MATH 4070: ARMA Models: Prediction and Forecasting

Whitney Huang, Clemson University

Contents

NOAA wind data example

This example is taken from Don Percival's time series course (UW Stat 519).

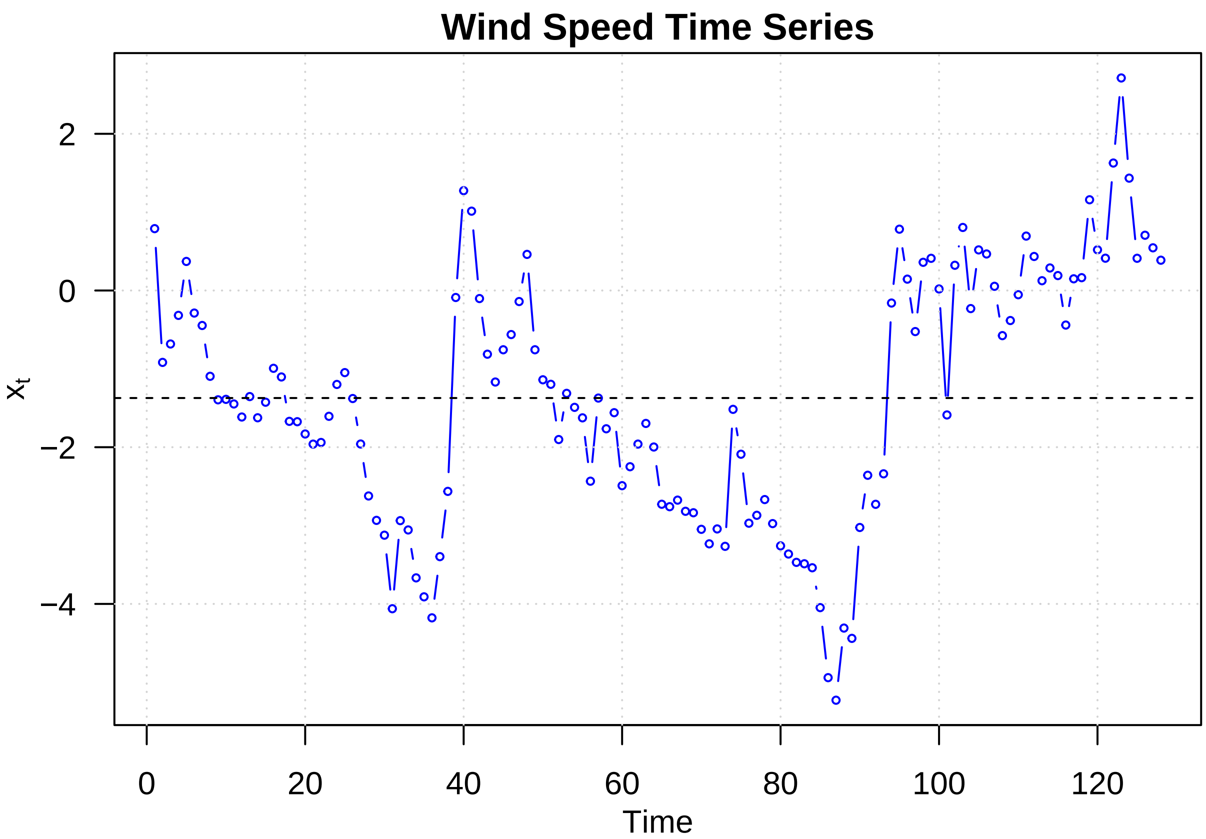
The one-step-ahead forecast of an AR(1) process is:

$$P_n X_{n+1} = \hat{\mu} + \hat{\phi}(X_n - \hat{\mu}),$$

where $\hat{\phi}$ is our estimate of ϕ, and $\hat{\mu}$ is an estimate of μ.

Load and plot the data

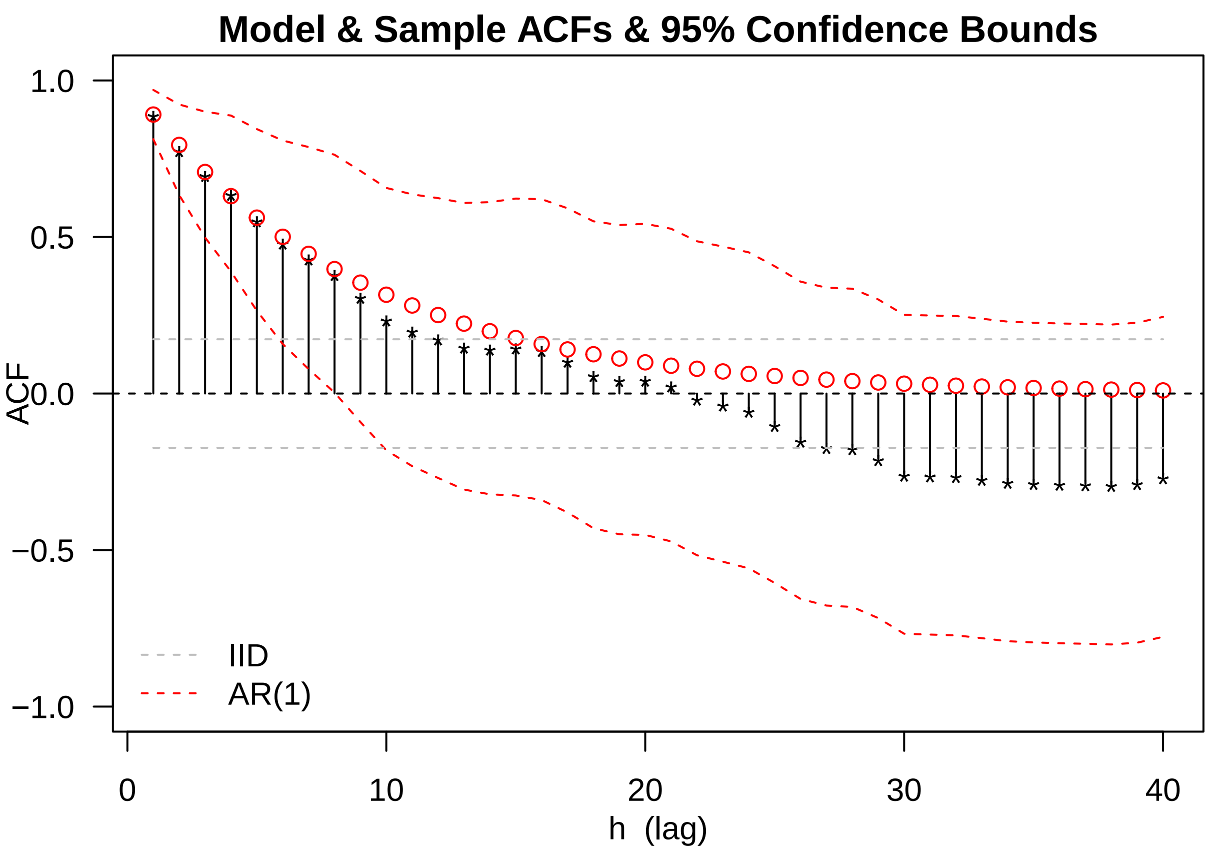
```
ws <- scan("http://faculty.washington.edu/dbp/s519/Data/wind-speed.txt")
n <- length(ws)
par(las = 1, mgp = c(2, 1, 0), mar = c(3.6, 3.6, 1.4, 0.6))
plot(ws, col = "blue", xlab = "Time", typ = "b",
     ylab = expression(x[t]), main = "Wind Speed Time Series", cex = 0.5)
grid()
xbar_ws <- mean(ws)
abline(h = xbar_ws, lty = 2)
```



“Estimate” ϕ using sample ACF and center the data

```
acf.ws <- acf(ws, lag.max = 40, plot = FALSE)$acf
phi.ws <- acf.ws[2] # this is an estimate for the coefficient of AR(1)

gen.whh.ar <- function(h, phi){
    p.2 <- phi^2; p.2h <- p.2^h
    - 2 * h * p.2h + (1 - p.2h) * (1 + p.2) / (1 - p.2)
}

plot.ACFbartlettAR <- function(ts, n.lags = 40){
    n.ts <- length(ts)
    lags <- 1:n.lags
    acf.est <- acf(ts, lag.max = n.lags, plot = FALSE)$acf[-1]
    acf.model <- acf.est[1]^lags
    plot(lags, acf.est, type = "h", xlab = "h  (lag)",
         ylab = "ACF", ylim = c(-1, 1),
         main = "Model & Sample ACFs & 95% Confidence Bounds", las = 1)
    points(lags, acf.est, pch = "*")
    points(lags, acf.model, col = "red")
    CI.AR <- 1.96 * sqrt(sapply(lags, function(h) gen.whh.ar(h, acf.est[1]))) / sqrt(n.ts)
    lines(lags, acf.est + CI.AR, col = "red", lty = 2)
    lines(lags, acf.est - CI.AR, col = "red", lty = 2)
    abline(h = 0, lty = "dashed")
    CI.IID <- rep(1.96 / sqrt(n), n.lags)
```

```r
    lines(lags, -CI.IID, col = "gray", lty = 2)
    lines(lags, CI.IID, col = "gray", lty = 2)
    legend("bottomleft", legend = c("IID", "AR(1)"), lty = "dashed",
           col = c("gray", "red"), bty = "n")
}

par(mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1.4, 0.6))
plot.ACFbartlettAR(ws)
```



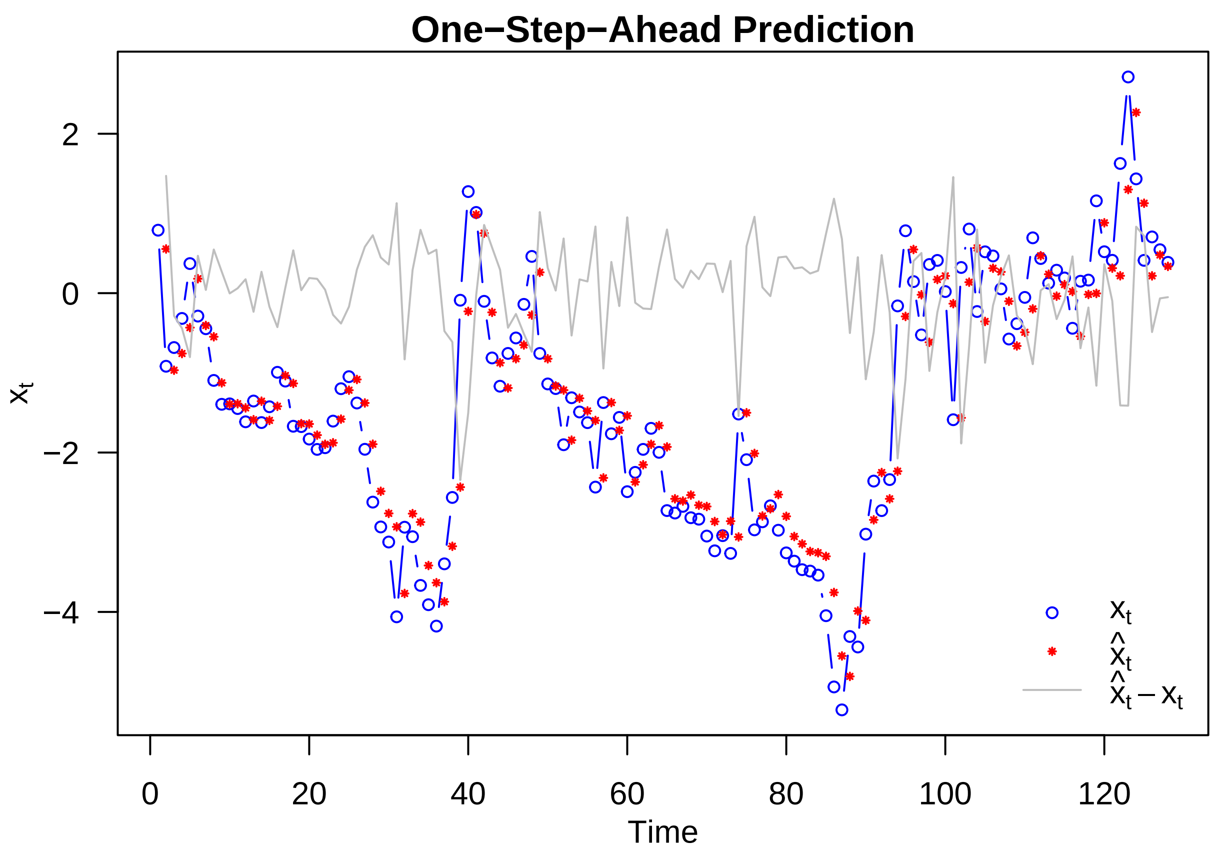

```r
## Alternatively, we can estimate phi using MLE
(phi_hat <- arima(ws, order = c(1, 0, 0)))
```

```
##
## Call:
## arima(x = ws, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##        0.906    -1.1136
## s.e.   0.037     0.6035
##
## sigma^2 estimated as 0.4615:  log likelihood = -132.99,  aic = 271.99
```

```r
ws.centered <- ws - xbar_ws
```

One-step-ahead forecast

```
ws.hat <- phi.ws * ws.centered[1:(n - 1)] + xbar_ws
## prediction errors
zt.ws <- ws.hat - ws[2:n]
## plot it
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1.2, 0.6))
plot(ws, col = "blue", xlab = "Time", type = "b", ylab = expression(x[t]),
     main = "One-Step-Ahead Prediction", cex = 0.75)
points(2:n, ws.hat, pch = 8, col = "red", cex = 0.375)
lines(2:n, zt.ws, col = "gray")
legend("bottomright", legend = expression(x[t], hat(x)[t], hat(x)[t] - x[t]),
       col = c("blue", "red", "gray"), pch = c(1, 8, NA),
       lty = c(NA, NA, "solid"), pt.cex = c(0.75, 0.375, 1), inset = 0.01,
       bty = "n")
```




```
var(zt.ws) # sample prediction variance
```

```
## [1] 0.4629379
```

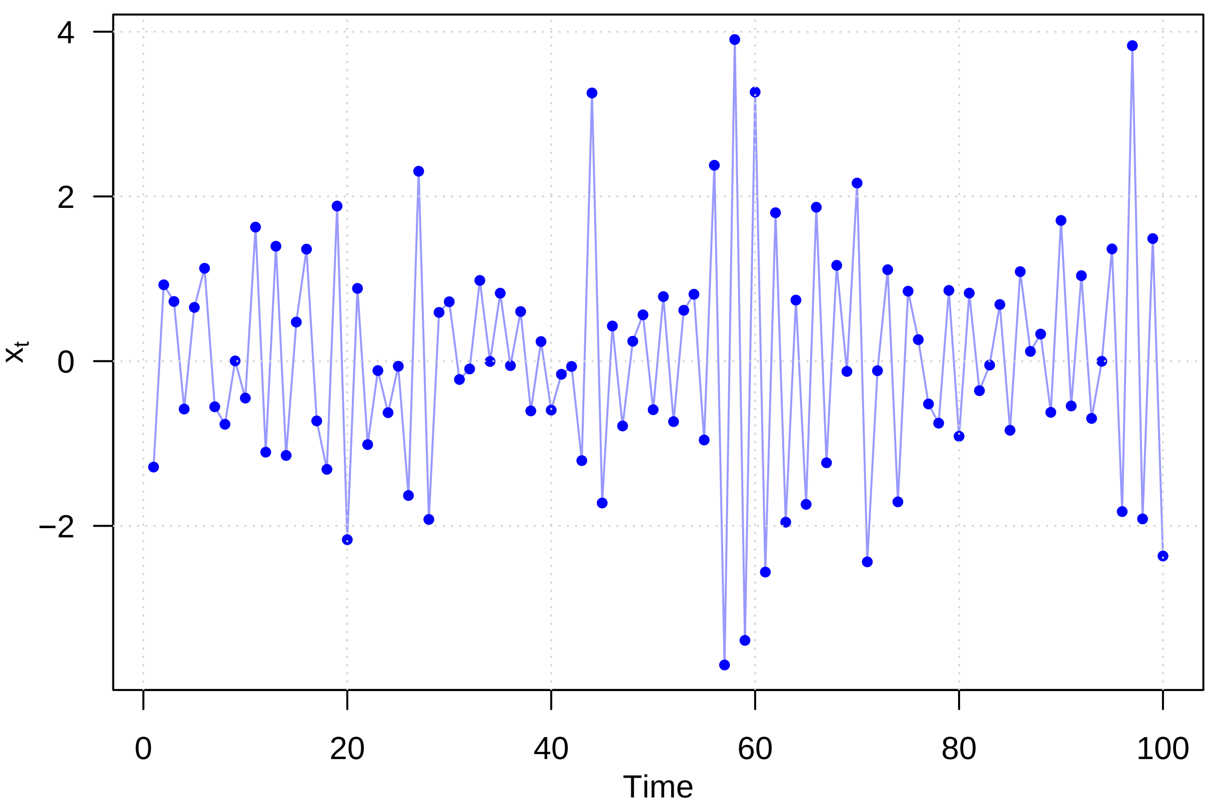
```
var(ws) # sample variance
```

```
## [1] 2.50251
```

Fill in missing value example

Simulate an AR(-0.9)

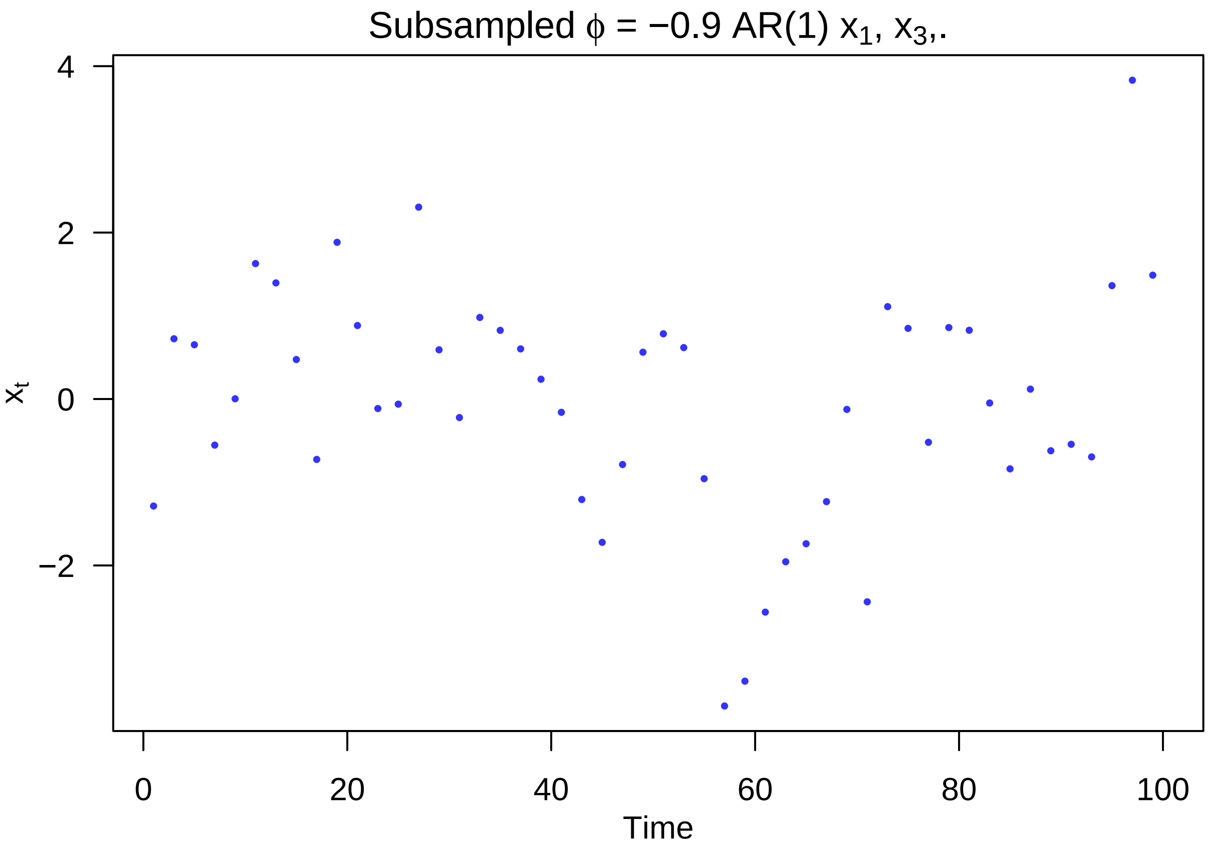
```
generate.AR1.ts <- function(phi = 0.0){
    ts <- rep(0, 100)
    ts[1] <- rnorm(1) / sqrt(1 - phi^2)
    for(i in 2:100) ts[i] <- phi * ts[i - 1] + rnorm(1)
    ts
}
set.seed(123)
ar1.ts <- generate.AR1.ts(-0.9)

library(scales)
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1.4, 0.6))
plot(ar1.ts, col = alpha("blue", 0.4), xlab = "Time", type = "l",
     ylab = expression(x[t]), cex = 0.5)
points(ar1.ts, pch = 16, cex = 0.75, col = "blue")
grid()
```



Let's remove some data to illustrate how to fill in missing values using forecasting algorithm

```
ar1.ts.subsampled <- ar1.ts
ar1.ts.subsampled[seq(2, 100, 2)] <- NA
```

```r
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1.4, 0.6))
plot(ar1.ts.subsampled, xlab = "Time", type = "b", ylab = expression(x[t]),
     main = expression(paste("Subsampled ", phi, " = -0.9 AR(1) ", x[1], ", ",x[3], ",.")),
     cex = 0.5, col = alpha("blue", 0.8), pch = 16)
```

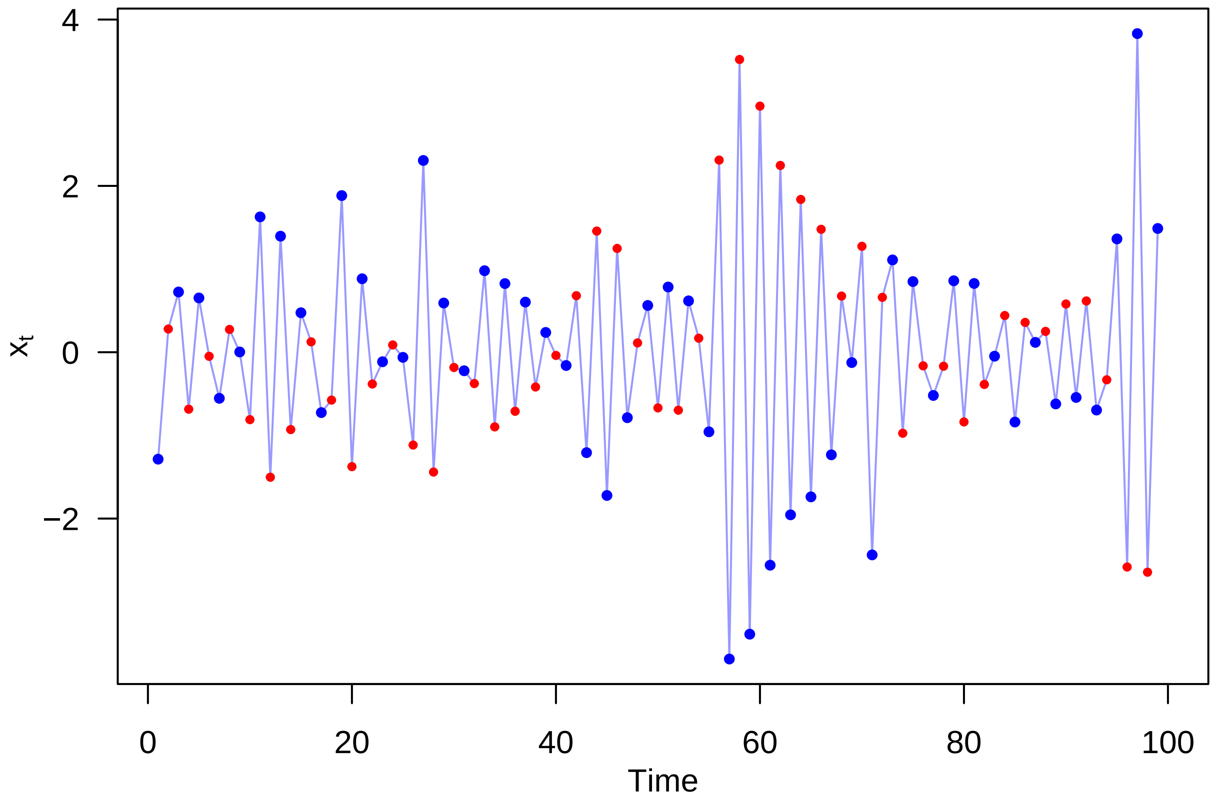


Fill in "missing" values

$\hat{X}_2 = \phi(X_1 + X_3)/(1 + \phi^2)$

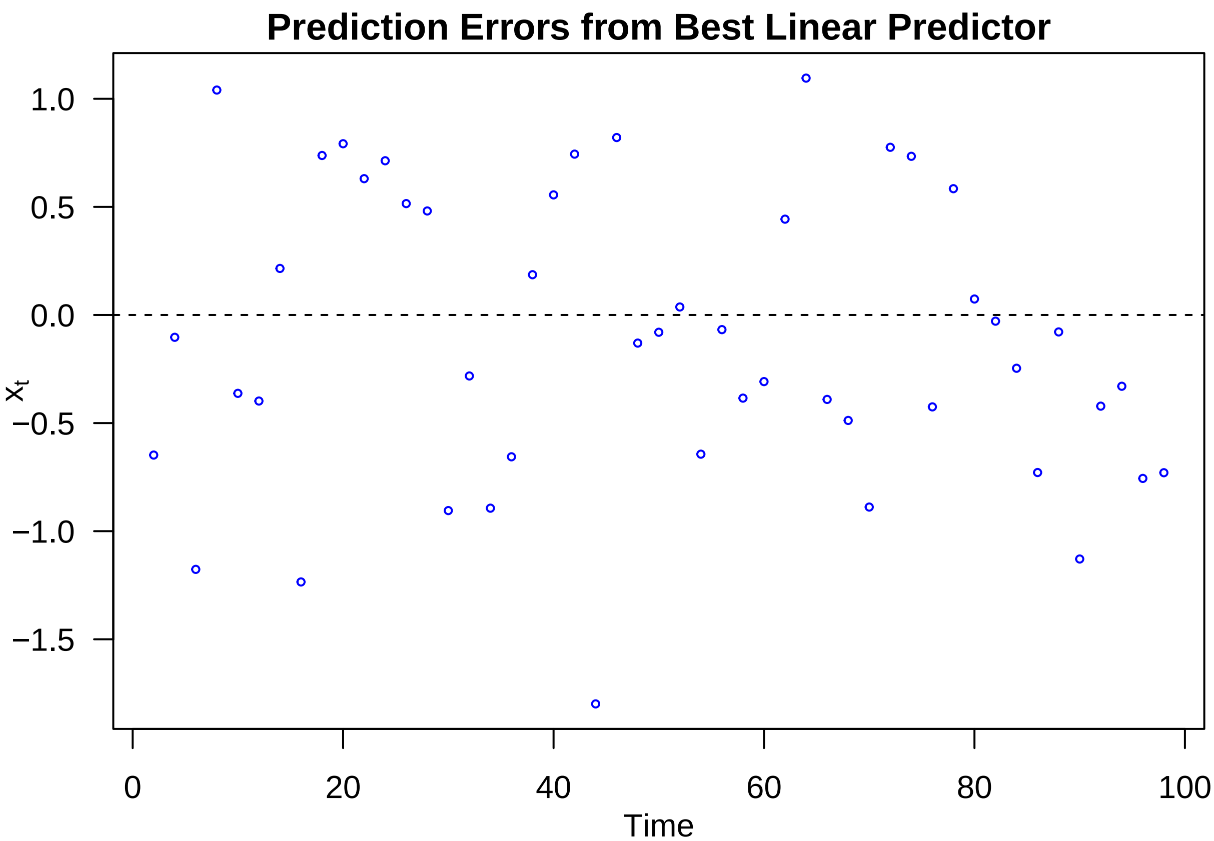
$\text{MSPE} = \frac{\sigma^2}{1+\phi^2}$

```r
ar1.ts.predicted <- ar1.ts
ar1.ts.predicted[seq(2, 98, 2)] <- -0.9 * (ar1.ts[seq(1, 97, 2)] + ar1.ts[seq(3, 99, 2)]) / 1.81
ar1.ts.predicted[100] <- NA


par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1.4, 0.6))
plot(ar1.ts.predicted, col = alpha("blue", 0.4), xlab = "Time", type = "l",
     ylab = expression(x[t]), cex = 0.5)
xs <- seq(2, 98, 2)
points(xs, ar1.ts.predicted[xs], pch = 19, col = "red", cex = 0.5)
xo <- seq(1, 99, 2)
points(xo, ar1.ts.subsampled[xo], col = "blue", cex = 0.75, pch = 16)
```



Prediction Errors from Best Linear Predictor

```
par(las = 1, mgp = c(2, 1, 0), mar = c(3.5, 3.5, 1.4, 0.6))
plot(xs, (ar1.ts.predicted - ar1.ts)[xs], col = "blue", xlab = "Time", type = "p",
     ylab = expression(x[t]), main = "Prediction Errors from Best Linear Predictor",
     cex=0.5)
abline(h = 0, lty = 2)
```



Prediction Errors from Best Linear Predictor
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_t
0
20
40
60
80
100
Time