This method then appears as a straightforward generalization of the Accept–Reject method in the sense that the instrumental distribution is the same density g as in the Accept–Reject method. Thus, the proposed values Y_t are the same, if not the accepted ones.

Metropolis–Hastings algorithms and Accept–Reject methods (Section 2.3), both being generic simulation methods, have similarities between them that allow comparison, even though it is rather rare to consider using a Metropolis–Hastings solution when an Accept–Reject algorithm is available. In particular, consider that

a. The Accept–Reject sample is iid, while the Metropolis–Hastings sample is not. Although the Y_t's are generated independently, the resulting sample is not iid, if only because the probability of acceptance of Y_t depends on $X^{(t)}$ (except in the trivial case when $f = g$).
b. The Metropolis–Hastings sample will involve repeated occurrences of the same value since rejection of Y_t leads to repetition of $X^{(t)}$ at time $t+1$. This will have an impact on tests like `ks.test` that do not accept ties.
c. The Accept–Reject acceptance step requires the calculation of the upper bound $M \geq \sup_x f(x)/g(x)$, which is not required by the Metropolis–Hastings algorithm. This is an appealing feature of Metropolis–Hastings if computing M is time-consuming or if the existing M is inaccurate and thus induces a waste of simulations.

Exercise 6.3 Compute the acceptance probability $\rho(x,y)$ in the case $q(y|x) = g(y)$. Deduce that, for a given value $x^{(t)}$, the Metropolis–Hastings algorithm associated with the same pair (f,g) as an Accept–Reject algorithm accepts the proposed value Y_t more often than the Accept–Reject algorithm.

The following exercise gives a first comparison of Metropolis–Hastings with an Accept–Reject algorithm already used in Exercise 2.20 when both algorithms are based on the same candidate.

Exercise 6.4 Consider the target as the $\mathcal{G}(\alpha,\beta)$ distribution and the candidate as the gamma $\mathcal{G}([\alpha],b)$ distribution (where $[a]$ denotes the integer part of a).

a. Derive the corresponding Accept–Reject method and show that, when $\beta = 1$, the optimal choice of b is $b = [\alpha]/\alpha$.
b. Generate 5000 $\mathcal{G}(4, 4/4.85)$ random variables to derive a $\mathcal{G}(4.85,1)$ sample (note that you will get less than 5000 random variables).
c. Use the same sample in the corresponding Metropolis–Hastings algorithm to generate 5000 $\mathcal{G}(4.85,1)$ random variables.
d. Compare the algorithms using (i) their acceptance rates and (ii) the estimates of the mean and variance of the $\mathcal{G}(4.85,1)$ along with their errors. (*Hint:* Examine the correlation in both samples.)

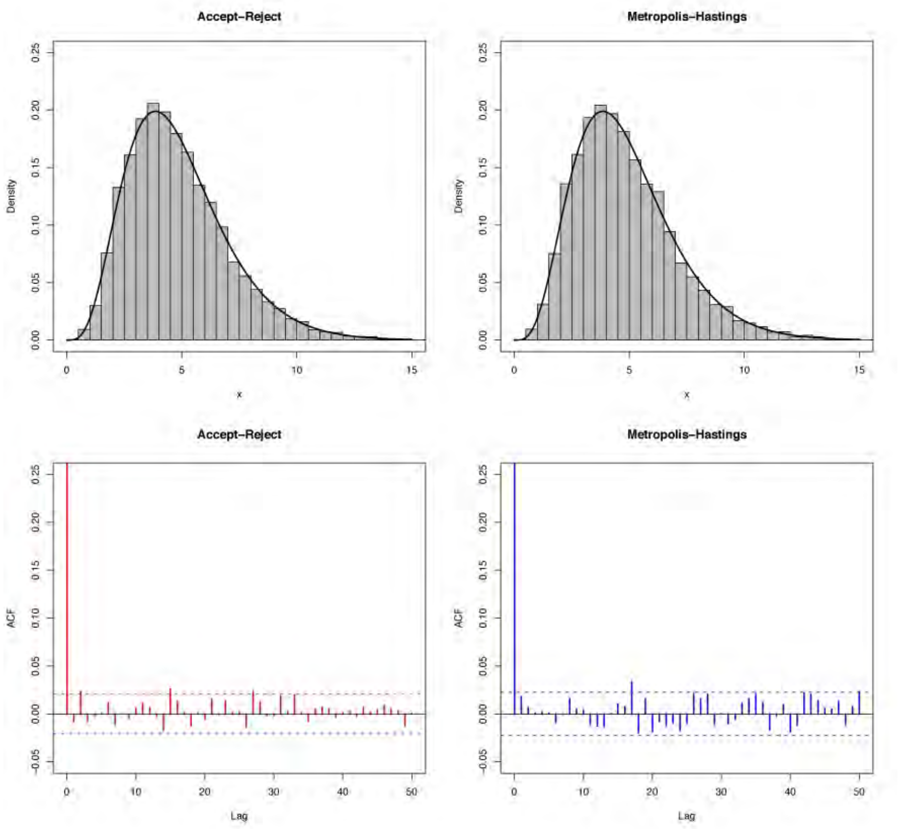

Fig. 6.3. Histograms and autocovariance functions from a gamma Accept–Reject algorithm (left panels) and a gamma Metropolis–Hastings algorithm (right panels). The target is a $\mathcal{G}(4.85, 1)$ distribution and the candidate is a $\mathcal{G}(4, 4/4.85)$ distribution. The autocovariance function is calculated with the `R` function `acf`.

Figure 6.3 illustrates Exercise 6.4 by comparing both Accept–Reject and Metropolis–Hastings samples. In this setting, operationally, the independent Metropolis–Hastings algorithm performs very similarly to the Accept–Reject algorithm, which in fact generates perfect and independent random variables.

Theoretically, it is also feasible to use a pair (f, g) such that a bound M on f/g does not exist and thus to use Metropolis–Hastings when Accept–Reject is not possible. However, as detailed in Robert and Casella (2004) and illustrated in the following formal example, the performance of the Metropolis–Hastings algorithm is then very poor, while it is very strong as long as $\sup f/g = M < \infty$.