

2 - Manipulation de Séries Temporelles en R

Webinaire: désaisonnalisation avec JDemetra+ en R

Anna Smyk



Objectifs (1/2)

Spécificité des séries temporelles: couple (index temps, valeur observée)

La date est liée aux valeurs, contenue dans l'objet

Fonctions spécifiques adaptées à cette structure : extractions, jointures, graphiques, autocorrélations, interpolations, lissage, modélisation, décomposition...

Besoin: utiliser des fonctions pré-codées dans des packages R

(éviter de recoder)

Nombreuses fonctions et packages disponibles:

voir CRAN Task View: <https://cran.r-project.org/web/views/Timeseries.html>

Objectifs (2/2)

Selon les besoins statistiques: différents packages requièrent différents formats

Deux exemples:

- rjdverse (famille autour de JDemetra+): objets de classe TS (très courant)

voir rjdverse: <https://github.com/rjdverse>

- fpp3 (forecasting principles and practice): objets de classe tsibble (prolonge la grammaire du tidyverse, permet de garder d'autres variables que la date et la valeur)

voir autour de fpp3: <https://robjhyndman.com/software/>

De multiples standards...

- objets ts : package stats
- objets tsibble : package tsibble: <https://CRAN.R-project.org/package=tsibble>
- objets zoo package zoo: <https://CRAN.R-project.org/package=zoo>
- objets xts package xts: <https://CRAN.R-project.org/package=xts>

...et un convertisseur

Convertisseur : package tsbox <https://CRAN.R-project.org/package=tsbox>

- conversion d'un format à l'autre
- nombreuses fonctions agnostiques

cf: cheat sheet

Manipulation de dates:

- package auxiliaire lubridate: <https://lubridate.tidyverse.org/>

cf. cheat sheet

Objets TS et principales opérations

On se concentre sur les objets TS utiles pour utiliser JDemetra+ !

création d'objets de classe TS (univariés et multivariés)

- conversions from and to data frames

Manipulations de données

- extractions de sous-séries
- extractions d'attributs
- jointures et création de séquences de dates

Fonctions statistiques

- sommes, moyennes
- imputation de valeurs manquantes

Création d'objets de classe TS univariés I

Fonction `ts(data = ., start = ., frequency = .)`

- à partir d'un vecteur numérique (colonne de data frame...)

Définition avec longueur, date de début et fréquence

```
ts1 <- ts((1:24) + 100, start = c(2020, 1), frequency = 12)  
print(ts1)
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
2020 101 102 103 104 105 106 107 108 109 110 111 112  
2021 113 114 115 116 117 118 119 120 121 122 123 124
```

```
class(ts1)
```

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

Création d'objets de classe TS univariés II

- frequency est le nombre d'observations par unité de temps (ici année) : 1=annuelle, 2=semestrielle, 4=trimestrielle, 6=bi-mestrielle, 12=mensuelle

Définition avec longueur, date de fin et fréquence

```
ts(3 + 5 * rnorm(60), frequency = 4, end = c(2002, 2)) #dernier point inclus
```

	Qtr1	Qtr2	Qtr3	Qtr4
1987			7.23778046	3.07995393
1988	-0.29829556	4.05432832	6.71066507	8.04711311
1989	4.89665764	2.01814454	0.03069370	3.43212817
1990	-0.89787257	5.46781147	3.22662912	-11.55519526
1991	8.92814619	1.84166746	6.61888632	-0.10872803
1992	4.03504959	-4.83211536	0.94580979	4.77525345
1993	-5.04660674	7.98617264	-5.35145149	1.82917179
1994	10.02151253	-4.00345366	-1.53755093	5.76356934
1995	11.32456201	10.71004854	9.09257760	-0.07463933
1996	3.75322074	4.05689175	8.06704517	-0.00651536

Création d'objets de classe TS univariés III

```
1997    0.49768600    2.60227460    2.90318209    4.24114499
1998   -0.73741938    0.02726382   11.60394672   -1.31203000
1999   -0.59685625    0.12238778    7.92786618    5.32463056
2000    3.97499357    1.34253334   -1.31870528    1.98109013
2001    8.14473970    9.97160467   -2.54022995    2.91264903
2002   10.57816851    4.36905225
```

Définition avec date de début et de fin

```
ts(3 + 5 * rnorm(72), frequency = 12, start = c(2000, 1), end = c(2004, 12)) #coupe le vecteur
```

	Jan	Feb	Mar	Apr	May	Jun
2000	6.2736349	1.4102836	-7.1563669	-3.4151331	-2.0702895	-1.7776001
2001	3.7150755	13.6398765	2.1754108	-1.9396335	0.5253450	5.3465519
2002	-6.8184837	5.4790938	2.6212904	0.2707564	-0.5583447	7.4729282
2003	4.5951147	3.9978757	6.0622074	1.8617877	2.3487366	3.6681279
2004	-2.2870798	0.2733633	14.4796688	4.4373565	7.6428670	-6.7599214
	Jul	Aug	Sep	Oct	Nov	Dec

Création d'objets de classe TS univariés IV

2000	-6.6519014	2.5403441	-3.6587036	6.7460616	1.6468717	7.4080740
2001	9.1532936	-7.1999808	4.1066791	-0.5991453	1.8750470	6.3143505
2002	-4.6157608	-0.7409484	6.8460221	0.2882827	-7.3826417	2.5099379
2003	6.1626281	-4.3213203	6.8262550	-4.1241952	-0.4820956	-3.7461198
2004	9.8915884	-3.8172813	-1.2639630	5.4319341	10.2779321	-1.1733619

Création d'objets de classe TS multivariés I

A partir d'une matrice

```
mts_object <- ts(  
  matrix(rnorm(30), 12, 3),  
  start = c(2000, 1),  
  frequency = 12  
)  
print(mts_object)
```

	Series 1	Series 2	Series 3
Jan 2000	-1.20511674	0.3300793	-0.8846672
Feb 2000	1.16948814	-0.8103935	0.1244415
Mar 2000	0.70039002	-2.8726856	-1.6320795
Apr 2000	1.31225763	-3.0457168	1.1509212
May 2000	1.15215034	-0.7976345	-0.9964734
Jun 2000	-0.37132613	0.2276553	0.3870087
Jul 2000	0.04308584	0.1094884	-1.2051167
Aug 2000	-1.24986882	1.6952969	1.1694881

Création d'objets de classe TS multivariés II

```
Sep 2000  0.42619883  1.0245760  0.7003900  
Oct 2000  0.65907213  0.9742932  1.3122576  
Nov 2000  1.07184391 -1.3300013  1.1521503  
Dec 2000  1.17145129 -1.4631521 -0.3713261
```

```
class(mts_object)
```

```
[1] "mts"      "ts"       "matrix"  "array"
```

```
is.mts(mts_object)
```

```
[1] TRUE
```

Création d'objets de classe TS multivariés I

A partir d'un data frame: on extrait les colonnes numériques (matrice de valeurs) et on respécifie les dates lors de la création de l'objet mts (attention à la date de début)

```
# data frame ipi
y_raw <- ts(ipi[, "RF3030"], start = c(1990, 1), frequency = 12)
y_raw

# start = c(1990,1): résulte de la connaissance du data frame
```

Récupération d'attributs (1/2) I

```
ts1 <- ts((1:24) + 100, start = c(2020, 1), frequency = 12)  
start(ts1)
```

```
[1] 2020    1
```

```
class(start(ts1))
```

```
[1] "numeric"
```

```
start(ts1)[2]
```

```
[1] 1
```

```
end(ts1)
```

```
[1] 2021    12
```

```
frequency(ts1)
```

```
[1] 12
```

Récupération d'attributs (2/2) I

création de la série des dates correspondante à un objet ts : fonction `time()`

```
time(ts1) #fractions: 1/frequency  
  
# fonctions pour retrouver un format date  
# exemple  
date <- zoo::as.Date(time(ts1))  
date  
class(date)
```

Récupération de la position dans l'année d'une observation : fonction `cycle()`

```
cycle(ts1)  
class(cycle(ts1))
```

Extraction et jointures I

Exemple avec deux objets ts

```
ts1 <- ts(1:15, start = c(2022, 1), frequency = 12)
ts2 <- ts(13:24, start = c(2023, 1), frequency = 12)
```

- extraction `ts.window` ou `tsbox::ts_span`

```
ts11 <- window(ts1, start = c(2022, 6), end = c(2022, 12))
ts11
```

```
      Jun Jul Aug Sep Oct Nov Dec
2022   6   7   8   9  10  11  12
```

```
ts12 <- ts_span(ts1, "-6 month")
ts12
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2022                10  11  12
2023  13  14  15
```


Extraction et jointures II

Union

```
# séries en tableau  
# on garde toute la couverture temporelle en rajoutant des NA  
  
ts.union(ts1, ts2) #classe mts  
  
#head(ts.union(ts1,ts2))
```

Intersection

```
# on ne garde que les périodes communes  
ts.intersect(ts1, ts2)
```

Extraction et jointures III

Conversions avec le package `tsbox`

`ts_c`: comme `ts.union`

`ts_bind`: on combine plusieurs séries en une, si chevauchement la première citée l'emporte (sauf si NA), cf. exemples infra

`ts_chain`: comme `ts_bind` mais avec interpolation

Listes de séries I

Format liste pratique pour appliquer des fonctions avec la famille 'lapply()'

```
ma_liste <- ts_tslist(mts_object)
ma_liste[2]
```

\$`Series 2`

	Jan	Feb	Mar	Apr	May	Jun
2000	0.3300793	-0.8103935	-2.8726856	-3.0457168	-0.7976345	0.2276553
	Jul	Aug	Sep	Oct	Nov	Dec
2000	0.1094884	1.6952969	1.0245760	0.9742932	-1.3300013	-1.4631521

```
class(ma_liste[2])
```

```
[1] "list"
```

```
ma_liste[[2]]
```

Listes de séries II

	Jan	Feb	Mar	Apr	May	Jun
2000	0.3300793	-0.8103935	-2.8726856	-3.0457168	-0.7976345	0.2276553
	Jul	Aug	Sep	Oct	Nov	Dec
2000	0.1094884	1.6952969	1.0245760	0.9742932	-1.3300013	-1.4631521

```
class(ma_liste[[2]])
```

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

Opérations arithmétiques sur les séries I

```
ts1 <- ts(1:6, start = c(2023, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2023                                1   2
2024    3   4   5   6
```

```
ts2 <- ts(10:15, start = c(2024, 1), frequency = 12)
ts2
```

```
      Jan Feb Mar Apr May Jun
2024   10  11  12  13  14  15
```

```
# opérations simples: sur périodes communes (coupe)
ts1 + ts2 # idem pour - * /
```

```
      Jan Feb Mar Apr
2024   13  15  17  19
```

Opérations arithmétiques sur les séries II

```
# avec ts box
```

```
# périodes communes
```

```
ts1 %ts+% ts2
```

```
      Jan Feb Mar Apr  
2024  13  15  17  19
```

```
# on peut forcer le format de la série figurant à gauche
```

```
ts_df(ts1) %ts+% ts2
```

```
      time value  
1 2024-01-01    13  
2 2024-02-01    15  
3 2024-03-01    17  
4 2024-04-01    19
```

Manipulation de dates I

création de séquences de dates sous R avec la fonction `seq()`

```
date <- seq(from = as.Date("2024-01-01"),  
            to = as.Date("2024-12-31"),  
            by = "month")
```

date

```
[1] "2024-01-01" "2024-02-01" "2024-03-01" "2024-04-01" "2024-05-01"  
[6] "2024-06-01" "2024-07-01" "2024-08-01" "2024-09-01" "2024-10-01"  
[11] "2024-11-01" "2024-12-01"
```

```
date <- seq(from = as.Date("2024-01-01"),  
            to = as.Date("2024-12-31"),  
            by = "quarter")
```

date

```
[1] "2024-01-01" "2024-04-01" "2024-07-01" "2024-10-01"
```

Manipulation de dates I

Manipulation avec le package lubridate (voir cheat sheet) qui contient de très nombreuses fonctions, ici deux exemples:

- conversion au format date d'une chaîne de caractères, fonctions `ymd()`, `ymd_hms`, `dmy()`, `dmy_hms`, `mdy()`

```
"Jan-2020"
```

```
[1] "Jan-2020"
```

```
"Jan-2020" %>% class()
```

```
[1] "character"
```

```
date <- lubridate::my("Jan-2020")  
date
```

```
[1] "2020-01-01"
```


Manipulation de dates II

```
class(date)
```

```
[1] "Date"
```

Manipulation de dates I

- extraction d'attributs/modification de la composante d'une date avec les fonctions `year()`, `month()`, `mday()`, `hour()`, `minute()` and `second()`

```
# création d'une variable date
date <- seq(from = as.Date("2024-01-01"),
            to = as.Date("2024-03-31"),
            by = "month")
date
```

```
[1] "2024-01-01" "2024-02-01" "2024-03-01"
```

```
month(date)
```

```
[1] 1 2 3
```

```
month(date) %>% class()
```

```
[1] "numeric"
```

Manipulation de dates II

```
month(date[2]) <- 11  
date
```

```
[1] "2024-01-01" "2024-11-01" "2024-03-01"
```

Série retardée I

Pour calculer la série retardée/avancée, il suffit d'utiliser la fonction `lag()`, mais attention au paramétrage selon le package

```
ts1 <- ts(1:6, start = c(2024, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun
2024   1   2   3   4   5   6
```

```
# package stats
stats::lag(ts1, k = -1) # attention période série finale
```

```
      Feb Mar Apr May Jun Jul
2024   1   2   3   4   5   6
```

```
# package dplyr sur vecteur numérique
dplyr::lag(as.vector(ts1), 1)
```

```
[1] NA  1  2  3  4  5
```

Série retardée II

```
# package tsbox  
tsbox::ts_lag(ts1) #k=1 par défaut
```

	Feb	Mar	Apr	May	Jun	Jul
2024	1	2	3	4	5	6

```
tsbox::ts_lag(ts1, 12)
```

	Jan	Feb	Mar	Apr	May	Jun
2025	1	2	3	4	5	6

```
tsbox::ts_lag(ts1, 4)
```

	May	Jun	Jul	Aug	Sep	Oct
2024	1	2	3	4	5	6

Différenciation I

Différenciation - à l'ordre k

$$Diff(k) = X_t - X_{t-k}$$

- le plus souvent à l'ordre 1 (tendance) et/ou à l'ordre 12,4... saisonnalité

$$Diff(1) = X_t - X_{t-1}$$

$$Diff(12) = X_t - X_{t-12}$$

```
ts1 <- ts(1:24, start = c(2024, 1), frequency = 12)
ts1
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	1	2	3	4	5	6	7	8	9	10	11	12
2025	13	14	15	16	17	18	19	20	21	22	23	24

```
# diff d'ordre 1
diff1 <- ts1 - lag(as.vector(ts1))
diff1
```

Différenciation II

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	NA	1	1	1	1	1	1	1	1	1	1	1
2025	1	1	1	1	1	1	1	1	1	1	1	1

```
diff1 <- ts1 - ts_lag(ts1) #attention NA et période de la série finale  
diff1
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024		1	1	1	1	1	1	1	1	1	1	1
2025	1	1	1	1	1	1	1	1	1	1	1	1

```
# ou fonction directe  
ts_diff(ts1)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	NA	1	1	1	1	1	1	1	1	1	1	1
2025	1	1	1	1	1	1	1	1	1	1	1	1

Différenciation III

```
# diff d'ordre 12
diff12 <- ts1 - ts_lag(ts1, 12)
diff12
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2025	12	12	12	12	12	12	12	12	12	12	12	12

```
# ou fonction directe
ts_diffy(ts1)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2024	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2025	12	12	12	12	12	12	12	12	12	12	12	12

Agrégation I

Passer à une fréquence plus basse avec une fonction spécifique (somme, moyenne, dernière valeur)

exemple de solution : `tsbox::ts_frequency`

```
ts1 <- ts((1:12) + 100, start = c(2024, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2024 101 102 103 104 105 106 107 108 109 110 111 112
```

```
ts_frequency(ts1, "quarter") #default: mean
```

```
      Qtr1 Qtr2 Qtr3 Qtr4
2024   102   105   108   111
```

```
ts_frequency(ts1, "quarter", "sum")
```

```
      Qtr1 Qtr2 Qtr3 Qtr4
2024   306   315   324   333
```

Agrégation II

```
ts_frequency(ts1, "quarter", "last")
```

	Qtr1	Qtr2	Qtr3	Qtr4
2024	103	106	109	112

Désagrégation temporelle vers une fréquence plus élevée : problème plus complexe, voir packages `rjd3bench`,...

Valeurs manquantes I

On peut utiliser des fonctions du package `zoo` ou `imputeTS` (par exemple) pour

- repérer les valeurs manquantes : fonction `is.na`
- les enlever: au début et/ou à la fin `zoo::na.trim()`
- les imputer
 - dernière valeur `zoo::na.locf`
 - interpolation linéaire `zoo::na.approx()`
 - autres méthodes: moyenne, splines, kalman filter

Voir package `imputeTS` (cheat sheet)

Valeurs manquantes I

```
ts1 <- ts((1:12) + 100, start = c(2024, 1), frequency = 12)
ts1
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2024 101 102 103 104 105 106 107 108 109 110 111 112
```

```
#ajout NA début
```

```
ts2 <- ts(as.numeric(rep(NA, 2)), start = c(2023, 12), frequency = 12)
ts2
```

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2023                                     NA
2024  NA
```

```
ts12 <- ts_bind(ts1, ts2)
ts12
```

Valeurs manquantes II

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2023												NA
2024	101	102	103	104	105	106	107	108	109	110	111	112

```
# ajout de NA au milieu  
month(as.Date(time(ts12))) # pas de NA ici
```

```
[1] 12  1  2  3  4  5  6  7  8  9 10 11 12
```

```
ts12[month(as.Date(time(ts12))) %in% c(3, 8)] <- NA  
ts12
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2023												NA
2024	101	102	NA	104	105	106	107	NA	109	110	111	112

```
#on enlève les valeurs manquantes du début  
ts12_i <- zoo::na.trim(ts12, sides = "left")  
ts12_i
```

Valeurs manquantes III

```
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2024 101 102  NA 104 105 106 107  NA 109 110 111 112
```

```
ts12_ii <- imputeTS::na_mean(ts12_i) # moyenne de la série sans NA
ts12_ii
```

```
      Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
2024 101.0 102.0 106.7 104.0 105.0 106.0 107.0 106.7 109.0 110.0 111.0 112.0
```