# Time Series with R

**Manuel Febrero–Bande**

manuel.febrero@usc.es

Dpt. of Statistics, Mathematical Analysis and Optimization
Univ. of Santiago de Compostela

X Xornada de Usuarios de R en Galicia
18 de Outubro de 2023, Santiago de Compostela



# Contents



X XORNADA DE USUARIOS DE R EN GALICIA

# Table of Contents



X XORNADA DE USUARIOS DE R EN GALICIA

# Time Series

A time series $\{X_t\}_{t \in T}$ 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: hourly, daily, weekly, monthly, quarterly, annually
- Irregular, 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)

| | Irregular | Regular daily- | Regular daily+ |
|---|---|---|---|
| Classes | `irts` | `timeSeries`, `ts` | `ts`, `timeSeries` |
| Packages | `tseries` | `timeSeries`, `zoo` | `stats`, `xts`, `zoo`, `tsibble` |

# Overview packages by topic

- Time/Date
  - `base`, `zoo`, `chron`, `lubridate`, `timeDate`
- Import data
  - `quantmod`, `fImport`
- ARIMA modeling
  - `stats`, `forecast`
  - Multivariate: `stats`, `vars`, `MTS`
  - `FinTS`, `StructTS`, `TSA`
- GARCH modeling
  - `rugarch`, `rmgarch`, `tseries`

# Basic Objects I

Typically, the Date/Time is stored as a number measuring days, seconds from a reference date (15/10/1582, 14/9/1752, 01/01/1970).

- `Date`: Days since 01/01/1970
- `POSIXct`: Seconds since 01/01/1970
- `POSIXlt`: Date/time in list format with components `sec:0-61`, `min:0-59`, `hour:0-23`, `mday:1-31`, `mon:0-11`, `year:` since `1900`, `yday:0-365` .
  `Sys.getlocale()`, `Sys.setlocale()`, language settings
  Formatting for read/write, see `?strptime` for more details.
  - Day: `%a`, `%A`: (Abbr., Compl) day of week. `%d`: Day (01–31). `%j`: Day of year (001–366)
  - Mes: `%b`, `%B`: (Abr., Compl.) Name of month. `%m`: mes (01–12)
  - Year: `%y`, `%Y`: Year (without/with) century
  - Hour: `%H`: (00–23). `%I` `%p`: (01–12, AM/PM)
  - Minute: `%M`: (00–59).
  - Seconds: `%S`: (00–61)
  - Number of day of week : `%w`: (0–6, Sunday=0)
  - Number of week: `%U`,`%W`: (00–53) (Sunday=1 or Monday=1)

# Example I

```
mybirth = "27/03/1967 13:45:00"
myb = strptime(mybirth, "%d/%m/%Y %H:%M:%S")  #reading
format(myb, "%A, %d-%B-%Y, Day:%j - Week:%W")  #writing

> [1] "lunes, 27-marzo-1967, Day:086 - Week:13"

Sys.setlocale("LC_TIME", "pt_BR.UTF-8")

> [1] "pt_BR.UTF-8"
> [1] "segunda-feira, 27-março-1967, Day:086 - Week:13"

x = Sys.Date()  #Today
difftime(x, myb)

> Time difference of 20653.47 days

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

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

# Example II

```
myb1 = as.Date(mybirth, format = "%d/%m/%Y", tz = "CET")
myb2 = as.POSIXlt(myb, format = "%d/%m/%Y %H:%M", tz = "CET")
today = as.POSIXlt(Sys.Date(), tz = "CET")
difftime(today, myb2, units = "weeks")

> Time difference of 2950.484 weeks

x1 = seq(myb1, as.Date(today), by = "1 year")
x2 = seq(myb2, today, by = "1 week")
head(x1, 3)

> [1] "1967-03-27" "1968-03-27" "1969-03-27"

head(x2, 3)

> [1] "1967-03-27 13:45:00 CET" "1967-04-03 13:45:00 CET"
> [3] "1967-04-10 13:45:00 CET"
```

# lubridate

lubridate is a package for manipulate dates, durations and intervals.

```
library(lubridate)
c(round_date(myb, "quarters"), floor_date(myb, "quarters"), ceiling_date(myb,
    "quarters"))

> [1] "1967-04-01 CET" "1967-01-01 CET" "1967-04-01 CET"

paste(wday(myb2, label = TRUE, abbr = FALSE), " Qday:", qday(myb1))

> [1] "lunes  Qday: 86"

xint = interval(myb2, today)
as.duration(xint)

> [1] "1784452500s (~56.55 years)"

(myb1 + 65 * 365.25) %within% xint

> [1] FALSE
```

# timeDate I

timeDate is a package integrated in Rmetrics for Date/Time manipulation.
S4 class timeDate includes three slots: @Data (POSIXct), @format and @FinCenter.

- Information: isWeekday, isWeekend, isBizday, isHoliday
- Alignment: time{First|Last}Dayin{Month|Quarter}(x); timeNdayOnOr{After|Before}
- Selection: start, end, length, window
- Reordering: cut, sort, sample, unique, rev
- Frequency: isDaily, isMonthly, isQuarterly, isRegular

# timeDate II

```
library(timeDate)
Dates <- c("1989-09-28", "2001-01-15")
Times <- c("23:12:55", "10:34:02")
tDates = timeDate(paste(Dates, Times), zone = "GMT")
tSeq = timeSequence(from = tDates[1], to = tDates[2], by = "2years")
# by= sec, min, hour, day, week, month, year (--'5mins'--)
dayOfWeek(tSeq[1:3])

> 1989-09-28 23:12:55 1991-09-28 23:12:55 1993-09-28 23:12:55
>               "Thu"               "Sat"               "Tue"

# Vacaciones ?holidayDate
c(EasterSunday(2023), Ascension(2023))

> GMT
> [1] [2023-04-09] [2023-05-18]

timeLastDayInQuarter("2023-05-13")

> GMT
> [1] [2023-06-30]
```

# Table of Contents

# Importing directly from Internet (Yahoo) I

The url must be revised from time to time.

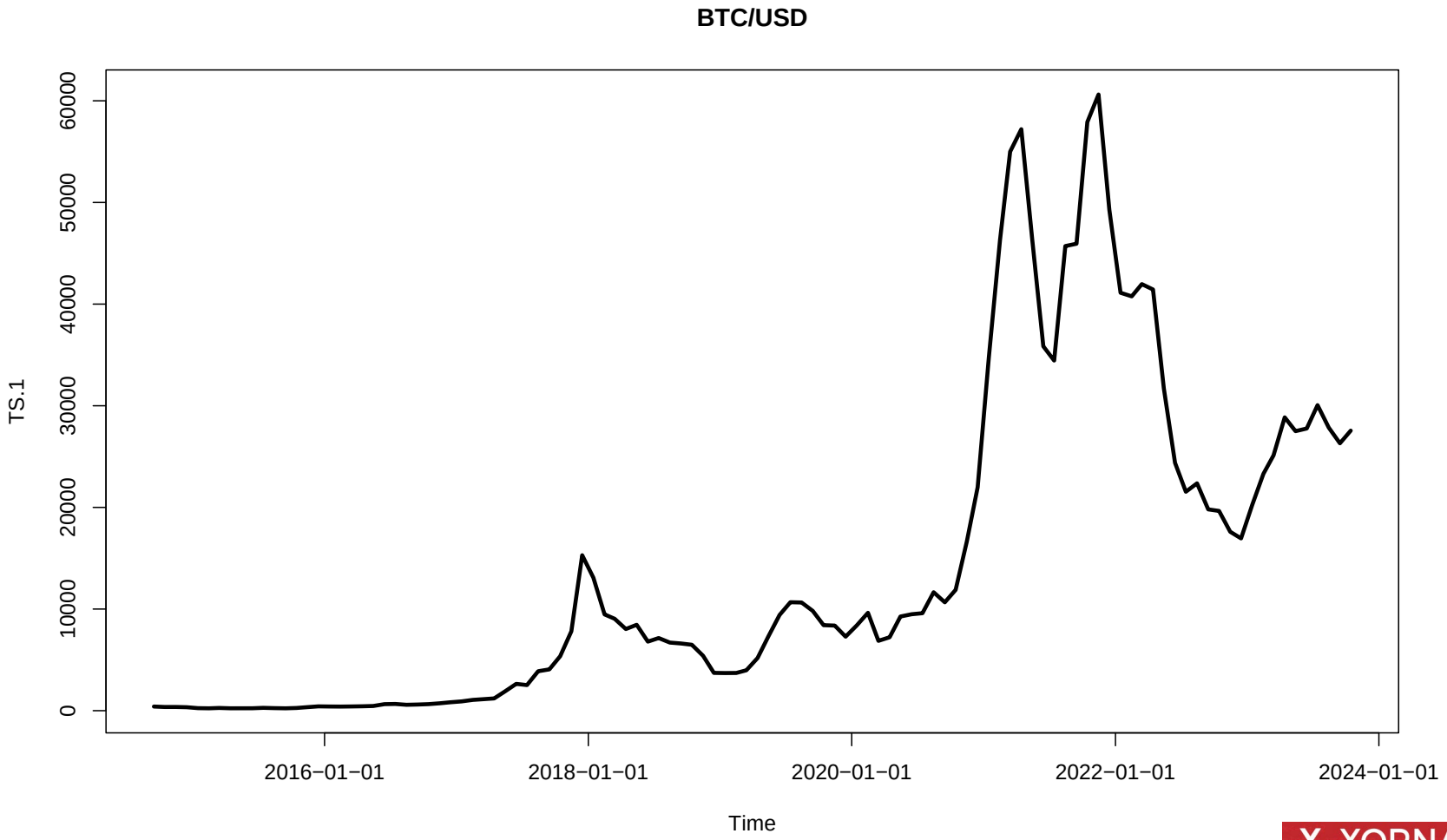
```
pre<-"https://query1.finance.yahoo.com/v7/finance/download/"
url<-paste0(pre,"BTC-USD?period1=aaa & period2=bbb &
interval=1d"
```

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

```
symbol = "BTC-USD"  # Bitcoin vs USD
prelim = "https://query1.finance.yahoo.com/v7/finance/download/"
ini = as.Date("01/01/2009", "%d/%m/%Y")
nini = as.numeric(ini) * 60 * 60 * 24  # in seconds
fin = Sys.Date() - 1
nfin = as.numeric(fin) * 60 * 60 * 24  #in seconds
url = paste(prelim, URLencode(symbol, reserved = TRUE), "?period1=",
    nini, "&period2=", nfin, "&interval=1d", sep = "")
```

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

# BTC-USD Monthly I



# library(quantmod)

Package for extracting financial information from different sources: yahoo, oanda, FRED or markets: FX, Metals. Also includes Technical Analysis. See ?TA

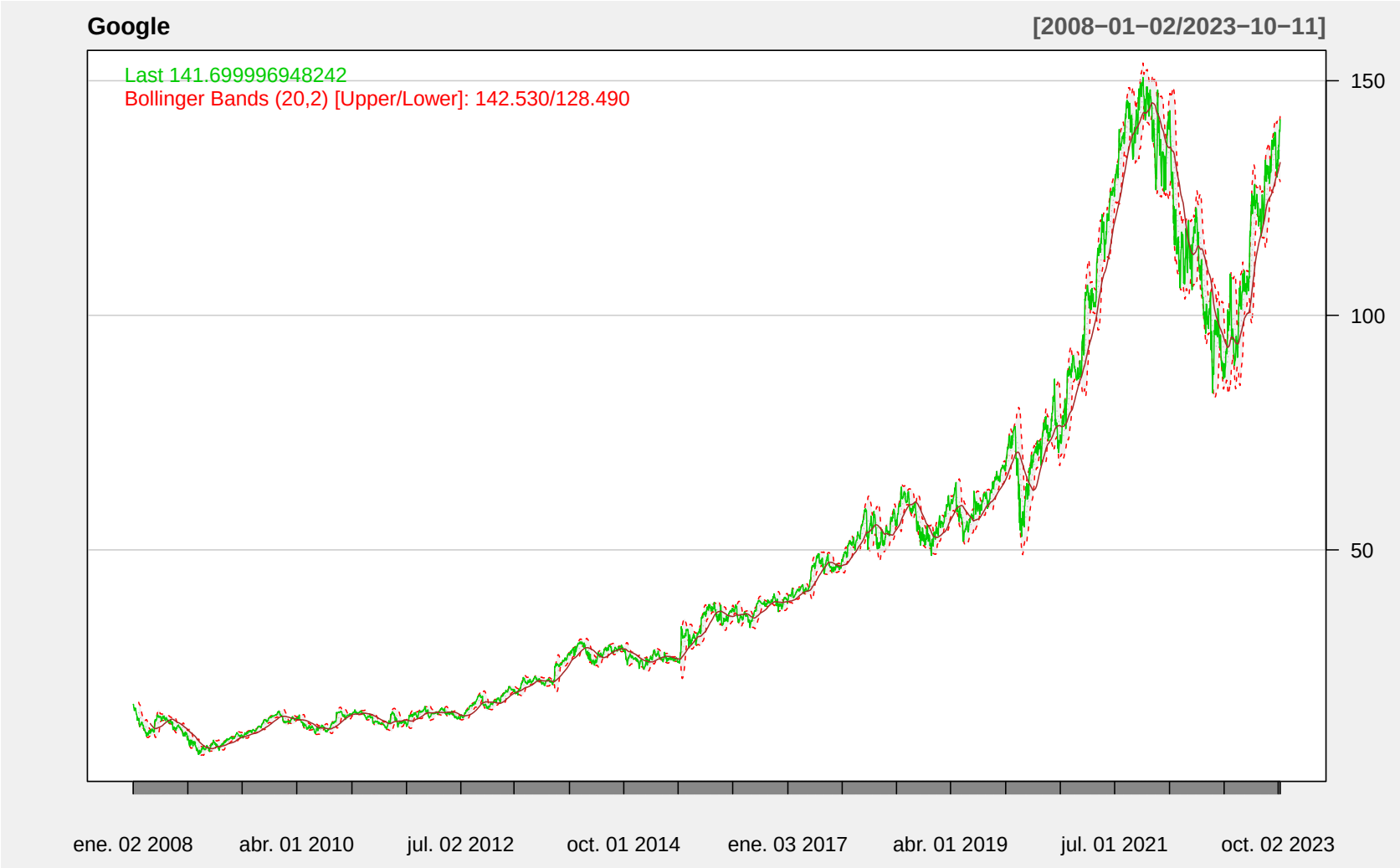
```
library(quantmod)
getSymbols("GOOG", src = "yahoo")  #New object GOOG of class xts

> [1] "GOOG"

# getQuote('GOOG')
getSplits("GOOG")

>             GOOG.spl
> 2014-03-27 0.4995005
> 2015-04-27 0.9972620
> 2022-07-18 0.0500000

getMetals("gold", from = Sys.Date() - 5 * 365)  # Metals object XAUUSD

> [1] "XAU/USD"

getFX("BTC/USD", from = Sys.Date() - 4 * 365, to = Sys.Date() -
    1)  # BTC/USD Only last four years

> [1] "BTC/USD"
```

# GOOG I

```
ini = as.Date("01/01/2008", "%d/%m/%Y")
fin = Sys.Date()
r = c(seq.Date(ini, fin, by = "5 year"), fin)
getSymbols("GOOG", from = r[1], to = r[2] - 1)

> [1] "GOOG"

goog = GOOG
for (i in 2:(length(r) - 1)) {
    getSymbols("GOOG", from = r[i], to = r[i + 1] - 1)
    goog = rbind(goog, GOOG)
}
googm = as.data.frame(aggregate(goog, by = list(strftime(index(goog),
    format = "%Y-%m")), FUN = mean))
colnames(googm) = c("mOpen", "mHigh", "mLow", "mClose", "mVolume",
    "mAdjusted")
```

```
chartSeries(goog$GOOG.Adjusted, name = "Google", TA = c(addBBands(),
    addSMA(n = 60)), theme = chartTheme("white"))
```

# GOOG II


Google
[2008−01−02/2023−10−11]
Last 141.699996948242
Bollinger Bands (20,2) [Upper/Lower]: 142.530/128.490
150
100
50
ene. 02 2008
abr. 01 2010
jul. 02 2012
oct. 01 2014
ene. 03 2017
abr. 01 2019
jul. 01 2021
oct. 02 2023

# Time Series objects

`stats::ts`: Most simple object for a regular time series.

**Example**

```
x=ts(data=googm[,"mAdjusted"],frequency=12,start=c(2008,1))
#Data by months beginning 2008, January
```

Utilities

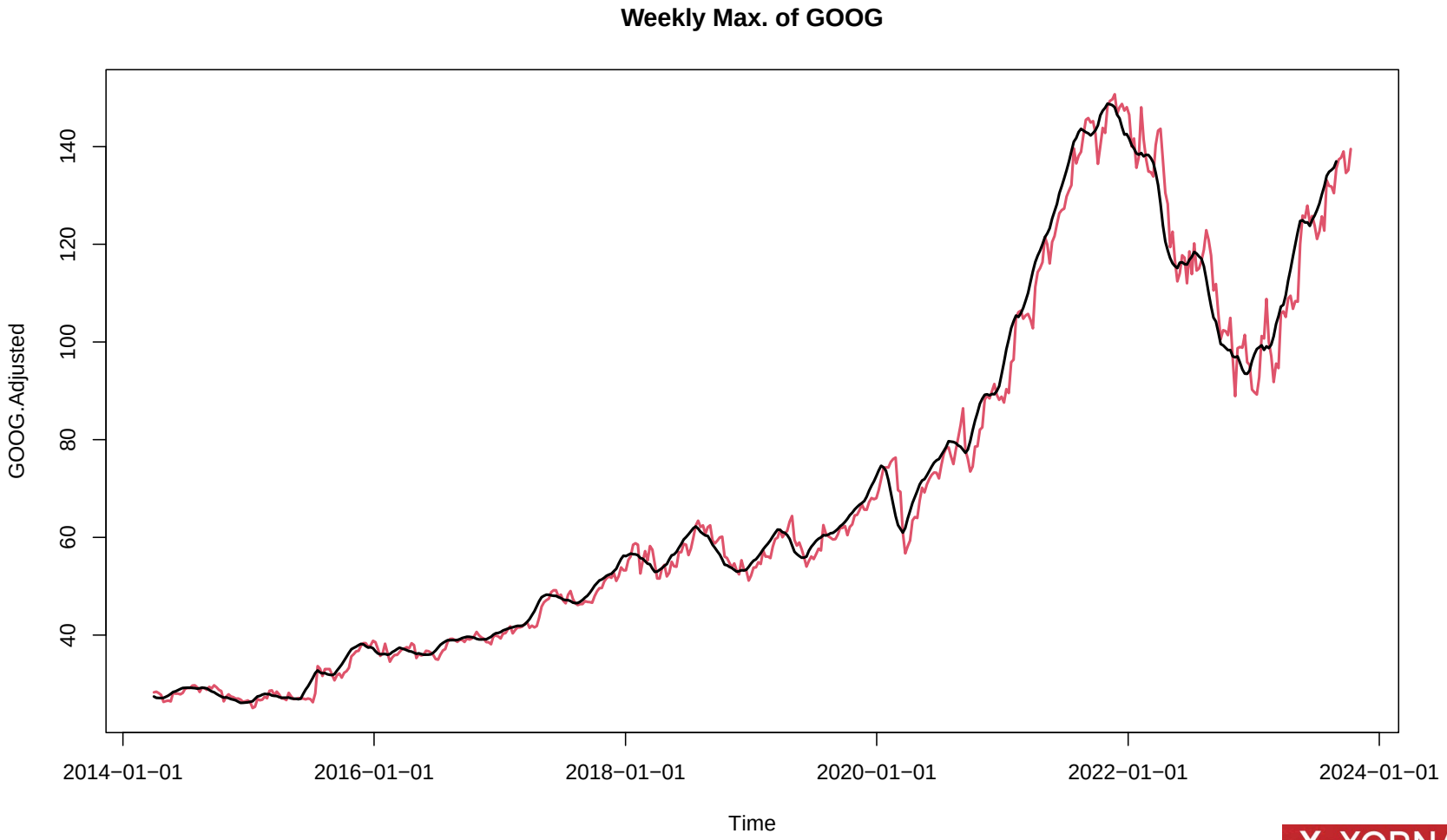
- `tsp(x)`: Start, end and frequency.
- `time(x)`: Date/Time.
- `cbind.ts`, `ts.union`, `ts.intersect`: Functions to mix time series.
- `embed(x,k)`: Vector of dim. $k$ $\{Z_t, Z_{t-1}, \ldots, Z_{t-k+1}\}$
- `filter(x,filter=c(1,1,1)/3)`: Filter (average of $Z_{t-1}, Z_t, Z_{t+1}$)
- `filter(x,c(1,1,1),method="recursive")`: $Y_t = Z_t + f_1 Z_{t-1} + \ldots + f_k Z_{t-k}$
- Descriptive: `lag.plot`, `monthplot`, `stl`, `HoltWinters`,

# timeSeries (S4) I

timeSeries: Provides a better representation of date/times (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

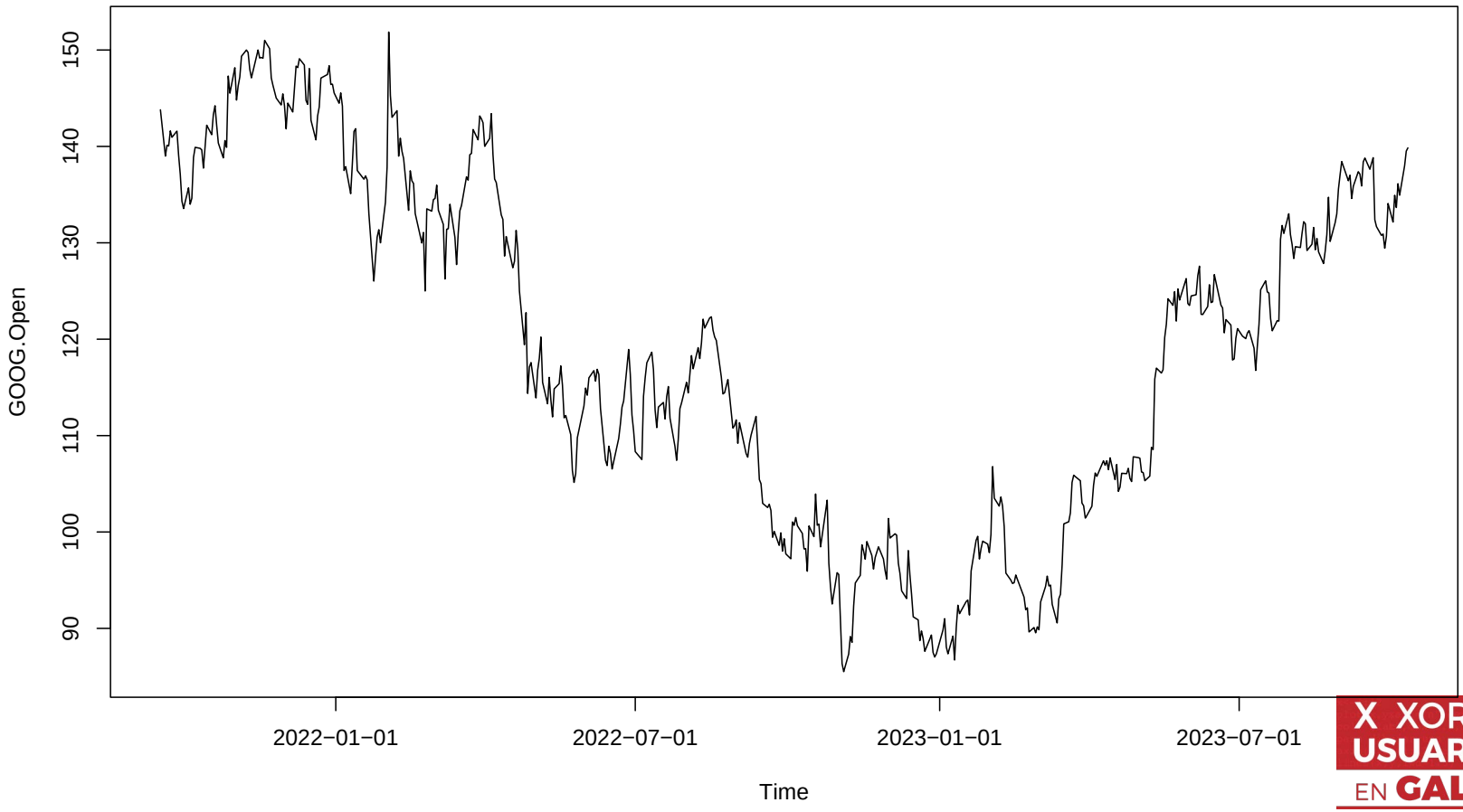
```
library(timeSeries)
googc = timeSeries(goog$GOOG.Adjusted, charvec = index(goog))
googd = window(googc, "2014-04-01", end(googc))  #Daily data
by = timeSequence(start(googd), end(googd), by = "week")
googw = aggregate(googd, by, FUN = max)  # Max. of Weekly data
head(series(googw), 3)

>            GOOG.Adjusted
> 2014-04-01      28.28036
> 2014-04-08      28.40900
> 2014-04-15      28.12977

plot(googw, lwd = 2, col = 2, main = "Weekly Max. of GOOG")
lines(rollStats(googw, 7, FUN = mean), lwd = 2)  # Mean of 7 weeks maximum
```

# timeSeries (S4) II



# Plot

```
goog.tS = timeSeries(goog[, 1:5], index(goog))
plot(tail(goog.tS[, "GOOG.Open"], 520), plot.type = "single")   #Last 2 years
```



# Table of Contents

# Stationary? Time Series

## 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).
$Z_t$ is usually a second order stationary time series

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

# Trend/Seasonality/Variance Stabilization I

- Fixed trend:

$$\hat{T}_t = \alpha_0 + \alpha_1 t + \ldots + \alpha_k t^k$$

$$\hat{T}_t = \nu_0 + \sum_{j=1}^{m} \left(\alpha_j \cos \omega_j t + \beta_j \sin \omega_j t\right), \omega_j = 2\pi / p_j$$

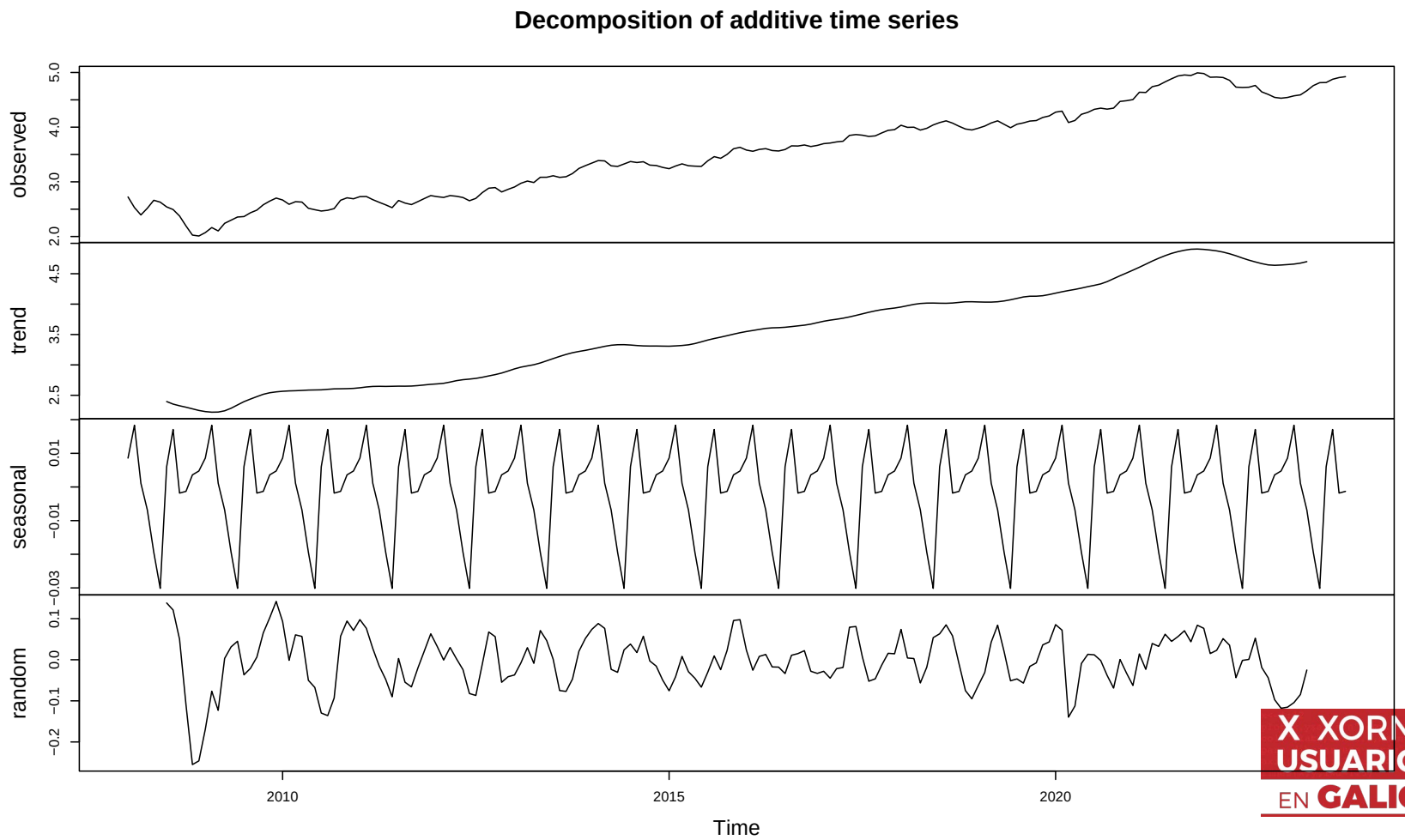
- Moving average:

$$\hat{T}_t = \sum_{j=-m}^{m} \alpha_j X_{t-j}$$

- Apply differences (remove polynomial trends):
  $\nabla^d X_t = \nabla^{d-1}(X_t - X_{t-1})$
- Seasonality: Any influence of an element of the cycle?
- Variance stabilization:
  $\mathrm{Var}\,(Z_t) = f(\mu_t) \implies T(Z_t) = \int \frac{1}{\sqrt{f(\mu_t)}} d\mu_t.$
  Compute means and variances by periods and seek for relationships.

# Decomposition of a time series

```
lgoog.ts = ts(log(googm[, "mAdjusted"]), frequency = 12, start = c(2008,
    1))
plot(decompose(lgoog.ts))
```



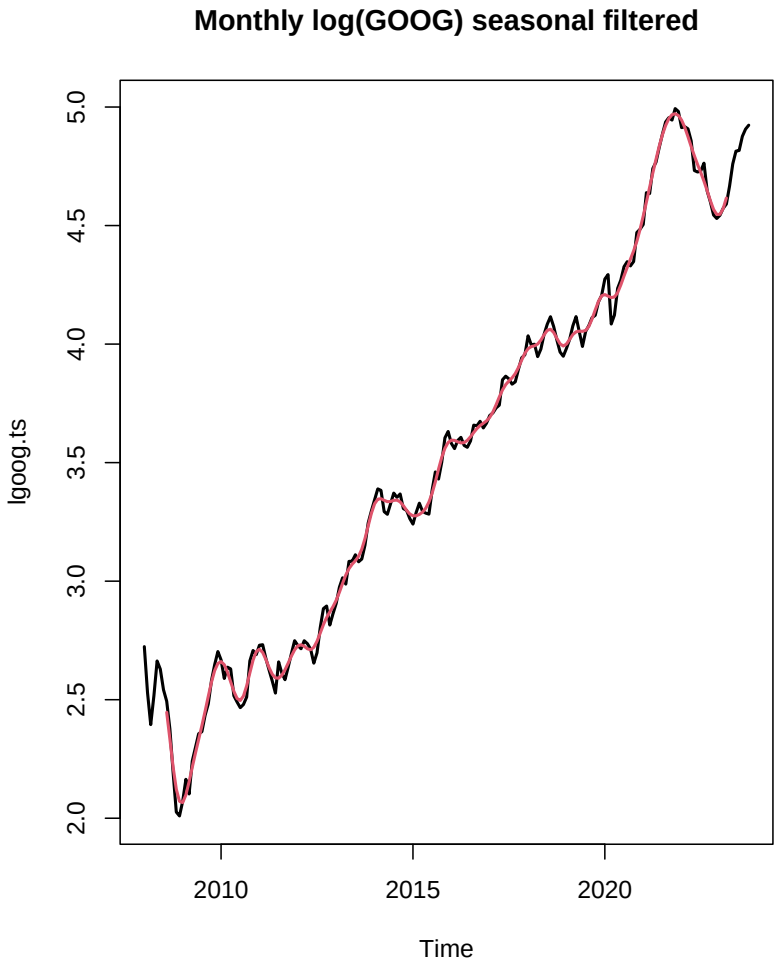
# Trend/... I

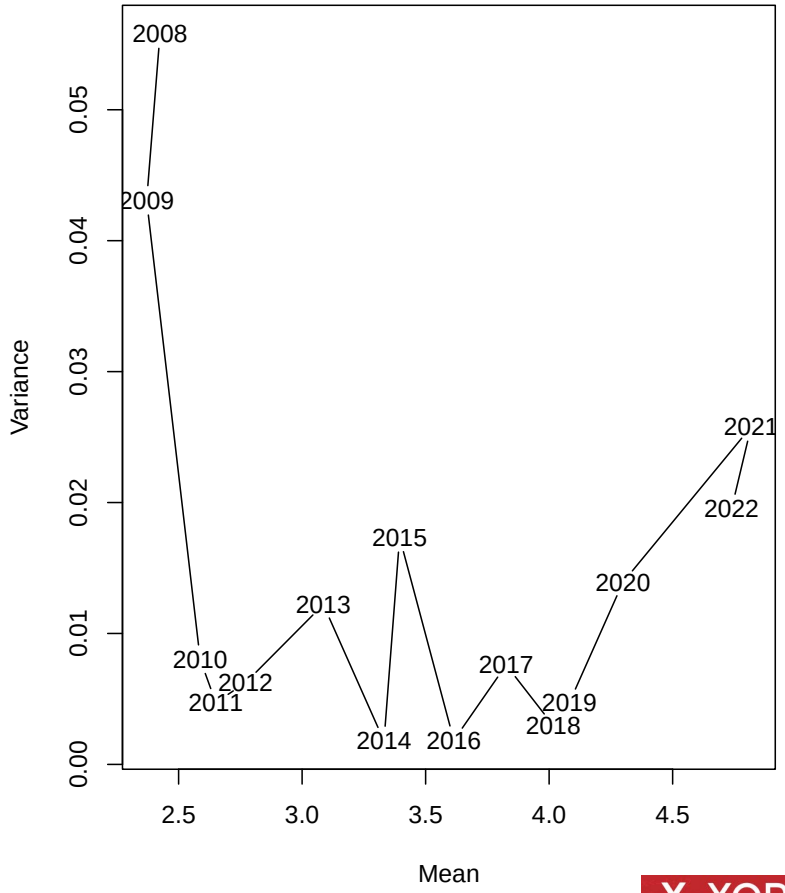
```
library(nlme)
t = index(lgoog.ts)
dummy = ifelse(time(lgoog.ts) < 2014.25, 0, 1)
z.gls = gls(lgoog.ts ~ poly(t, 3) + dummy, correlation = corAR1())
summary(z.gls)

....

> Correlation Structure: AR(1)
>  Formula: ~1
>  Parameter estimate(s):
>       Phi
> 0.9999928
>
> Coefficients:
>                 Value Std.Error   t-value p-value
> (Intercept)  3.599935 16.132969  0.223141  0.8237
> poly(t, 3)1 11.821902  3.688708  3.204890  0.0016
> poly(t, 3)2  1.738115  1.511871  1.149645  0.2518
> poly(t, 3)3 -1.742278  0.997052 -1.747431  0.0822
> dummy       -0.107655  0.061469 -1.751360  0.0815
....

# lgoog.ts=lgoog.ts-dummy*z.gls$coefficients[['dummy']]
```

# Trend/... II

```
Sp.15 = filter(lgoog.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(residuals(z.gls) ~ -1 + as.factor(cycle(lgoog.ts)))
summary(z.seas)

....

> Coefficients:
>                               Estimate Std. Error t value
> as.factor(cycle(lgoog.ts))1   0.0075022  0.0477218   0.157
> as.factor(cycle(lgoog.ts))2   0.0051024  0.0477218   0.107
> as.factor(cycle(lgoog.ts))3  -0.0183382  0.0477218  -0.384
> as.factor(cycle(lgoog.ts))4  -0.0094798  0.0477218  -0.199
> as.factor(cycle(lgoog.ts))5  -0.0058790  0.0477218  -0.123
> as.factor(cycle(lgoog.ts))6  -0.0129112  0.0477218  -0.271
> as.factor(cycle(lgoog.ts))7   0.0055437  0.0477218   0.116
> as.factor(cycle(lgoog.ts))8   0.0175580  0.0477218   0.368
> as.factor(cycle(lgoog.ts))9   0.0005594  0.0477218   0.012
> as.factor(cycle(lgoog.ts))10  0.0011097  0.0477218   0.023
> as.factor(cycle(lgoog.ts))11  0.0049291  0.0492869   0.100
> as.factor(cycle(lgoog.ts))12  0.0049192  0.0492869   0.100
....
```

# Trend/... III


Monthly log(GOOG) seasonal filtered
lgoog.ts
2.0
2.5
3.0
3.5
4.0
4.5
5.0
2010
2015
2020
Time


Variance
0.00
0.01
0.02
0.03
0.04
0.05
2.5
3.0
3.5
4.0
4.5
Mean
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022

# 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

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

## MA(q) models

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

## ARMA(p,q) models

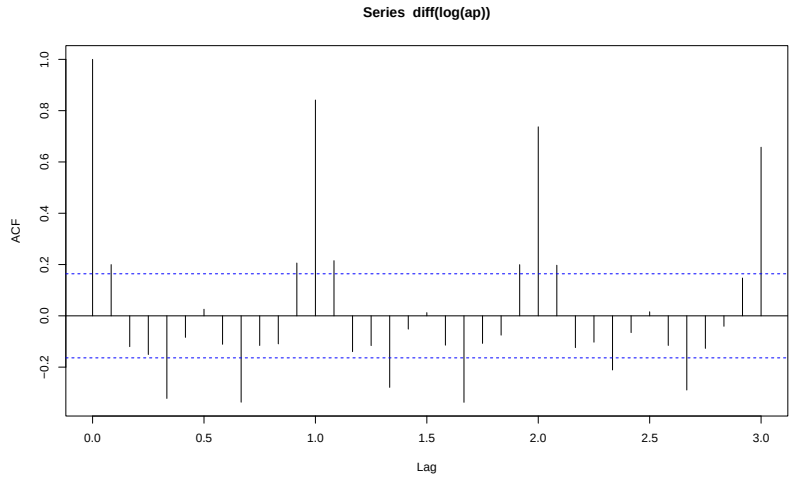
$z_t = c + \phi_1 z_{t-1} + \ldots + \phi_p z_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_p \epsilon_{t-q}$
$\Phi_p(B) z_t = c + \Theta_q(B)\epsilon_t$

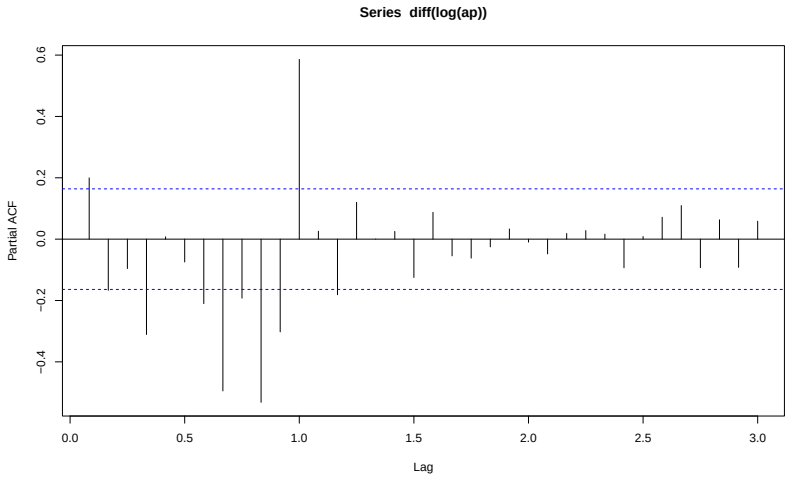
# Seasonal ARIMA Models

The extension to seasonal models is straightforward. Only take into account that the usual tools (ACF, PACF, ...) reflects seasonality at lags $s, 2s, 3s, ....$

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

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



# Steps for ARMA modeling

- Identify trends and/or seasonal components.
- Identify order of ARMA(p,q) using ACF and PACF functions.
- Propose and estimate a tentative model.
- Check stationarity and invertibility conditions.
- Check randomness of the residuals.
- Reconsider a new model if any of the checks fails.

# 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

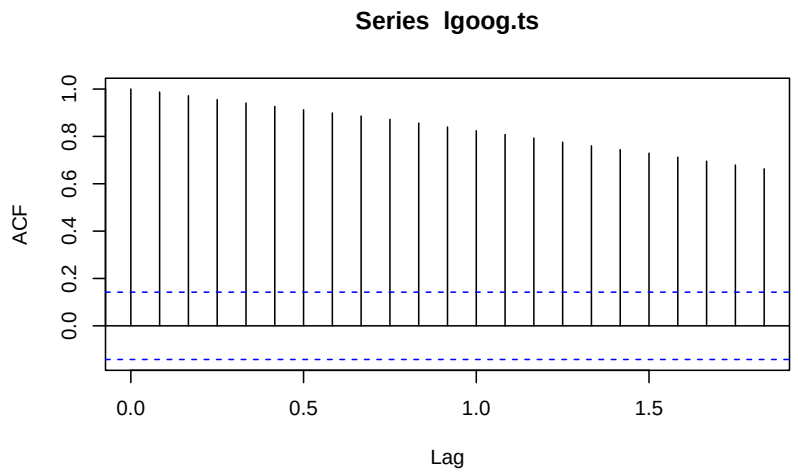
- ACF: $\rho_k \neq 0, k \leq q$, and $\rho_k = 0, k > q$
- PACF: Mixture of exponential and/or sinusoidal decays.

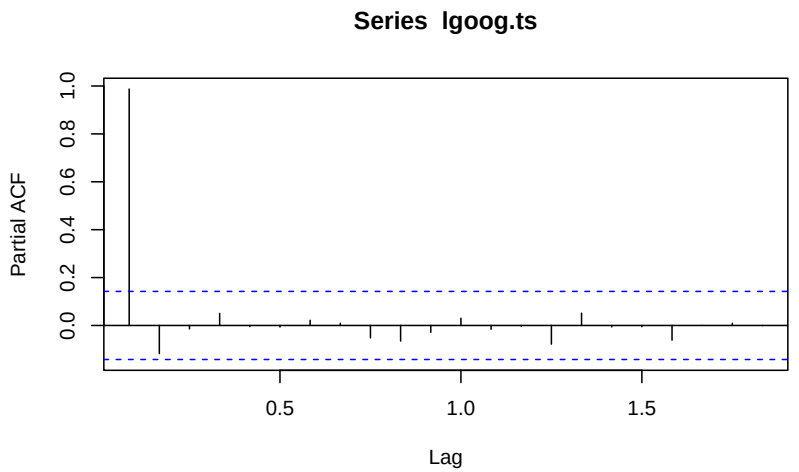
## ARMA(p,q) models

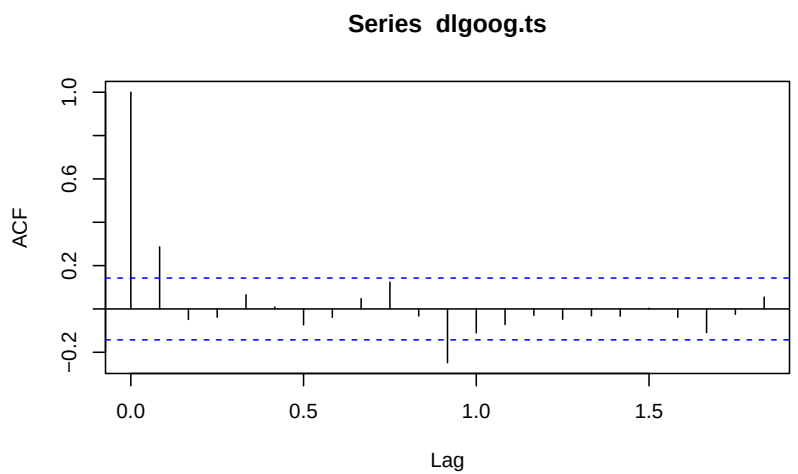
- ACF: First $q$ non zero autocorrelations and then mixture of exponential and/or sinusoidal decays.
- PACF: First $p$ non zero autocorrelations and then mixture of exponential and/or sinusoidal decays.

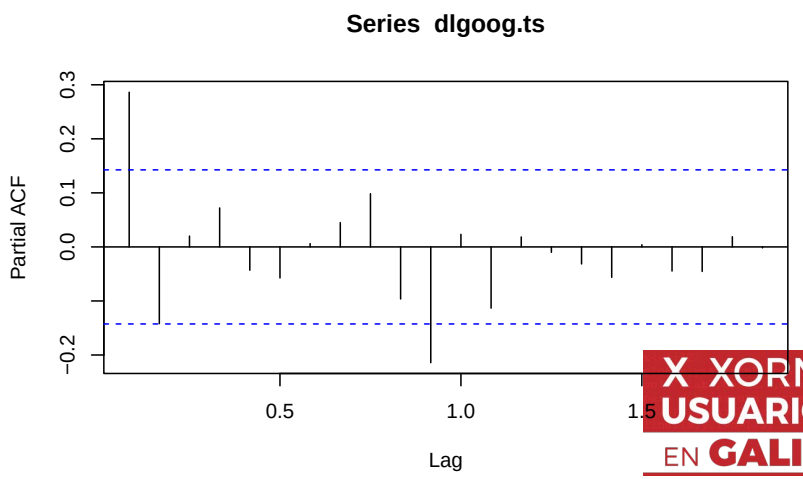
# Model identification

```
acf(lgoog.ts, ci.type = "white") pacf(lgoog.ts)
```



# Simple AR Estimation

## Example (Code)

```
res.ar = ar(dlgoog.ts, order.max = 10, method = "mle")   #meth=c('yw','burg','ols')
res.ar

>
> Call:
> ar(x = dlgoog.ts, order.max = 10, method = "mle")
>
> Coefficients:
>       1        2
>  0.3399  -0.1433
>
> Order selected 2  sigma^2 estimated as  0.003401
```

# Ljung–Box test

## Example (Box–Ljung test)

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

>
>  Box-Ljung test
>
> data:  res.ar$resid
> X-squared = 6.0338, df = 10, p-value = 0.8124

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

> [1] 2.641797 2.641797
```

# ARIMA Estimation I

## ARIMA(2,1,0)

```
res.arma = arima(lgoog.ts, order = c(2, 1, 0))

>
> Call:
> arima(x = lgoog.ts, order = c(2, 1, 0))
>
> Coefficients:
>          ar1      ar2
>       0.3633  -0.1169
> s.e.  0.0733   0.0741
>
> sigma^2 estimated as 0.00348:  log likelihood = 266.68,  aic = -527.36
>
>  Box-Ljung test
>
> data:  res.arma$resid
> X-squared = 17.429, df = 12, p-value = 0.1342
```

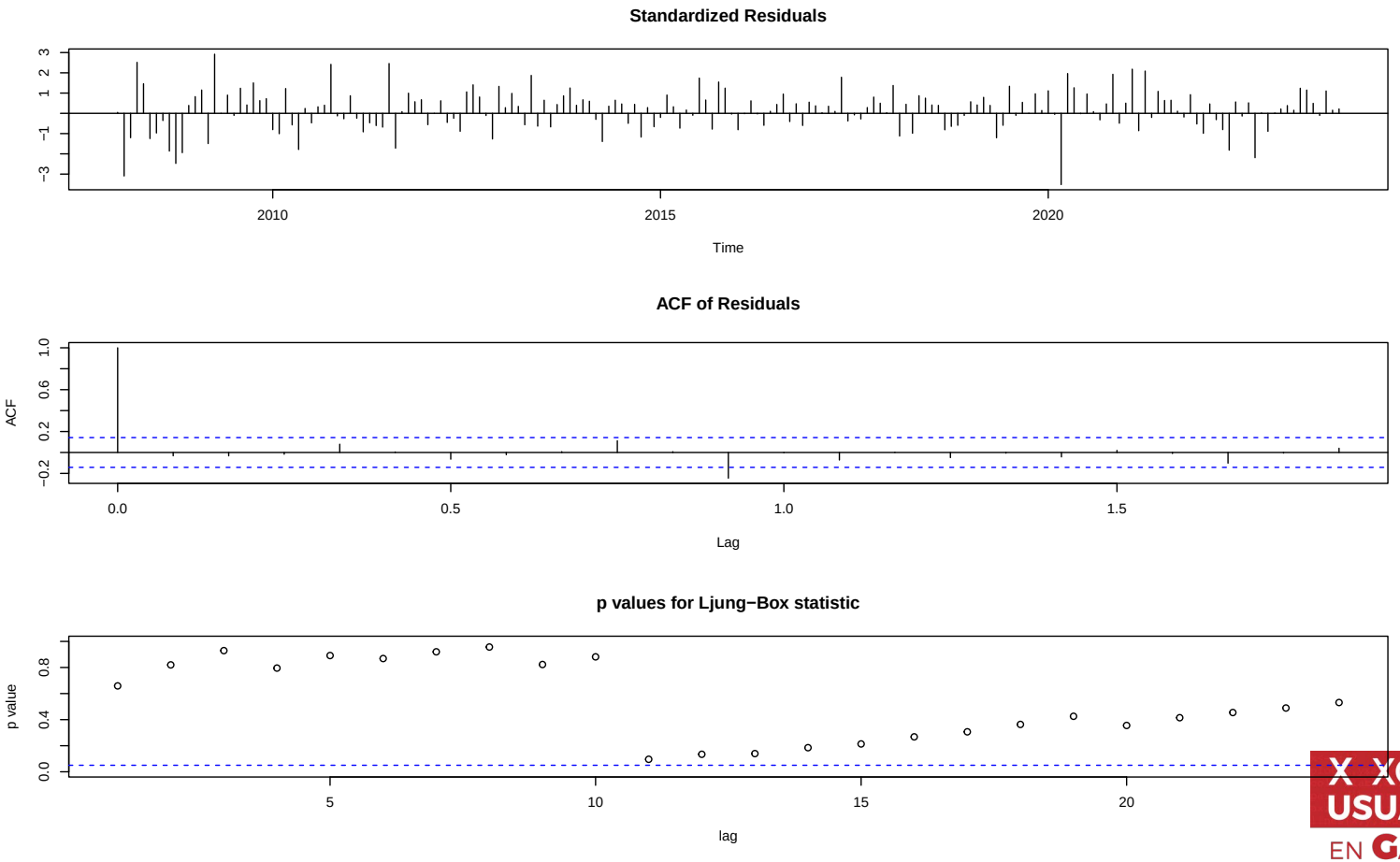
# ARIMA Estimation II

## ARIMA$(2,1,0)\times(0,0,1)_{12}$

```
res.arma2 = arima(lgoog.ts, order = c(2, 1, 0), seasonal = list(order = c(0,
    0, 1), period = 12))  #Worse AIC

>
> Call:
> arima(x = lgoog.ts, order = c(2, 1, 0), seasonal = list(order = c(0, 0, 1),
>     period = 12))
>
> Coefficients:
>          ar1      ar2    sma1
>       0.3703  -0.1217  0.0263
> s.e.  0.0766   0.0756  0.0834
>
> sigma^2 estimated as 0.003478:  log likelihood = 266.73,  aic = -525.46
>
>  Box-Ljung test
>
> data:  res.arma2$resid
> X-squared = 17.499, df = 12, p-value = 0.1318
```

# Diagnosis ARIMA

```r
tsdiag(res.arma, gof.lag = 24)
# tsdiag(res.arma2,gof.lag=24)
```



# Package forecast

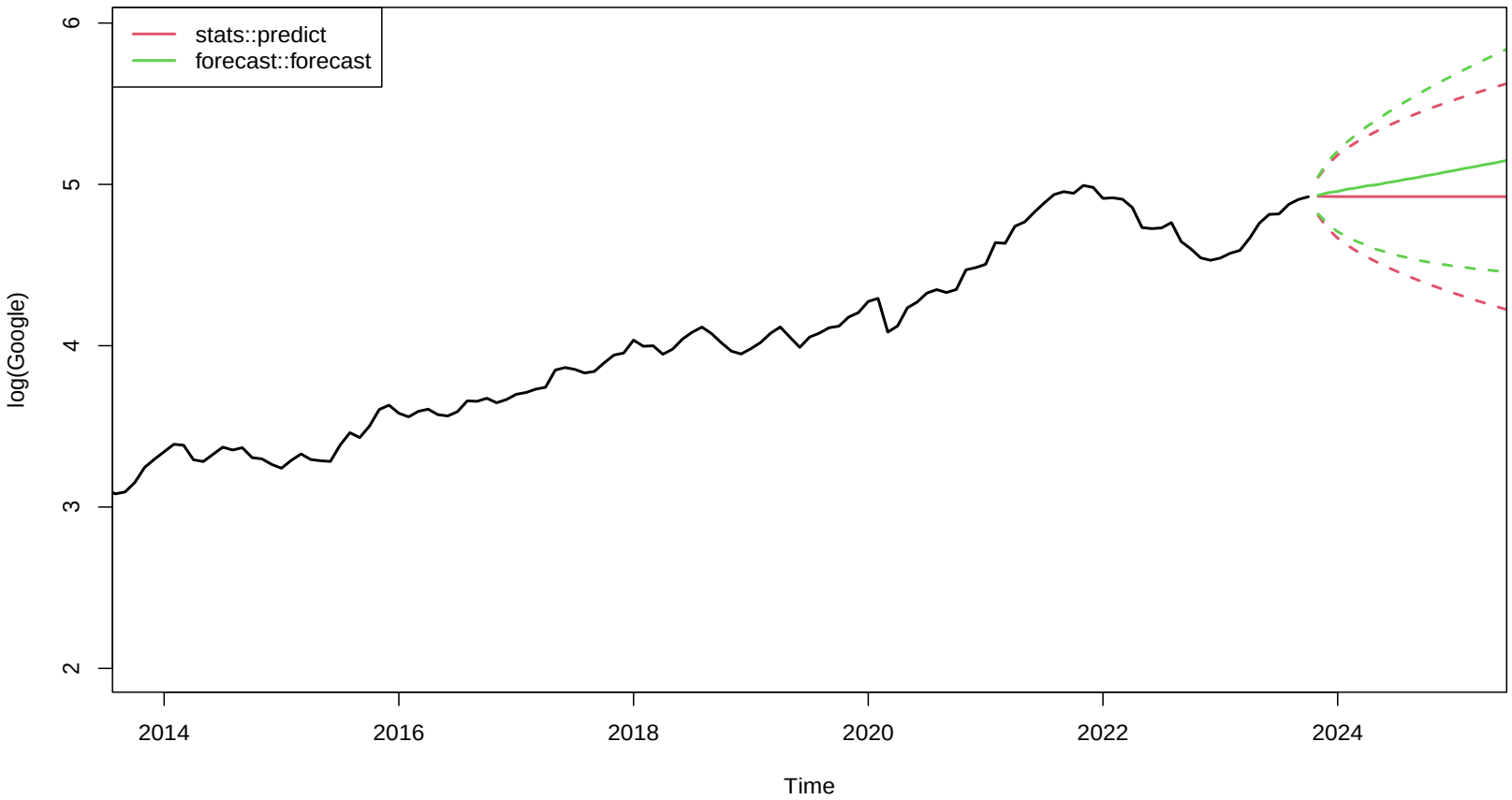
Several utilities that summarizes/ functions from other packages.
Includes exploratory and diagnostic plots and ARFIMA modelling.

- `auto.arima`: Selects an ARIMA model fulfilling some diagnostics.
- `tbats`: Exp. smoothing, State Space Model with Box-Cox transformation, ARMA errors, Trend and Seasonal Components.
- `forecast`: Alternative to classical `predict` for several models (not only ARMA)
- `HoltWinters`: Prediction with Holt-Winters.
- `StructTS`: Structural time series.
- `seasonplot`: Seasonal plot
- `tsdisplay`: Common plot for time series and ACF and PACF functions.

# Automatic ARIMA.

```
library(forecast)
aufit = auto.arima(lgoog.ts, d = 1, max.p = 5, max.q = 5, max.order = 6,
    max.d = 2)

> Series: lgoog.ts
> ARIMA(1,1,2)(1,0,0)[12] with drift
>
> Coefficients:
>           ar1     ma1     ma2     sar1   drift
>       -0.9393  1.2839  0.3410  -0.0293  0.0114
> s.e.   0.0878  0.1119  0.0758   0.0846  0.0056
>
> sigma^2 = 0.00349:  log likelihood = 268.95
> AIC=-525.91   AICc=-525.45   BIC=-506.46
>
>  Box-Ljung test
>
> data:  aufit$residuals
> X-squared = 17.805, df = 12, p-value = 0.1217
```

# ARIMA Prediction I



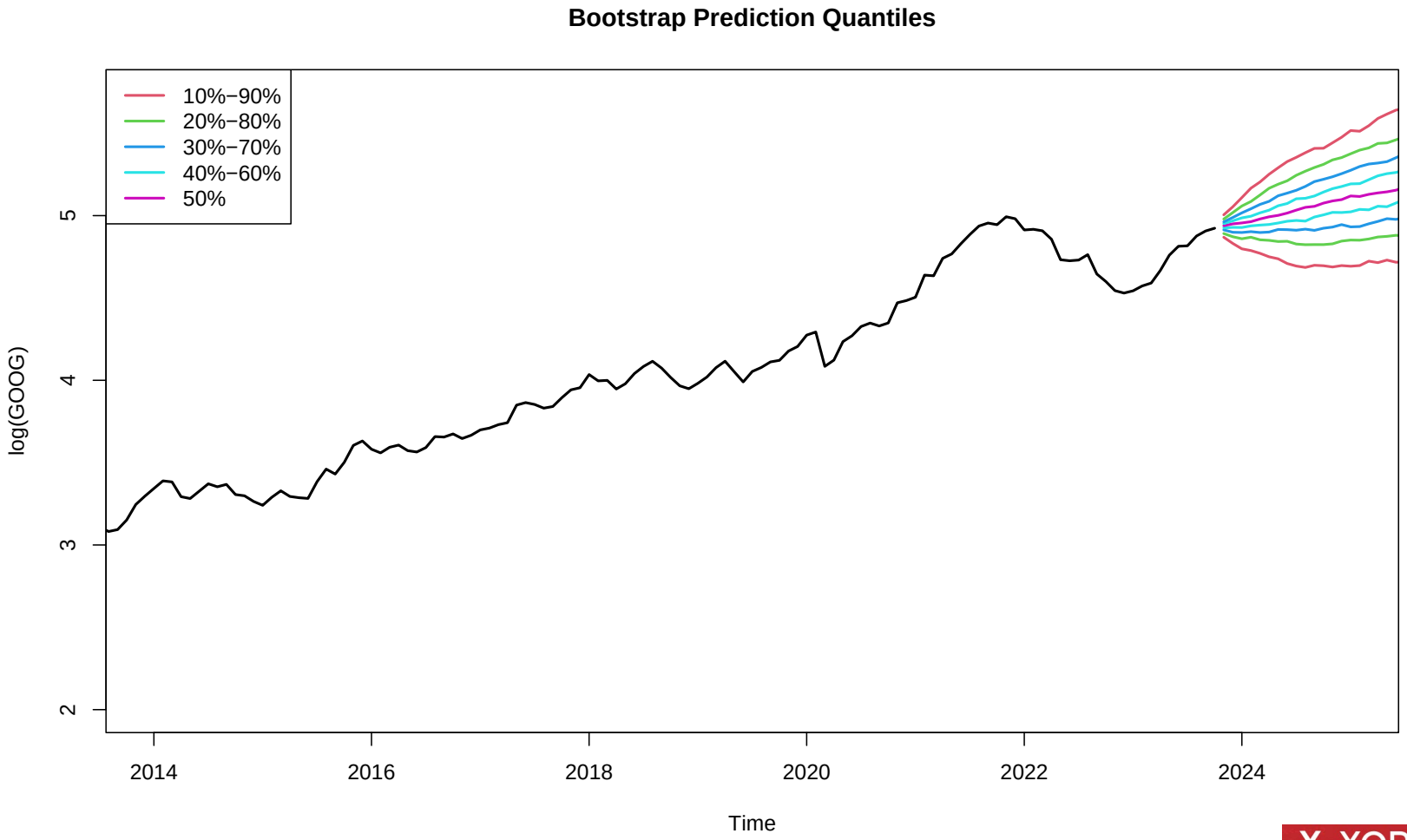
# ARIMA Simulation I

## 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)
```

```
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 II



# Other packages

This package extends the classical tools of ARIMA modelization including TAR and Transfer models.

- Package `TSA`: Extensions to TAR and Transfer models.
  - `eacf`: Extended ACF.
  - `arima, arimax`: ARIMA and transfer models.
  - `detectAO, detectIO`: Outliers
  - Tests: `runs`:Ind, `McLeod.Li.test`:Garch, `Keenan.test`:NLin, `Tsay.test`:Cuad.AR, `tlrt`:TAR
  - `tar, predict.tar`: Threshold AR Models.
- `FinTS`: Associated with Tsay's book with shortcuts to other packages.
  - `ARIMA`$\Longrightarrow$ `stats::arima + stats::Box.test`, `Unitroot`$\Longrightarrow$ Shortcut to `tseries::adf.test` `fUnitRoots::adfTest` and `urca::ur.df`
- `PerformanceAnalytics`: Mostly devoted to graphical tools and summaries.
- `fArma`: From Rmetrics.org is now outdated (v. 3.x to 4.x)

# Table of Contents

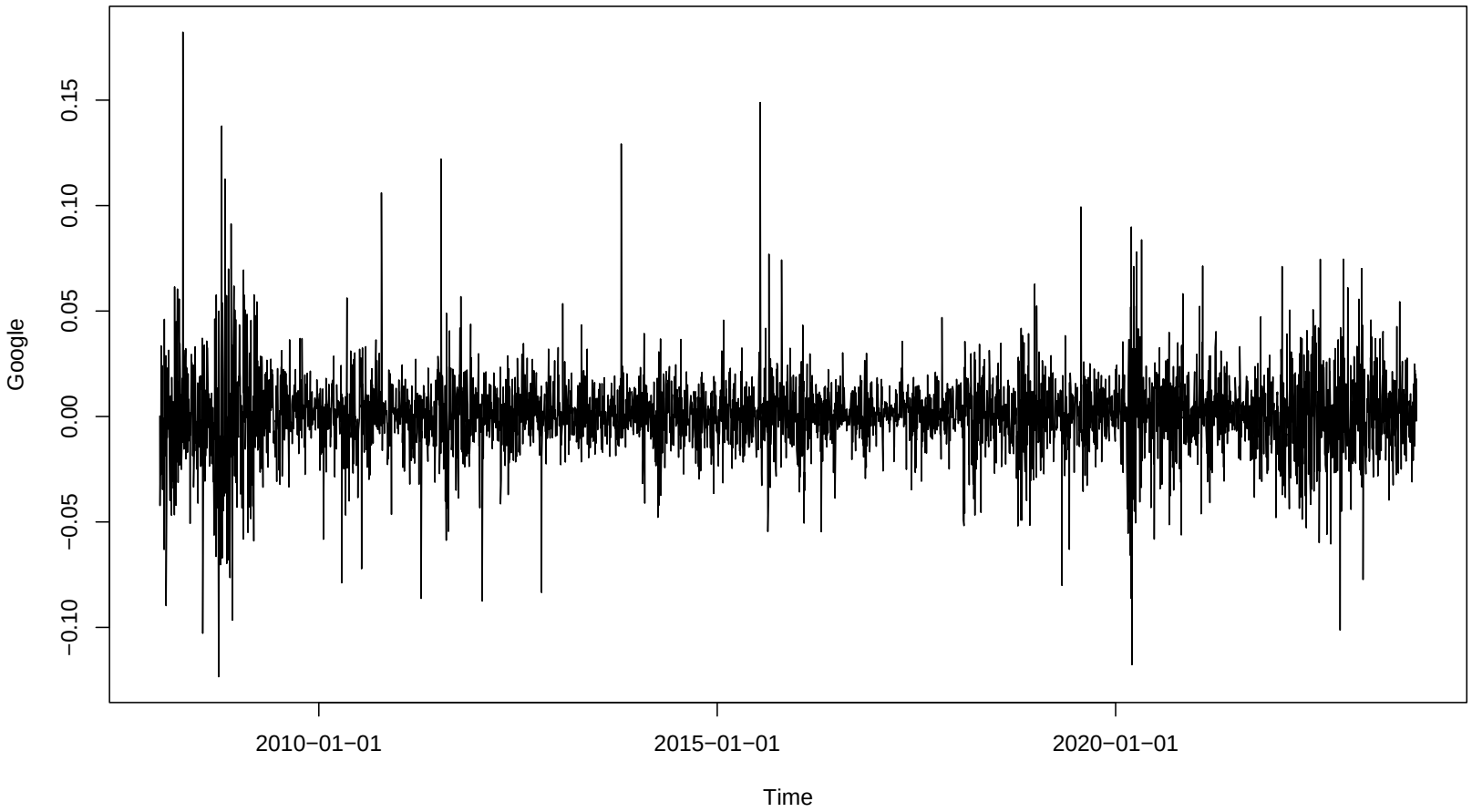
# Returns

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




Google Returns: Skewness=0.314, Kurtosis=8.792

# GARCH Models

## GARCH(p,q)

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

where $\{w_t\}$ is a standard white noise and $\sigma_t^2 = \mathrm{Var}\left(\epsilon_t | \mathcal{F}_{t-1}\right)$
$\epsilon_t \sim GARCH(p,q) \longrightarrow \epsilon_t^2 \sim ARMA(\max(p,q), q)$

# 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

- SGARCH: $\sigma_t^2=\underbrace{\left(\omega+\sum_{j=1}^{m}\varsigma_j\nu_{j,t}\right)}_{\omega_t}+\sum_{j=1}^{p}\beta_j\sigma_{t-j}^2+\sum_{j=1}^{q}\alpha_j\epsilon_{t-j}^2$

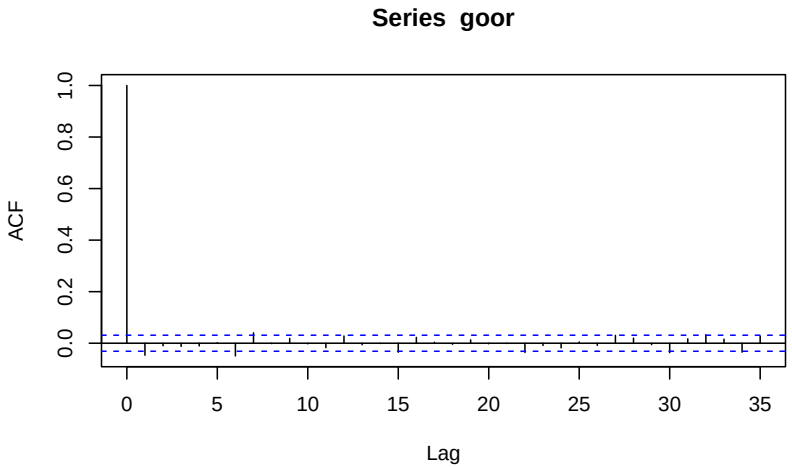
# Extension ARIMA-GARCH Models II

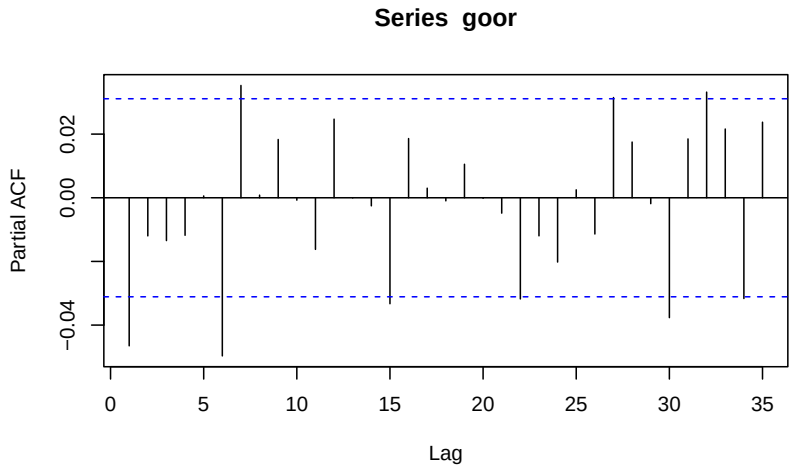
- EGARCH: $\log \sigma_t^2 = \omega_t + \sum_{j=1}^{p} \beta_j \log \sigma_{t-j}^2 + \sum_{j=1}^{q} \left( \alpha_j \frac{|\epsilon_{t-j}| + \gamma_i \epsilon_{t-j}}{\sigma_{t-i}} \right)$
- GJR-GARCH:
  $\sigma_t^2 = \omega_t + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 + \sum_{j=1}^{q} \left( \alpha_j \epsilon_{t-j}^2 + \gamma_j I_{(\epsilon_{t-j}<0)} \epsilon_{t-j}^2 \right)$
- 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:
  $\sigma_t^\lambda =$
  $\omega_t + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^\lambda + \sum_{j=1}^{q} \alpha_j \sigma_{t-j}^\lambda \left( \left| \frac{\epsilon_{t-j}}{\sigma_{t-j}} - \eta_{2j} \right| - \eta_{1j} \left( \frac{\epsilon_{t-j}}{\sigma_{t-j}} - \eta_{2j} \right) \right)^\delta$
- Component GARCH:
  $\sigma_t^\lambda = q_t + \sum_{j=1}^{p} \beta_j \left( \sigma_{t-j}^2 - q_{t-j}^2 \right) + \sum_{j=1}^{q} \alpha_j \left( \sigma_{t-j}^2 - q_{t-j} \right)$
  $q_t = \omega + \rho q_{t-1} + \phi \left( \epsilon_{t-1}^2 - \sigma_{t-1}^2 \right)$

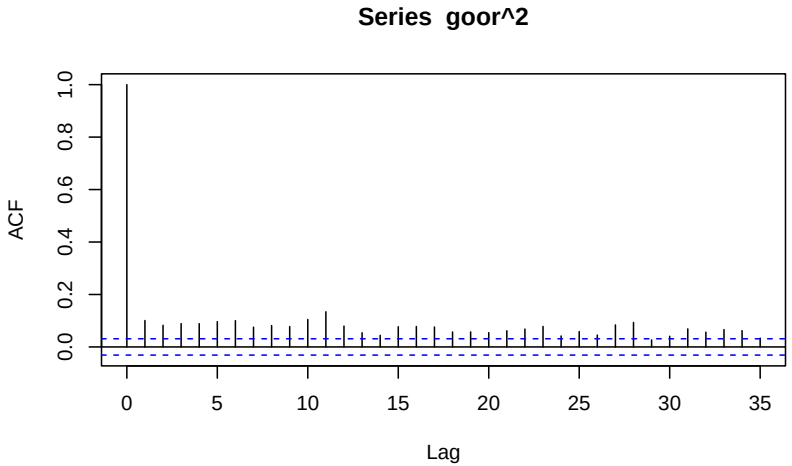
# rugarch I

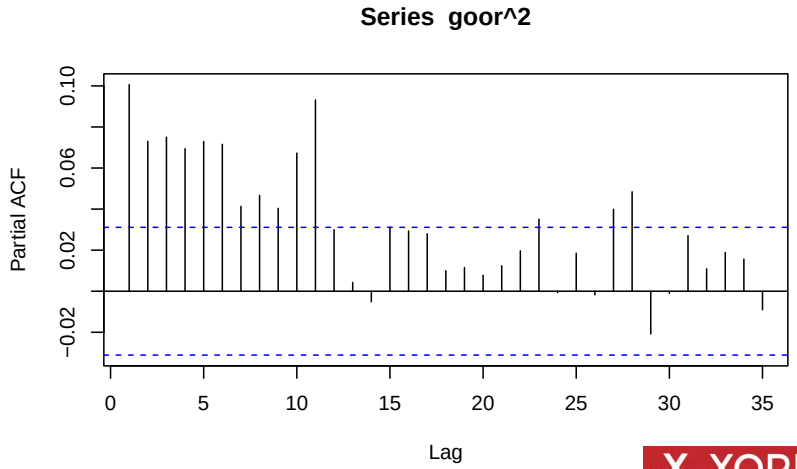
```
# ar(goor,order.max=10)
par(mfrow = c(2, 2))
acf(goor)
pacf(goor)
acf(goor^2)
pacf(goor^2)
monday = ifelse(dayOfWeek(time(goor)) == "Mon", 1, 0)
# spec=ugarchspec(mean.model=list(armaOrder=c(0,0)),
# variance.model=list(model='sGARCH', garchOrder=c(1,1)))
# fit=ugarchfit(spec,data=goor)
```

# rugarch II


Series goor
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35


Series goor
Partial ACF
Lag
−0.04
0.00
0.02
0
5
10
15
20
25
30
35


Series goor^2
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35


Series goor^2
Partial ACF
Lag
−0.02
0.02
0.06
0.10
0
5
10
15
20
25
30
35

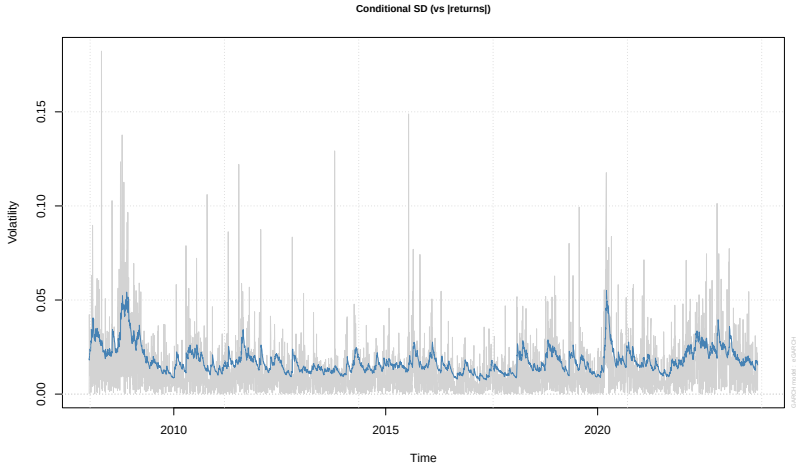
# Best model

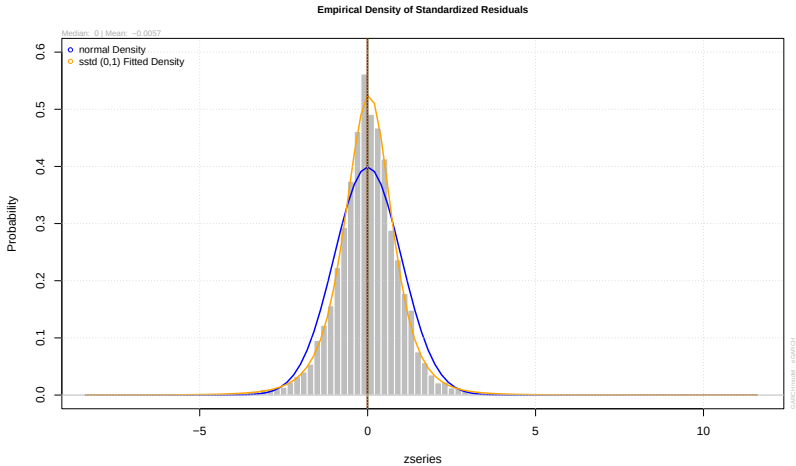
```
spec2 = ugarchspec(variance.model = list(model = "eGARCH", garchOrder = c(1,
    1), external.regressors = cbind(monday)), mean.model = list(armaOrder = c(0,
    0)), distribution.model = "sstd")
fit2 = ugarchfit(spec2, data = goor)
```

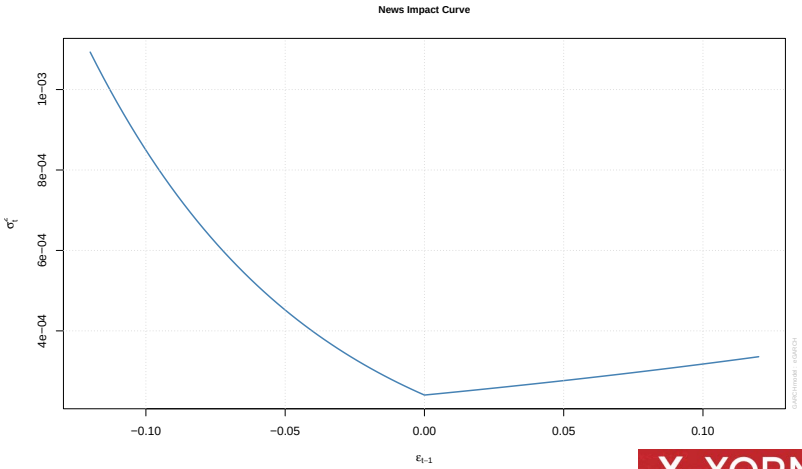
```
....

> Optimal Parameters
> ------------------------------------
>         Estimate  Std. Error    t value Pr(>|t|)
> mu      0.000668    0.000211    3.15865 0.001585
> omega  -0.117437    0.013970   -8.40652 0.000000
> alpha1 -0.079441    0.010113   -7.85515 0.000000
> beta1   0.984953    0.000617 1596.30318 0.000000
> gamma1  0.124388    0.015109    8.23260 0.000000
> vxreg1 -0.035975    0.068203   -0.52746 0.597872
> skew    0.966323    0.020567   46.98477 0.000000
> shape   4.089307    0.272240   15.02095 0.000000
....
```

# GARCH plots


Series with 2 Conditional SD Superimposed
Returns
Time
2010
2015
2020


Conditional SD (vs |returns|)
Volatility
Time
2010
2015
2020


Empirical Density of Standardized Residuals
normal Density
sstd (0,1) Fitted Density
Probability
zseries


News Impact Curve

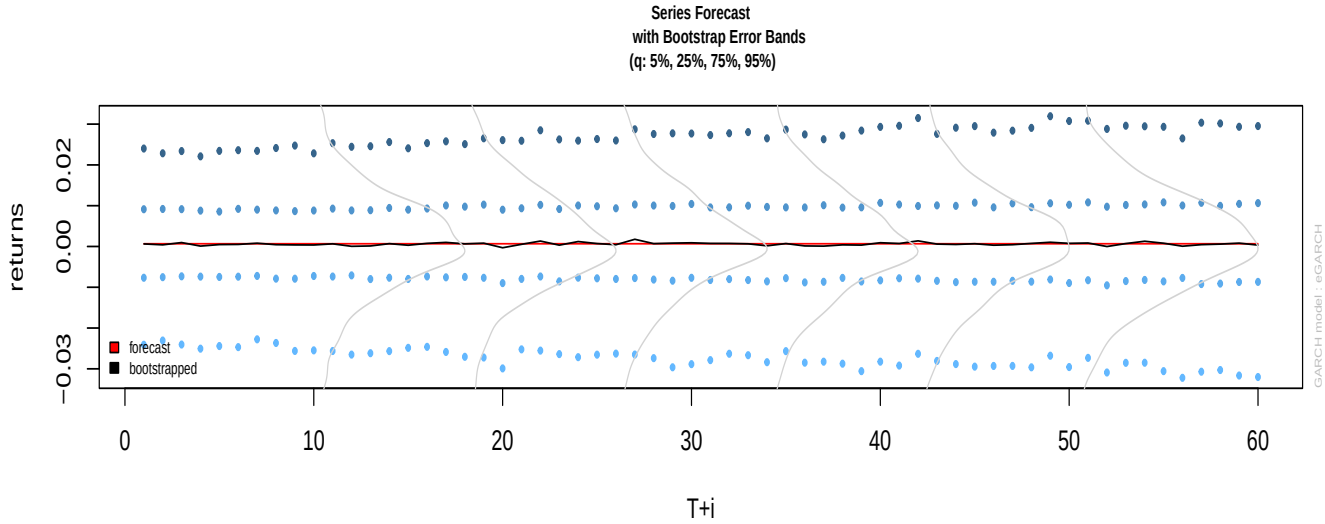
# GARCH forecasting I

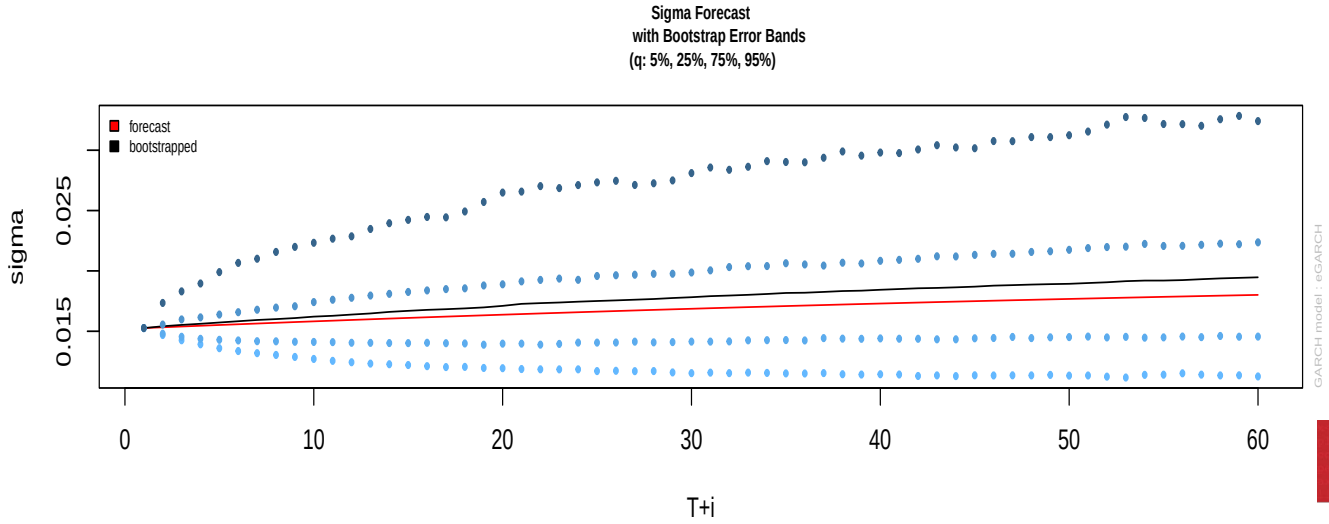
```
nmonday = rep(c(1, 0, 0, 0, 0), 12)
fit2.fore = ugarchforecast(fit2, n.ahead = 60, external.forecasts = list(mregfor = cbind(nm
# fit2.fore
```

## *Bootstrap*

```
fit2.boot = ugarchboot(fit2, method = "Partial", sampling = "spd",
    n.ahead = 60, external.forecasts = list(mregfor = cbind(nmonday)),
    n.bootpred = 2000, mexsimdata = t(rbind(replicate(2000, nmonday))))
```

# Bootstrap Plot



# More on Time Series

| Topic | Packages |
|---|---|
| Data Retrieval | `pdfetch`, `Quandl`, `alfred`, `readabs` |
| New classical | `dygraphs`, `fable`, `feasts`, `ggTimeSeries`, `modeltime`, `TSstudio` |
| Dynamic Regression | `dyn`, `dynlm`, `tsDyn`, `dse` |
| Multivariate TS | `vars`, `MTS`, `BigVAR`, `fsMTS`, `HDTSA`, `fMultivar` |
| Kalman Filter | `astsa`, `KFAS`, `FKF` |
| ARFIMA | `fracdiff` |
| Unit Root Test | `urca` |
| GARCH | `rugarch`, `rmgarch`, `fGarch` |
| Bayesian | `bayesforecast`, `bayesGARCH`, `mbsts`, `bmgarch` |

Many other topics are also represented in R: Outliers, ML forecast, Frequency analysis, Nonlinear models,...

# References


- Chan, K.S. and Ripley, B. (2022) TSA: Time Series Analysis (1.3.1). `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. (2022) urca: Unit root and cointegration tests for time series data (1.3-3). `http://cran.r-project.org/web/packages/urca/`.
- Maechler, M. (2022) fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models (1.5-2). `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.