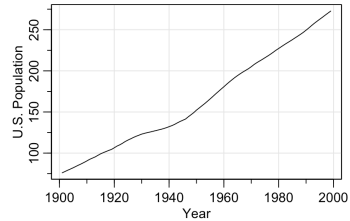


Time Series Cheat Sheet

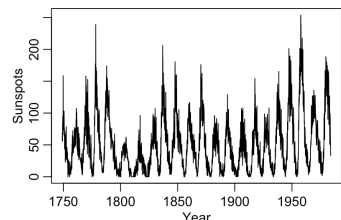


Plot Time Series

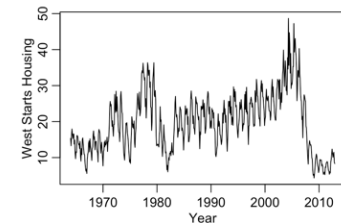
1. `tsplot(x=time, y=data)`



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



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



Simulation

Autoregression of Order p

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

Moving Average of Order q

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

ARMA (p, q)

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

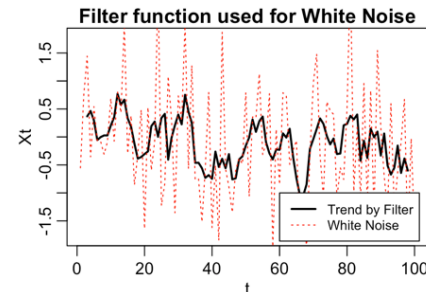
Simulation of ARMA (p, q)

`arima.sim(model=list(ar=c(ϕ_1, \dots, ϕ_p),
ma=c($\theta_1, \dots, \theta_q$)), n=n)`

Filters

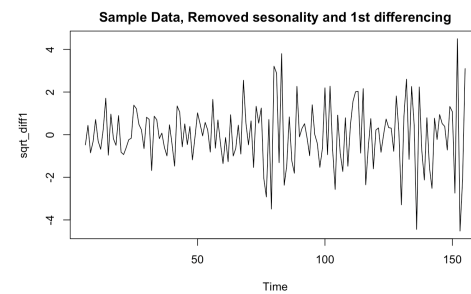
Linear Filter: filter()

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



Differencing Filter: diff()

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

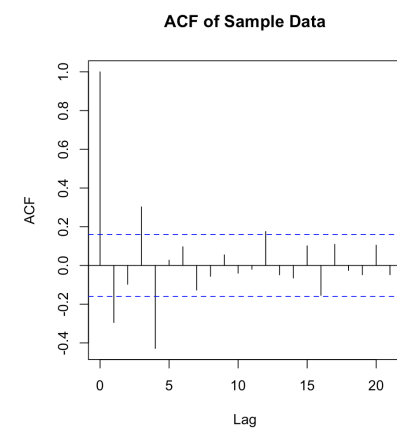


Auto-correlation

Use ACF and PACF to detect model

(Complete) Auto-correlation function: acf()

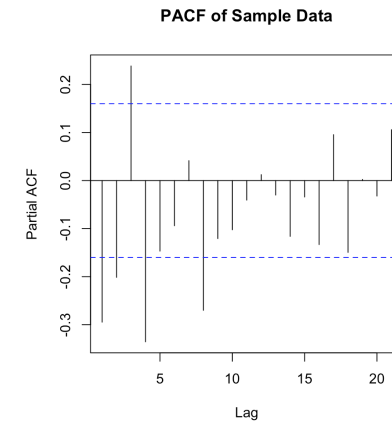
`acf(data, type='correlation', na.action=na.pass)`



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, o, 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

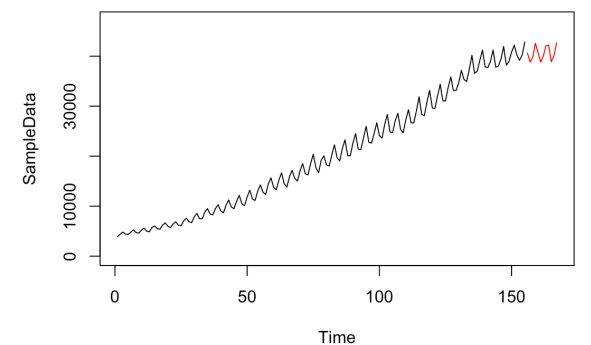
predict(): Predict future observations given a fitted ARMA model

`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))`

