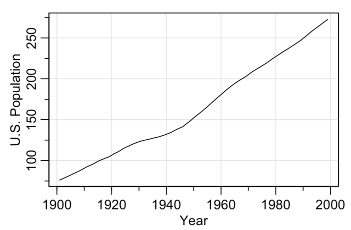
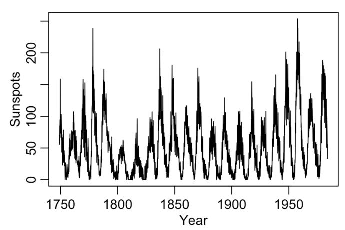
Time Series Cheat Sheet

Plot Time Series

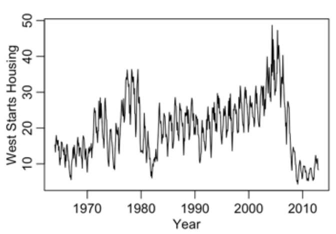
1. tsplot(x=time, y=data)



2. plot(ts(data, start=start_time, frequency=gap))



3. ts.plot(ts(data, start=start_time, frequency=gap))



Simulation

Autoregression of Order p

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + W_t$$

Moving Average of Order q

$$X_t = Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2} + \ldots + \theta_q Z_{t-p}$$

ARMA (p, q)

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2} + \ldots + \theta_q Z_{t-p}$$

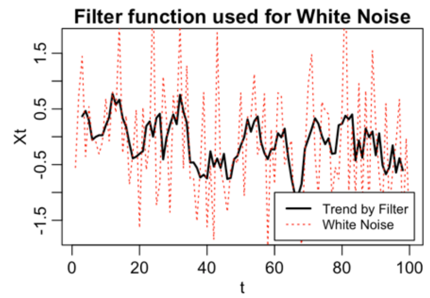
Simulation of ARMA (p, q)

arima.sim(model=list(ar=c($\phi_1, \ldots, \phi_p$), ma=c($\theta_1, \ldots, \theta_q$)), n=n)

Filters

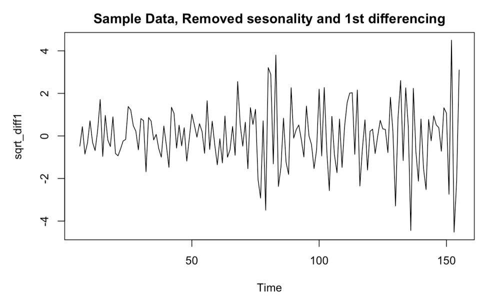
Linear Filter: filter()

filter(data, filter=filter_coefficients, sides=2, method="convolution", circular=F)



Differencing Filter: diff()

diff(data, lag=4, differences=1)

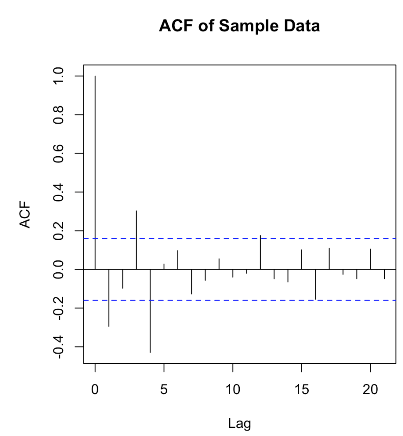


Auto-correlation

Use ACF and PACF to detect model

(Complete) Auto-correlation function: acf()

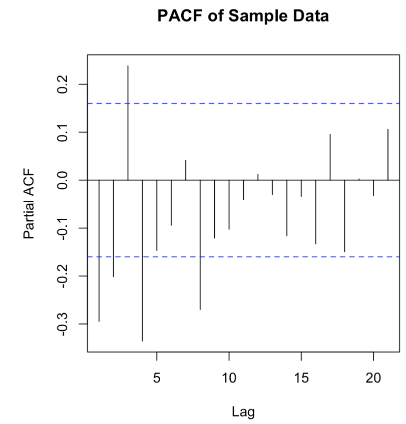
acf(data, type='correlation', na.action=na.pass)



Partial Auto-correlation function: pacf()

pacf(data, na.action=na.pass)

OR: acf(data, type='partial', na.action=na.pass)



Parameter Estimation

Fit an ARMA time series model to the data

ar(): To estimate parameters of an AR model

ar(x=data, aic=T, order.max = NULL, c("yule-walker", "burg", "ols", "mle", "yw"))

```
Call:
ar(x = sqrt1, aic = TRUE, order.max = NULL, method = c("yule-walker",     "burg", "ols", "mle", "yw"))

Coefficients:
      1        2        3        4        5        6        7        8        9
-0.3066  -0.1903   0.0793  -0.5065  -0.1873  -0.1149  -0.0580  -0.3031  -0.1207

Order selected 9  sigma^2 estimated as  1.52
```

arima(): To estimate parameters of an AM or ARMA model, and build model

arima(data, order=c(p, 0, q),method=c('ML'))

```
Call:
arima(x = sqrt1, order = c(2, 0, 6), method = c("ML"))

Coefficients:
          ar1      ar2      ma1     ma2      ma3      ma4     ma5      ma6  intercept
      -0.1658  -0.7951  -0.1536  0.7524  -0.2101  -0.7680  0.0605  -0.6812    -0.0048
s.e.   0.0918   0.0754   0.0916  0.1047   0.0794   0.0743  0.0812   0.0895     0.0062

sigma^2 estimated as 1.193:  log likelihood = -230.78,  aic = 481.55
```

AICc(): Compare models using AICC

AICc(fittedModel)

Forecasting

Forecasting future observations given a fitted ARMA model

predict(): Predict future observations given a fitted ARMA model

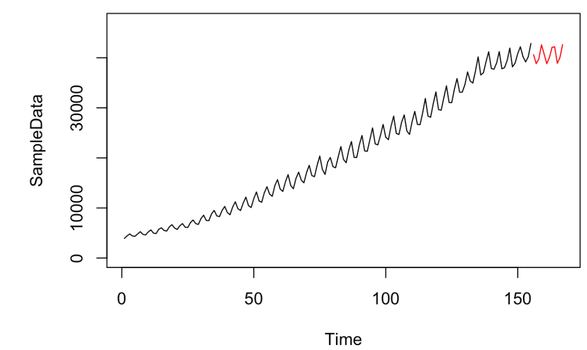
predict(arima_model, number_to_predict)

Plot Predicted values and Confidence Interval:

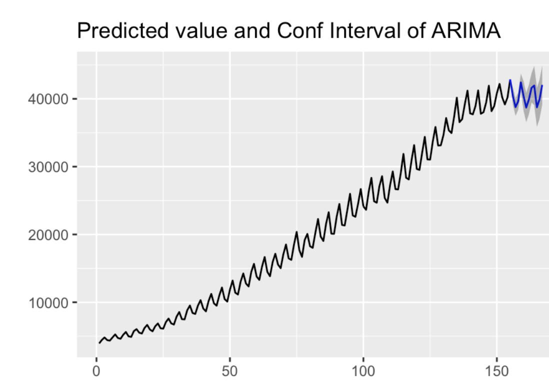
fit<-predict(arima_model, number_to_predict)

ts.plot(data,
xlim=c(1, length(data)+number_to_predict),
ylim=c(0, max(fit$pred+1.96*fit$se)))

lines(length(data)+1:length(data)+
number_to_predict, fit$pred)



OR: autoplot(forecast(arima_model, level=c(95), h=number_to_predict))



 Yunjun Xia, Shuyu Huang • yx2569@columbia.edu, sh3967@columbia.edu • Updated: 2019-10

Class Agnostic Time Series with tsbox : : CHEAT SHEET

Basics

IDEA

tsbox provides a time series toolkit which:

1. works identically with most time series **classes**
2. handles regular and irregular **frequencies**
3. **converts** between classes and frequencies

Most functions in tsbox have the same structure:

function starts with ts_

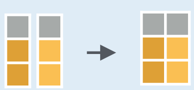
first argument is any ts-boxable object

```
a <- ts_pc(AirPassengers)
```

returns a ts-boxable obect of the same class as input

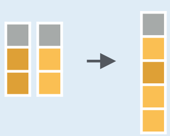
COMBINE TIME SERIES

collect time series of **all classes** and **frequencies** as multiple time series



```
ts_c(mdeaths, austres)
```

combine time series to a new, single time series (first series wins if overlapping)



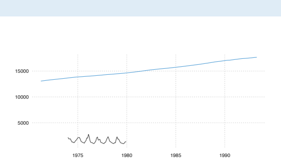
```
ts_bind(mdeaths, austres)
```

like ts_bind, but extra- and retropolate, using growth rates

```
ts_chain(mdeaths, austres)
```

PLOT AND SUMMARIZE

Plot time series of **all classes** and **frequencies**



```
ts_plot(mdeaths, austres)
ts_ggplot(mdeaths, austres)
```

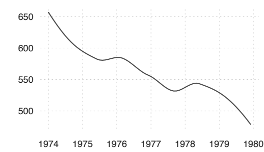
```
ts_summary(ts_c(mdeaths, austres))
```

	id	obs	diff	freq	start	end
1	mdeaths	72	1 month	12	1974-01-01	1979-12-01
2	austres	89	3 month	4	1971-04-01	1993-04-01

Helper Functions

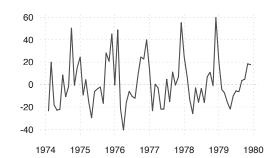
Transform time series of **all classes** and **frequencies**

TRANSFORM



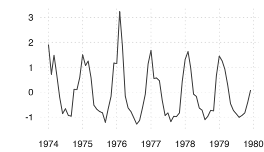
ts_trend(): Trend estimation based on loess

```
ts_trend(fdeaths)
```



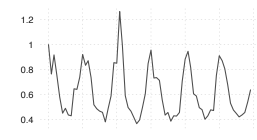
ts_pc(), ts_pcy(), ts_pca(), ts_diff(), ts_diffy(): (annualized) Percentage change rates or differences to previous period, year

```
ts_pc(fdeaths)
```



ts_scale(): normalize mean and variance

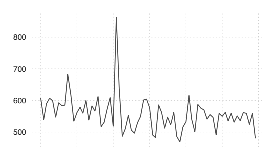
```
ts_scale(fdeaths)
```



ts_index():Index, based on levels

ts_compound(): Index, based on growth rates

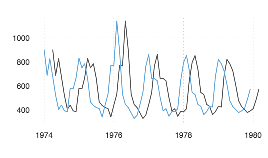
```
ts_index(fdeaths, base = 1976)
```



ts_seas(): seasonal adjustment using X-13

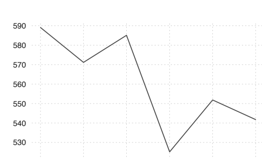
```
ts_seas(fdeaths)
```

SPAN AND FREQUENCY



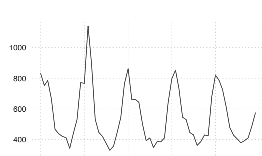
ts_lag(): Lag or lead of time series

```
ts_lag(fdeaths, 4)
```



ts_frequency(): convert to frequency

```
ts_frequency(fdeaths, "year")
```



ts_span(): filter time series for a time span.

```
ts_span(fdeaths, "1976-01-01")
ts_span(fdeaths, "-5 year")
```

Class Conversion

tsbox is built around a set of converters, which convert time series of the following **supported classes** to each other:

converter function	ts-boxable class
ts_ts()	ts, mts
ts_data.frame(), ts_df()	data.frame
ts_data.table(), ts_dt()	data.table
ts_tbl()	df_tbl, "tibble"
ts_xts()	xts
ts_zoo()	zoo
ts_tibbletime()	tibbletime
ts_timeSeries()	timeSeries
ts_tsibble()	tsibble
ts_tslist()	a list with ts objects

Time Series in data frames

LONG STRUCTURE

Default structure to store multiple time series in long data frames (or data tables, or tibbles)

```
ts_df(ts_c(fdeaths, mdeaths))
```

id	time	value
fdeaths	1974-01-01	901
fdeaths	1974-02-01	689
fdeaths	1974-03-01	827
...	...	...

AUTO-DETECT COLUMN NAMES

tsbox auto-detects a *value*-, a *time*- and zero, one or several *id*-columns. Alternatively, the *time*- and the *value*-column can be explicitly named **time** and **value**.

ts_default(): standardize column names in data frames

RESHAPE

ts_wide(): convert default long structure to wide

ts_long(): convert wide structure to default long

USE WITH PIPE

tsbox plays well with tibbles and with `%>%`, so it can be easily integrated into a dplyr/pipe workflow

```
library(dplyr)
ts_c(fdeaths, mdeaths) %>%
  ts_tbl() %>%
  ts_trend() %>%
  ts_pc()
```

pass return value as first argument to the next function