# Univariate Time Series with R

**Manuel Febrero–Bande**

Dpt. de Estadística e Inv. Operativa
Univ. de Santiago de Compostela

Niteroi, 2016


DEPARTAMENTO DE ESTATÍSTICA
E INVESTIGACIÓN OPERATIVA

# Table of Contents

# Table of Contents

# Date/Time Basics Objects

- `Date`: Calendar dates (int: nº of days since 01/01/1970)
- `POSIXct, POSIXlt, POSIXt`: Calendar dates and times (seconds since 1970-01-01, readable list).
  Format, see `?strptime` for complete details
  - Day: `%a, %A`: (Abbr., Full) weekday name. `%d`: Day (01–31). `%j`: Day of year (001–366)
  - Month: `%b, %B`: (Abbr., Full) month name. `%m`: month (01–12)
  - Year: `%y, %Y`: Year (without, with) century (01–12)
  - Hour: `%H`: (00–23). `%I %p`: (01–12, AM/PM)
  - Minute: `%M`: (00–59).
  - Seconds: `%S`: (00–61)
  - Weekday: `%w`: (0–6, Sunday=0)
  - Week number: `%U,%W`: (00–53) (1st Sunday, 1st Monday)

# Example I

```
x = Sys.Date() + 0:5
format(x, "%Y-%B-%A")

> [1] "2016-mayo-viernes"   "2016-mayo-sábado"
> [3] "2016-mayo-domingo"   "2016-mayo-lunes"
> [5] "2016-mayo-martes"    "2016-mayo-miércoles"
```

```
dates = c("2012/05-01", "2012/06-15", "2012/07-01")
x = as.Date(dates, format = "%Y/%m-%d")
x

> [1] "2012-05-01" "2012-06-15" "2012-07-01"
```

# Example II

```
bir = "01/01/2003 12:01"
b1 = as.Date(bir, format = "%d/%m/%Y")
b2 = as.POSIXlt(bir, format = "%d/%m/%Y %H:%M")
today = as.POSIXlt(Sys.Date())
difftime(today, b2)

> Time difference of 4887.541 days

x1 = seq(b1, as.Date(today), by = "1 year")
x2 = seq(b2, today, by = "1 week")
head(x1)

> [1] "2003-01-01" "2004-01-01" "2005-01-01" "2006-01-01"
> [5] "2007-01-01" "2008-01-01"

head(x2)

> [1] "2003-01-01 12:01:00 CET" "2003-01-08 12:01:00 CET"
> [3] "2003-01-15 12:01:00 CET" "2003-01-22 12:01:00 CET"
```

# Other classes

- `library(date)` Simple functions for dates.(origin:1/1/1960)
- `library(chron)` Objects for representing dates and times.
  `x=chron(dates=c("02/27/92"),times=c("14:00:00"))`
  - `dates(x);times(x)` Extract dates and times
  - `seq.dates(fr,to,by="days")` # "weeks","months","years"
  - `cut.dates(x,breaks,start.on.monday=TRUE)`
  - `years(x); quarters(x); months(x); weekdays(x); days(x); hours(x); minutes(x); seconds(x)`
  - `is.weekend(x); is.holiday(x,holidays)`

# timeDate I

timeDate is a complete package for the managing of dates and times related with Rmetrics bundle.
The S4 class timeDate includes three slots: @Data (POSIXct), @format and @FinCenter.

- ```
  Dates <- c("1989-09-28","2001-01-15")
  Times <- c("23:12:55", "10:34:02")
  charvec = paste(Dates, Times)
  x=timeDate(charvec,zone="GMT")
  ```
- ```
  x=timeSequence(from = "2008-01-01", to =
  "2008-01-31", by = "day")
  # sec, min, hour, day, week, month, year (–"5mins"–)
  ```
- Information: dayOfWeek, isWeekday, isWeekend, isBizday, isHoliday

# timeDate II

- Align: `time{First|LastDay}in{Month|Quarter}(x)`; `timeNdayOnOr{After|Before}`
- Subsetting: `start`, `end`, `length`, `window`
- Reordering: `cut`, `sort`, `sample`, `unique`, `rev`
- Frequency: `isDaily`, `isMonthly`, `isQuarterly`, `isRegular`
- A lot of functions for computing holidays: `EasterSunday(2013)`; `Easter(2012:2016)`; `Ascension(2013)`; `?holidayDate`

# Table of Contents

# Time Series

A time series $\{Z_t\}_{t \in T}$ is an ordered sequence of random variables that are supposed to be dependent or correlated with respect to the index $t$ (typically time).

- Continuous: The observations are continuously observed
- Discrete: The observations are taken only at specific time intervals
  - Equal intervals: Regular Time Series
  - Unequal intervals: Irregular Time Series

## Goals

- Understanding the generating mechanism
- Optimal control of a system
- Forecasting of future values

# Considerations

- Type of time scale: daily, hourly, monthly, ...
- Recorded at regular points (daily price of a commodity) or aggregated over regular intervals (monthly industrial production)?
- Calendar problems: Months of different lengths, movable feasts and public holidays (Easter)
- Changes in the value of the money –> Deflating the value series by a suitable price index.
- Length of time series related with its features (seasonality, economic cycle)

# IBEX example

Example reading a file from http://www.bolsamadrid.es/esp/aspx/Indices/InfHistorica.aspx?grupo=IBEX

```
IB35 = read.csv("IBEX2005.dat", dec = ",", header = FALSE, sep = ";",
    as.is = TRUE)
colnames(IB35) = c("date", "Close", "Prev", "Min", "Max", "Mean")
head(IB35)
```

```
>         date  Close   Prev    Min    Max   Mean
> 1 03/01/2005 9124.1 9080.8 9154.3 9057.5 9128.6
> 2 04/01/2005 9120.0 9120.0 9139.9 9094.2 9113.1
> 3 05/01/2005 9024.4 9024.4 9066.2 9018.7 9041.8
> 4 07/01/2005 9041.1 9041.1 9072.9 9008.9 9044.0
> 5 10/01/2005 9028.5 9028.5 9058.8 9003.6 9022.3
> 6 11/01/2005 8966.4 8966.4 9052.2 8952.1 8991.3
```

```
IB35$date = as.Date(IB35$date, format = "%d/%m/%Y")
```

# Intra-day example I

```
IB35id = read.csv2("IBEX-35.csv", header = FALSE, dec = ".",
    sep = ";")
colnames(IB35id) = c("date", "hour", "value")
head(IB35id)
```

```
>       date  hour   value
> 1 20101101 90100 10881.7
> 2 20101101 90200 10884.2
> 3 20101101 90300 10880.3
> 4 20101101 90400 10885.0
> 5 20101101 90500 10878.3
> 6 20101101 90600 10877.5
```

# Intra-day example II

```
IB35id$hour = ifelse(nchar(IB35id$hour) == 5, paste("0", IB35id$hour,
    sep = ""), IB35id$hour)
IB35id$time = paste(IB35id$date, IB35id$hour, sep = "")
IB35id$time = as.POSIXct(IB35id$time, format = "%Y%m%d%H%M%S")
IB35id$fecha = as.Date(as.character(IB35id$date), format = "%Y%m%d")
head(IB35id)
```

```
>       date   hour   value                time      fecha
> 1 20101101 090100 10881.7 2010-11-01 09:01:00 2010-11-01
> 2 20101101 090200 10884.2 2010-11-01 09:02:00 2010-11-01
> 3 20101101 090300 10880.3 2010-11-01 09:03:00 2010-11-01
> 4 20101101 090400 10885.0 2010-11-01 09:04:00 2010-11-01
> 5 20101101 090500 10878.3 2010-11-01 09:05:00 2010-11-01
> 6 20101101 090600 10877.5 2010-11-01 09:06:00 2010-11-01
```

# Obtaining from Internet (Yahoo) I

`url<-"http://chart.yahoo.com/table.csv?s=BBVA.MC &a=01 &b=01 &c=2002 &d=12&e=31&f=2012 &g=d&x=.csv"`

Note 1: The *symbol* must be URLencoded.
See also `library(fImport)`

```
symbol = "BBVA.MC"  # BBVA
ini = as.Date("01/01/2002", "%d/%m/%Y")
cadini = paste("&a=", as.numeric(format(ini, "%m")) - 1, "&b=",
    format(ini, "%d"), "&c=", format(ini, "%Y"), sep = "")
fin = Sys.Date() - 1
cadfin = paste("&d=", as.numeric(format(fin, "%m")) - 1, "&e=",
    format(fin, "%d"), "&f=", format(fin, "%Y"), sep = "")
url = paste("http://chart.yahoo.com/table.csv?s=", URLencode(symbol,
    reserved = TRUE), cadini, cadfin, "&g=d&x=.csv", sep = "")
```

# Obtaining from Internet (Yahoo) II

```
bbva <- read.table(url, header = TRUE, sep = ",")
bbva <- bbva[order(bbva[, "Date"]), ]
bbva$ym = strftime(bbva$Date, format = "%Y-%m")
bbvam = aggregate(bbva$Close, by = list(bbva$ym), FUN = mean)
colnames(bbvam) = c("Date", "mClose")
tail(bbvam)
```

```
>         Date   mClose
> 168 2015-12 7.068043
> 169 2016-01 6.098762
> 170 2016-02 5.614381
> 171 2016-03 6.170435
> 172 2016-04 5.970952
> 173 2016-05 5.603714
```

# Obtaining from Internet (Google) I

```
url <-"http://www.google.com/finance/historical?q=symbol
&startdate=MMM+DD%2C+YYYY &enddate=MMM+DD%2C+YYYY
&histperiod={daily|weekly}&num=200&output=csv"
```

Note 1: The *symbol* and *dates* must be URLencoded.

# Obtaining from Internet (Google) II

```
lct = Sys.getlocale("LC_TIME")
Sys.setlocale("LC_TIME", "C")  #Names of months in English

> [1] "C"

symbol = "NASDAQ:GOOGL"  # Google
ini = as.Date("01/01/2005", "%d/%m/%Y")
cadini = paste("&startdate=", URLencode(format(ini, "%b+%d,+%Y")),
    sep = "")
fin = Sys.Date() - 1
cadfin = paste("&enddate=", URLencode(format(fin, "%b+%d,+%Y")),
    sep = "")
url = paste("http://www.google.com/finance/historical?q=", URLencode(sy
    cadini, cadfin, "&histperiod=daily&num=200&output=csv", sep = "")
```

# Obtaining from Internet (Google) III

```
goog = read.table(url, header = TRUE, sep = ",", as.is = TRUE)
colnames(goog) = c("Date", "Open", "High", "Low", "Close", "Volume")
goog$Date = as.Date(goog$Date, format = "%d-%b-%y")
Sys.setlocale("LC_TIME", lct)

> [1] "Spanish_Spain.1252"

tail(goog)

>            Date   Open   High   Low  Close   Volume
> 2859 2005-01-10  97.35  99.15 96.01  97.63  7554795
> 2860 2005-01-07  95.42  97.22 94.48  97.02  9666175
> 2861 2005-01-06  97.72  98.05 93.95  94.37 10389803
> 2862 2005-01-05  96.82  98.55 96.21  96.85  8239545
> 2863 2005-01-04 100.77 101.57 96.84  97.35 13762396
> 2864 2005-01-03  98.80 101.92 97.83 101.46 15860692

# for (i in 2:6){goog[,i]=as.numeric(goog[,i])} # Check
# variates
```

# library(quantmod) I

This package has functions to consult financial data from several sources: yahoo, google, oanda, FRED,...
Also includes functions for Technical Analysis. See `?TA`

```
library(quantmod)
getSymbols("GOOG", src = "yahoo")  #New object GOOG of class xts

> [1] "GOOG"

# getQuote('YHOO') #getSplits('YHOO') #getDividends('YHOO')
getMetals("gold", from = "2012/03/01")  # Metals object XAUUSD

> [1] "XAUUSD"

getFX("EUR/USD", from = "2005-01-01")  #Only last five years

> [1] "EURUSD"
```

# EUR/USD example I

```
ini = as.Date("01/01/2002", "%d/%m/%Y")
fin = Sys.Date()
r = c(seq.Date(ini, fin, by = "5 year"), fin)
getFX("EUR/USD", from = r[1], to = r[2] - 1)

> [1] "EURUSD"

euro = EURUSD
for (i in 2:(length(r) - 1)) {
    getFX("EUR/USD", from = r[i], to = r[i + 1] - 1)
    euro = rbind(euro, EURUSD)
}
eurom = as.data.frame(aggregate(euro, by = list(strftime(index(euro),
    format = "%Y-%m")), FUN = mean))
colnames(eurom) = "EURUSD"
```

# EUR/USD example II

```
chartSeries(euro, TA = c(addBBands(), addSMA(n = 20)), theme = chartThe
```

# Time Series Objects

`stats::ts`: This is the most simple object for regular time series.

**Example**

```
x=ts(data=1:24,frequency=4,start=c(2001,1))
#Data by quarters begining Q1, 2001
```

**Parameters**

- `data`: Vector (univariate) or Matrix (multivariate)
- `frequency`| `deltat`: Number of observations per unit of time| fraction of the sampling period between successive observations.
- `start`|`end`: Time of the first observation | Time of the last observation.

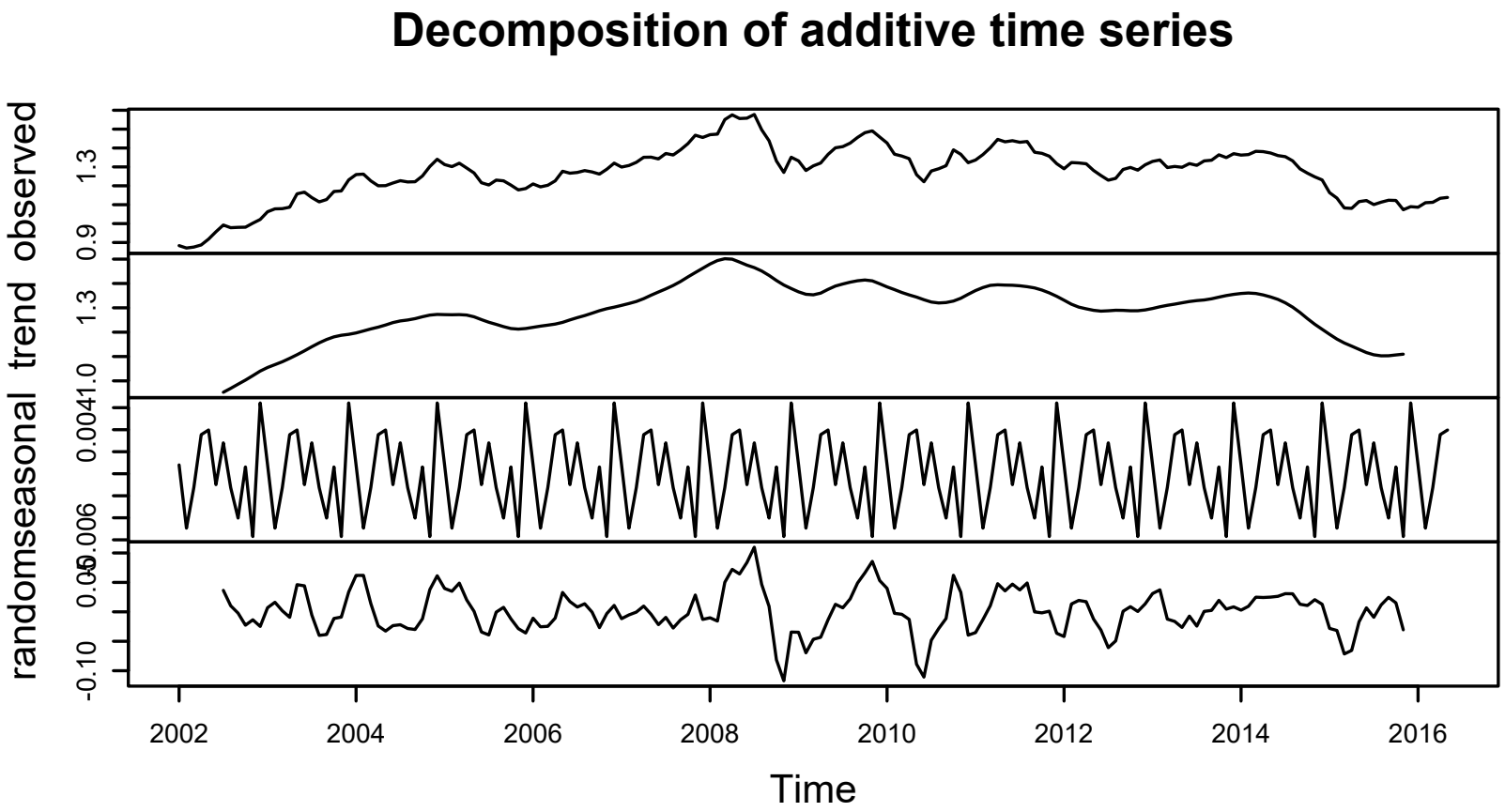
# Functions for ts objects

- `tsp(x)`: Start, End and Frequency
- `time(x)`: Times for the time series.
- `cycle(x)`: Position in the cycle.
- `frequency(x)`: Number of samples per unit time.
- `window(x,start=c(2001,3),end=c(2003,4))`: Subset of a time series.
- `cbind.ts, ts.union, ts.intersect` : Merge time series
- `embed(x,3)`:
- `filter(x,filter=c(1,1,1)/3)`: Average Mean
- `filter(x,c(1,1,1),method="recursive")`: AR filter

# Example ts objects

Exploratory: `lag.plot`, `monthplot`, `stl`, `HoltWinters`,...

```
euro.ts = ts(eurom, start = c(2002, 1), frequency = 12) plot(decompose(
```

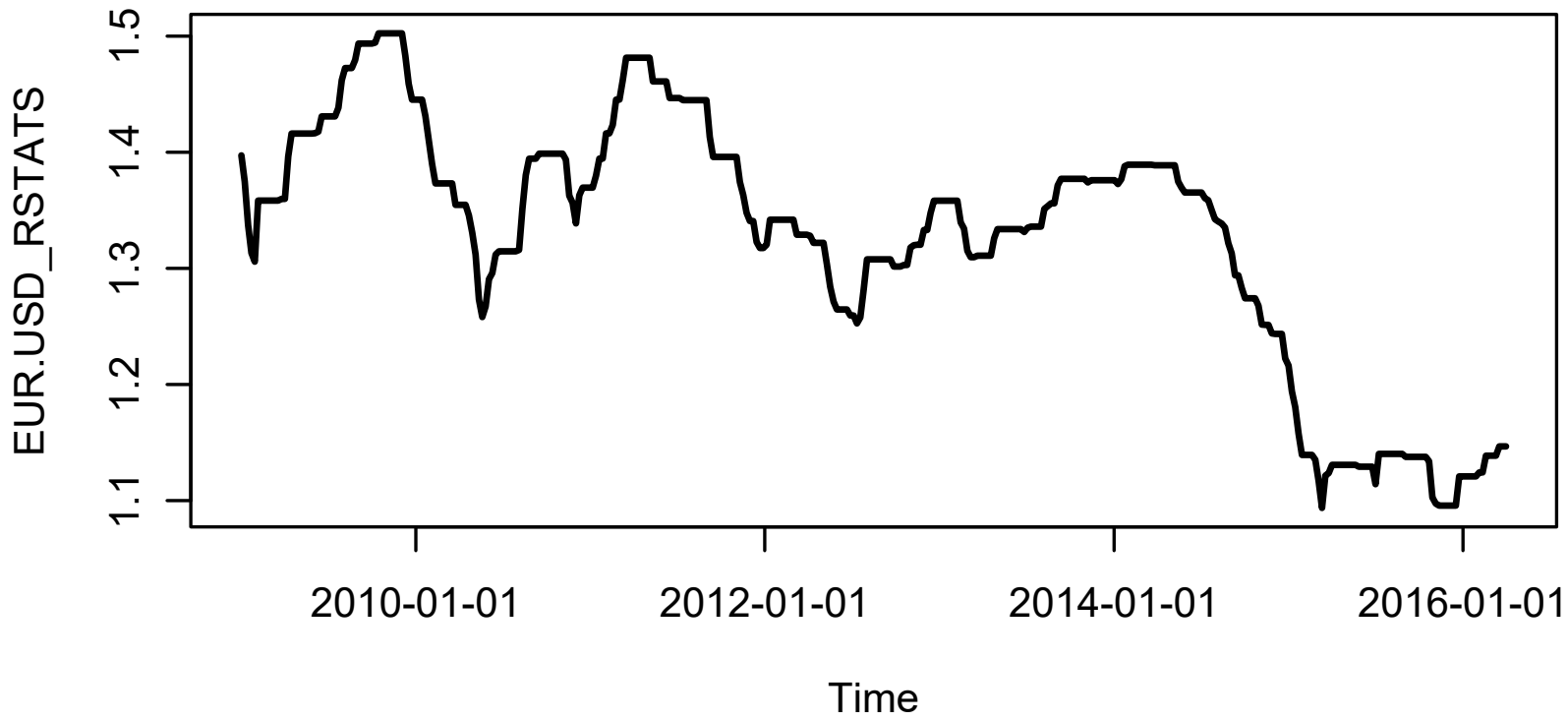


# timeSeries objects (S4) I

The `timeSeries` object: Provides better date representation (timeDate, fBasics).
Slots: `positions`, `.Data`, `format`, `unit`, `title`, `FinCenter`
Utilities: `durations`, `returns`, `spreads`, `scale`
Exploratory: `colCum{maxs|mins|prods|returns|sums}`, `colStats`, `rowStats`, `rollStats`, `ranks`, `runlengths`

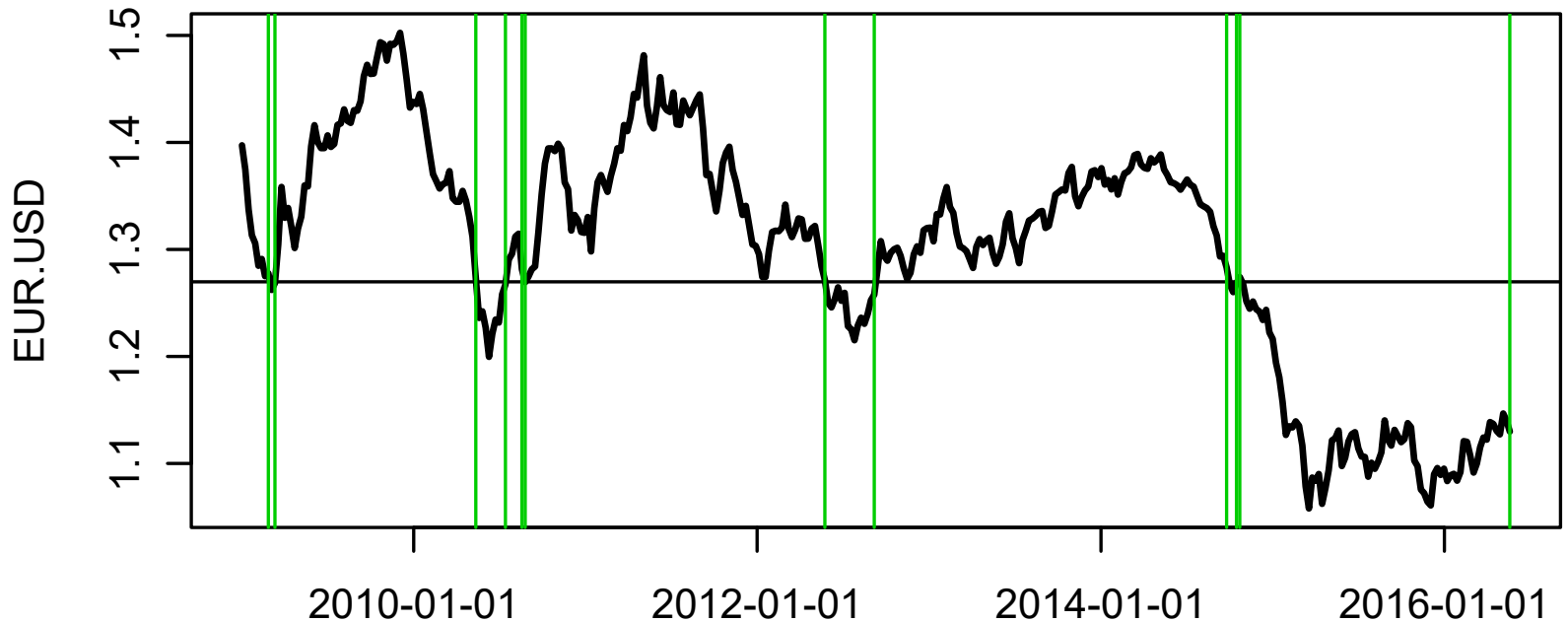
```
library(timeSeries)
eurusd = timeSeries(euro, charvec = index(euro))
eusd = window(eurusd, "2009-01-01", end(eurusd))  #Daily data
by = timeSequence(start(eusd), end(eusd), by = "week")
eusdw = aggregate(eusd, by, FUN = mean)  #Weekly data
plot(rollStats(eusdw, 8, FUN = max), lwd = 2)  # Max. 8 weeks
```

# timeSeries objects (S4) II



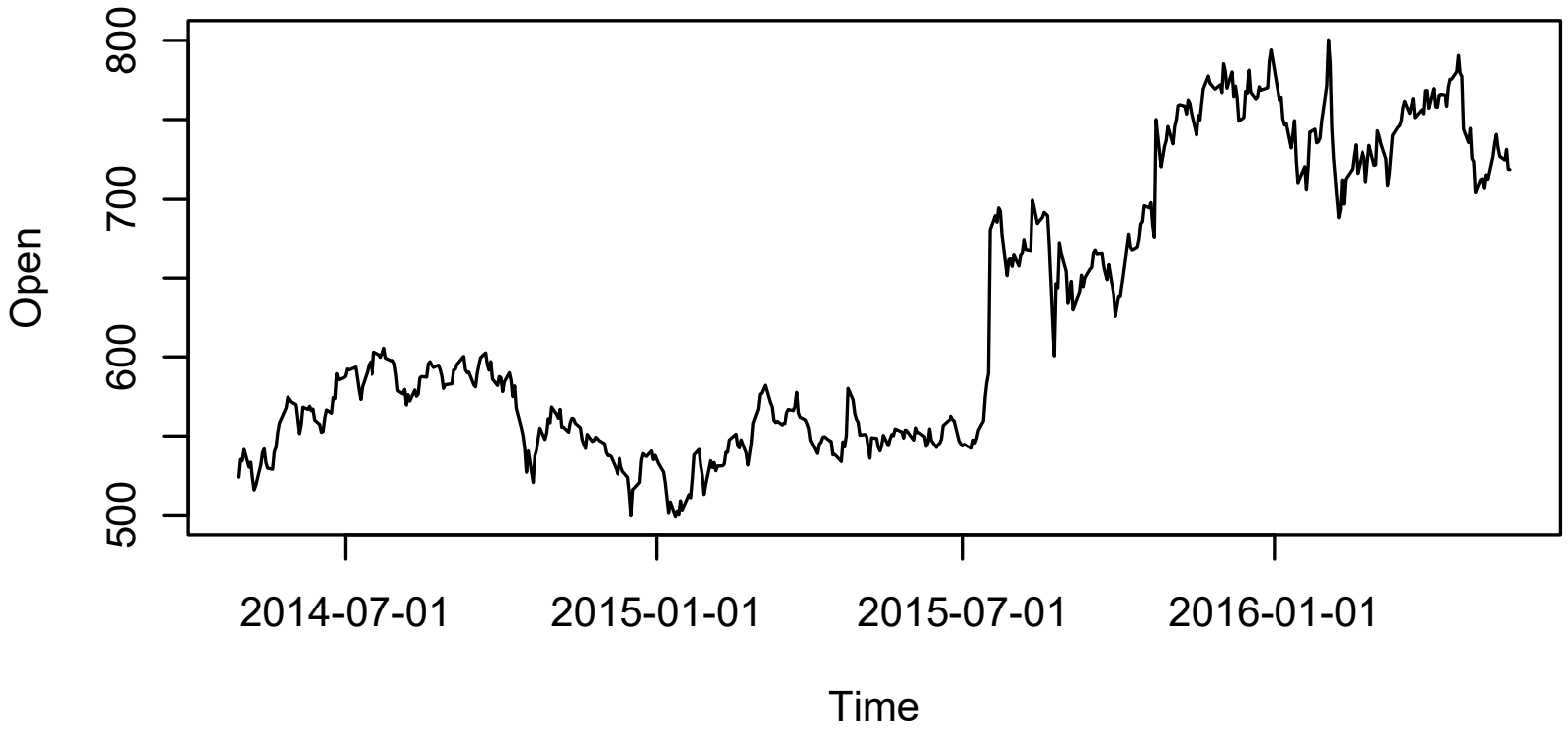
# Examples timeSeries

```
x = eusdw - median(eusdw)
r = runlengths(x)  # Runs below/over median
plot(eusdw, lwd = 2)
abline(h = median(eusdw))
abline(v = time(r), col = 3)
```



# Plot

```
goog.tS = timeSeries(goog[, 2:5], goog[, "Date"])
plot(head(goog.tS[, "Open"], 520), plot.type = "single")   #Last 2 years
```



# Stationary Time Series

A trajectory of length $n$ is observed: $\{z_t\}_{t=1}^n$.
A time series is said *stationary (2nd order)* if

$$\begin{aligned} \mathbb{E}[Z_t] &= \mu \quad \forall t \\ \mathrm{Cov}(Z_t, Z_{t-j}) &= \mathbb{E}[(Z_t-\mu)(Z_{t-j}-\mu)] = \gamma_j \quad \forall t, j \\ \rho_j &= \frac{\mathrm{Cov}(Z_t, Z_{t-j})}{\sqrt{\mathrm{Var}(Z_t)\,\mathrm{Var}(Z_{t-j})}} = \frac{\gamma_j}{\gamma_0} \end{aligned}$$

Classical decomposition:
$X_t = T_t + S_t + Z_t$ where $T_t$ is the long-term trend (deterministic), $S_t$ is the seasonal component (deterministic) and $Z_t$ is the residual, irregular or random effect (stochastic).

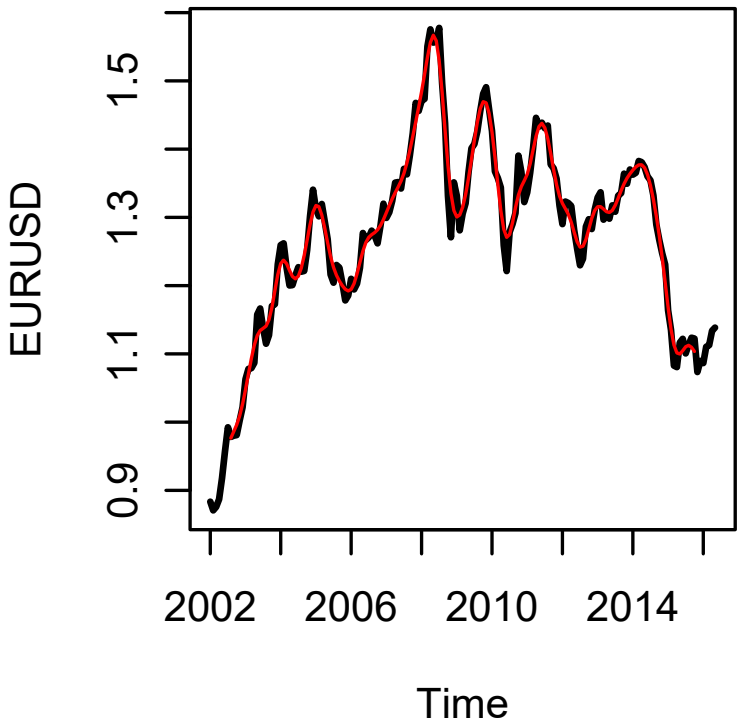
# Trend/Seasonality/Variance Stabilization

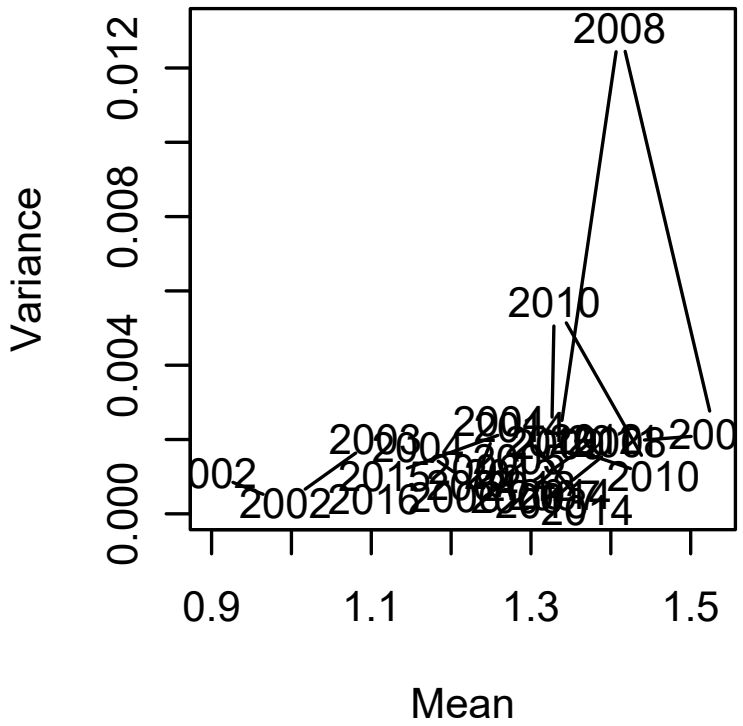
- Deterministic: $\hat{T}_t = \alpha_0 + \alpha_1 t + \ldots + \alpha_k t^k$,
  $\hat{T}_t = \nu_0 + \sum_{j=1}^{m} (\alpha_j \cos \omega_j t + \beta_j \sin \omega_j t)$, $\omega_j = 2\pi/p_j$
  z=lm(euro.ts~poly(as.numeric(time(euro.ts)),3)
  z=gls(euro.ts~poly(as.numeric(time(euro.ts)),3),
  correlation=corARMA(p=2))#Better with gls
- Moving averages: $\hat{T}_t = \sum_{j=-m}^{m} \alpha_j z_{t-j}$
  filter(euro.ts,c(-3,-6,-5,3,21,46,67,74,67,46,21,3,-5,-6,-3)/320)
  #Spencer's 15 point
- Seasonality:
  z=lm(euro.ts~time(euro.ts)+as.factor(cycle(euro.ts))
- Variance stabilization:
  $\mathrm{Var}\,(Z_t) = f(\mu_t) \implies T(Z_t) = \int \frac{1}{\sqrt{f(\mu_t)}} d\mu_t$
  m=aggregate(euro.ts,nfrequency=2,FUN=mean) #6m.
  v=aggregate(euro.ts,nfrequency=2,FUN=var)
  plot(m,v) # Any relationship? Box-Cox transformation?

# Example Trend/Seasonality/Variance Stabilization I

```
library(nlme)
z.lm = lm(euro.ts ~ poly(as.numeric(time(euro.ts)), 2))
z.gls = gls(euro.ts ~ poly(as.numeric(time(euro.ts)), 2), correlation =
Sp.15 = filter(euro.ts, c(-3, -6, -5, 3, 21, 46, 67, 74, 67,
    46, 21, 3, -5, -6, -3)/320)  #Spencer's 15 point
z.seas = lm(euro.ts ~ time(euro.ts) + as.factor(cycle(euro.ts)))
m = aggregate(euro.ts, nfrequency = 2, FUN = mean)
v = aggregate(euro.ts, nfrequency = 2, FUN = var)
# summary(z.seas)
```

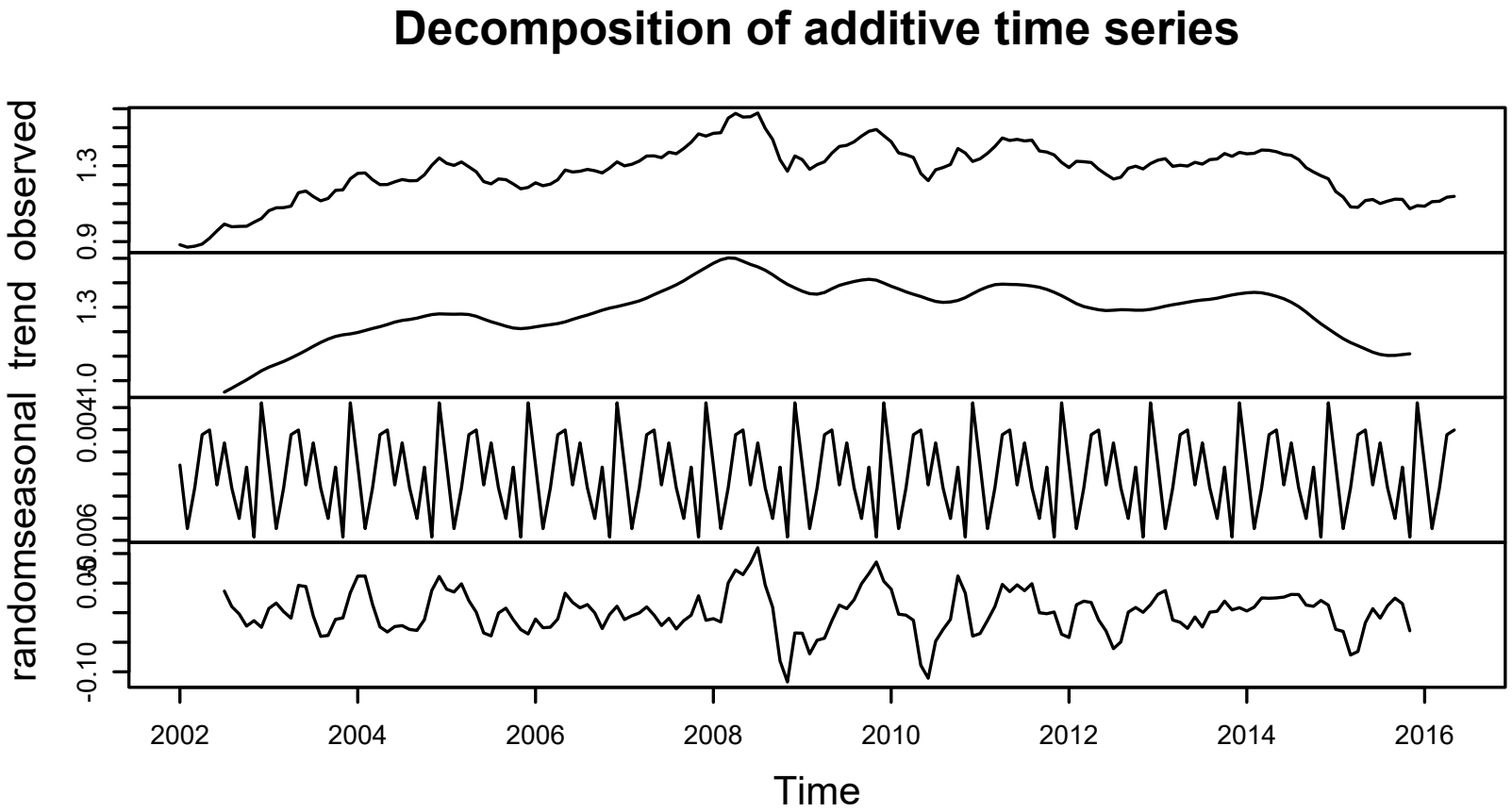
# Example Trend/Seasonality/Variance Stabilization II



# Decomposition of a Time Series

```
plot(decompose(euro.ts))
```



# Sample Estimators

- Sample mean: $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} z_t$
- Sample variance: $\hat{\gamma}_0 = \frac{1}{n}\sum_{i=1}^{n}(z_t - \hat{\mu})^2$
- Sample autocovariance: $\hat{\gamma}_j = \frac{1}{n-j}\sum_{i=1}^{n-j}(z_t - \hat{\mu})(z_{t+j} - \hat{\mu})$
- Sample autocorrelation: $\hat{\rho}_j = \frac{\hat{\gamma}_j}{\hat{\gamma}_0}$

A series is said ergodic if all sample moments converge to its theoretical counterparts.

- $\mathrm{Var}\,(\hat{\mu}) = \frac{\gamma_0}{n}\left\{1 + 2\sum_{j=1}^{n-1}\left(1 - \frac{j}{n}\right)\rho_j\right\}$
- $\mathbb{E}\,[\hat{\gamma}_j] \approx \gamma_j - \mathrm{Var}\,(\hat{\mu})$;
  $\mathrm{Var}\,(\hat{\gamma}_j) \approx \frac{1}{n-j}\sum_{i=-\infty}^{\infty}\left(\gamma_i^2 + \gamma_{i+j}\gamma_{i-j}\right)$;
  $\mathrm{Cov}\,(\hat{\gamma}_j, \hat{\gamma}_{j+k}) \approx \frac{1}{n-j}\sum_{i=-\infty}^{\infty}\left(\gamma_i\gamma_{i+k} + \gamma_{i+j+k}\gamma_{i-j}\right)$
- $\mathrm{Var}\,(\hat{\rho}_j) \approx \frac{1}{n}\sum_{i=-\infty}^{\infty}\left(\rho_i^2 + \rho_{i+j}\rho_{i-j} - 4\rho_j\rho_i\rho_{i-j} + 2\rho_j^2\rho_i^2\right)$
  Approx. Bartlett: If $\rho_j = 0, j > m$, then
  $\mathrm{Var}\,(\hat{\rho}_j) \approx \frac{1}{n}\left(1 + 2\rho_1^2 + \ldots + 2\rho_m^2\right)$

# Partial autocorrelation function (PACF)

The PACF is the correlation between $(z_t, z_{t+j})$ given the information of $(z_{t+1}, \ldots, z_{t+j-1})$

$$P_j = \frac{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{j-2} & \rho_1 \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{j-3} & \rho_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho_{j-1} & \rho_{j-2} & \rho_{j-3} & \cdots & \rho_1 & \rho_j \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{j-2} & \rho_{j-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{j-3} & \rho_{j-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho_{j-1} & \rho_{j-2} & \rho_{j-3} & \cdots & \rho_1 & 1 \end{vmatrix}}$$

# Sample PACF

The sample PACF is obtained substituting $\rho_i$ by $\hat{\rho}_i$ in the previous equation or using the Durbin algorithm

$$\hat{P}_{j+1} = \frac{\hat{\rho}_{j+1} - \sum_{i=1}^{j} p_{ji}\hat{\rho}_{j+1-i}}{1 - \sum_{i=1}^{j} p_{ji}\hat{\rho}_j}$$

with $p_{j+1,i} = p_{ji} - \hat{P}_{j+1}p_{j,j+1-i}$, and $p_{ii} = \hat{P}_i$ for $i = 1, \ldots, j$.

Under the hypothesis of non temporal dependence

$$\mathrm{Var}\left(\hat{P}_j\right) \approx \frac{1}{n}$$

# Table of Contents

# ARMA Models

Let $\epsilon_t$ a white noise process (i.e. $\mathbb{E}[\epsilon_t] = 0$, $\mathrm{Var}(\epsilon_t) = \sigma^2$, and $\rho_j = 0, \forall j \neq 0$ ) and $B$ the backward operator $B^j(z_t) = z_{t-j}$

## AR(p) models

$$z_t = c + \phi_1 z_{t-1} + \ldots + \phi_p z_{t-p} + \epsilon_t$$
$$\Phi_p(B) z_t = c + \epsilon_t \text{ with } \Phi_p(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$$

## MA(q) models

$$z_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_p \epsilon_{t-q}$$
$$z_t = \mu + \Theta_q(B)\epsilon_t \text{ with } \Theta_q(B) = 1 + \theta_1 B + \ldots + \theta_q B^q$$

## ARMA(p,q) models

$$z_t = c + \phi_1 z_{t-1} + \ldots + \phi_p z_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_p \epsilon_{t-q}$$
$$\Phi_p(B) z_t = c + \Theta_q(B)\epsilon_t$$

# Stationarity and Invertibility

The conditions about stationarity and invertibility depends on the roots of the characteristic polynomials $(\Phi_p(B), \Theta_q(B))$

## Stationarity

The roots of $\Phi_p(B)$ lie outside unit circle.

Non stationarity

- Any root greater than one –> Explosive AR model
- Unit root (i.e. $|r_i| = 1$) –> Difference $(\nabla z_t = z_t - z_{t-1})$
- Deterministic trend –> Estimate or difference. (Differencing $d$ times can neutralize polynomial trends up to degree $d$)

## Invertibility

The roots of $\Theta_q(B)$ lie outside unit circle.

With no unit roots an equivalent invertible formulation can be obtained.

# ARMA Identification

## AR(p) models

- ACF: $\rho_k = \sum_{i=1}^{m} G_i^k \sum_{j=1}^{d_i-1} A_{ij} k^j$ with $G_i^{-1}$ the roots of $\Phi_p(B) = 0 \rightarrow$ Mixture of exponential and/or sinusoidal decays.
- PACF: $P_k \neq 0, k \leq p$, and $P_k = 0, k > p$

## MA(q) models

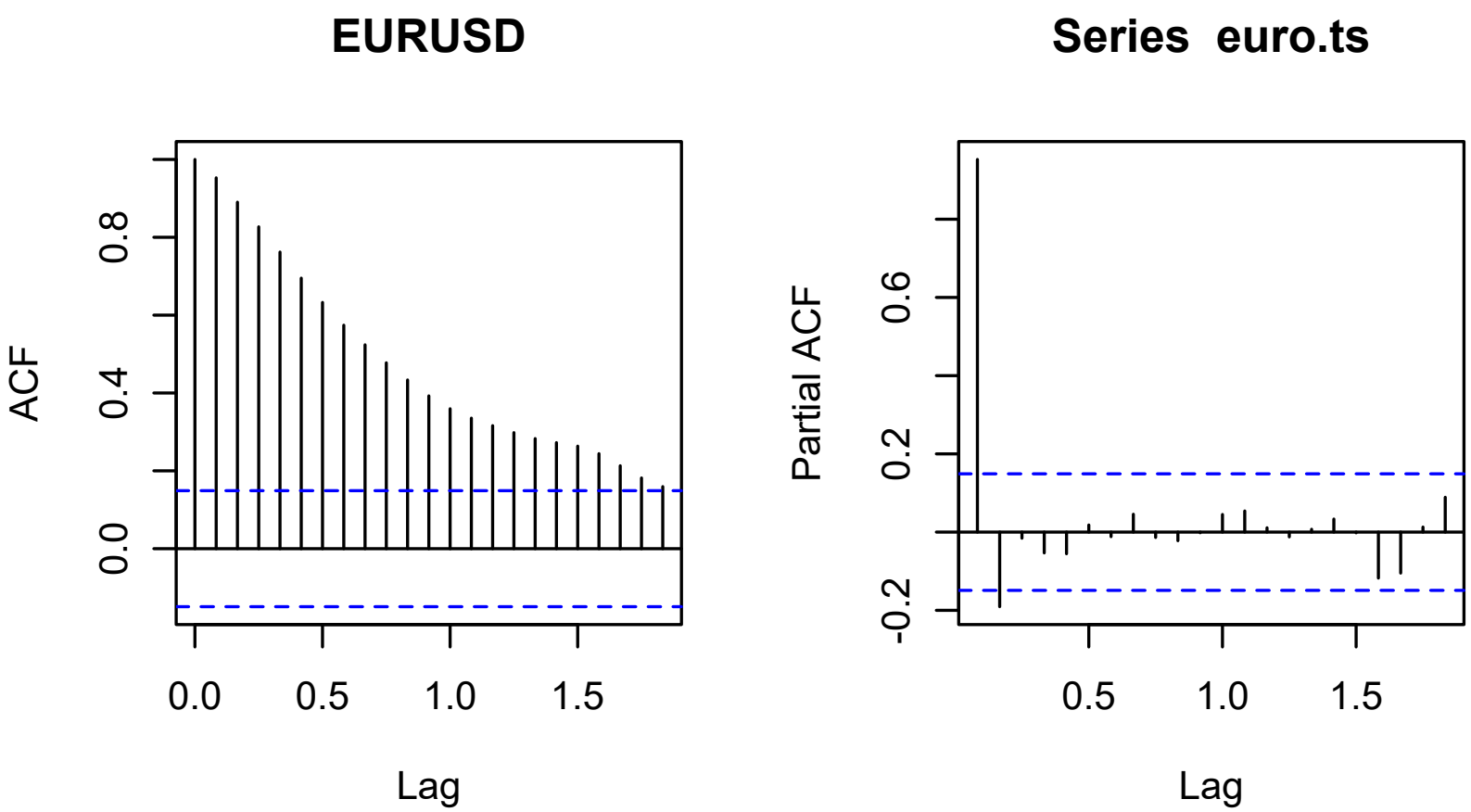
- ACF: $\rho_k \neq 0, k \leq q$, and $\rho_k = 0, k > q$
- PACF: Mixture of exponential and/or sinusoidal decays.

## ARMA(p,q) models

- ACF: First $q$ non zero autocorrelations and then mixture of exponential and/or sinusoidal decays.
- PACF: First $p$ non zero autocorrelations and then mixture of exponential and/or sinusoidal decays.

# Model Identification

```
par(mfrow = c(1, 2)) acf(euro.ts, ci.type = "white") pacf(euro.ts)
```



# Simple AR Estimation

## Example (Code)

```
res.ar = ar(euro.ts[, 1], 6, method = "mle")   #meth=c('yw','burg','ols'
res.ar

>
> Call:
> ar(x = euro.ts[, 1], aic = 6, method = "mle")
>
> Coefficients:
>       1        2
>  1.2881  -0.3099
>
> Order selected 2  sigma^2 estimated as  0.000855
```

# Box test and diagnostic

## Example (Box Ljung test)

```
Box.test(res.ar$resid, lag = 10, "Ljung-Box")

>
>  Box-Ljung test
>
> data:  res.ar$resid
> X-squared = 10.703, df = 10, p-value = 0.3811

abs(polyroot(c(1, -res.ar$ar)))

> [1] 1.033128 3.123633
```

# Unit Root Tests (I)

Augmented Dickey-Fuller test:

$z_t = \beta^t \mathbf{D}_t + \phi z_{t-1} + \sum_{j=1}^{p} \psi_j \Delta z_{t-j} + \epsilon_t$ or

$\Delta z_t = \beta^t \mathbf{D}_t + \pi z_{t-1} + \sum_{j=1}^{p} \psi_j \Delta z_{t-j} + \epsilon_t$

where $\mathbf{D}_t$ is a vector or deterministic terms (constant, trend) and $\{\epsilon_t\}$ is an homoskedastic noise process.

$H_0 : z_t \sim I(1) \rightarrow \phi = 1 \rightarrow \pi = 0$ vs. $H_1 : z_t \sim I(0) \rightarrow |\phi| < 1$

Test:

$$ADF_t = \frac{(\hat{\phi} - 1)}{SE(\phi)} \xrightarrow{H_0} \frac{\int_0^1 W(r) dW(r)}{\left( \int_0^1 W(r)^2 dr \right)^{1/2}}$$

$$ADF_n = \frac{n(\hat{\phi} - 1)}{1 - \sum_{j=1}^{p} \psi_j} \xrightarrow{H_0} \frac{\int_0^1 W(r) dW(r)}{\int_0^1 W(r)^2 dr}$$

# Example Unit Root

```
library(urca)
summary(ur.df(eusdw, type = "drift", selectlags = "AIC"))

....

> Coefficients:
>              Estimate Std. Error t value Pr(>|t|)
> (Intercept)  0.012398   0.009378   1.322 0.186920
> z.lag.1     -0.009928   0.007188  -1.381 0.168014
> z.diff.lag   0.195395   0.050244   3.889 0.000119 ***
> Value of test-statistic is: -1.3812 1.1741
>
> Critical values for test statistics:
>       1pct  5pct 10pct
> tau2 -3.44 -2.87 -2.57
> phi1  6.47  4.61  3.79
```

# Unit Root Tests (II)

Philips-Perron test: $\Delta z_t = \beta^t \mathbf{D}_t + \pi z_{t-1} + u_t$ where $\{u_t\}$ is an $I(0)$ process (possibly serial correlated and heteroskedastic).
$H_0 : z_t \sim I(1) \rightarrow \phi = 1 \rightarrow \pi = 0$ vs. $H_1 : z_t \sim I(0) \rightarrow |\phi| < 1$
Test:

$$Z_t = \left(\frac{\hat{\sigma}^2}{\hat{\lambda}^2}\right)^{1/2} ADF_t - \frac{1}{2}\left(\frac{\hat{\lambda}^2 - \hat{\sigma}^2}{\hat{\lambda}^2}\right)\left(\frac{nSE(\hat{\pi})}{\hat{\sigma}^2}\right)$$

$$Z_\pi = n\hat{\pi} - \frac{n^2 SE(\hat{\pi})}{2\hat{\sigma}^2}\left(\hat{\lambda}^2 - \hat{\sigma}^2\right)$$

where $\hat{\sigma}^2$ and $\hat{\lambda}^2$ are consistent estimates of the residual variance.
$\sigma^2 = \lim_{n\to\infty} n^{-1} \sum_{i=1}^n \mathbb{E}\left[u_i^2\right]$, and long-run variance
$\lambda^2 = \lim_{n\to\infty} \sum_{i=1}^n \mathbb{E}\left[n^{-1} S_n^2\right]$ with $S_n = \sum_{i=1}^n u_i$

# Example Unit Root (II)

```
summary(ur.pp(eusdw, model = "constant", lags = "short"))

....

> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) 0.009751   0.009516   1.025    0.306
> y.l1        0.991966   0.007294 136.000   <2e-16 ***
>
>          aux. Z statistics
> Z-tau-mu             1.3085
```

# Unit Root Tests (III)

Elliot, Rothenberg and Stock P-test: $z_t = \beta^t \mathbf{D}_t + u_t$,
$u_t = \phi u_{t-1} + v_t$ where $\{v_t\}$ is 1-summable linear process with long-run variance $\lambda^2$.
$H_0 : z_t \sim I(1) \rightarrow \phi = 1$ vs. $H_1 : z_t \sim I(0) \rightarrow \bar{\phi} = 1 - \bar{c}/n, \bar{c} < 0$
Test:

$$\begin{aligned}
\mathbf{z}_\phi &= (z_1, z_2 - \phi z_1, \ldots, z_n - \phi z_{n-1})^t \\
\mathbf{D}_\phi &= \left(\mathbf{D}_1^t, \mathbf{D}_2^t - \phi \mathbf{D}_1^t, \ldots, \mathbf{D}_n^t - \phi \mathbf{D}_{n-1}^t\right)^t \\
\mathbf{S}(\phi) &= \tilde{\mathbf{z}}_\phi^t \tilde{\mathbf{z}}_\phi \\
P_n &= \left[\mathbf{S}(\bar{\phi}) - \bar{\phi}\mathbf{S}(1)\right] / \hat{\lambda}^2 \xrightarrow{H_0} \text{Tabulated}
\end{aligned}$$

with $\tilde{\mathbf{z}}_\phi = \mathbf{z}_\phi - \mathbf{D}_\phi \hat{\beta}_\phi$ and $\hat{\beta}_\phi = \left(\mathbf{D}_\phi^t \mathbf{D}_\phi\right)^{-1} \mathbf{D}_\phi^t z_\phi$

# Example Unit Root (III)

```
summary(ur.ers(eusdw, type = "P-test", model = "constant"))
```

```
>
> ###############################################
> # Elliot, Rothenberg and Stock Unit Root Test #
> ###############################################
>
> Test of type P-test
> detrending of series with intercept
>
> Value of test-statistic is: 10.5199
>
> Critical values of P-test are:
>                 1pct 5pct 10pct
> critical values 1.99 3.26  4.48
```

# Unit Root Tests (IV)

Elliot, Rothenberg and Stock DF-GLS test:

Detrend data: $z_t^d = z_t - \beta_{\bar{\phi}}^t \mathbf{D}_t$ and then construct the same model as in ADF test: $\Delta z_t^d = \pi z_{t-1}^d + \sum_{j=1}^p \psi_j \Delta z_{t-j}^d + \epsilon_t$

$H_0 : z_t \sim I(1) \rightarrow \pi = 0$ vs. $H_1 : z_t \sim I(0) \rightarrow \pi = -\bar{c}/n, \bar{c} < 0$

Test: Compute the $t$-statistic for testing $\pi = 0$ $(ADF_t)$.

When $\mathbf{D}_t = 1$, ERS showed that the asymptotic distribution of the DF-GLS test under the null is the same as the ADF $t$-test but with higher asymptotic power under local alternatives.

When $\mathbf{D}_t = (1, t)$, the asymptotic distribution is different from ADF $t$-test and also this test has higher power against local alternatives.

# Example Unit Root (IV)

```
summary(ur.ers(eusdw, type = "DF-GLS", model = "constant"))

....

>
> Coefficients:
>              Estimate Std. Error t value Pr(>|t|)
> yd.lag       -0.003981   0.005804  -0.686 0.493201
> F-statistic:  2.54 on 5 and 376 DF,  p-value: 0.0281
>
>
> Value of test-statistic is: -0.6859
>
> Critical values of DF-GLS are:
>                  1pct  5pct 10pct
> critical values -2.57 -1.94 -1.62
```

# Stationarity Test

Kwiatkowski, Phillips, Schmidt & Shin test:

$$\begin{aligned} z_t &= \beta^t \mathbf{D}_t + \mu_t + u_t \\ \mu_t &= \mu_{t-1} + \epsilon_t \end{aligned}$$

where $\{u_t\}$ is an $I(0)$ process (possibly serial correlated and heteroskedastic) and $\{\epsilon_t\}$ is a noise process.
Test: $H_0 : z_t \sim I(0) \rightarrow \sigma_\epsilon = 0$ vs. $H_1 : z_t \sim I(1) \rightarrow \sigma_\epsilon > 0$

$$\text{KPSS} = \left( n^{-2} \sum_{i=1}^{n} \hat{S}_i^2 \right) / \hat{\lambda}^2 \xrightarrow{d} \int_0^1 V_k(r) dr$$

$(\mathbf{D}_t = 1)$, $V_1(r) = W(r) - rW(1)$ and
$(\mathbf{D}_t = (1, t)^t)$, $V_2(r) = W(r) + r(2 - 3r)W(1) + 6r(r^2 - 1) \int_0^1 W(s) ds$

# Example Stationarity Test

```
summary(ur.kpss(eusdw, type = "mu", lags = "short"))
```

```
>
> ######################
> # KPSS Unit Root Test #
> ######################
>
> Test is of type: mu with 5 lags.
>
> Value of test-statistic is: 3.3411
>
> Critical value for a significance level of:
>                 10pct  5pct 2.5pct  1pct
> critical values 0.347 0.463  0.574 0.739
```

# ARIMA Estimation I

## AR(3)

```
res.arma = arima(euro.ts, order = c(3, 0, 0))
res.arma
```

```
>
> Call:
> arima(x = euro.ts, order = c(3, 0, 0))
>
> Coefficients:
>          ar1      ar2     ar3  intercept
>       1.3053  -0.3897  0.0649     1.1928
> s.e.  0.0756   0.1215  0.0767     0.1035
>
> sigma^2 estimated as 0.0008493:  log likelihood = 364.35,  aic = -718
```

# ARIMA Estimation II

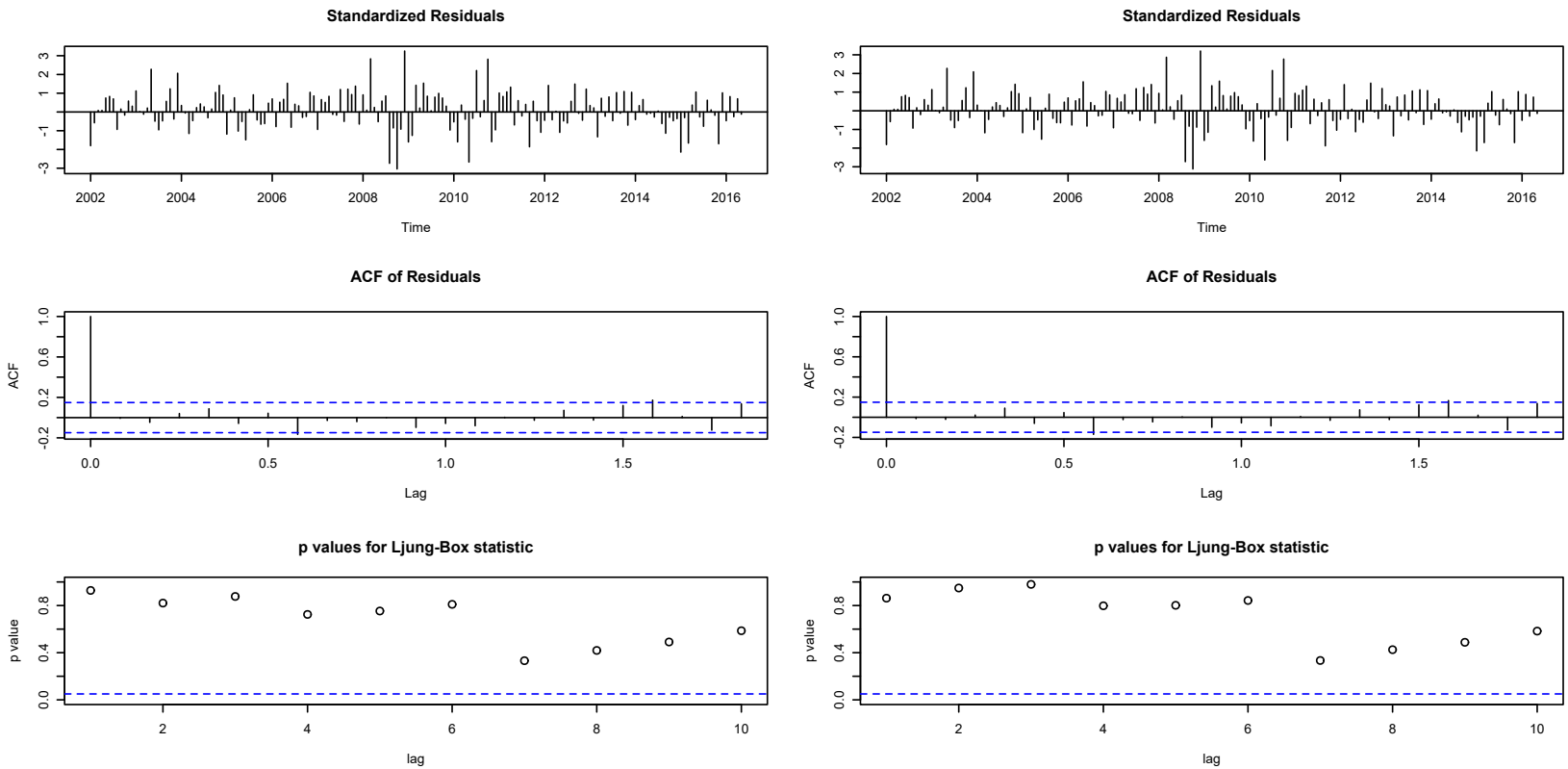
## ARMA(2,1)

```
res.arma2 = arima(euro.ts, order = c(2, 0, 1))
res.arma2  #Better AIC
```

```
>
> Call:
> arima(x = euro.ts, order = c(2, 0, 1))
>
> Coefficients:
>          ar1      ar2     ma1  intercept
>       1.0388  -0.0638  0.2723     1.1941
> s.e.  0.2180   0.2167  0.2086     0.1019
>
> sigma^2 estimated as 0.0008472:  log likelihood = 364.56,  aic = -719
```

# ARIMA diagnosis

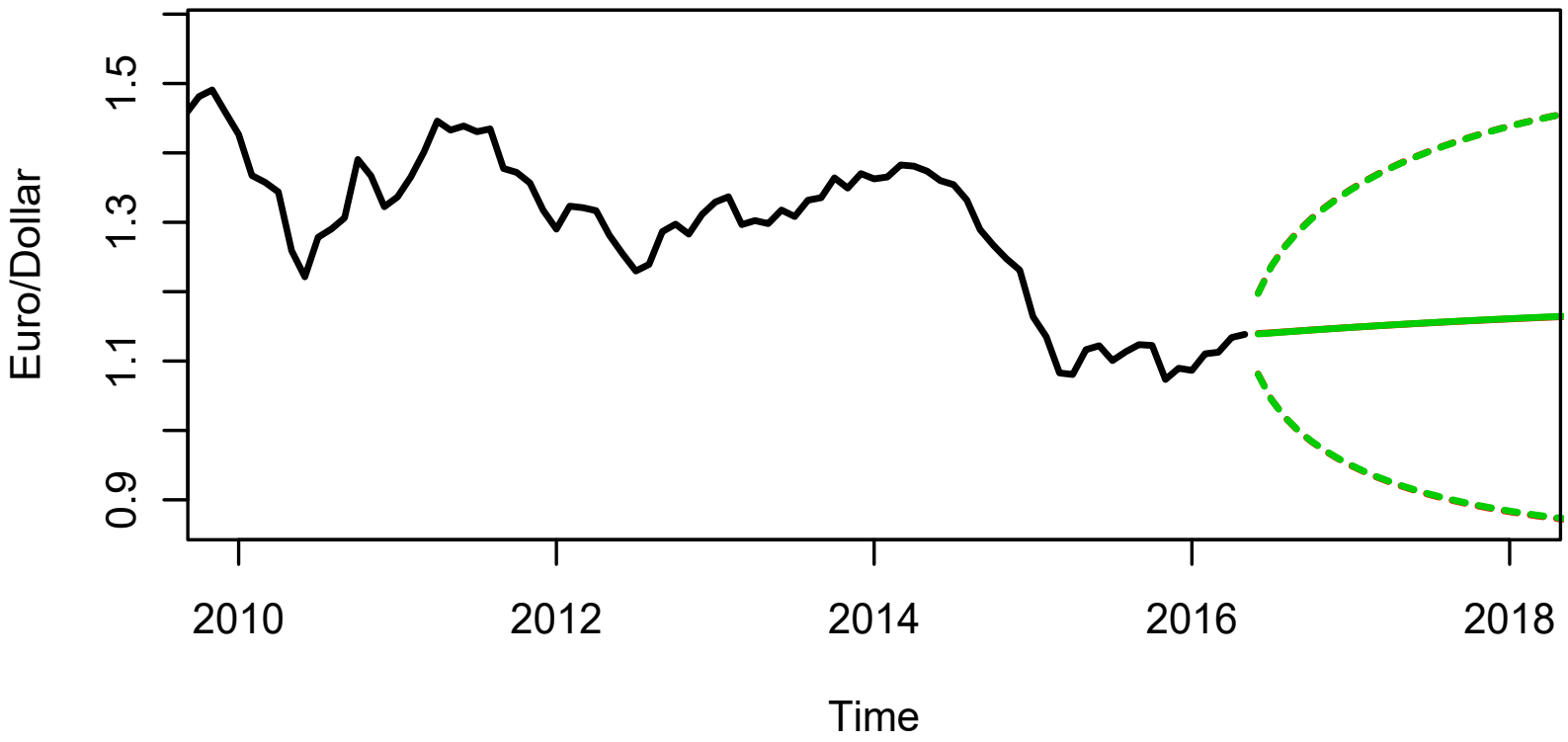
```
tsdiag(res.arma)
tsdiag(res.arma2)
```



# ARIMA Forecasting I

```
pr = predict(res.arma, n.ahead = 24)
pr2 = predict(res.arma2, n.ahead = 24)
A = pr[["se"]] %*% matrix(c(-2, 2), nrow = 1)
A2 = pr2[["se"]] %*% matrix(c(-2, 2), nrow = 1)
pr.ts = cbind(pr[["pred"]], pr[["pred"]] + A[, 1], pr[["pred"]] +
    A[, 2])
pr2.ts = cbind(pr2[[1]], pr2[[1]] + A2[, 1], pr2[[1]] + A2[,
    2])
```

# ARIMA Forecasting II



# ARIMA Simulation I

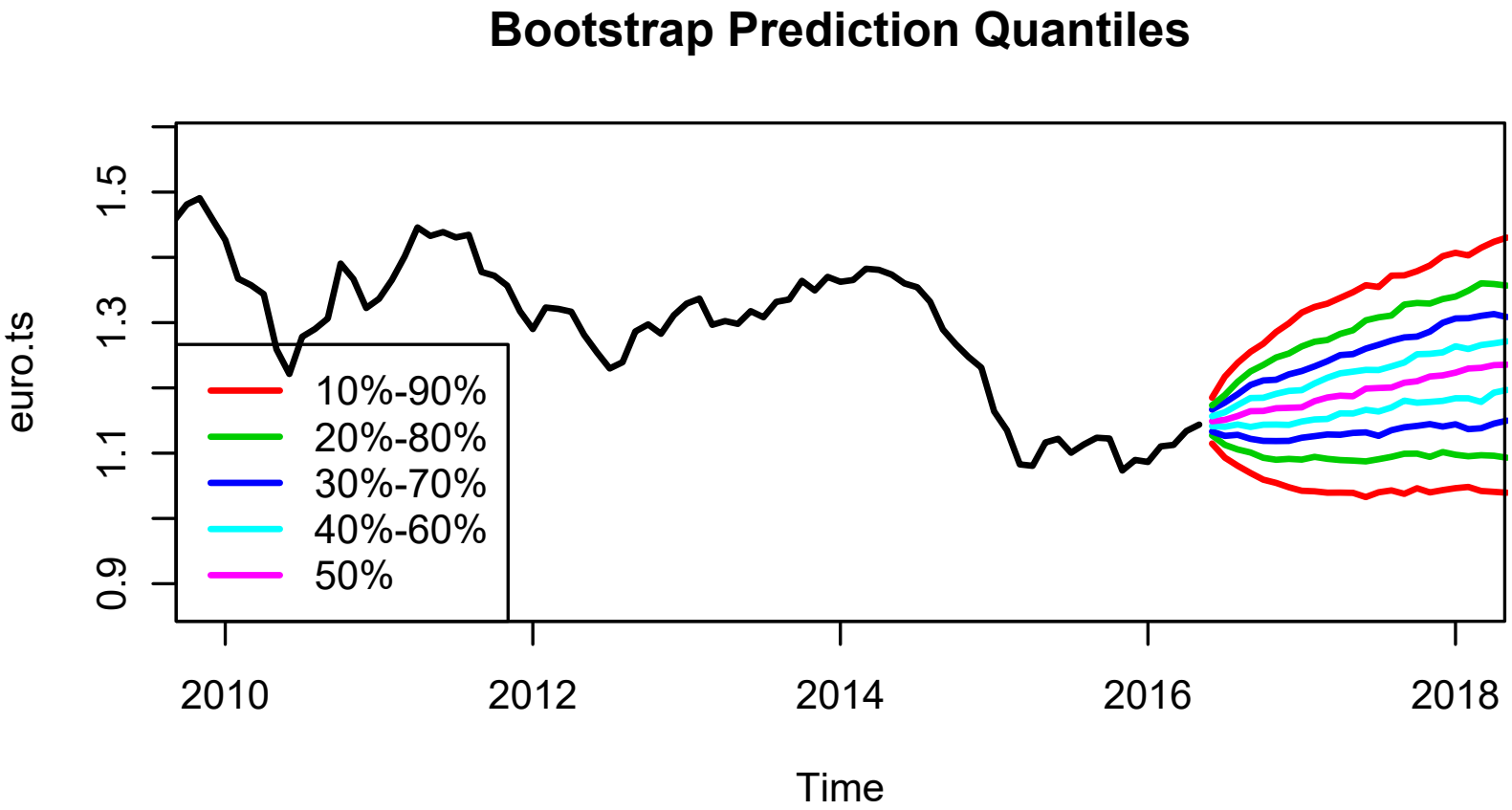
Simple simulation

```
x = arima.sim(model = list(order = c(2, 1, 0), ar = c(0.75, -0.25)),
    n = 200, rand.gen = rnorm, sd = 0.3)
```

# ARIMA Simulation II

```
source("arima.simforecast.R")
myboot = function(n, vec, prob = NULL) {
    sample(vec, size = n, replace = TRUE, prob = prob)
}  #Replace from residuals
# Sim. of the forecasts using default gaussian distribution
xsim = arima.simforecast(res.arma, n.ahead = 24, std = TRUE,
    nrep = 500)
# Sim. of the forecasts using residuals (needs std=FALSE)
xsim2 = arima.simforecast(res.arma, n.ahead = 24, rand.gen = myboot,
    std = FALSE, vec = residuals(res.arma), nrep = 500)
# Compute the quantiles
quan.ts = ts(t(apply(xsim2, 1, quantile, prob = seq(0.1, 0.9,
    by = 0.1))), start = start(xsim2), frequency = frequency(xsim2))
```

# ARIMA Simulation III

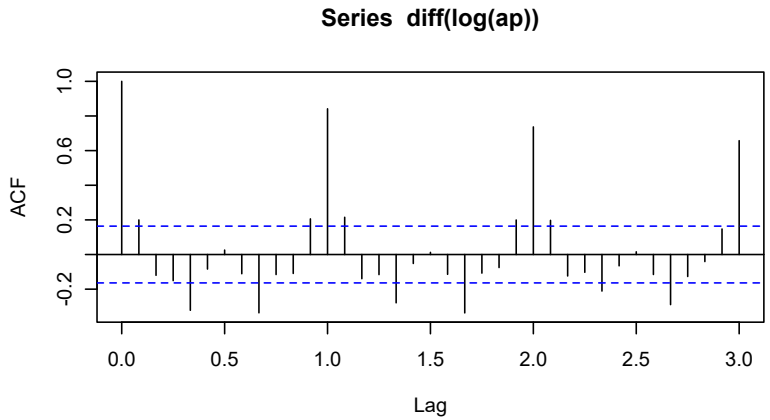


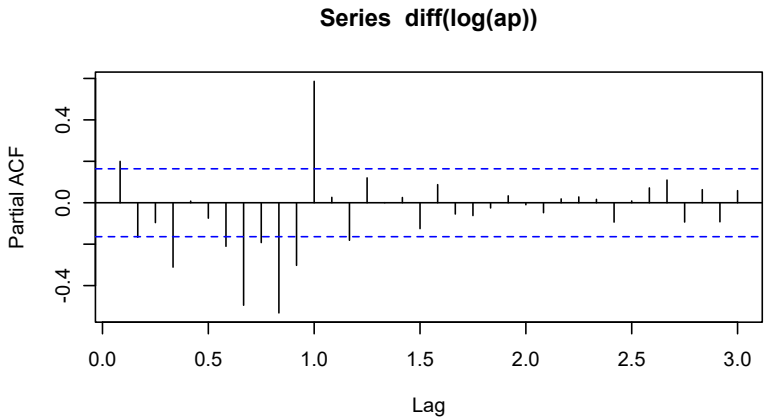
# Seasonal ARIMA Models

The extension to seasonal models is straightforward. Only take into account that the usual tools (ACF, PACF, ...) reflect seasonality at multiples of $s$ lags.

## $\text{ARIMA}(p,d,q) \times \text{ARIMA}_s(P,D,Q)$ models

$$\Phi_p(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D z_t = c + \Theta_q(B)\Theta_Q(B^s)\epsilon_t$$



# Seasonal Models

## AirPassengers: $\mathrm{IMA}(1,1) \times \mathrm{IMA}_{12}(1,1)$

```
ap.res = arima(log(ap), order = c(0, 1, 1), seasonal = list(order = c(0
    1, 1), frequency = 12))

....

> Coefficients:
>           ma1     sma1
>       -0.4018  -0.5569
> s.e.   0.0896   0.0731
>
> sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -483.4

Box.test(residuals(ap.res), lag = 36, type = "Ljung-Box")

....

> X-squared = 37.874, df = 36, p-value = 0.3838
```

# Package TSA

This package extends the classical tools for ARIMA modelling and also includes TAR and Transfer Models.

- `harmonic, season`: Dummy variates (harmonic or cycle)
- `BoxCox.ar`: Determine optimal power transformation
- `armasubsets`: Selection of ARMA Models
- `eacf`: Sample Extended ACF
- `arima, arimax`: ARIMA and Transfer Models.
- `detectAO, detect IO`: Outliers
- Tests: `runs`:Ind, `McLeod.Li.test`:Garch, `Keenan.test`:NLin, `Tsay.test`:Cuad.AR, `tlrt`:TAR
- `tar, predict.tar`: Threshold AR Models

# Simple packages

## PerformanceAnalytics

This package is mainly devoted to create exploratory charts and tables: `chart.RollingEstimation`, `chart.Boxplot`, ..., `table.Stats`, `table.Correlation`, ...

## forecast

Package mainly devoted to simple univariate time series modelling through state space models (exp. smoothing and ARIMA)

## FinTS

Package associated with Tsay's book with many data examples and wrappers to functions of other packages.

- `ARIMA` $\Longrightarrow$ `stats::arima` + `stats::Box.test`, `Unitroot` $\Longrightarrow$ Wrapper for `tseries::adf.test` `fUnitRoots::adfTest` and `urca::ur.df`
- `runscript`: Run (or Open) a script of the chapter of the book.

# Package fArma

Works with timeSeries objects extending other packages.
`library(fArma)`

## ARMA(3,2)

```
res=armaFit(euro.ts~arma(3,2),data=euro.ts,
fixed=c(NA,0,NA,0,NA,NA))
armaRoots(coef(res)[1:3])
armaTrueacf(res)
```

Some generic methods change: `print`, `summary`, `plot`

## ARFIMA(2,0)

```
fit2=armaFit(~arfima(2,0),data=eusdw[,1])
```

# Long Memory Time Series

Sometimes, the ACF does not decay as fast as expected. When $\rho(k) \to C_\rho k^{-\alpha}$ as $k \to \infty$, the time series has *long-memory*. This behaviour can be modeled allowing non integer differentiation in ARIMA.

$$(1-B)^d = \sum_{k=0}^{\infty} \frac{\Gamma(d+1)}{\Gamma(k+1)\Gamma(d-k+1)}(-1)^k B^k$$

being $d = (1-\alpha)/2$ (or $H = d + 1/2$, the *Hurst coefficient*)

- $|d| > 1/2$: Non-stationary series.
- $0 < d < 1/2$: Stationary and Long Memory
- $-1/2 < d < 0$: Stationary and Short Memory (anti-persistent)

# FARIMA in practice

```
library(fracdiff)
res.fr = fracdiff(eusdw, nar = 2, nma = 1, ar = c(1.4, -0.5),
    ma = 0.6)

....

> Coefficients:
>     Estimate
> d      0.305
> ar1    1.523
> ar2   -0.539
> ma     0.702
> sigma[eps] = 0.01492949
> [d.tol = 0.0001221, M = 100, h = 1.134e-05]
> Log likelihood:  1075 ==> AIC = -2139.678 [5 deg.freedom]
```

Methods: `summary`, `coef`, `confint`, `residuals`, `fitted`, `sim` but no `predict`

# References


- Chan, K.S. and Ripley, B. (2012) TSA: Time Series Analysis (1.01). `http://cran.r-project.org/web/packages/TSA/`.
- Cryer, J.D. and Chan, K.S. (2010) Time Series Analysis: With Applications in R. Springer.
- Cowpertwait, P. and Metcalfe A.V. (2009) Introductory Time Series with R. Springer.
- Pfaff, B. (2012) urca: Unit root and cointegration tests for time series data (1.2-7). `http://cran.r-project.org/web/packages/urca/`.
- Maechler, M. (2012) fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models (1.4-2). `http://cran.r-project.org/web/packages/fracdiff/`.
- Shumway, R.H. & Stoffer, D.S. (2006) Time Series Analysis and its Applications: With R Examples. Springer
- Tsay, R. (2010) Analysis of Financial Time Series. 3ed. Wiley.
- Zivot, E. and Wang, J. (2006) Modeling Financial Time Series with S-Plus. Springer.

# Table of Contents

# Returns

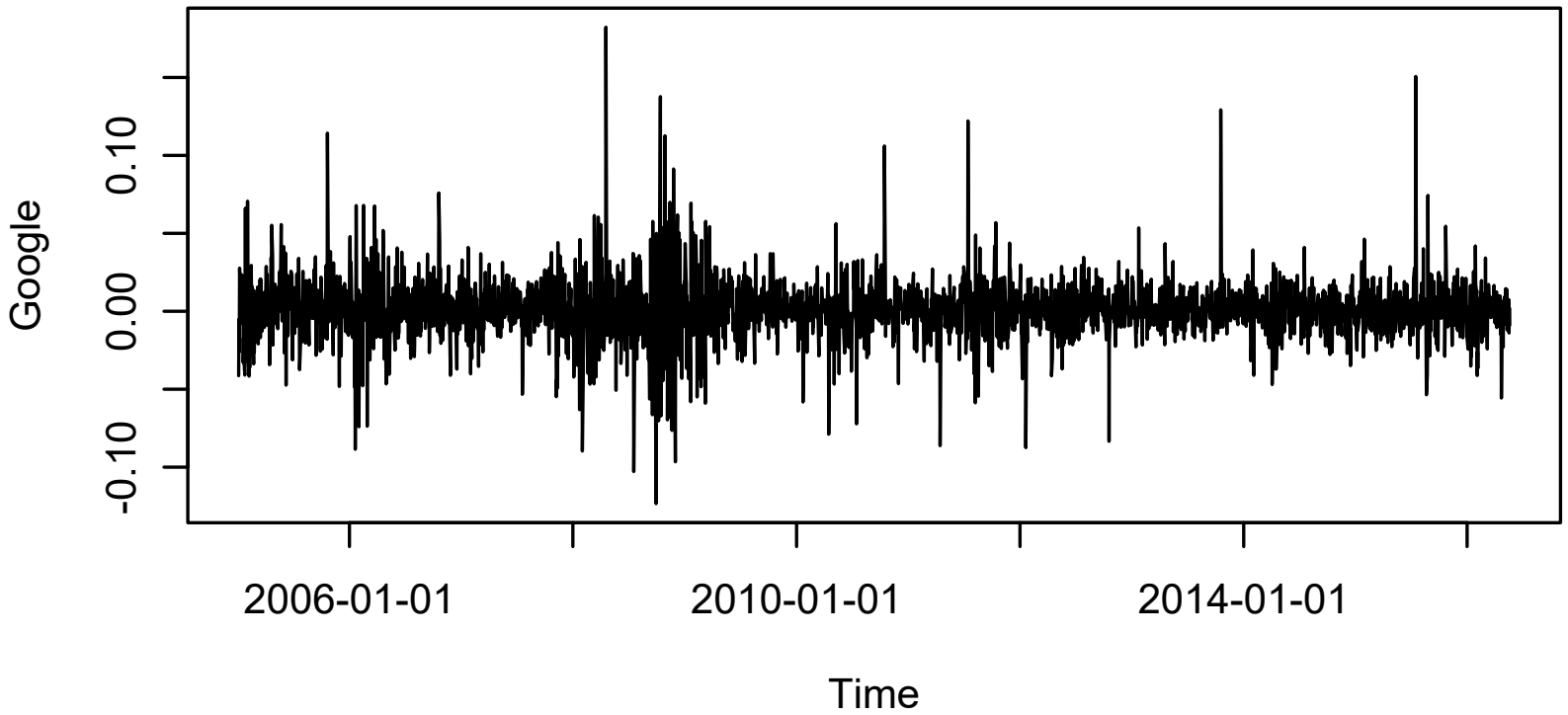
$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$ (discrete), $r_t = \log\left(P_t / P_{t-1}\right)$ (continuous)

Characteristics you may encounter when modelling returns

- Constant mean with reduced dynamic structure.(ARMA(1,1))
- High Kurtosis.
- Volatility Clustering.
- Persistent Volatility (long memory).
- Leverage effect (asymmetric impact of good/bad news).
- Changes in volatility regime due to seasonal, external or unexpected facts.
- Volatility influenced by other volatility asset returns.

# Example

```
> Error:  package or namespace load failed for 'rugarch'
```




Google Returns:Skewness=0.541, Kurtosis=9.513

# ARCH Models

A series $\{\epsilon_t\}$ is ARCH($p$), if

$$\begin{aligned} \epsilon_t &= w_t \sigma_t \\ \sigma_t^2 &= \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 \end{aligned}$$

where $\{w_t\}$ is a standard white noise, $\sigma_t^2 = \mathrm{Var}\left(\epsilon_t / \mathcal{F}_{t-1}\right)$ and $\mathcal{F}_{t-1}$ is the $\sigma$-algebra generated by $\{\epsilon_{t-1}, \epsilon_{t-2}, \ldots\}$
Example in ARCH(1)

$$\begin{aligned} \mathbb{E}\left[\epsilon_t\right] &= \mathbb{E}\left[\mathbb{E}\left[\epsilon_t / \mathcal{F}_{t-1}\right]\right] = \mathbb{E}\left[\sigma_t \mathbb{E}\left[w_t / \mathcal{F}_{t-1}\right]\right] = 0 \\ \mathrm{Var}\left(\epsilon_t\right) &= \mathbb{E}\left[\epsilon_t^2\right] = \mathbb{E}\left[\mathbb{E}\left[\epsilon_t^2 / \mathcal{F}_{t-1}\right]\right] = \alpha_0 + \alpha_1 \mathbb{E}\left[\epsilon_t^2\right] \\ &\longrightarrow \mathrm{Var}\left(\epsilon_t\right) = \frac{\alpha_0}{1 - \alpha_1} \end{aligned}$$

# GARCH Models

A series $\{\epsilon_t\}$ is GARCH$(p, q)$, if

$$\begin{aligned} \epsilon_t &= w_t \sigma_t \\ \sigma_t^2 &= \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \end{aligned}$$

$\epsilon_t \sim GARCH(p,q) \longrightarrow \epsilon_t^2 \sim ARMA(\max(p,q), q)$

Example in GARCH(1,1)

$$\begin{aligned} \mathrm{Var}\left(\epsilon_t\right) &= \mathbb{E}\left[\epsilon_t^2\right] = \mathbb{E}\left[\mathbb{E}\left[\epsilon_t^2/\mathcal{F}_{t-1}\right]\right] = \alpha_0 + \alpha_1 \mathbb{E}\left[\epsilon_t^2\right] + \beta_1 \mathbb{E}\left[\epsilon_{t-1}^2\right] \\ &\longrightarrow \mathrm{Var}\left(\epsilon_t\right) = \frac{\alpha_0}{1 - \alpha_1 - \beta_1} \\ \frac{\mathbb{E}\left[\epsilon_t^4\right]}{\left(\mathbb{E}\left[\epsilon_t^2\right]\right)^2} &= \frac{3\left[1 - (\alpha_1 + \beta_1)^2\right]}{1 - 2\alpha_1^2 - (\alpha_1 + \beta_1)^2} > 3 \end{aligned}$$

# Extension ARIMA-GARCH Models I

Dynamics:

$$\begin{aligned}\Phi_p(B)(1-B)^d(z_t-\mu_t) &= \Theta_q(B)\epsilon_t,\\ \text{with } \mu_t &= \mu+\sum_{i=1}^{m_1}\delta_i x_{i,t}+\sum_{i=m_1+1}^{m}\delta_i x_{i,t}\sigma_t+\xi\sigma_t^k\end{aligned}$$

being $\{x_i\}$ external regressors (last $m-m_1+1$ are multiplied by $\sigma_t$) and $\{\epsilon_t\}$ a standard noise distribution (Normal, GED, Student, ...).
Also, long-memory models ($0<d<1$) are possible.

Volatility: Being $\{\nu_j\}$ external regressors

# Extension ARIMA-GARCH Models II

- SGARCH:
$$\sigma_t^2 = \underbrace{\left(\omega + \sum_{j=1}^{m} \varsigma_j \nu_{j,t}\right)}_{\omega_t} + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 + \sum_{j=1}^{q} \alpha_j \epsilon_{t-j}^2$$
- EGARCH:
$$\log \sigma_t^2 = \omega_t + \sum_{j=1}^{p} \beta_j \log \sigma_{t-j}^2 + \sum_{j=1}^{q} \left(\alpha_j \frac{|\epsilon_{t-j}| + \gamma_i \epsilon_{t-j}}{\sigma_{t-i}}\right)$$
- GJR-GARCH:
$$\sigma_t^2 = \omega_t + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 + \sum_{j=1}^{q} \left(\alpha_j \epsilon_{t-j}^2 + \gamma_j I_{(\epsilon_{t-j}<0)} \epsilon_{t-j}^2\right)$$
- APGARCH:
$$\sigma_t^\delta = \omega_t + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^\delta + \sum_{j=1}^{q} \alpha_j \left(|\epsilon_{t-j}| - \gamma_j \epsilon_{t-j}\right)^\delta$$
- family GARCH:
$$\sigma_t^\lambda = \omega_t + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^\lambda +$$
$$\sum_{j=1}^{q} \alpha_j \sigma_{t-j}^\lambda \left(\left|\frac{\epsilon_{t-j}}{\sigma_{t-j}} - \eta_{2j}\right| - \eta_{1j}\left(\frac{\epsilon_{t-j}}{\sigma_{t-j}} - \eta_{2j}\right)\right)^\delta$$

# Extension ARIMA-GARCH Models III

- Component GARCH:

$$\sigma_t^{\lambda} = q_t + \sum_{j=1}^{p} \beta_j \left( \sigma_{t-j}^2 - q_{t-j}^2 \right) + \sum_{j=1}^{q} \alpha_j \left( \sigma_{t-j}^2 - q_{t-j} \right)$$

$$q_t = \omega + \rho q_{t-1} + \phi \left( \epsilon_{t-1}^2 - \sigma_{t-1}^2 \right)$$

# rugarch I

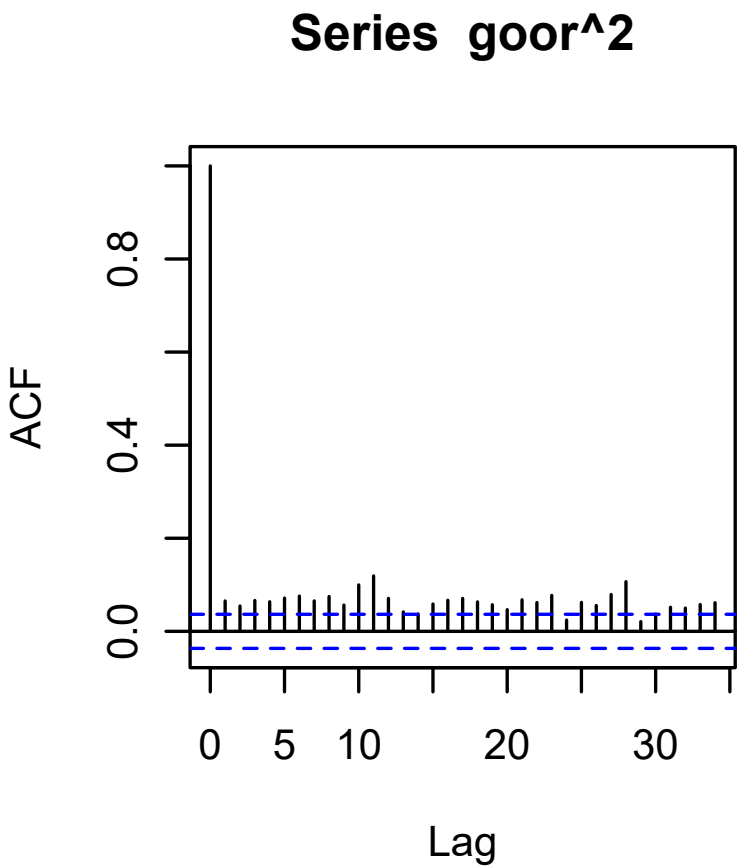
```
# ar(goor,order.max=10)
par(mfrow = c(1, 2))
# acf(goor);pacf(goor)
acf(goor^2)
pacf(goor^2)
monday = ifelse(dayOfWeek(time(goor)) == "Mon", 1, 0)
spec = ugarchspec(mean.model = list(armaOrder = c(0, 0)), variance.mode
    garchOrder = c(1, 1)))
```

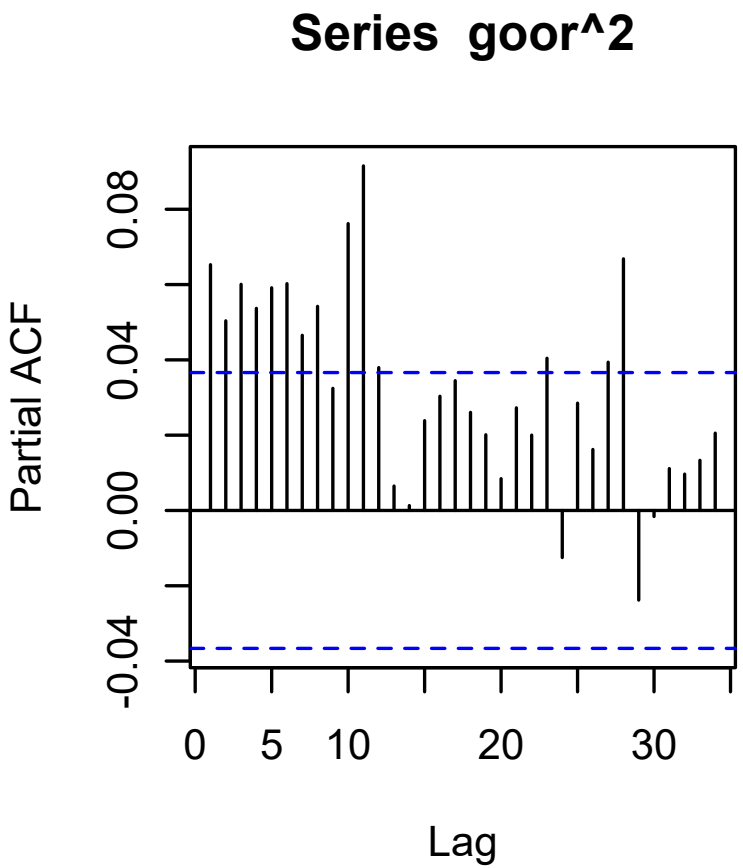
```
> Error in eval(expr, envir, enclos):  no se pudo encontrar la
función "ugarchspec"
```

```
fit = ugarchfit(spec, data = goor)
```

```
> Error in eval(expr, envir, enclos):  no se pudo encontrar la
función "ugarchfit"
```

# rugarch II


Series goor^2
ACF
0.0
0.4
0.8
0
5
10
20
30
Lag


Series goor^2
Partial ACF
-0.04
0.00
0.04
0.08
0
5
10
20
30
Lag

# GARCH fit I

```
> Error in eval(expr, envir, enclos):  objeto 'fit' no encontrado
```

```
> Error in eval(expr, envir, enclos):  objeto 'fit' no encontrado
```

```
> Error in eval(expr, envir, enclos):  objeto 'fit' no encontrado
```

```
> Error in eval(expr, envir, enclos):  objeto 'fit' no encontrado
```

```
> Error in eval(expr, envir, enclos):  objeto 'fit' no encontrado
```

```
> Error in eval(expr, envir, enclos):  objeto 'fit' no encontrado
```

# GARCH plots

```
> Error in plot(fit, which = 1):  objeto 'fit' no encontrado
> Error in plot(fit, which = 2):  objeto 'fit' no encontrado
> Error in plot(fit, which = 3):  objeto 'fit' no encontrado
```

```
> Error in plot(fit, which = 8):  objeto 'fit' no encontrado
> Error in plot(fit, which = 9):  objeto 'fit' no encontrado
> Error in plot(fit, which = 12):  objeto 'fit' no encontrado
```

# Best model

```
spec2 = ugarchspec(variance.model = list(model = "eGARCH", garchOrder =
    1), external.regressors = cbind(monday)), mean.model = list(armaOrd
    0)), distribution.model = "sstd")

> Error in eval(expr, envir, enclos):  no se pudo encontrar la
función "ugarchspec"

fit2 = ugarchfit(spec2, data = goor)

> Error in eval(expr, envir, enclos):  no se pudo encontrar la
función "ugarchfit"
```

# Results I

```
> Error in eval(expr, envir, enclos): objeto 'fit2' no
encontrado
```

```
> Error in eval(expr, envir, enclos): objeto 'fit2' no
encontrado
```

```
> Error in eval(expr, envir, enclos): objeto 'fit2' no
encontrado
```

```
> Error in eval(expr, envir, enclos): objeto 'fit2' no
encontrado
```

# Results II

```
> Error in eval(expr, envir, enclos):  objeto 'fit2' no
encontrado
```


```
> Error in eval(expr, envir, enclos):  objeto 'fit2' no
encontrado
```

# GARCH plots

```
> Error in plot(fit2, which = 1):  objeto 'fit2' no encontrado
> Error in plot(fit2, which = 2):  objeto 'fit2' no encontrado
> Error in plot(fit2, which = 3):  objeto 'fit2' no encontrado
```

```
> Error in plot(fit2, which = 8):  objeto 'fit2' no encontrado
> Error in plot(fit2, which = 9):  objeto 'fit2' no encontrado
> Error in plot(fit2, which = 12):  objeto 'fit2' no encontrado
```

# GARCH forecasting I

```
nmonday = rep(c(1, 0, 0, 0, 0), 8)
fit2.fore = ugarchforecast(fit2, n.ahead = 40, external.forecasts = lis

> Error in eval(expr, envir, enclos):  no se pudo encontrar la
función "ugarchforecast"

# fit2.fore
```

*Bootstrap*

```
fit2.boot = ugarchboot(fit2, method = "Partial", sampling = "spd",
    n.ahead = 40, n.bootpred = 2000, external.forecasts = list(mregfor
    mexsimdata = t(rbind(replicate(2000, nmonday))))

> Error in eval(expr, envir, enclos):  no se pudo encontrar la
función "ugarchboot"
```

# Bootstrap Plot

```
> Error in plot(fit2.boot, which = "all"):  objeto 'fit2.boot' no
encontrado
```

# References I



Cryer, J.D. and Chan, K.S. (2010) Time Series Analysis: With Applications in R. Springer.


Chan, N.H. (2010) Time Series: Applications to Finance with R and S-Plus, 2ed. Wiley.


Ghalanos, A. (2012) rugarch: Univariate GARCH models (1.0-16). `http://cran.r-project.org/web/packages/rugarch/`


Gouriéroux, Ch. (1997) ARCH Models and Financial Applications. Springer.


Zivot, E. and Wang, J. (2006) Modeling Financial Time Series with S-Plus. Springer.