# Signal + Noise

|  |
| --- |
| t = seq(1:100) a = rnorm(n = 100, mean = 0, sd = 1) X = ts(3\*cos(2\*pi\*0.05\*t) + 1.5\*cos(2\*pi\*0.35\*t + 2) + a) plotts.sample.wge(X)  *# Equivalently* X2 = **gen.sigplusnoise.wge**(n = 100, coef = **c**(3,1.5), freq = **c**(0.05, 0.35), psi = **c**(0, 2)) |

# Plottign a Time Series

|  |
| --- |
| **plotts.wge**(airlog) *# equivalent to using native plot (shown below)*  **plot**(airlog, type = 'l') |
| *# Converting vector to Time Series and plotting*  t = **seq**(1,250) y1 = **ts**(**sin**(2**\***pi**\***0.1**\***t)) **plot**(y1) |

# Determining Stationarity of a Time Series

|  |
| --- |
| **library**(tswge)  **source**("common\_functions.R") |
| **acf**(lavon[1**:**56]) *# Computes ACF* |
| **check\_stationarity**(whale) *# All-inclusive function to compute stationarity* |

# Diagnostic Plots in a Time Series

## Spectral Density (Parzen Window)

|  |
| --- |
| p = **parzen.wge**(fig1.21a, trunc = 31, dbcalc = FALSE) *# Plots Spectral Density in nominal terms*  p = **parzen.wge**(fig1.21a, trunc = 31, dbcalc = TRUE) *# Plots Spectral Density in dB* |

## All Diagnostic Plots together (Realization + ACF + Spectral Density)

|  |
| --- |
| p = **plotts.sample.wge**(fig1.21a, trunc = 31) *# Plots Spectral Density in dB by default. Use parzen.wge to plot parzen plot with more flexibility* |

## True ACF + Spectral Density

|  |
| --- |
| p = **plotts.true.wge**(n = 200, phi = 0.9) *# AR(1) with phi = 0.9* |

# Filtering

## Moving Average (Low Pass)

|  |
| --- |
| *# 5 point MA*  ma = stats**::filter**(x, **rep**(1,5))**/**5 p = **plotts.sample.wge**(**na.omit**(ma)) *# Can not have NA values created by the moving average filters* |

## Difference Filer (High Pass)

|  |
| --- |
| dif = **diff**(fig1.21a, lag = 1) p = **plotts.sample.wge**(dif) |

## Butterworth Filter

|  |
| --- |
| *# Low Pass*  xa = **butterworth.wge**(x, order = 4, type = "low", cutoff = **c**(.2))  p = **plotts.sample.wge**(xa**$**x.filt)  *# High Pass*  xb = **butterworth.wge**(x, order = 4, type = "high", cutoff = **c**(.2))  p = **plotts.sample.wge**(xb**$**x.filt)  *# Band Pass*  xf = **butterworth.wge**(x, order=4, type='pass', cutoff=**c**(0.1,0.2)) |

# Autoregressive Models

## AR(1)

|  |
| --- |
| *# generating an AR(1) realization*  phi = **c**(0.9)  x\_0.9 = **gen.arma.wge**(n=200, phi=phi, vara=1, sn=2) |
| *# Computing the True variance of the realization (provided by tswge)*  *# (use this one)*  phi = **c**(0.9)  pt\_0.9 = **plotts.true.wge**(n = 200, phi = phi) *# AR(1) with phi = 0.9*  pt\_0.9**$**acv[1]  *Alternatively (my calculations)*  *# var\_x = var\_a / (1 – phi^2)*  **calculate\_ar1\_varx**(phi = phi, vara = 1) *# or use the common ar(p) function*  **calculate\_arp\_varx**(phi = phi, p = pt\_0.9, vara = 1) |

## AP(p)

|  |
| --- |
| *# generating an AR(p) realization with Mean of 10*  phi = **c**(1.5, **-**0.9) x = 10 **+** **gen.arma.wge**(n=200, phi = phi, sn = 101, plot = FALSE) px = **plotts.sample.wge**(x) |
| *# Computing the True variance of the realization (provided by tswge)*  *# (use this one)*  pt = **plotts.true.wge**(phi = phi)  varX = p**$**acv[1]  *Alternatively (my calculations)*  **calculate\_arp\_varx**(phi = phi, p = pt, vara = 1) |
| *# Computing the mean and variance of the realization*  *# Since the model is stationary, we can use all the data to compute mean and variance* muX\_est = **mean**(x) varX\_est = **var**(x) |
| *# Computing the factors of an AR(p) model*  phi = **c**(1.9, **-**2.3, 2, **-**1.2, 0.4) **factor.wge**(phi) |