A Comment on Forecast Uncertainty

April 20, 2020

In class and in homework assignments, we have often discussed constructing forecasts, and discussing the uncertainty of forecasts and future values.

Let's introduce some notation to make things clearer. Suppose we observe a time series $\boldsymbol{y} = (y_1, \dots, y_n)$, and we want to forecast $y_{n+1}, \dots, y_{n+h}$ for some $h > 0$. We assume a mean zero ARIMA(p, d, q) model with parameters, $\boldsymbol{\phi} = (\phi_1, \dots, \phi_p)$, $\boldsymbol{\theta} = (\theta_1, \dots, \theta_q)$, and σ^2_w. We define:

- The average value of the future observation y_{n+k}: $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2_w\right]$.
- The variance of observed future values about the average: $\mathbb{V}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2_w\right] = \mathbb{E}\left[\left(y_{n+k} - \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2_w\right]\right)^2\right]$.

When we use the `predict` function in `R`, the standard errors returned are the square roots of estimates $\mathbb{V}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2_w\right]$, obtained by plugging in the estimates of the ARIMA(p, d, q) parameters returned by `arima`, denoted by $\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}$, and $\hat{\sigma}^2_w$. A 95% interval based on $\mathbb{V}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2_w\right]$ is a prediction interval that treats the average value of the future observation y_{n+k} as known given estimates of the ARIMA(p, d, q) parameters and equal to $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2_w\right]$. It gives us an interval that would contain y_{n+k} simulated according to an ARIMA(p, d, q) process with parameters $hat\boldsymbol{\phi}, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2_w$ in 95% of simulations.

In practice, we may want to acknowledge that the estimates of the ARIMA(p, d, q) parameters are themselves estimates, and that the average value of the future observation y_{n+k} is not known. We denote the variance of our estimate of the average value of the future observation y_{n+k} as:

$$\mathbb{V}\left[\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2_w\right]\right] = \mathbb{E}\left[\left(\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2_w\right] - \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2_w\right]\right)^2\right]$$

.

To my knowledge, this variance is not returned by any `R` function directly. However, it can be approximated using a parametric bootstrap with B bootstrap samples. For each $b = 1, \dots, B$, we:

- Simulate $\boldsymbol{y}^{(b)} = \left(y_1^{(b))}, \dots, y_n^{(b))}\right)$
- Compute $\hat{\boldsymbol{\phi}}^{(b)} = \left(\hat{\phi}_1^{(b)}, \dots, \hat{\phi}_p^{(b)}\right)$, $\hat{\boldsymbol{\theta}}^{(b)} = \left(\hat{\theta}_1^{(b)}, \dots, \hat{\theta}_q^{(b)}\right)$, and $\hat{\sigma}^{2(b)}_w$ from an ARIMA(p, q) model for $\boldsymbol{y}^{(b)}$
- Compute $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}^{2(b)}_w\right]$

Unfortunately, the last step is tricky because, as far as I am aware, there are no `R` functions that allow you to compute forecasted values for an observed time series $\boldsymbol{y}$ using arbitrary ARIMA(p, d, q) parameter values. If we define the `R` object that stores the output from using the `arima` function to fit an ARIMA model to $\boldsymbol{y}^{(b)}$ as `arima.sim`, the predicted values that are returned by applying the `predict` function to `arima.sim` are $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}^{(b)}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}^{2(b)}_w\right]$, which is fundamentally a different quantity than the one we are interested in, $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}^{2(b)}_w\right]$. To obtain $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}^{2(b)}_w\right]$, we can use the ideas we discussed in the context of forecasting by hand. The technical details of doing this in a computationally efficient way go beyond what we learned when we discussed forecasting, and involve use of the innovations algorithm, which is similar to the Durbin-Levinson algorithm. For this reason, I have written a few functions that allow you to compute $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}^{2(b)}_w\right]$ in `R` easily. They are described at the end of this document.

Given a way of computing $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}_w^{2(b)}\right]$, we can approximate the variance $\mathbb{V}\left[\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]\right]$ with the bootstrap sample variance,

$$\widehat{\mathbb{V}}\left[\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]\right] = \frac{1}{B-1}\sum_{b=1}^{B}\left(\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}_w^{2(b)}\right] - \frac{1}{B}\sum_{b=1}^{B}\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}^{(b)}, \hat{\boldsymbol{\theta}}^{(b)}, \hat{\sigma}_w^{2(b)}\right]\right)^2.$$

A 95% interval based on $\mathbb{V}\left[\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]\right]$ is a confidence interval which refers to uncertainty about the the average value of the future observation y_{n+k}, $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]$.

In practice, we may want to combine both sources of uncertainty when we construct prediction intervals. We are interested in understanding how a single future observation y_{n+k} deviates from the estimated average value $\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]$. This can be written as:

$$\begin{aligned}
&\mathbb{E}\left[\left(y_{n+k} - \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]\right)^2\right] = \\
&\mathbb{E}\left[\left(y_{n+k} - \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma_w^2\right] + \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma_w^2\right] - \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]\right)^2\right] = \\
&\mathbb{V}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma_w^2\right] + \mathbb{V}\left[\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]\right] - \\
&2\mathbb{E}\left[\left(y_{n+k} - \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma_w^2\right]\right)\left(\mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{\theta}, \sigma_w^2\right] - \mathbb{E}\left[y_{n+k}|\boldsymbol{y}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}_w^2\right]\right)\right]
\end{aligned}$$

It is common to assume the last term is equal to zero in practice. Thus an approximate variance of future values about the estimated forecasts can be obtained, and a corresponding 95% prediction interval that accounts for both parameter estimation uncertainty and uncertainty about the future can be constructed.

A More Flexible Function for Computing Forecasts

First, as we know from learning about forecasting, computing forecasts requires computing the autocovariance function. Below are functions for computing the autocovariance function of an ARMA(p, q) process.

```r
# A first helper function for computing the autocovariance function
# of an ARMA(p, q) process
psi <- function(l, theta = NULL) {
  if (l < 0) {l <- abs(l)}
  q <- length(theta)
  if (!is.null(theta)) {
    theta <- c(1, theta)
  } else {
    theta <- c(1)
  }
  if (l <= q) {
    sum(theta[(l + 1):length(theta)]*theta[1:(length(theta)-l)])
  } else {
    0
  }
}

# A second helper function for computing the autocovariance function
# of an ARMA(p, q) process
```

```r
chi <- function(j, inv.root = NULL, phi = NULL) {
  if (is.null(inv.root) & !is.null(phi)) {
    inv.root <- 1/polyroot(c(1, -phi))
  }
  (inv.root[j]*prod(1 - inv.root*inv.root[j])*prod(inv.root[j] - inv.root[-j]))^(-1)
}

# Computes the autocovariance function of an ARMA(p, q) process
arma.acv <- function(lag.max = 10, theta = NULL,
                     phi = NULL, corr = TRUE, max.iter = Inf,
                     sig.sq = 1) {

  if (length(theta) == 1) {
    if (theta == 0) {
      theta <- NULL
    }
  }

  if (length(phi) > 1) {
    for (i in length(phi):2) {
      if (phi[i] == 0) {
        phi <- phi[1:(i - 1)]
      }
    }
  }

  if (length(phi) == 1) {
    if (phi == 0) {
      phi <- NULL
    }
  }




  q <- ifelse(!is.null(theta), length(theta), 0)
  p <- ifelse(!is.null(phi), length(phi), 0)

  if (!is.null(theta)) {
    ls <- seq(-q, q, by = 1)
    psis <- numeric(length(ls))
  } else {
    ls <- c(0)
    psis <- c(NA)
  }

  for (l in unique(abs(ls))) {
    psis[abs(ls) == l] <- psi(l = l, theta = theta)
  }


  if (!is.null(phi)) {
    inv.root <- 1/polyroot(c(1, -phi))
    chis <- numeric(p)
    for (j in 1:p) {
      chis[j] <- chi(j, inv.root = inv.root)
    }
    hgfs <- matrix(NA, nrow = (p + lag.max + q)*2 + 1, ncol = p)
    hs <- (-((p + lag.max + q))):(p + lag.max + q)
  }

  gam <- rep(0, 1 + lag.max)
  for (s in 0:lag.max) {
    for (l in ls) {
      if (length(ls) == 1) {
        l = 0
      }
      if (p > 0) {
        for (j in 1:p) {

            if (p + s - l > 0) {
              gam[s + 1] <- gam[s + 1] +
                chis[j]*psis[l == ls]*inv.root[j]^(p + s - l)
            } else {
              gam[s + 1] <- gam[s + 1] +
                chis[j]*psis[l == ls]*inv.root[j]^(2*p - (p + s - l))
            }

        }
      } else {
        gam[s + 1] <- ifelse(abs(s) <= q, psis[s == ls], 0)
      }
    }
  }
  if (!corr) {
    return(sig.sq*Re(gam))
  } else {
    return(list("rho" = Re(gam)/Re(gam[1]), "var" = Re(gam[1])))
  }
}
```

Next is a function that implements the innovations algorithm, which finds the innovation coefficients d_{ij} which minimize the expected squared forecast error for a future observation y_{m+1} of a mean-zero stationary process: $\mathbb{E}\left[\left(y_{m+1} - \sum_{j=1}^{m} d_{ij}\left(y_{m+1-j} - \hat{y}_{m+1-j}\right)\right)^2\right]$ for $i = 1, \dots, m$ and $j = 1, \dots, i$, where $\hat{y}_j = \sum_{k=1}^{j} d_{jk}\left(y_{j+1-k} - \hat{y}_{j+1-k}\right)$.

```
solve.innov <- function(n, phi = 0, theta = 0, sig.sq.w = 1) {
  D <- matrix(nrow = n, ncol = n)
  v <- rep(NA, n + 1)
  acv <- arma.acv(lag.max = n, phi = phi, theta = theta,
                  sig.sq = sig.sq.w, corr = FALSE)
  gamma.x.0 <- acv[1]
  v[1] <- gamma.x.0
  for (i in 1:n) {
    for (j in 0:(i-1)) {
      D[i, i - j] <- acv[abs(i - j) + 1]
```

```r
      if (j > 0) {
        for (k in 0:(j - 1)) {
          D[i, i - j] <- D[i, i - j] -
            D[j, j - k]*D[i, i - k]*v[k + 1]
        }
      }
      D[i, i - j] <- D[i, i - j]/v[j + 1]
    }
    v[i + 1] <- gamma.x.0 - sum(D[i, i:1]^2*v[1:i], na.rm = TRUE)
  }
  return(list("D"=D, "d.n" = D[nrow(D), ],
              "v" = v, "v.n" = v[length(v)]))
}
```

Last is a function that takes an observed time series $\boldsymbol{y}$, ARMA(p, q) parameter values, and a number of steps ahead and returns forecasts and variances of future values about the forecasts.

```r
get.preds <- function(y, phi, theta, sig.sq.w, h = 0) {
  innov <- solve.innov(n = length(y) - 1 + h,
                       phi = phi, theta = theta,
                       sig.sq.w = sig.sq.w)
  pred.vars <- pred.vals <- rep(NA, length(y) + h)
  pred.vals[1] <- 0
  pred.vars[1:length(y)] <- innov$v[1:length(y)]
  for (i in 2:(length(y))) {
    pred.vals[i] <- sum(innov$D[i - 1, 1:(i - 1)]*(y[(i-1):1] - pred.vals[(i - 1):1]))
  }
  if (h > 0) {
    for (i in 1:h) {
      pred.vals[length(y) + i] <- sum(innov$D[length(y) + i - 1,
                                              (1 + i - 1):(length(y) + i - 1)]*
                                        (y[length(y):1] - pred.vals[length(y):1]))
      pred.vars[length(y) + i] <- innov$v[1] -
        sum(innov$D[length(y) + i - 1,
                    (1 + i - 1):(length(y) + i - 1)]^2*(innov$v[length(y):1]))
    }
  }

  return(list("preds" = pred.vals, "ses" = sqrt(pred.vars)))
}
```