR

```
reccusum <- stability(pibgr.VAR, type = "OLS-CUSUM")
plot(reccusum)
```

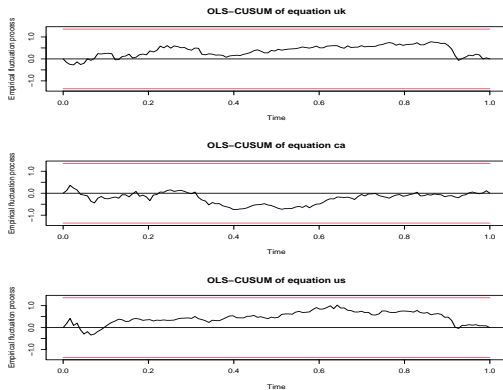


Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Test structurel (2)

■ Le test ne détecte aucune instabilité structurelle

```
reccusum <- stability(pibgr.VAR, type = "OLS-CUSUM")
plot(reccusum)
```



Modèle VAR

Test de causalité

- Le **test de Granger** est utilisé pour la détection de **causalités** entre variables
- Attention, le test de causalité ne permet pas de conclure à une causalité à proprement parler (voir par exemple PFAFF [4])
- On parle de **causalité au sens de Granger** (Granger causality) si la connaissance du passé d'une série temporelle y_{1t} entraîne une prévision de y_{2t} distincte de celle fondée uniquement sur le passé de y_{2t} , i.e. y_{1t} aide à prédire y_{2t}
- On peut également réaliser un test de causalité instantannée (instantaneous causality test, voir PFAFF [4])

Modèle VAR

Taux de croissance du PIB (UK, Canada, US) : Test de causalité

- La fonction `causality` de la librairie `vars` permet d'effectuer le test de Granger et le test de causalité immédiate
- On indique la liste des variables de la série dont on veut tester la causalité de Granger

```
causality(pibgr.VAR, cause=c("us", "ca"))  
  
## $Granger  
##  
##   Granger causality H0: ca us do not Granger-cause uk  
##  
## data:  VAR object pibgr.VAR  
## F-Test = 2.2372, df1 = 4, df2 = 348, p-value = 0.06467  
##  
##  
## $Instant  
##  
##   H0: No instantaneous causality between: ca us and uk  
##  
## data:  VAR object pibgr.VAR  
## Chi-squared = 6.4042, df = 2, p-value = 0.04068
```

Modèle VAR

Données Canada (1)

- Les données sont fournies avec la librairie `vars` (voir l'article [ici](#))
- Les séries utilisées représentent des indicateurs macro-économiques du marché du travail au Canada (données OCDE) :
 - `prod` : productivité du travail (log différence entre le GDP et nombre d'actifs)
 - `e` : nombre d'actifs (employment)
 - `U` : taux de chômage (unemployment rate)
 - `rw` : log de l'index des salaires réels (real wage index)
- Les séries s'étendent du premier trimestre 1980 aux quatrième trimestre 2004

Modèle VAR

Données Canada (2)

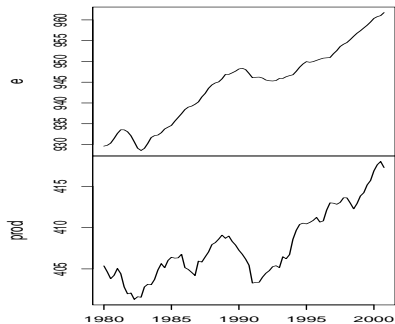
- On charge la librairie et les données (objet `mts`)

```
data("Canada")
window(Canada, start=c(1980,1), end=c(1981,4))

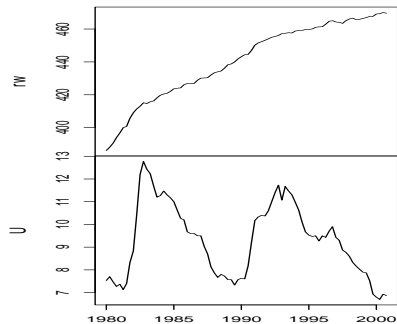
##           e      prod      rw      U
## 1980 Q1 929.6105 405.3665 386.1361 7.53
## 1980 Q2 929.8040 404.6398 388.1358 7.70
## 1980 Q3 930.3184 403.8149 390.5401 7.47
## 1980 Q4 931.4277 404.2158 393.9638 7.27
## 1981 Q1 932.6620 405.0467 396.7647 7.37
## 1981 Q2 933.5509 404.4167 400.0217 7.13
## 1981 Q3 933.5315 402.8191 400.7515 7.40
## 1981 Q4 933.0769 401.9773 405.7335 8.33
```

Modèle VAR

Données Canada (3)



Canada



Modèle VAR

Données Canada (4)

- Le modèle VAR ne permet de modéliser que des séries **stationnaires**
- Un test de stationarité ADF (Augmented Dickey-Fuller) est réalisé sur les 4 séries
- Toutes les séries sont rendues stationnaires avec une intégration d'ordre 1

Modèle VAR

Test de stationarité sur la série prod (1)

- La série prod présente une tendance, on utilise `type='trend'`
 - Hypothèse nulle H_0 : la série est **non stationnaire avec dérive**
 - Hypothèse alternative H_1 : la série est **stationnaire avec trend déterministe**

```
prod.adf1 <- ur.df(Canada[, "prod"], type = 'trend', lags = 2)
```

- Valeur de la statistique du test

```
attr(prod.adf1, "teststat")
##          tau3      phi2      phi3
## statistic -1.987512 2.300006 2.381689
```

- Valeurs critiques

```
attr(prod.adf1, "cval")
##      1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

- La valeur tau3 est supérieure à la valeur critique à 10%, on accepte H_0 , la série peut être rendue stationnaire par différenciation

Modèle VAR

Test de stationarité sur la série prod intégrée (1)

- On vérifie que la série est stationnaire après différentiation

```
prod.diff <- diff(Canada[, "prod"])
prod.diff.adf1 <- ur.df(prod.diff, type = "drift", lags = 1)
```

- Valeur de la statistique du test

```
attr(prod.diff.adf1, "teststat")
##          tau2      phi1
## statistic -5.160425 13.31839
```

- Valeurs critiques

```
attr(prod.diff.adf1, "cval")
##      1pct  5pct 10pct
## tau2 -3.51 -2.89 -2.58
## phi1  6.70  4.71  3.86
```

- La valeur de tau2 est inférieure à la valeur critique à 1%, on rejete H_0 et on conclut à la stationarité de la série

Modèle VAR

Test de stationnarité sur la série prod intégrée (2)

■ Test de stationnarité avec la fonction `adf.test`

```
adf.test(diff(Canada[, "prod"]), k = 1)
## Warning in adf.test(diff(Canada[, "prod"]), k = 1): p-value smaller than printed p-value
##
## Augmented Dickey-Fuller Test
##
## data: diff(Canada[, "prod"])
## Dickey-Fuller = -5.1952, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

Modèle VAR

Données Canada : Sélection du lag

- La longueur optimale du lag est déterminée avec la fonction `VARselect`
- La valeur optimale selon le critère AIC est de 3

```
VARselect(Canada, lag.max = 8, type = "both")

## $selection
## AIC(n)  HQ(n)  SC(n)  FPE(n)
##      3      2      1      3
##
## $criteria
##           1           2           3           4           5
## AIC(n) -6.272579064 -6.636669705 -6.771176872 -6.634609210 -6.398132246
## HQ(n)  -5.978429449 -6.146420347 -6.084827770 -5.752160366 -5.319583658
## SC(n)  -5.536558009 -5.409967947 -5.053794411 -4.426546046 -3.699388378
## FPE(n)  0.001889842  0.001319462  0.001166019  0.001363175  0.001782055
##           6           7           8
## AIC(n) -6.307704843 -6.070727259 -6.06159685
## HQ(n)  -5.033056512 -4.599979185 -4.39474903
## SC(n)  -3.118280272 -2.390621985 -1.89081087
## FPE(n)  0.002044202  0.002768551  0.00306012
```

Modèle VAR

Données canada : Estimation du modèle

- On commence par un modèle d'ordre $p = 1$
- L'argument `type = 'both'` indique qu'on inclut la constante et la tendance dans le modèle

```
p1ct <- VAR(Canada, p = 1, type = 'both')
p1ct

##
## VAR Estimation Results:
## =====
##
## Estimated coefficients for equation e:
## =====
## Call:
## e = e.l1 + prod.l1 + rw.l1 + U.l1 + const + trend
##
##           e.l1      prod.l1      rw.l1      U.l1      const
## 1.23892283    0.19465028   -0.06776277    0.62301475  -278.76121138
##      trend
## -0.04066045
##
##
## Estimated coefficients for equation prod:
## =====
## Call:
## prod = e.l1 + prod.l1 + rw.l1 + U.l1 + const + trend
##
##           e.l1      prod.l1      rw.l1      U.l1      const      trend
## 0.01291155    0.96313671  -0.03909399    0.21108918  16.24340747    0.04613085
##
##
##
```


Modèle VAR

Données canada : Equation pour la série e

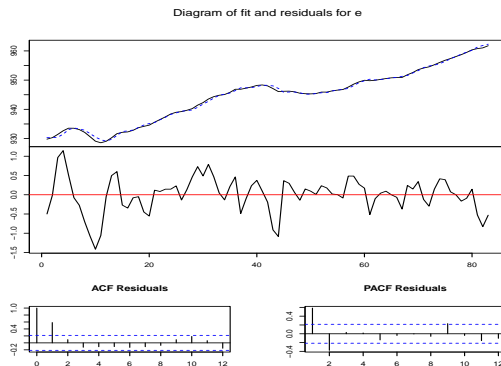
```
summary(pict, equation = "e")

##
## VAR Estimation Results:
## =====
## Endogenous variables: e, prod, rw, U
## Deterministic variables: both
## Sample size: 83
## Log Likelihood: -207.525
## Roots of the characteristic polynomial:
## 0.9504 0.9504 0.9045 0.7513
## Call:
## VAR(y = Canada, p = 1, type = "both")
##
##
## Estimation results for equation e:
## =====
## e = e.l1 + prod.l1 + rw.l1 + U.l1 + const + trend
##
##           Estimate Std. Error t value Pr(>|t|)
## e.l1      1.23892    0.08632   14.353 < 2e-16 ***
## prod.l1    0.19465    0.03612    5.389 7.49e-07 ***
## rw.l1     -0.06776    0.02828   -2.396 0.018991 *
## U.l1       0.62301    0.16927    3.681 0.000430 ***
## const    -278.76121   75.18295   -3.708 0.000392 ***
## trend     -0.04066    0.01970   -2.064 0.042378 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.4701 on 77 degrees of freedom
## Multiple R-Squared: 0.9975, Adjusted R-squared: 0.9973
## F-statistic: 6088 on 5 and 77 DF, p-value: < 2.2e-16
##
##
```

Modèle VAR

Données canada : Valeurs prédites et résidus

```
plot(pict, names = "e")
```



Modèle VAR

Données canada : Diagnostics

```
ser11 <- serial.test(p1ct, lags.pt = 16, type = "PT.asymptotic")
ser11$serial

##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object p1ct
## Chi-squared = 233.5, df = 240, p-value = 0.606
```

Modèle VAR

Exercices

- Modélisez l'évolution conjointe des PIB du Royaume-Uni, du Canada, des USA et de la France à l'aide d'un modèle VAR

Section 13

Cointégration et modèle à correction d'erreur

Cointégration

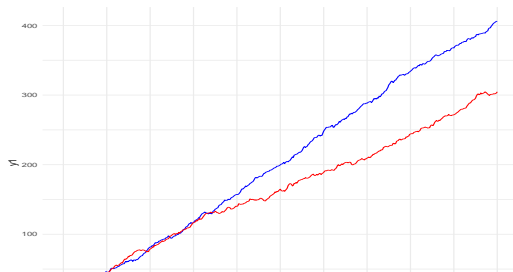
Introduction

- Nous abordons la modélisation de séries temporelles multivariées lorsque une ou plusieurs séries sont intégrées (d'ordre 1), i.e. non-stationnaires

Régression fallacieuse

- Un modèle de régression linéaire impliquant des séries non-stationnaires de type stochastique (difference stationary data) produit des résultats erronés (voir par exemple PFAFF [4])
- Le code suivant simule deux marches aléatoires avec dérive

```
library(lmtest)
set.seed(123456)
e1 <- rnorm(500)
e2 <- rnorm(500)
trd <- 1:500
y1 <- 0.8 * trd + cumsum(e1)
y2 <- 0.6 * trd + cumsum(e2)
spdata <- data.frame(index=1:500, y1,y2)
ggplot(spdata) +
  geom_line(aes(x=index, y=y1), color="blue") +
  geom_line(aes(x=index, y=y2), color="red")
```



Régression fallacieuse (2)

- Le modèle de régression suivant indique que les coefficients sont significativement différents de 0, et le R^2 est proche de 1

```
sr.reg <- lm(y1 ~ y2)
sr.dw <- dwtest(sr.reg)$statistic
summary(sr.reg)

##
## Call:
## lm(formula = y1 ~ y2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30.6541 -11.5262  0.3589  11.1423  31.0058
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -29.326969   1.367155  -21.45  <2e-16 ***
## y2           1.440787   0.007519  191.62  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.71 on 498 degrees of freedom
## Multiple R-squared:  0.9866, Adjusted R-squared:  0.9866
## F-statistic: 3.672e+04 on 1 and 498 DF,  p-value: < 2.2e-16
```

- Il s'agit d'un exemple de régression fallacieuse (**spurious regression**)

Régression fallacieuse (3)

- Pour éviter ce problème, on pourrait utiliser les séries intégrées dans la régression
- Cependant, de nouveaux problèmes apparaissent Pfaff-2008 :
 - 1 La différenciation atténue les fortes valeurs positives d'autocorrélation des résidus, affectant l'inférence sur les coefficients de la régression
 - 2 La validation des théories économiques basées sur les relations de long-terme entre niveaux des variables est compliquée si on utilise les séries différenciées

Cointégration

Introduction

- Pour éviter ce problème, on pourrait utiliser les séries intégrées dans la régression
- Cependant, de nouveaux problèmes apparaissent Pfaff-2008 :
 - 1 La différenciation atténue les fortes valeurs positives d'autocorrélation des résidus, affectant l'inférence sur les coefficients de la régression
 - 2 L'analyse des relations d'équilibre à long terme entre niveaux des variables est compliquée si on utilise les séries différenciées

Cointégration

Définition

- Le concept de **cointégration** introduit par Engle-1987 permet l'étude des relations stables à long terme de variables non-stationnaires
- Deux ou plusieurs séries chronologiques cointégrées partagent une dérive stochastique commune, leur combinaison linéaire donne lieu à une série stationnaire
- Les composants du vecteur y_t sont cointégrés d'ordre d, b (noté $y_t \sim CI(d, b)$) si
 - 1 Tous les composants de y_t sont intégrés d'ordre d ($I(d)$)
 - 2 Un vecteur $\alpha (\neq 0)$ existe tel que $z_t = \alpha' y_t \sim I(db), b > 0$
- Le vecteur *alpha* est appelé vecteur de cointégration

Cointégration

Test de cointégration

- Si deux séries non-stationnaires paraissent posséder des caractéristiques communes, on peut tester l'hypothèse d'une composante non-stationnaire commune au moyen d'un test de cointégration [cf.]Kleiber-2008
- Pour tester la cointégration, une première méthode en deux étapes a été proposée par Engle-1987 :
 - 1 Modèle de régression d'une série par l'autre
 - 2 Test de racine unitaire (ADF) sur les résidus

Section 14

Modèles à correction d'erreur

Modèle ECM

Introduction

- Pour modéliser une variable non stationnaire à l'aide d'une seule variable explicative elle aussi non stationnaire, on utilise un modèle à correction d'erreur (Error Correction Model, ECM)

Modèle ECM

Données USIncExp

- Nous allons utiliser les données USIncExp et suivre l'exemple de l'article présentant la librairie `strucchange` Zeileis-2002

```
USIncExp2 <- window(USIncExp, start = c(1985,12))
```

- La fonction de consommation (consumption function) est modélisée par un modèle ECM (cf. Hansen, 1992a) :

$$\Delta c_t = \beta_1 + \beta_2 e_{t-1} + \beta_3 \Delta i_t + \mu_t \quad (1)$$

$e_t = c_t - \alpha_1 - \alpha_2 i_t$ (2) où c_t représente les dépenses de consommation et i_t le revenu

Modèle ECM

Données USIncExp : Equation de cointégration

- L'équation de cointégration (2) est estimée à l'aide d'un modèle OLS

$$c_t = \alpha_1 + \alpha_2 i_t + e_t$$

- Les résidus \hat{e}_t seront utilisés comme variable indépendante dans la modélisation de la fonction de consommation (1)

```
coint.res <- residuals(lm(expenditure ~ income, data = USIncExp2))
```


Modèle ECM

Données USIncExp : Préparation des données

- On utilise la fonction `lag` car les résidus dans l'équation (1) sont à $t - 1$
- On utilise la fonction `diff` pour intégrer les séries (obtenir Δc_t et Δi_t)

```
coint.res <- stats::lag(ts(coint.res, start = c(1985,12), freq = 12), k = -1)
USIncExp2 <- cbind(USIncExp2, diff(USIncExp2), coint.res)
USIncExp2 <- window(USIncExp2, start = c(1986,1), end = c(2001,2))
colnames(USIncExp2) <- c("income", "expenditure", "diff.income", "diff.expenditure", "coint.res")
window(USIncExp2, start=c(1986,1), end=c(1986,12))
```

	income	expenditure	diff.income	diff.expenditure	coint.res
## Jan 1986	3630.1	2829.3	6.5	11.0	27.510616
## Feb 1986	3647.7	2821.4	17.6	-7.9	33.081616
## Mar 1986	3674.8	2824.6	27.1	3.2	10.481554
## Apr 1986	3674.3	2838.6	-0.5	14.0	-8.953200
## May 1986	3686.6	2863.1	12.3	24.5	5.464415
## Jun 1986	3703.7	2869.3	17.1	6.2	19.691076
## Jul 1986	3721.3	2891.0	17.6	21.7	11.608630
## Aug 1986	3734.6	2909.9	13.3	18.9	18.608568
## Sep 1986	3752.0	2984.8	17.4	74.9	26.399998
## Oct 1986	3756.6	2947.5	4.6	-37.3	86.766982
## Nov 1986	3770.6	2945.6	14.0	-1.9	45.624921
## Dec 1986	3797.0	3016.9	26.4	71.3	32.031690

Modèle ECM

Données USIncExp : Estimation

- La variable dépendante est l'augmentation des dépenses et les variables indépendantes sont les résidus de cointegration et l'augmentation des revenus (plus une constante)

```
ecm.model <- diff.expenditure ~ coint.res + diff.income
summary(lm(ecm.model, data=USIncExp2))

##
## Call:
## lm(formula = ecm.model, data = USIncExp2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -70.555 -11.947  -0.528  10.452  55.576
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.82428     1.88918   9.435  < 2e-16 ***
## coint.res    -0.06843     0.03318  -2.063   0.0406 *
## diff.income   0.19001     0.04282   4.438 1.59e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.09 on 179 degrees of freedom
## Multiple R-squared:  0.1021, Adjusted R-squared:  0.09203
## F-statistic: 10.17 on 2 and 179 DF,  p-value: 6.542e-05
```

Modèle ECM

Données GermanM1

■ ECM for German M1 money demand (cf Kleiber-2008)

```
data("GermanM1")
```

```
LTW <- dm ~ dy2 + dR + dR1 + dp + m1 + y1 + R1 + season
```

Bibliographie I

- [1] Charles R. NELSON et Charles R. PLOSSER. "Trends and random walks in macroeconomic time series : Some evidence and implications". In : *Journal of Monetary Economics* 10.2 (1982), p. 139-162. ISSN : 0304-3932. DOI : [https://doi.org/10.1016/0304-3932\(82\)90012-5](https://doi.org/10.1016/0304-3932(82)90012-5). URL : <https://www.sciencedirect.com/science/article/pii/0304393282900125>.
- [2] Robert L. HETZEL. "The Federal Reserve System and Control of the Money Supply in the 1970s". In : *Journal of Money, Credit and Banking* 13.1 (fév. 1981), p. 31. ISSN : 0022-2879. DOI : [10.2307/1991806](https://doi.org/10.2307/1991806).
- [3] Walter KRÄMER et Harald SONNBERGER. *The Linear Regression Model Under Test*. Physica-Verlag HD, 1986. ISBN : 9783642958762. DOI : [10.1007/978-3-642-95876-2](https://doi.org/10.1007/978-3-642-95876-2).
- [4] Bernhard PFAFF. *Analysis of Integrated and Cointegrated Time Series with R*. Springer New York, 2008. ISBN : 9780387759678. DOI : [10.1007/978-0-387-75967-8](https://doi.org/10.1007/978-0-387-75967-8). URL : <https://www.semanticscholar.org/paper/92290935b077d8c917303037d35ce7e205404d9b>.

Bibliographie II

- [5] Anindya BANERJEE. *Co-integration, error correction, and the econometric analysis of non-stationary data*. Anindya Banerjee, Juan J. Dolado, John W. Galbraith, and David F. Hendry. *Advanced texts in econometrics*. Oxford : Clarendon Press. 329 p. ISBN : 0191521582.
- [6] Christian KLEIBER et Achim ZEILEIS. *Applied Econometrics with R. Use R!* Springer New York, 2008. ISBN : 9780387773186. DOI : [10.1007/978-0-387-77318-6](https://doi.org/10.1007/978-0-387-77318-6).
- [7] Jonathan D. CRYER et Kung-Sik CHAN. *Time Series Analysis : With Applications in R*. Springer New York, 2010. ISBN : 9780387759593. DOI : [10.1007/978-0-387-75959-3](https://doi.org/10.1007/978-0-387-75959-3).
- [8] R. TSAY. *Multivariate Time Series Analysis : With R and Financial Applications*. Wiley, 2014. URL : <https://www.semanticscholar.org/paper/d4c0cfa6bf6f3a17fad19d48e98781fdc7f4174d>.
- [9] James D. HAMILTON. *Time series analysis*. James D. Hamilton. [Nachdr.]09. Princeton, N. J. : Princeton Univ. Press, 1994. 799 p. ISBN : 0691042896.
- [10] Helmut LÜTKEPOHL. *New introduction to multiple time series analysis*. English. Berlin : Springer, 2005. ISBN : 3-540-40172-5.

Bibliographie III

- [11] Achim ZEILEIS et al. "Monitoring structural change in dynamic econometric models". In : *Journal of Applied Econometrics* 20.1 (jan. 2005), p. 99-121. ISSN : 1099-1255. DOI : [10.1002/jae.776](https://doi.org/10.1002/jae.776).