Analyse de séries temporelles avec R

Alexis Gabadinho

2023-10-05

Section 1

Environnement de travail (rappels)

Importation de fichiers csv (1)

- PIB et ses composants, valeurs aux prix courants - Source: INSEE
- On importe les données à partir du fichier csv
- A noter: dans le fichier csv, le séparateur décimal est un point virgule

```
pib <- read.csv("../data/pib_fr.csv", header=TRUE, 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
```

Importation de fichiers csv (1)

- P7 = Importations
- P3M = Dépenses de consommation des ménages
- P3 = Dépenses de consommation totale
- P51 = FBCF (formation brute de capital fixe total)
- P51M = FBCF (formation brute de capital fixe ménages)
- P6 = Exportations

Importation de fichiers csv (2)

```
summary(pib)
```

```
##    PERIODE               PIB               P7               P3M
##  Length:298         Min.   :  3.20   Min.   :  0.40   Min.   :  1.90
##  Class :character   1st Qu.: 22.73   1st Qu.:  2.90   1st Qu.: 12.25
##  Mode  :character   Median :200.50   Median : 43.85   Median :109.85
##                     Mean   :238.27   Mean   : 64.82   Mean   :125.30
##                     3rd Qu.:430.40   3rd Qu.:111.95   3rd Qu.:225.50
##                     Max.   :701.60   Max.   :268.70   Max.   :356.60
##       P3P              P31G              P32G              P3
##  Min.   : 0.10   Min.   :  0.20   Min.   : 0.20   Min.   :  2.40
##  1st Qu.: 0.30   1st Qu.:  2.10   1st Qu.: 1.60   1st Qu.: 16.27
##  Median : 2.80   Median : 26.20   Median :18.75   Median :157.55
##  Mean   : 4.42   Mean   : 34.56   Mean   :20.14   Mean   :184.41
##  3rd Qu.: 7.80   3rd Qu.: 63.25   3rd Qu.:35.83   3rd Qu.:332.40
##  Max.   :14.50   Max.   :106.30   Max.   :56.20   Max.   :533.70
##       P51S             P51B             P51G             P51M
##  Min.   : 0.40   Min.   :0.00   Min.   : 0.100   Min.   : 0.100
##  1st Qu.: 3.00   1st Qu.:0.10   1st Qu.: 1.200   1st Qu.: 1.625
##  Median :22.20   Median :0.95   Median : 8.600   Median :10.500
##  Mean   :28.82   Mean   :1.79   Mean   : 9.422   Mean   :12.892
##  3rd Qu.:48.12   3rd Qu.:3.10   3rd Qu.:17.200   3rd Qu.:23.125
##  Max.   :99.20   Max.   :8.00   Max.   :25.800   Max.   :40.300
##       P51P              P51               P54               P6
##  Min.   :0.0000   Min.   :  0.600   Min.   :-6.100   Min.   :  0.40
##  1st Qu.:0.0000   1st Qu.:  5.925   1st Qu.: 0.200   1st Qu.:  3.00
##  Median :0.3000   Median : 42.450   Median : 0.700   Median : 42.95
##  Mean   :0.4302   Mean   : 53.356   Mean   : 1.401   Mean   : 63.92
##  3rd Qu.:0.8000   3rd Qu.: 92.375   3rd Qu.: 2.200   3rd Qu.:114.22
##  Max.   :1.5000   Max.   :174.600   Max.   :11.800   Max.   :237.40
```

Importation de fichiers csv (1)

- Advance retail sales (commerce de détail) - données mensuelles
- 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.csv("../data/RSXFSN.csv", header=TRUE, sep=",", dec = ".")
head(retail)
```

```
##         DATE RSXFSN
## 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
```

- Convesion de la date

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

Le tidyverse

- Collection de librairies

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## 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
```

Importation de fichiers csv (2)

- Pour importer les données à partir du fichier csv dans un objet tibble
- read_csv2() uses ; for the field separator and , for the decimal point. This format is common in some European countries.

```
pib_tbl <- read_csv2("../data/pib_fr.csv")
```

```
## i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.

## Rows: 298 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr  (1): PERIODE
## dbl (15): PIB, P7, P3M, P3P, P31G, P32G, P3, P51S, P51B, P51G, P51M, P51P, P...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(pib_tbl)
```

```
## # A tibble: 6 x 16
##   PERIODE   PIB    P7   P3M   P3P  P31G  P32G    P3  P51S  P51B  P51G  P51M
##   <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1949T1    3.2   0.4   1.9   0.1   0.2   0.2   2.4   0.4     0   0.1   0.1
## 2 1949T2    3.2   0.4   2     0.1   0.3   0.2   2.5   0.4     0   0.1   0.1
## 3 1949T3    3.3   0.4   2.1   0.1   0.3   0.2   2.6   0.4     0   0.1   0.1
## 4 1949T4    3.4   0.4   2.1   0.1   0.3   0.3   2.6   0.4     0   0.1   0.1
## 5 1950T1    3.6   0.5   2.1   0.1   0.3   0.3   2.7   0.5     0   0.1   0.1
## 6 1950T2    3.8   0.5   2.2   0.1   0.3   0.3   2.8   0.5     0   0.1   0.1
## # i 4 more variables: P51P <dbl>, P51 <dbl>, P54 <dbl>, P6 <dbl>
```

Extraction de l'année

```
pib_tbl <- pib_tbl %>% mutate(ANNEE=substring(PERIODE, 1, 4), TRIMESTRE=substring(PERIODE, 6, 7))
pib_tbl %>% select("PIB", "ANNEE", "TRIMESTRE")
```

```
## # A tibble: 298 x 3
##       PIB ANNEE TRIMESTRE
##     <dbl> <chr> <chr>
##  1   3.2 1949  1
##  2   3.2 1949  2
##  3   3.3 1949  3
##  4   3.4 1949  4
##  5   3.6 1950  1
##  6   3.8 1950  2
##  7   4   1950  3
##  8   4.2 1950  4
##  9   4.4 1951  1
## 10   4.8 1951  2
## # i 288 more rows
```

Matrice de corrélation

```
datanum <- pib_tbl %>% select(-PERIODE, -ANNEE, -TRIMESTRE)
cormat <- round(cor(datanum),2)
cormat
```

```
##        PIB   P7  P3M  P3P P31G P32G   P3 P51S P51B P51G P51M P51P  P51  P54   P6
## PIB  1.00 0.98 1.00 0.99 1.00 1.00 1.00 0.99 0.96 0.99 0.99 0.99 1.00 0.58 0.99
## P7   0.98 1.00 0.98 0.99 0.99 0.97 0.98 0.99 0.99 0.96 0.99 0.99 0.99 0.64 1.00
## P3M  1.00 0.98 1.00 0.99 1.00 1.00 1.00 0.99 0.95 0.99 0.99 0.99 0.99 0.57 0.99
## P3P  0.99 0.99 0.99 1.00 1.00 0.99 0.99 0.99 0.98 0.97 0.99 1.00 0.99 0.60 1.00
## P31G 1.00 0.99 1.00 1.00 1.00 0.99 1.00 0.99 0.97 0.98 0.99 1.00 1.00 0.58 0.99
## P32G 1.00 0.97 1.00 0.99 0.99 1.00 1.00 0.99 0.95 0.99 0.98 0.98 0.99 0.55 0.98
## P3   1.00 0.98 1.00 0.99 1.00 1.00 1.00 0.99 0.96 0.99 0.99 0.99 1.00 0.57 0.99
## P51S 0.99 0.99 0.99 0.99 0.99 0.99 0.99 1.00 0.98 0.98 0.99 0.99 1.00 0.61 0.99
## P51B 0.96 0.99 0.95 0.98 0.97 0.95 0.96 0.98 1.00 0.93 0.97 0.98 0.97 0.65 0.98
## P51G 0.99 0.96 0.99 0.97 0.98 0.99 0.99 0.98 0.93 1.00 0.98 0.97 0.99 0.52 0.97
## P51M 0.99 0.99 0.99 0.99 0.99 0.98 0.99 0.99 0.97 0.98 1.00 0.99 1.00 0.59 0.99
## P51P 0.99 0.99 0.99 1.00 1.00 0.98 0.99 0.99 0.98 0.97 0.99 1.00 0.99 0.60 0.99
## P51  1.00 0.99 0.99 0.99 1.00 0.99 1.00 1.00 0.97 0.99 1.00 0.99 1.00 0.60 0.99
## P54  0.58 0.64 0.57 0.60 0.58 0.55 0.57 0.61 0.65 0.52 0.59 0.60 0.60 1.00 0.63
## P6   0.99 1.00 0.99 1.00 0.99 0.98 0.99 0.99 0.98 0.97 0.99 0.99 0.99 0.63 1.00
```

Heatmap

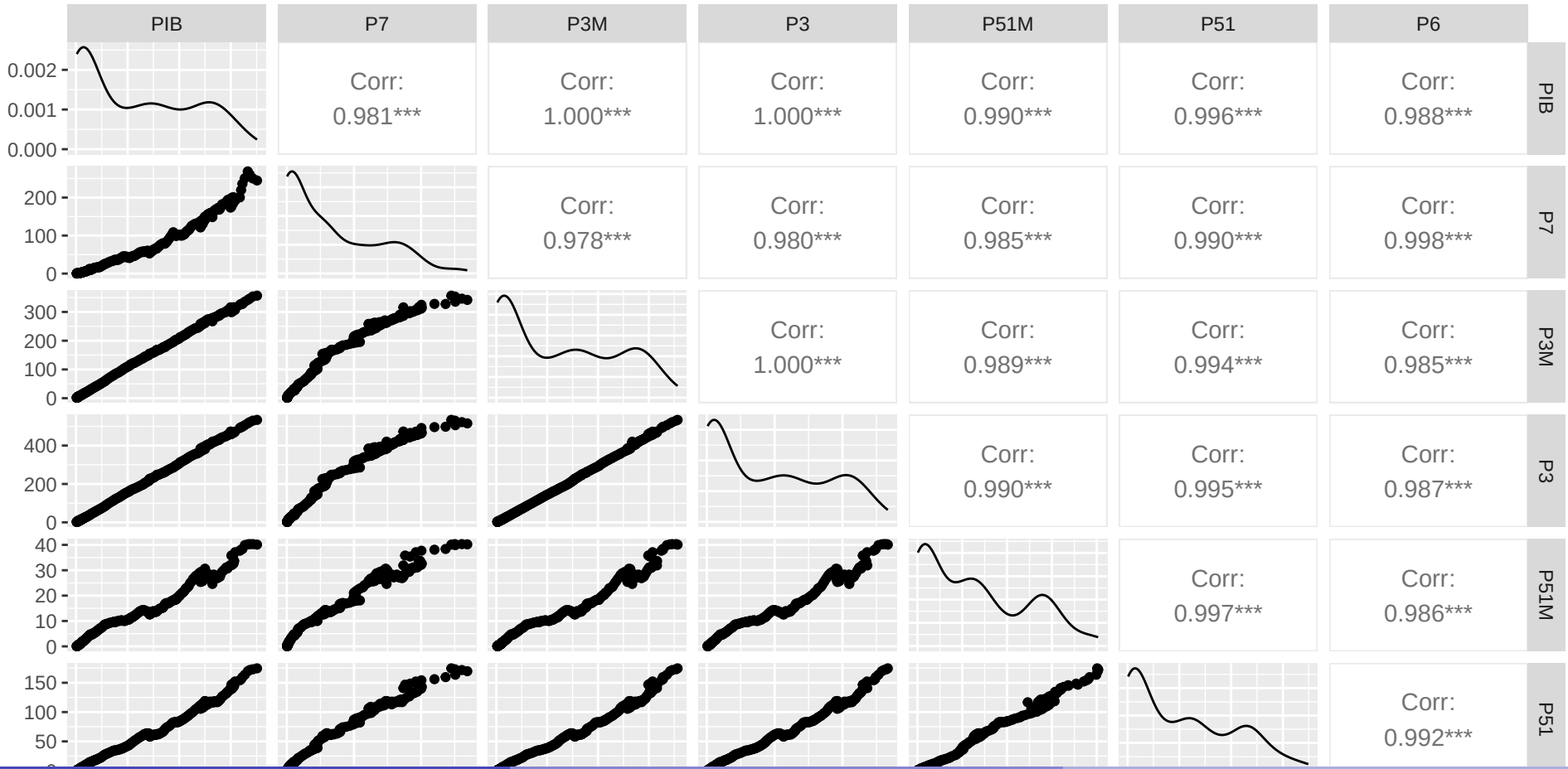
```
library(reshape2)
cormat %>% melt() %>% ggplot(aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```

Pairplot

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
pib_tbl %>% select(PIB, P7, P3M, P3, P51M, P51, P6) %>% ggpairs()
```



Section 2

Librairies spécialisées et structures de séries temporelles dans R

Séries temporelles avec R (1)

- La classe de base fournie par R pour représenter des séries temporelles s'appelle 'ts' (abréviation de l'anglais time series). Cette classe est définie dans le package stats.
- Elle concerne des séries temporelles qui sont échantillonnées à des périodes équidistantes dans le temps.

Séries temporelles avec R (2)

- Un objet de classe ts possède trois paramètres caractéristiques :
  - frequency désigne le 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 désigne la date de début de la série temporelle. Elle est exprimée comme un nombre unique ou comme un vecteur de deux entiers qui représentent respectivement une unité temporelle (comme une année) et une subdivision de cette unité (comme un mois ou un trimestre selon la valeur du paramètre frequency) ;
  - end désigne la date de fin de la série temporelle. Sa valeur est exprimée comme pour le paramètre start

Transformation des données pib en objet ts

- Pour les données pib, frequency=4 (trimestres)

```
pib_ts <- ts(pib['PIB'], frequency=4, start=c(1949,1))
class(pib_ts)
```

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

```
head(pib_ts, 12)
```

```
##       PIB
##  [1,] 3.2
##  [2,] 3.2
##  [3,] 3.3
##  [4,] 3.4
##  [5,] 3.6
##  [6,] 3.8
##  [7,] 4.0
##  [8,] 4.2
##  [9,] 4.4
## [10,] 4.8
## [11,] 5.0
## [12,] 5.4
```

Transformation des données pib en objet ts

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

```
retail_ts <- ts(retail['RSXFSN'], frequency=12, start=c(1992,1))
head(retail_ts, 36)
```

```
##       RSXFSN
##  [1,] 130683
##  [2,] 131244
##  [3,] 142488
##  [4,] 147175
##  [5,] 152420
##  [6,] 151849
##  [7,] 152586
##  [8,] 152476
##  [9,] 148158
## [10,] 155987
## [11,] 154824
## [12,] 191347
## [13,] 137020
## [14,] 134462
## [15,] 153025
## [16,] 158615
## [17,] 163519
## [18,] 162964
## [19,] 164590
## [20,] 163989
## [21,] 159298
## [22,] 163992
## [23,] 169980
## [24,] 206174
## [25,] 145276
```

Manipulation des objets ts

- La fonction window permet d'extraire une portion d'une série temporelle. Elle possède des arguments start et end pour indiquer les dates de début et de fin de la série extraite. On peut aussi utiliser l'argument optionnel frequency pour réétalonner la nouvelle série selon une fréquence différente. L'argument optionnel extend prend une valeur logique (TRUE ou FALSE) : il autorise l'extension d'une série temporelle à des dates qui ne figurent pas dans la série initiale

```
pib_ts_S21 <- window(pib_ts, start=2018)
print(pib_ts_S21, calendar=FALSE)
```

```
## Time Series:
## Start = c(2018, 1)
## End = c(2023, 2)
## Frequency = 4
##          PIB
##  [1,] 584.8
##  [2,] 588.5
##  [3,] 593.4
```

Manipulation des objets ts

Extraction de l'index

```
time(pib_ts_S21)
```

```
##          Qtr1    Qtr2    Qtr3    Qtr4
## 2018 2018.00 2018.25 2018.50 2018.75
## 2019 2019.00 2019.25 2019.50 2019.75
## 2020 2020.00 2020.25 2020.50 2020.75
## 2021 2021.00 2021.25 2021.50 2021.75
## 2022 2022.00 2022.25 2022.50 2022.75
## 2023 2023.00 2023.25
```

Manipulation des objets ts

```
library(lubridate)
as.numeric(time(pib_ts_S21))
```

```
##  [1] 2018.00 2018.25 2018.50 2018.75 2019.00 2019.25 2019.50 2019.75 2020.00
## [10] 2020.25 2020.50 2020.75 2021.00 2021.25 2021.50 2021.75 2022.00 2022.25
## [19] 2022.50 2022.75 2023.00 2023.25
```

Séries multiples

- Le package stats définit aussi une notion de série temporelle multiple.
- Ce sont des objets de classe mts (multiple time series) qui représentent simultanément plusieurs séries temporelles dont les observations correspondent au même découpage du temps : elles ont les mêmes paramètres start, end et frequency.

```
pib_ts <- ts(pib[2:ncol(pib)], frequency=4,start=c(1949,1))
window(pib_ts, start=2020)
```

```
##           PIB    P7   P3M  P3P  P31G P32G    P3 P51S P51B P51G P51M P51P   P51
## 2020 Q1 587.1 184.0 301.2 12.3  91.3 50.3 455.1 75.0  5.9 21.7 30.7  1.2 134.5
## 2020 Q2 528.7 148.5 267.4 11.6  91.6 49.6 420.2 66.9  4.8 19.3 24.6  1.2 116.7
## 2020 Q3 598.9 173.5 316.0 12.5  95.3 49.4 473.1 80.0  5.8 21.9 31.9  1.3 140.9
## 2020 Q4 602.2 176.1 299.0 12.5  98.0 50.4 459.9 81.9  5.8 22.0 35.8  1.3 146.8
## 2021 Q1 608.2 183.7 302.5 12.7  99.9 50.3 465.5 83.1  6.5 22.2 35.4  1.3 148.5
## 2021 Q2 616.5 192.1 307.4 12.9 101.2 51.0 472.5 84.5  6.8 22.4 37.1  1.3 152.2
## 2021 Q3 635.0 200.4 324.8 13.3 101.8 51.6 491.5 85.9  6.9 22.4 37.7  1.3 154.2
## 2021 Q4 640.0 219.8 328.0 13.6 102.3 52.3 496.1 86.8  7.0 22.9 38.1  1.3 156.1
## 2022 Q1 645.1 236.4 327.7 13.9 103.5 53.0 498.1 88.8  7.3 23.8 38.4  1.4 159.6
## 2022 Q2 654.2 250.9 335.6 14.1 102.4 53.9 506.1 90.6  7.5 24.4 39.9  1.4 163.7
## 2022 Q3 665.8 268.7 341.9 14.3 104.3 55.1 515.6 95.4  7.7 24.9 40.2  1.4 169.7
## 2022 Q4 673.2 260.6 346.2 14.4 105.7 55.6 521.9 96.9  7.8 25.3 40.3  1.4 171.7
## 2023 Q1 685.3 250.3 353.6 14.5 105.8 55.7 529.6 97.8  7.9 25.5 40.3  1.4 173.0
## 2023 Q2 701.6 244.9 356.6 14.5 106.3 56.2 533.7 99.2  8.0 25.8 40.1  1.5 174.6
##           P54    P6
## 2020 Q1   5.7 175.7
## 2020 Q2   9.7 130.6
```

Les objets tsibble

- On créé un objet tsibble avec la fonction as_tsibble()
- To coerce a data frame to tsibble, we need to declare key and index.
- Ici on créé une colonne QUARTER de type yearquarter contenant le trimestre qui va être utilisée automatiquement comme index
- Other columns can be considered as measured variables.

```
library(tsibble)

pib_tsbl <- pib_tbl %>% mutate(ANNEE=as.numeric(ANNEE), TRIMESTRE=as.numeric(TRIMESTRE)) %>%
  mutate(QUARTER=make_yearquarter(year=ANNEE, quarter=TRIMESTRE)) %>%
  select(-ANNEE, -TRIMESTRE, -PERIODE) %>% as_tsibble()
pib_tsbl
```

```
## # A tsibble: 298 x 16 [1Q]
##      PIB    P7   P3M   P3P  P31G  P32G    P3  P51S  P51B  P51G  P51M  P51P   P51
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1   3.2   0.4   1.9   0.1   0.2   0.2   2.4   0.4     0   0.1   0.1     0   0.6
##  2   3.2   0.4   2     0.1   0.3   0.2   2.5   0.4     0   0.1   0.1     0   0.6
##  3   3.3   0.4   2.1   0.1   0.3   0.2   2.6   0.4     0   0.1   0.1     0   0.6
##  4   3.4   0.4   2.1   0.1   0.3   0.3   2.6   0.4     0   0.1   0.1     0   0.7
##  5   3.6   0.5   2.1   0.1   0.3   0.3   2.7   0.5     0   0.1   0.1     0   0.7
##  6   3.8   0.5   2.2   0.1   0.3   0.3   2.8   0.5     0   0.1   0.1     0   0.7
##  7   4     0.5   2.4   0.1   0.3   0.3   3.1   0.5     0   0.1   0.1     0   0.7
##  8   4.2   0.6   2.5   0.1   0.3   0.3   3.2   0.5     0   0.1   0.1     0   0.7
##  9   4.4   0.6   2.7   0.1   0.3   0.3   3.4   0.6     0   0.1   0.1     0   0.8
## 10   4.8   0.7   2.8   0.1   0.4   0.3   3.6   0.6     0   0.1   0.2     0   0.9
## # i 288 more rows
```

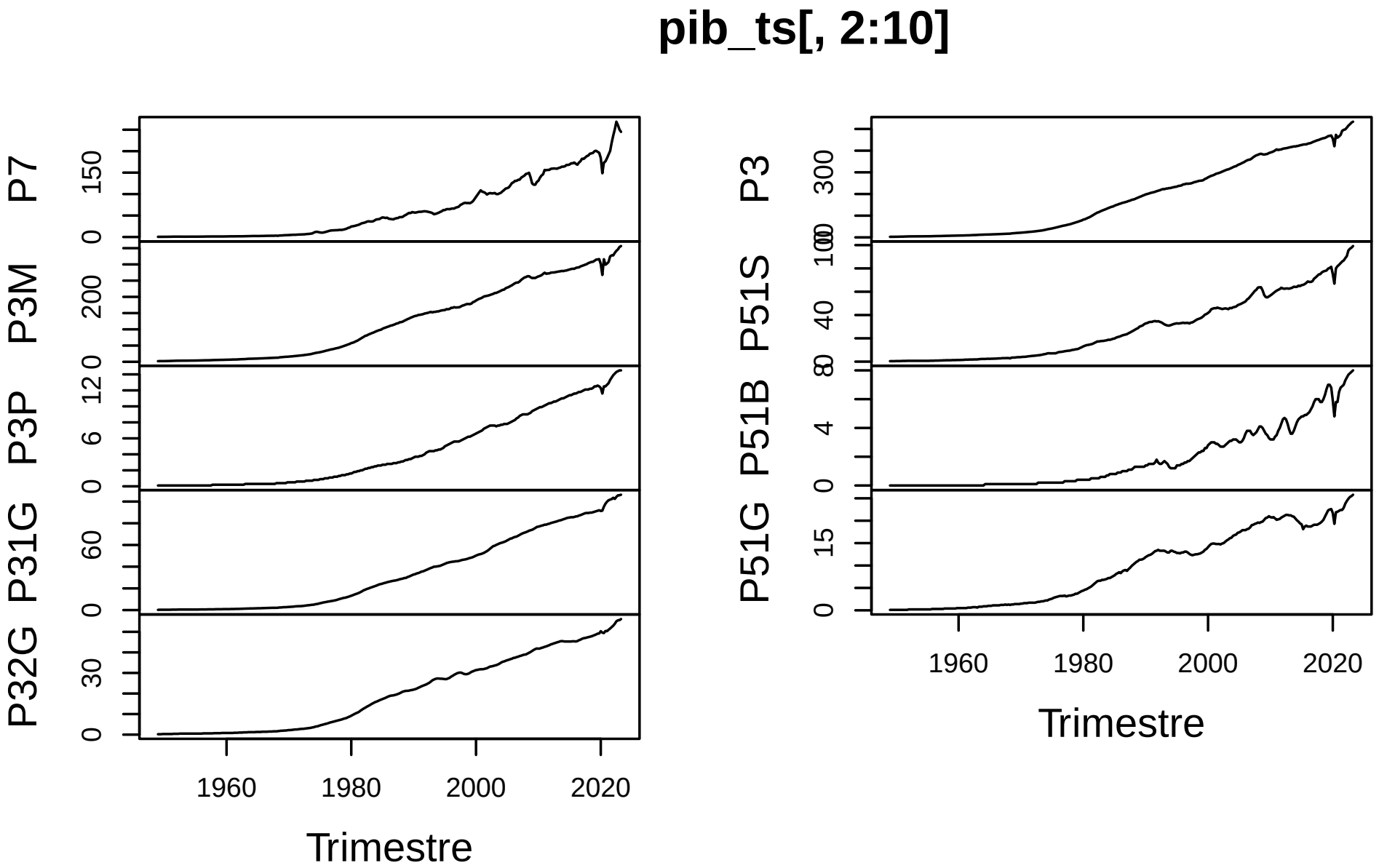
Librairies spécialisées

- feasts (Feature Extraction And Statistics for Time Series) provides a collection of tools for the analysis of time series data. The package name is an acronym comprising of its key features: .
- The package works with tidy temporal data provided by the tsibble package to produce time series features, decompositions, statistical summaries and convenient visualisations.
- These features are useful in understanding the behaviour of time series data, and closely integrates with the tidy forecasting workflow used in the fable package.

Section 3

Analyse descriptive et représentations graphiques

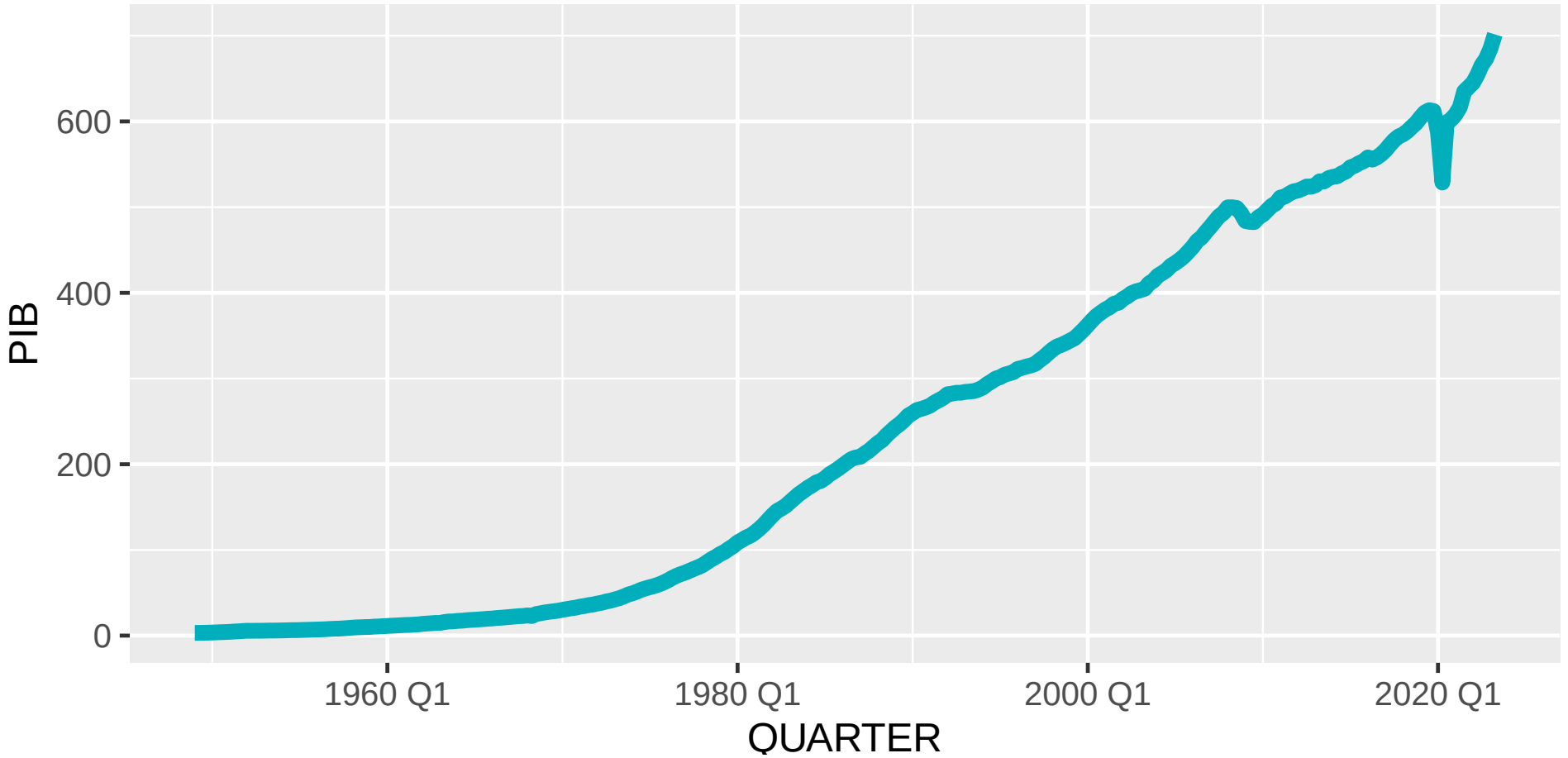
Représenter des séries temporelles



Représentation avec ggplot

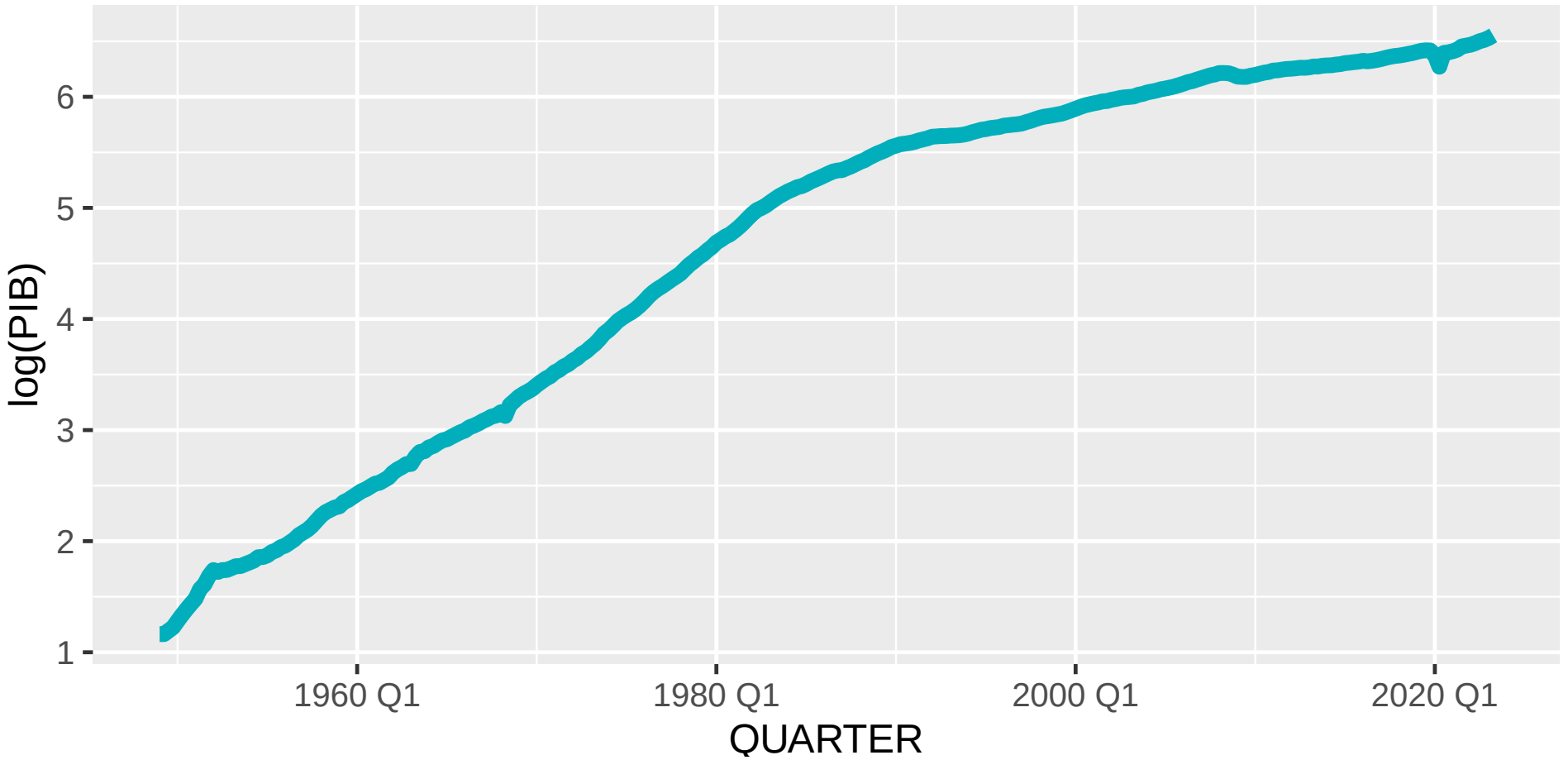
- Basic line plot

```
ggplot(data = pib_tsbl, aes(x = QUARTER, y = PIB))+
  geom_line(color = "#00AFBB", size = 2)
```



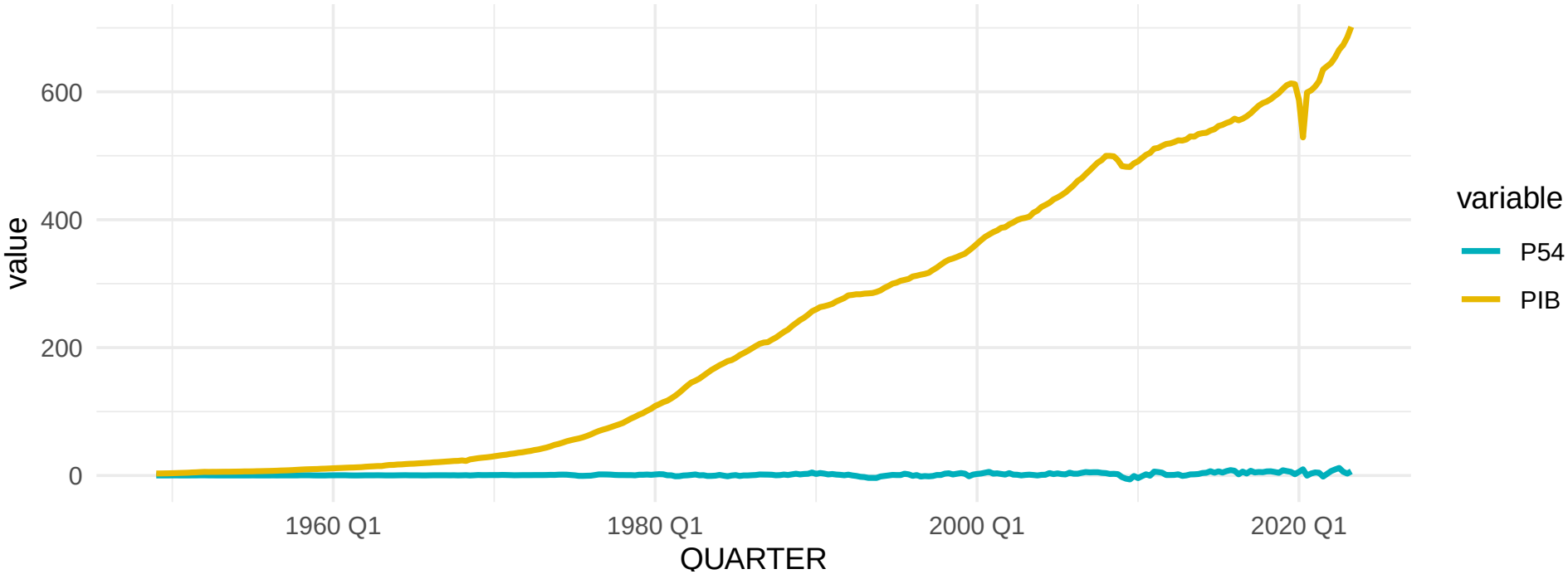
Représentation avec ggplot - Echelle logarithmique

```
# Basic line plot
ggplot(data = pib_tsbl, aes(x = QUARTER, y = log(PIB))) +
  geom_line(color = "#00AFBB", size = 2)
```



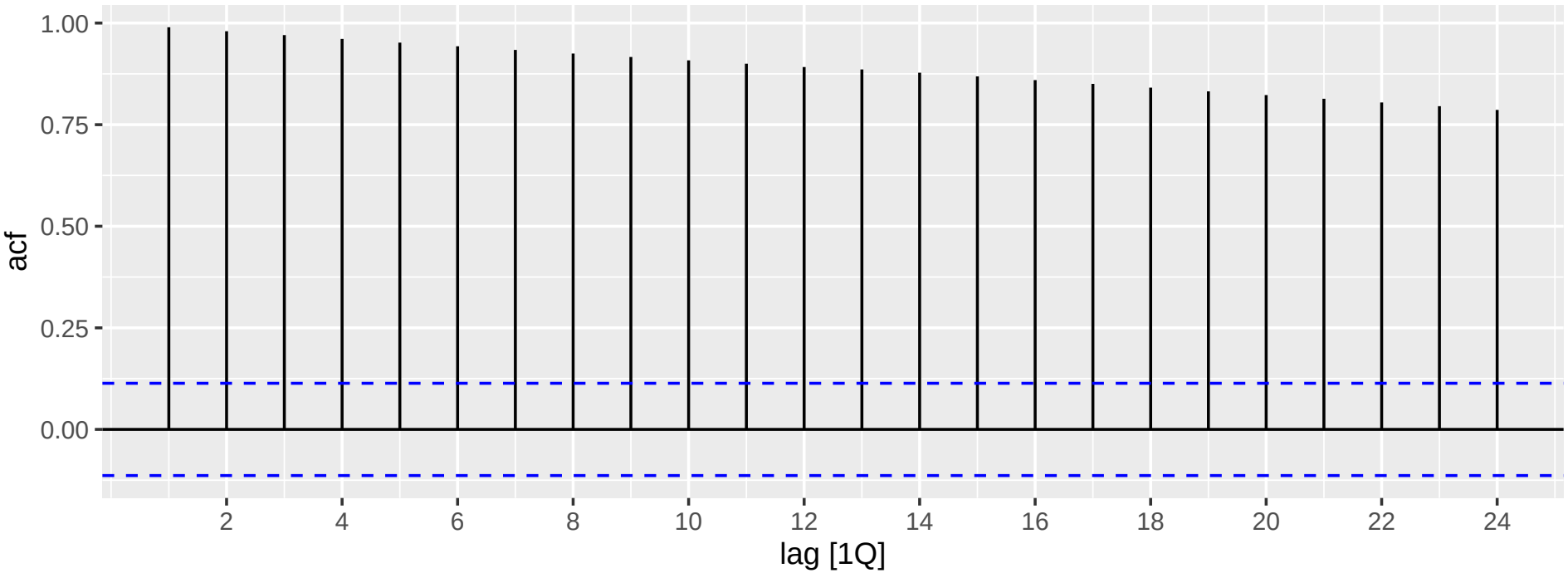
Plot multiple time series data (2)

```
# Multiple line plot
ggplot(df, aes(x = QUARTER, y = value)) +
  geom_line(aes(color = variable), size = 1) +
  scale_color_manual(values = c("#00AFBB", "#E7B800")) +
  theme_minimal()
```



Autocorrélation plot

```
library(feasts)
pib_tsbl %>% ACF(PIB) %>% autoplot()
```



Section 4

Transformation des données et stabilisation de la variance

D'{e}saisonnalisez ‘{a} l’aide de la r'{e}gression lin'{e}aire

- On souhaite désaisonnaliser la série temporelle retail l’aide de la régression linéaire.
- On créé les bases tendancielle et saisonnière :

```
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   0
##  [2,] 131244  0  1  0  0  0  0  0  0  0   0   0   0
##  [3,] 142488  0  0  1  0  0  0  0  0  0   0   0   0
##  [4,] 147175  0  0  0  1  0  0  0  0  0   0   0   0
##  [5,] 152420  0  0  0  0  1  0  0  0  0   0   0   0
##  [6,] 151849  0  0  0  0  0  1  0  0  0   0   0   0
##  [7,] 152586  0  0  0  0  0  0  1  0  0   0   0   0
##  [8,] 152476  0  0  0  0  0  0  0  1  0   0   0   0
```

Section 5

Décomposition d'une série temporelle

Decompositions (1)

- A common task in time series analysis is decomposing a time series into some simpler components.

Trend and Seasonality

In general, trends in the data can be linear:

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

or exponential:

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

Note that β_1 in the exponential time trend model is the average annual growth rate (assuming t is in years).

Trend and Seasonality

Often, data can be decomposed into three components:

- Trend
- Season
- Random component

The seasonal component can be included via dummy variables. For example, for quarterly data the following model can be used:

$$y_t = \beta_0 + \delta_1 \cdot Q1_t + \delta_2 \cdot Q2_t + \delta_3 \cdot Q3_t + \beta_1 \cdot x_{1,t} + \cdots + \beta_k \cdot x_{k,t} + \epsilon_t$$

One seasonal dummy must be dropped. That is, quarterly and yearly data require three and eleven dummy variables, respectively.

Trend and Seasonality

- Données `retail` sur le commerce de détail
- On créé une variable catégorielle (factor) pour le mois
- Le temps est un index T de 1 à ...

```
retail$MONTH = factor(month(retail$DATE), labels=month(1:12, label=TRUE))
retail$T     = c(1:nrow(retail))
bhat         = lm(RSXFSN ~ MONTH+T, data=retail)
summary(bhat)
```

```
##
## Call:
## lm(formula = RSXFSN ~ MONTH + T, data = retail)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -95922 -22769    753  10775 101781
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 99954.45    6105.34  16.372  < 2e-16 ***
## MONTHfévr   -3682.36    7744.75  -0.475 0.634738
## MONTHmars   36178.00    7744.79   4.671 4.20e-06 ***
## MONTHavril  28100.95    7744.86   3.628 0.000326 ***
## MONTHmai    47512.13    7744.96   6.135 2.21e-09 ***
## MONTHjuin   38870.58    7745.08   5.019 8.12e-07 ***
## MONTHjuil   37742.22    7745.23   4.873 1.64e-06 ***
## MONTHaoût   44347.86    7745.40   5.726 2.14e-08 ***
## MONTHsept   21660.47    7745.60   2.796 0.005437 **
## MONTHoct    28471.16    7807.07   3.647 0.000304 ***
```

Trend and Seasonality

- Prédiction du modèle

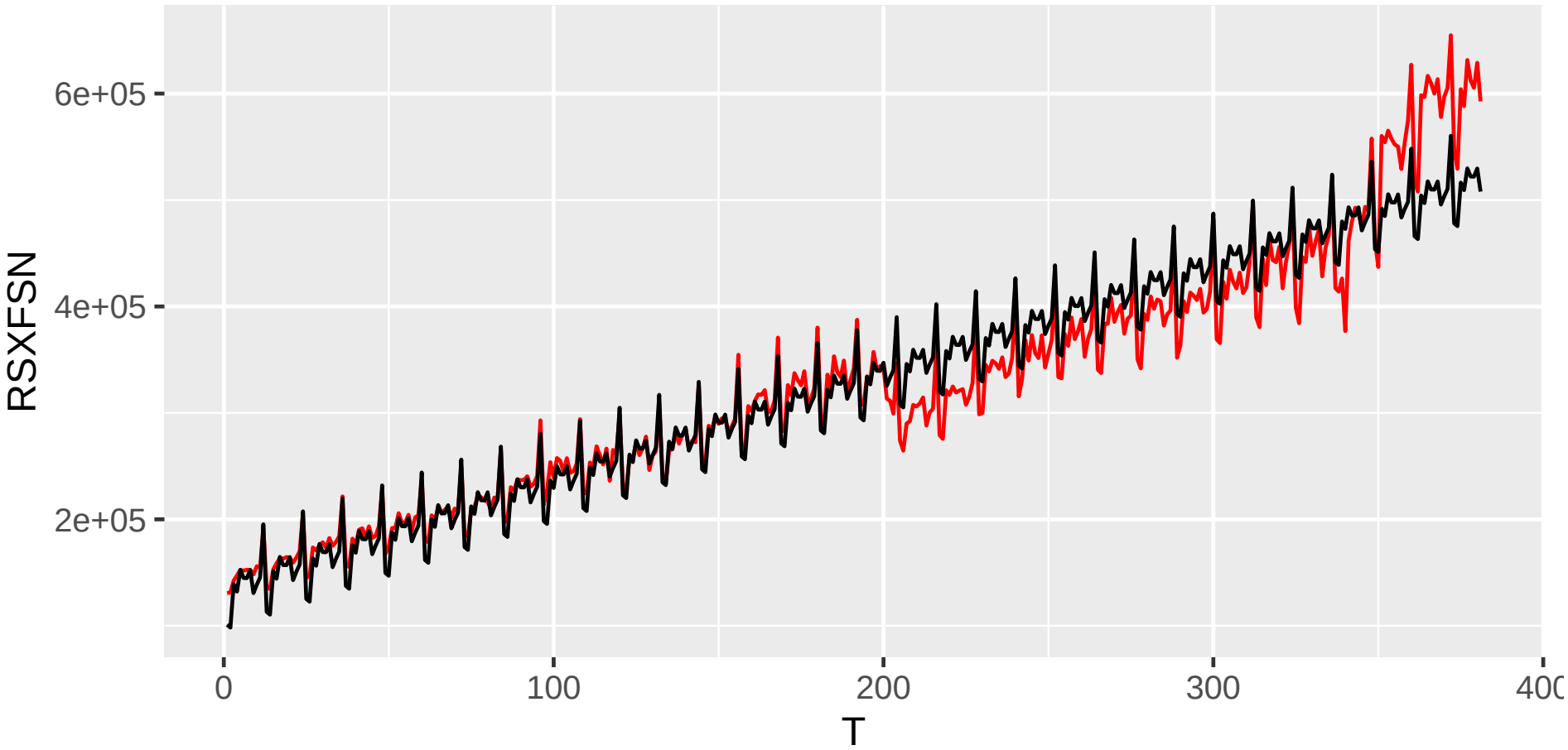
```
retail$fit   = predict.lm(bhat)
retail$fit
```

```
##   [1] 100968.37  98299.93 139174.21 132111.09 152536.18 144908.56 144794.12
##   [8] 152413.68 130740.21 138564.83 145424.05 195318.22 113135.43 110466.99
##  [15] 151341.27 144278.15 164703.24 157075.62 156961.18 164580.74 142907.27
##  [22] 150731.89 157591.12 207485.28 125302.49 122634.05 163508.33 156445.21
##  [29] 176870.30 169242.68 169128.24 176747.80 155074.33 162898.95 169758.18
##  [36] 219652.34 137469.55 134801.11 175675.39 168612.27 189037.36 181409.74
##  [43] 181295.30 188914.86 167241.39 175066.01 181925.24 231819.40 149636.61
##  [50] 146968.17 187842.46 180779.33 201204.42 193576.80 193462.36 201081.92
##  [57] 179408.46 187233.07 194092.30 243986.46 161803.67 159135.24 200009.52
##  [64] 192946.39 213371.49 205743.86 205629.42 213248.99 191575.52 199400.13
##  [71] 206259.36 256153.52 173970.73 171302.30 212176.58 205113.45 225538.55
##  [78] 217910.92 217796.48 225416.05 203742.58 211567.19 218426.42 268320.58
##  [85] 186137.79 183469.36 224343.64 217280.51 237705.61 230077.98 229963.54
##  [92] 237583.11 215909.64 223734.25 230593.48 280487.64 198304.86 195636.42
##  [99] 236510.70 229447.57 249872.67 242245.04 242130.61 249750.17 228076.70
## [106] 235901.32 242760.54 292654.70 210471.92 207803.48 248677.76 241614.64
## [113] 262039.73 254412.10 254297.67 261917.23 240243.76 248068.38 254927.60
## [120] 304821.76 222638.98 219970.54 260844.82 253781.70 274206.79 266579.17
## [127] 266464.73 274084.29 252410.82 260235.44 267094.66 316988.82 234806.04
## [134] 232137.60 273011.88 265948.76 286373.85 278746.23 278631.79 286251.35
## [141] 264577.88 272402.50 279261.72 329155.89 246973.10 244304.66 285178.94
## [148] 278115.82 298540.91 290913.29 290798.85 298418.41 276744.94 284569.56
## [155] 291428.79 341322.95 259140.16 256471.72 297346.00 290282.88 310707.97
## [162] 303080.35 302965.91 310585.47 288912.00 296736.62 303595.85 353490.01
## [169] 271307.22 268638.78 309513.06 302449.94 322875.03 315247.41 315132.97
## [176] 322752.53 301079.06 308903.68 315762.91 365657.07 283474.28 280805.84
```

Trend and Seasonality

- Données `retail` sur le commerce de détail

```
ggplot(retail)+
  geom_line(mapping=aes(x=T,y=RSXFSN),color="red")+
  geom_line(mapping=aes(x=T,y=fit))
```



Trend and Seasonality

The function tslm from the package forecast is used next. The function fits a linear model including seasonality and a trend component (and a trend-squared component if desired).

```
library(forecast)
```

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

```
bhat = tslm(pib_ts[,'PIB']~trend+I(trend^2)+season)
summary(bhat)
```

```
##
## Call:
## tslm(formula = pib_ts[, "PIB"] ~ trend + I(trend^2) + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -113.906  -20.681    7.124   19.532   47.396
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.604e+01  4.897e+00  -5.317 2.10e-07 ***
## trend        4.617e-01  6.593e-02   7.003 1.73e-11 ***
## I(trend^2)   6.570e-03  2.135e-04  30.767  < 2e-16 ***
## season2     -7.874e-01  3.984e+00  -0.198    0.843
## season3      1.995e-01  3.997e+00   0.050    0.960
## season4     -9.013e-02  3.997e+00  -0.023    0.982
```

Décomposition avec la librairie *feasts* (1)

- The feasts package supports two common time series decomposition methods:
  - Classical decomposition
  - STL decomposition

```
dcmp <- pib_tsbl %>%
  model(STL(PIB ~ season(window = Inf)))
components(dcmp)
```

```
## # A dable: 298 x 7 [1Q]
## # Key:     .model [1]
## # :        PIB = trend + season_year + remainder
##    .model                QUARTER   PIB trend season_year remainder season_adjust
##    <chr>                   <qtr> <dbl> <dbl>       <dbl>     <dbl>          <dbl>
##  1 STL(PIB ~ season(win~ 1949 Q1   3.2  3.27       0.113    -0.181           3.09
##  2 STL(PIB ~ season(win~ 1949 Q2   3.2  3.29      -0.674     0.580           3.87
##  3 STL(PIB ~ season(win~ 1949 Q3   3.3  3.36       0.413    -0.471           2.89
##  4 STL(PIB ~ season(win~ 1949 Q4   3.4  3.40       0.148    -0.151           3.25
##  5 STL(PIB ~ season(win~ 1950 Q1   3.6  3.63       0.113    -0.143           3.49
##  6 STL(PIB ~ season(win~ 1950 Q2   3.8  3.83      -0.674     0.645           4.47
##  7 STL(PIB ~ season(win~ 1950 Q3   4    3.99       0.413    -0.406           3.59
##  8 STL(PIB ~ season(win~ 1950 Q4   4.2  4.18       0.148    -0.126           4.05
##  9 STL(PIB ~ season(win~ 1951 Q1   4.4  4.49       0.113    -0.204           4.29
## 10 STL(PIB ~ season(win~ 1951 Q2   4.8  4.78      -0.674     0.696           5.47
## # i 288 more rows
```

Décomposition avec la librairie *feasts* (2)

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