RjpWiki アーカイブス

【時系列オブジェクト一覧 (06.06.27) 】

1　時間情報

時系列オブジェクトは (等間隔) 観測時間情報を持つため、時間に関する以下のような幾つかの特殊な情報を持つ:

自然な時間単位	年、月、週、一時間等
観測開始時間	start
観測終了時間	end
頻度	frequency：単位時間内の観測値の数。月別データなら、自然な時間単位「年」に対する周期は 12
サンプリング比率	deltat：　自然な時間単位に対する、サンプリング間隔を表す比率。月別データなら、自然な時間単位「年」に対する deltat は 1/12。周期とサンプリング比率どちらか一方を与えれば良い。
周期	cycle：　各データの観測時間情報を表す、自然な時間単位とその中での頻度を表す対。例えば年と月 (1993、5)、年と四半期 (2002、Qtr2)、月と日 (12、13)、週と曜日 (43、Fri) 等

1.1　R の組込み時系列データ

airmiles	米国の商業航空路線の有償旅客マイル数 (revenue passenger miles、ある期間中の運賃払乗客数をその乗客らの飛行距離でかけたもの)。 1937 -- 1960 年の年別時系列、24 観測値
co2	大気中の CO2 濃度 (ppm)。1997 年度の ISO モル分率圧力計スケールによる。1959--1997 年度の月別データ。468 観測値。1964 年の 2,3,4 月分は 欠損しており、1964 年の 1,5 月のデータから線形補間で補われている
nhtemp	コネチカット州 New Haven の年平均気温 (華氏)。 1912--1971 年の年別データ。60 観測値
presidents	米国の (ほぼ) 四半期毎の大統領指示率 (ギャラップ社調査)。 1945 年第 1 四半期から 1974 年第 4 四半期。120 観測値。実際はかなりでっち上げのデータ
sunspots	月別の平均相対太陽黒点数。1960 年まではスイス連邦観測所、それ以降は東京天文台による。 1749--1983 年の月別データ。stats パッケージ中の sunspot.month データはより長期で少し異なる黒点数データである

uspop	10 年毎の国勢調査による米国人口 (単位百万人)。1790--1970 年。19 観測値
AirPassengers	古典的な Box-Jenkins の航空路線データ。1949--1960 年の月別国際航空路線総顧客数 (単位千人)
BJsales	二つのデータ、Box-Jenkins の売上高データ BJsales と、景気の先行指標データ BJsales.lead を含む。各々150 観測値からなる。
EuStockMarkets	ヨーロッパの主要株式指標 (Germany DAX (Ibis), Switzerland SMI, France CAC, そして UK FTSE) の毎日の引け値。データは週末と祭日を除く。4 変量で、1860 観測値
JohnsonJohnson	1960--1980 年の Johnson-Johnson の一株当たり四半期毎利益 (単位ドル)
LakeHuron	ヒューロン湖の水位の年別測定値 (単位フィート)。98 観測値
Nile	1871--1970 年のアスワンに於けるナイル川の年流量測定値。100 観測値
UKDriverDeaths	1969--1984 年の英国の、死亡もしくは重傷を負った車の運転者の月別数。 シートベルトの着用が 1983 年 1 月 31 日に義務付けられた。データ Seatbelts はより詳しい情報を含む 8 変量時系列データである
UKLungDeaths	呼吸器系疾患 (肺癌、肺気腫、喘息) による 1974--1979 年の英国の月別死亡者数。両性 ldeaths、男性 mdeaths、女性 fdeaths の 3 時系列からなる
UKgas	1960-1968 年の英国の四半期毎のガス消費量。108 観測値
USAccDeaths	1973--1978 年の米国の月別事故死亡者数
WWWusag	あるサーバー経由でインターネットに接続したユーザの分毎の数。観測数 100
austres	1971 年 3 月から 1994 年 5 月に至るオーストラリアの四半期毎の居住者数 (千人単位)
beavers	2 匹の雌のビーバ (Castor canadensis) の体温を 10 分毎にテレメータで測定したデータ。データフレーム beaver1 は 4 変数 114 観測値 (12 月 12,13 日、22:20 分のデータは欠損)、beaver2 は 4 変数 100 観測値 (10 月 3,4 日)
lh	女性の 10 分間隔の血液中の黄体形成ホルモン量の時系列。48 標本量。
lynx	1821--1934 年にカナダで罠で捕獲され大山猫 (lynx) の年別数。Campbell-Walker が JRSS, series A, vol.140, 411-431(1977) で解析したデータと思われる
nottem	ノッティンガム城で 1920--1939 年に測定された月別平均気温 (華氏)
sunspot	月別と年別の太陽黒点数。年別時系列 sunspot.year と月別時系列 sunspot.monneth はそれぞれ 289, 2988 観測値を含む。月別データは、1988 年までのデータである base パッケージ中の sunspots データよりも長い
treering	(単位が与えられていない) 正規化されたカリフォルニアの Bristlecone pine (Bristlecone pine は樹齢 4 千年を越える個体が存在する世界でも最長級の寿命を持つ植物) の年輪の幅。7981 観測値

2　時系列オブジェクトの生成と処理

2.1　時系列オブジェクトを生成する、ts(), as.ts(), is.ts()

クラス ”ts” のオブジェクトをベクトルもしくは行列から生成する基本関数は ts() である。時系列データには、一年およびそれを 12 分割した月、一週間とそれを 7 分割した日、等の自然な時間単位と、それの分割単位が備わっていることが多い。

```
ts(data = NA, start = 1, end = numeric(0), frequency = 1,
   deltat = 1, ts.eps = getOption("ts.eps"), class = , names = )
as.ts(x)
is.ts(x)
```

2.2　時系列オブジェクトに対する数値演算

時系列オブジェクトに対しては、スカラーによる算術演算、関数の適用等が可能であり、結果は同じ時間範囲を持つ時系列オブジェクトになる。行列風の鈎括弧演算子による部分抽出ができる。また二つの時系列同士の算術演算も可能で、共通時間範囲で各時間毎に演算を行った結果からなる時系列になる。

2.3　時系列オブジェクトに対する演算

時系列オブジェクトに対して、その階差 (difference) を取る diff() 関数がある。diff() は既定のメソッドを持つ総称的関数であり、クラス ”ts” と ”POSIXt” (R は二種類の基本日付・時刻クラス POSIXct と POSIXlt を扱う。前者は 1970 年始めからの経過秒数を表す (符号付き) 数値ベクトルであり、後者は秒・分・時・日にち・月・(1900 年以来の) 経過年数からなる名前付きベクトルリストである。両者は時間帯を表す ”tzone” 属性を持つこともある (OS 依存)。POSIXt は両クラスを継承する日付・時刻クラスであり、論理比較、加減等の限られた算術演算ができる) に対するメソッドを持つ。欠損値 NA があれば伝播する。

```
diff(x, lag = 1, differences = 1, ...)
## クラス "POSIXt" に対する S3 メソッド
diff(x, lag = 1, differences = 1, ...)
```

2.4　時系列オブジェクトに対する演算

ラグ階差関数 diff() の逆処理 (離散積分)。diffinv() はクラス ”ts” に対するメソッドを持つ総称的関数であり、ベクトルと行列に対する既定動作を持つ。欠損値は処理できない。

```
diffinv(x, lag = 1, differences = 1,
        xi = rep(0.0, lag*differences*NCOL(x)), ...)
```

2.5 時系列の合併と共通部分、ts.union(), ts.intersection()

頻度が同じ複数の時系列オブジェクトを、時間範囲の合併と共通部分上の多変量時系列に変換する関数 ts.union(), ts.intersection() がある。ts.union() は該当観測値が無ければ欠損値 NA で埋める。ts.intersect() は全ての時系列に共通の期間に制限した時系列を与える。

```
ts.intersect(..., dframe = FALSE)
ts.union(..., dframe = FALSE)
```

2.6 時系列に関する情報を得る、start(), end(), frequncy(), cycle(), tsp()

時系列オブジェクトに関する情報を得る関数には start(),end(), frequency(), time(), cycle(), tsp() 等がある。時系列オブジェクトは付加情報を持つベクトルもしくは行列であるから、長さ関数 length() 関数や、次元関数 dim() を用いて、サイズを知ることができる。

2.7 時系列オブジェクトの部分時系列を得る

window() 関数は、一つの時系列オブジェクトから、時間範囲を指定して部分時系列を取り出す。

```
window(x, start = NULL, end = NULL, frequency = NULL,
       deltat = NULL, extend = FALSE, ...)
```

2.8 時系列のラグ演算

時系列のラグ (遅延演算) を取る。与えられた観測値数分、時間軸を過去にずらす。

```
lag(x, k = 1, ...)
```

2.9 時系列オブジェクトに対するメソッド、diff(), na.omit()

クラス ”ts” の時系列オブジェクト (普通関数 ts() の結果) に対するメソッド。

```
## クラス "ts" に対する S3 メソッド
diff(x, lag=1, differences=1, ...)
## クラス "ts" に対する S3 メソッド
na.omit(object, ...)
```

2.10 時系列を部分的に集約する

データを部分集合に分割し、各々に対する要約統計量を計算し、結果を適切な形式で返す。

```
aggregate(x, ...)
## クラス "data.frame" 用メソッド
aggregate(x, by, FUN, ...)
## クラス "ts" 用メソッド
aggregate(x, nfrequency = 1, FUN = sum, ndeltat = 1,
          ts.eps = getOption("ts.eps"), ...)
```

2.11 欠損値を含まない最長の部分時系列を取り出す

引き続いて欠損値を含まないような、最長の部分時系列を取り出す。もしそうしたものが複数あれば、最初のものを返す。引き数 frame は一次元、もしくは多次元時系列である。frame のクラスは保存される。

```
na.contiguous(frame)
```

3 時系列の自己相関、スペクトル密度

3.1 時系列の自己共分散と自己相関係数 acf(), pacf(), ccf()

時系列データに関する最も基本的な統計量は自己共分散 (auto-covariance) と自己相関係数 (auto-correlation) である。R の関連する関数は acf(), pacf(), ccf() である。関数 acf() は時系列オブジェクトの自己共分散と自己相関係数を計算し、既定でそれをプロットする。関数 pacf() は 偏自己相関係数 (partial autocorrelations) を計算する。関数 ccf() は二つの一次元時系列間のクロス相関係数 (cross-correlation) とクロス共分散 (cross-cavarinace) を計算する。

```
acf(x, lag.max = NULL,
    type = c("correlation", "covariance", "partial"),
    plot = TRUE, na.action = na.fail, demean = TRUE, ...)
pacf(x, lag.max = NULL, plot = TRUE, na.action = na.fail, ...)
ccf(x, y, lag.max = NULL, type = c("correlation", "covariance"),
    plot = TRUE, na.action = na.fail, ...)
```

4 スペクトル密度関数

4.1 時系列のスペクトル密度関数の推定

時系列のスペクトル密度関数 (spectral density) を推定する。

```
spectrum(x, method = c("pgram","ar"), plot = TRUE,
         na.action = na.fail, ...)
```

4.2 AR モデルの当てはめによりスペクトル密度を推定する

x に AR モデルを当てはめ (もしくは既に存在する当てはめ結果) により、スペクトル密度を推定する。返り値はクラス "spec" のオブジェクトで、もし plot = TRUE ならコンソールには出力されない。

```
spec.ar(x, n.freq, order = NULL, plot = TRUE, na.action = na.fail,
        method = "yule-walker", ...)
```

4.3 ペリオドグラムの計算

高速フーリエ変換によりペリオドグラムを計算する。オプションとして結果を修正 Daniel 平滑法 (終端の重みを半分にする移動平均) による系列で平滑化する。もし plot = TRUE なら結果はコンソールに表示されない。

```
spec.pgram(x, spans = NULL, kernel, taper = 0.1,
           pad = 0, fast = TRUE, demean = FALSE, detrend =TRUE,
           plot = TRUE, na.action = na.fail, ...)
```

4.4 時系列の両端を削る

cosine-bell 関数を、時系列 x[,i] の最初と最後の p[i]/2 分の観測値に適用する。返り値は新しい時系列である。cosine bell とは関数 (1 - cos(x))/ 2 を指す。この 0 から 1 まで値が変化する関数を時系列の両端部分に掛けることにより、時系列の値を両端で徐々に 0 に近づける操作が cosine taper である。

```
spec.taper(x, p=0.1)
```

5 AR モデル

5.1 時系列への AR モデルの当てはめ

時系列オブジェクトへの AR モデル (自己回帰モデル、auto-regressive model) の当てはめを様々な手法で行う関数群。

```
ar(x, aic = TRUE, order.max = NULL,
   method=c("yule-walker", "burg", "ols", "mle", "yw"),
   na.action, series, ...)
ar.burg(x, aic = TRUE, order.max = NULL, na.action = na.fail,
        demean = TRUE, series, var.method = 1, ...)
ar.yw(x, aic = TRUE, order.max = NULL, na.action = na.fail,
      demean = TRUE, series, ...)
ar.mle(x, aic = TRUE, order.max = NULL, na.action = na.fail,
       demean = TRUE, series, ...)
```

5.2 当てはめモデルによる予測

当てはめられた AR モデルにより、予測を行う関数 predict() がある。predict() は総称的関数であり、実際は predict.ar() 関数が実行される。

```
predict(object, newdata, n.ahead = 1, se.fit = TRUE, ...)
```

5.3 最小自乗法による AR モデルの当てはめ

通常最小自乗法 (OLS, Ordinary Least Square) により、時系列に AR モデルを当てはめる。既定では AIC 法で次数を決める。

```
ar.ols(x, aic = TRUE, order.max = NULL, na.action = na.fail,
       demean = TRUE, intercept = demean, series, ...)
```

6 ARMA、ARIMA モデル

6.1 ARIMA モデルのシミュレーション

ARIMA モデルをシミュレーションする。返り値はクラス ”ts” のオブジェクトである。

```
arima.sim(model, n, rand.gen = rnorm, innov = rand.gen(n, ...),
          n.start = NA, ...)
```

6.2 ARIMA モデルの当てはめ

一変量時系列に ARIMA モデルを当てはめる。

```
arima(x, order = c(0, 0, 0),
      seasonal = list(order = c(0, 0, 0), period = NA),
      xreg = NULL, include.mean = TRUE, transform.pars = TRUE,
      fixed = NULL, init = NULL, method = c("CSS-ML", "ML", "CSS"),
      n.cond, optim.control = list(), kappa = 1e6)
```

6.3 ARMA モデルの自己相関係数

ARMA モデルの理論自己相関係数もしくは部分自己相関係数を計算する。

```
ARMAacf(ar = numeric(0), ma = numeric(0), lag.max = r, pacf = FALSE)
```

6.4 ARMA モデルを無限次数 MA モデルに変換する

ARMA モデルを無限次数 MA モデルに変換する。

```
ARMAtoMA(ar = numeric(0), ma = numeric(0), lag.max)
```

7 時系列に対する検定

7.1 時系列の独立性帰無仮説に対する検定

与えられた時系列に対する「独立性帰無仮説」を調べる Box-Pierce もしくは Ljung-Box 検定統計量を計算する。欠損値は許されない。

```
Box.test(x, lag = 1, type = c("Box-Pierce", "Ljung-Box"))
```

7.2 Phillips-Perron の単位根検定

x が単位根を持つという帰無仮説を、定常対立仮説に対して検定する Phillips-Perron 検定を実行する。欠損値は許されない。

```
PP.test(x, lshort = TRUE)
```

7.3 時系列解析の診断図

時系列の当てはめ結果の診断図を描く総称的関数である。

```
tsdiag(object, gof.lag, ...)
```

8 時系列の成分への分解

8.1 移動平均による古典的な時系列の成分への分解

時系列を、移動平均により季節、傾向、不規則成分に分解する。加法、乗法モデルをともに扱える。

```
decompose(x, type = c("additive", "multiplicative"), filter = NULL)
```

8.2 loess を用いた時系列の成分への分解

loess() 関数を用いた時系列の、季節、傾向、不規則成分への分割。STL と略される。

```
stl(x, s.window, s.degree = 0,
    t.window = NULL, t.degree = 1,
    l.window = nextodd(period), l.degree = t.degree,
    s.jump = ceiling(s.window/10),
    t.jump = ceiling(t.window/10),
    l.jump = ceiling(l.window/10),
    robust = FALSE,
    inner = if(robust)  1 else 2,
    outer = if(robust) 15 else 0,
    na.action = na.fail)
```

9 フィルタリング、平滑化

9.1 線形フィルタ

一変量時系列、もしくは多変量時系列の各時系列に線形フィルタを適用する。返り値は時系列オブジェクトである。

```
filter(x, filter, method = c("convolution", "recursive"),
       sides = 2, circular = FALSE, init)
```

9.2 各関数による平滑化

入力時系列の、特定の核関数による畳み込みによる平滑化を計算する。返り値は平滑化された結果の系列である。

```
kernapply(x, k, circular = FALSE, ...)
kernapply(k1, k2)
```

9.3 平滑化核関数オブジェクト

クラス ”tskernel” は、離散的で対称な正規化された平滑化核関数を表現するようにデザインされている。これらの核関数は、ベクトル、行列、時系列オブジェクトを平滑化するのに使われる。

```
kernel(coef, m, r, name)
df.kernel(k)
bandwidth.kernel(k)
is.tskernel(k)
```

9.4 Holt-Winters フィルタリング

時系列に対する Holt-Winters フィルタリングを計算する。未知パラメータは自乗予測誤差の最小化で決定される。

```
HoltWinters(x, alpha = NULL, beta = NULL, gamma = NULL,
            seasonal = c("additive", "multiplicative"),
            start.periods = 3, l.start = NULL, b.start = NULL,
            s.start = NULL,
            optim.start = c(alpha = 0.3, beta = 0.1, gamma = 0.1),
            optim.control = list())
```

10 時系列関連の作図関数

10.1 プロット関数

プロット用の総称的関数である。作図パラメータの詳細に付いては par() を参照せよ。

```
plot(x, y, ...)
```

10.2 時系列のラグプロット

時系列をそのラグ版に対してプロットする。自己相関が低い場合でも、「自己従属性を」を可視化するのに役立つ。

```
lag.plot(x, lags = 1, layout = NULL, set.lags = 1:lags,
         main = NULL, asp = 1,
         font.main=par("font.main"), cex.main=par("cex.main"),
         diag = TRUE, diag.col="gray", type="p", oma =NULL, ask =NULL,
         do.lines = n <= 150, labels = do.lines, ...)
```

10.3 時系列の季節成分等をプロットする

時系列の季節成分 (または、他の副系列) をプロットする。各季節 (もしくは他のカテゴリ) 毎に時系列がプロットされる。現在の作図デバイスにプロットが行われるが、返り値は無い。

```
monthplot(x, labels = NULL, times, phase, base, choice, ...)
```

10.4 複数の時系列を一つの画面にプロット

複数の時系列を一つの画面にプロットする。plot.ts とは異なり、各系列は異なった時間情報を持っても良いが、頻度は同じでなければならない。一つの時系列に対しても使えるが plot() の使用が簡単で好ましい。

```
ts.plot(..., gpars = list())
```

10.5 累積ペリオドグラムをプロットする

累積ペリオドグラムをプロットする。

```
cpgram(ts, taper=0.1, main=
       paste("Series: ", deparse(substitute(ts))), ci.col="blue")
```

11 時系列の予測

11.1 時系列の予測

predict() は様々なモデル当てはめ関数の結果から予測を行う総称的関数である。この関数は最初の引き数の class に応じて、特定のメソッドを起動する。関数 predict.lm() は線形予測関数 lm() の結果に基づいて予測を行う。

```
predict(object, ...)
```

12 状態空間モデルとカルマンフィルタ

12.1 時系列の構造モデル

時系列に構造モデル (structual model) を当てはめる。

```
StructTS(x, type = c("level", "trend", "BSM"), init = NULL,
         fixed = NULL, optim.control = NULL)
```

12.2 固定区間平滑化

状態空間モデルを用い、一変量時系列に固定区間平滑化を実行する。固定区間平滑化は、全ての観測値に基づき、各時点での最良の状態の推定を与える。

```
tsSmooth(object, ...)
```

12.3 構造モデル用補助関数

カルマンフィルタを用いて、(ガウシアン) 対数尤度を求める、また予測や平滑化を行う。

```
KalmanLike(y, mod, nit = 0)
KalmanRun(y, mod, nit = 0)
KalmanSmooth(y, mod, nit = 0)
KalmanForecast(n.ahead = 10, mod)
makeARIMA(phi, theta, Delta, kappa = 1e6)
```

13 その他

13.1 時系列を低次元空間に埋め込む

時系列 x を低次元ユークリッド空間に埋め込む。

```
embed(x, dimension = 1)
```

13.2 対称テプリッツ行列

最初の行を与えて対称テプリッツ (Toeplitz) 行列を作る。唯一の引き数 x はテプリッツ行列の最初の行である。

```
toeplitz(x)
```