Time-domain Astronomy

Tutorials

Tiago Campante

1 A sample R session

R is a public-domain statistical software environment used by researchers in many scientific fields. R can be downloaded from `http://www.r-project.org/`. The installation includes help files and some user manuals. Furthermore, the Comprehensive R Archive Network (CRAN) contains a plethora of add-on packages that can be installed on-the-fly within an R session. It also includes links to online tutorials.

Here, we analyze a hypothetical data set at the same time as we illustrate some of the uses of elementary R functions. We start by obtaining some basic information about the session environment. The working directory can be changed using `setwd`. We create a simple vector using the `c` function to concatenate values. This new vector is then displayed in several ways.

```
# I. Query the session environment and make a vector of three numbers

getwd()
setwd("/Users/myself/newdir")
sessionInfo() ;  citation()
a.test <- c(33, 44, 55)
ls()  ;  class(a.test) ;  str(a.test)
a.test ;  write(file="output", a.test)
```

We now create a set of 500 data pairs with a given nonlinear relationship and heteroscedastic errors (i.e., the scatter depends on the independent variable). Several useful R functions are used: the `seq` function produces a regular sequence of values between specified limits; the `sample` function draws random values from a data set; and the `rnorm` function returns normal random deviates. We fix the random number sequence to allow the calculation to be repeated.

```
# II. Create a 500x2 bivariate data set where the X values are evenly
# distributed between 0 and 3 and the Y values have a nonlinear
```

```
# dependence on X with heteroscedastic Gaussian scatter

help(seq) ; help(sample)  ; help(rnorm)
set.seed(1)
x <- sample(seq(0.01, 3, length.out=500))
y <- 0.5*x + 0.3^(x^2) + rnorm(500, mean=0, sd=(0.05*(1+x^2)))
```

We go on to present operations commonly known as exploratory data analysis, where we consider the distributions of the X and Y variables separately. The `summary` function presents the minimum, maximum, quartiles, median, and mean of a distribution directly on the console. We save this information using `write.table`. The univariate summary information is then plotted using the `boxplot` function. The box-and-whisker plot is a visualization of Tukey's five-number summary of a distribution. Though rarely used in astronomy, it is a compact and visually effective display of robust measures of location and spread, appropriate for both small and moderately large univariate data sets. The X distribution is symmetrical (by construction), while the Y distribution is skewed with outliers. We also plot the univariate empirical distribution function (edf) for each variable using `ecdf`. We produce PostScript figures using the `dev.copy2eps` function.

```
# III. Examine, summarize and produce box-and-whisker plots
# of univariate distributions

summary(x) ; summary(y)
str(summary(x))
write.table(rbind(summary(x),summary(y)),file="summary.txt")

help(par) ; help(boxplot)
par(mfrow=c(1,2))  # set up two-panel figure
boxplot(x,  notch=T, main="Boxplot for X")
boxplot(y,  notch=T, pch=20, cex=0.5, main="Boxplot for Y")
par(mfrow=c(1,1))
dev.copy2eps(file="box.eps")

plot(ecdf(x),pch=20,cex=0.3,main="",ylab="EDF",xlab="")
plot(ecdf(y),pch=20,cex=0.3,add=T)
text(2.0,0.5,"X") ; text(1.4,0.8,"Y")
dev.copy2eps(file="edf.eps")
```

The two variables are from now on treated as a multivariate data set. We bind the vectors by columns into a matrix with `cbind` and coerce the result to a data frame with `as.data.frame`. The data frame is made readily available for further analysis using the `attach` function.

```
# IV. Put X and Y into a data frame
```

```
help(cbind) ; help(as.data.frame) ; help(names)
xy <- cbind(x, y)  ;  str(xy)
xy <- as.data.frame(xy)
names(xy) <- c("Xvar", "Yvar")  ; str(xy)
attach(xy)  ;  ls()
```

We use `plot` to produce standard bivariate scatter plots of the variables in their original format and after logarithmic transformation. The `cor.test` function gives parametric and nonparametric correlation coefficients with associated p-values.

```
# V. Bivariate plots and correlation tests

help(plot)
par(mfrow=c(2,2))
plot(xy, pch=20, cex=0.5)
plot(log10(xy), pch=20, cex=0.5, xlab="log(Xvar)", ylab="log(Yvar)")

length(x[x>2.5])
cor.test(x[x>2.5],y[x>2.5], method="pearson")
cor.test(x[x>2.5],y[x>2.5], method="kendall")
```

We use the function `smoothScatter` to produce a two-dimensional kernel density estimator. We superpose a cubic B-spline fit to the data with `smooth.spline`.

```
# VI. Kernel density estimator with spline fit

smoothScatter(log10(xy), lwd=4, pch=20, cex=0.2, xlab="log(Xvar)",
    ylab="log(Yvar)", colramp = colorRampPalette(c("white",gray(20:5/20))))
lines(smooth.spline(log10(xy)), lwd=3)
```

We use the function `lm` to fit a fifth-order polynomial function by unweighted least squares to the logarithmic bivariate distribution.

```
# VII. Least-squares polynomial regression

logx <- log10(x) ; logx2 <- (log10(x))^2 ; logx3 <- (log10(x))^3
logx4 <- (log10(x))^4 ; logx5 <- (log10(x))^5
yfit <- lm(log10(y) ~ logx + logx2 + logx3 + logx4 + logx5)
str(yfit)
plot(log10(x), log10(y), pch=20, col=grey(0.5), cex=0.3,
    xlab="log(Xvar)", ylab="log(Yvar)")
lines(sort(log10(x)), yfit$fit[order(log10(x))], lwd=3)
```

```
dev.copy2eps(file="smooth.eps")
par(mfrow=c(1,1))
```

2 Time series primer with R

An introduction on using R for time series analysis is given here. Visit the following link: `https://github.com/nickpoison/astsa/blob/master/fun_with_astsa/fun_with_astsa.md`. It contains a useful tutorial that will guide you through several important steps in the analysis of (real and simulated) time series using R. You will need to have the `astsa` package installed and loaded before you start.

None of the real data sets you will be analyzing as part of this tutorial are astrophysical. In fact, they concern topics as diverse as quarterly earnings per share, weekly cardiovascular mortality, or global warming. This serves as a reminder that the concepts and tools employed during this course are truly interdisciplinary and widely used.

3 R application: the GX 5-1 X-ray binary-star system

We use the variability of Galactic X-ray source GX 5-1 to illustrate many of the time series capabilities of R. GX 5-1 is an X-ray binary-star system with gas from a normal companion accreting onto a neutron star. Highly variable X-rays are produced in the inner accretion disk, often showing stochastic red noise and quasi-periodic oscillations of uncertain origin.

The data set consists of 65 536 measurements of photon counts in evenly spaced 1/128-second bins obtained with the Japanese *Ginga* satellite during the 1980s. Although strictly a Poisson process, the count rates are sufficiently high that they can be considered to be normally distributed.

```
# I. Read in GX 5-1 data and create time series

GX.dat <- scan("https://www.dropbox.com/s/ugd0kgq9f2t7eqc/GX.dat?dl=0")
GX.time <- seq(from=0, to=512, length.out=65536)
GX.ts <-  ts(GX.dat, GX.time) ; GX.ts.offset <- ts(GX.dat-30, GX.time)
```

Exploratory data analysis. We start with the visual exploration of the data set in a variety of displays. The histogram of counts in each bin shows that the standard deviation is 24 % larger than expected for a white noise process, and that asymmetry about the mean is present.

```
# II. Compare histogram of counts to normal distribution

hist(GX.dat, breaks=100, xlim=c(40,100), ylim=c(0,3500), xlab="GX 5-1 counts",
    font=2, font.lab=2, main="")
curve(dnorm(x,mean=mean(GX.dat), sd=sqrt(mean(GX.dat)))*65536, lwd=3, add=T)
sd(GX.dat) / sqrt(mean(GX.dat))  # result is 1.24
```

Examination of the time series in its raw form and after various smoothing operations does not reveal obvious nonstationary structure to account for the extra variance, although some of the smoothers give a hint of autocorrelated variations with a characteristic timescale around 20–50 seconds. GX 5-1 thus appears to exhibit stochastic, possibly autocorrelated and quasi-periodic, variability that can now be investigated in detail.

```
# III. Examine raw and smoothed time series

plot.ts(GX.ts[1:6000], ylab="GX 5-1 counts", xlab="Time (1/128 sec)",
    cex.lab=1.3, cex.axis=1.3)

plot(GX.time,GX.dat, ylim=c(-10,115), xlab="Time (sec)", ylab="GX 5-1 counts",
    cex.lab=1.3, cex.axis=1.3, type="n")
lines(ksmooth(GX.time, GX.dat+30, "normal", bandwidth=7), lwd=2)
text(450, 110, "Normal kernel")
lines(filter(GX.ts, sides=2, rep(1,7)/7), lwd=2)
text(450, 85, "Moving average")
lines(kernapply(GX.ts.offset, kernel("modified.daniell", 7)), lwd=2)
text(450, 50, "Modified Daniell")
lines(supsmu(GX.time, GX.dat-60), lwd=2)
text(400, 20, "Friedman's super smoother")
lines(lowess(GX.time, GX.dat-80, 0.05), lwd=2)
text(450, 0, "Local regression")
```

Spectral analysis. We start with a manual construction of the Schuster periodogram based on the script provided by Shumway and Stoffer (2011), before using the automated function `spec.pgram`. Note that entries in the raw periodogram can be very uncertain (e.g., the 500th spectral value is 119 with 95 % confidence interval $[32, 4702]$ assuming an asymptotic χ^2_2 distribution).

```
# IV. Raw periodogram

f <- 0:32768/65536
I <- (4/65536) * abs(fft(GX.ts) / sqrt(65536))^2
plot(f[2:60000], I[2:60000], type="l", ylab="Power", xlab="Frequency")

Pergram <- spec.pgram(GX.ts,log="no",main="")
summary(Pergram)
Pergram$spec[500] # value of 500th point
2*Pergram$spec[500] / qchisq(c(0.025,0.975),2)
```

We then proceed to experiment with various smoothers to reduce the variance of the raw periodogram. Smoothing has a dramatic effect on the periodogram appearance and much is learned

about the excess variance of the time series. It appears to arise from two distinct processes: a noise component below $\sim$0.05 rising toward lower frequencies and a strong but broadened spectral peak around 0.17–0.23. These are the red noise and quasi-periodic oscillations addressed in many studies of accretion binary-star systems like GX 5-1.

```
# V. Raw and smoothed periodogram

par(mfrow=c(3,1))
spec.pgram(GX.ts, log="no", main="", sub="")
spec.pgram(GX.ts, spans=50, log="no", main="", sub="")
spec.pgram(GX.ts, spans=500, log="no", main="", sub="")
```

Modeling as an autoregressive process. The red noise and quasi-periodic structure in the GX 5-1 data are clearly seen in plots of the autocorrelation (ACF) and partial autocorrelation (PACF) functions. The PACF is more informative: the strongest predictor of a current value are the values separated by 4–6 time increments, while oscillations have periods around 5–6 increments. The envelope of significant autocorrelation extends to lags of 20–30 increments.

```
# VI. Autocorrelation functions of the GX 5-1 time series

par(mfrow=c(1,2))
acf(GX.ts, 40, main="", ci.col="black", ylim=c(-0.05,0.3), lwd=2)
pacf(GX.ts, 40, main="", ci.col="black", ylim=c(-0.05,0.3), lwd=2)
```

The presence of autocorrelation and (from the smoothed time series) absence of changes in the average level motivates an AR(p) model. The `ar` function selects a model using the minimum Akaike information criterion (AIC). Several computational options are provided and we chose ordinary least squares. AR(27) is the best model, although any model with order $p>20$ is adequate.

```
# VII. Autoregressive modeling

ARmod <- ar(GX.ts, method="ols")
ARmod$order # model selection based on AIC
ARmod$ar # best-fit parameter values
ARmod$asy.se.coef$ar # parameter standard errors
par(mfrow=c(1,1))
plot(0:29, log10(ARmod$aic[1:30]), xlab="AR model parameters",
    ylab="log(AIC)", pch=20)
arrows(27, 0.4, 27, 0.0, length=0.1)
```

Since the AR model is stochastic, we cannot meaningfully compare its time series to the observed GX 5-1 data. However, the Fourier spectrum of the model can be compared to the GX 5-1 spectrum.

The function `spec.ar` runs `ar`, chooses the model with the minimum AIC, and computes the Fourier transform of the model. The spectral properties of GX 5-1 are remarkably well reproduced by the model: the steep red noise signal at frequencies below 0.05, the shallow red noise around 0.05–0.1, the height and width of the peak around 0.19, the slight asymmetry around the peak, the weak harmonic around 0.38, and the spectral noise level at high frequencies.

```
# VIII. Spectrum of AR model

ARspec <- spec.ar(GX.ts, plot=F)
GXspec <- spec.pgram(GX.ts, span=101, main="", sub="", lwd=2)
lines(ARspec$freq, ARspec$spec, col="red", lwd=4)
legend(0.23, 550, c("Periodogram, Daniell smooth", "AR(27) model"),
    lty=c(1,1), lwd=c(2,4), col=c("black","red"))
```

Modeling as a long-memory process. The only spectral feature not accounted for by the AR model is the excess signal at the very lowest frequency. This indicates that the GX 5-1 behavior has an additional long-term memory component.

Several estimators of the long-memory parameters d and H are provided by CRAN packages based both on frequency- and time-domain techniques. Recall that $d = \alpha/2$ and Hurst's self-similarity parameter is given by $H = d + 1/2$, where the spectral power is $P \propto (1/f)^{\alpha}$.

The function `fdGPH` calculates the Geweke–Porter-Hudak estimator from linear regression on the raw log-periodogram. It gives $d = 0.10 \pm 0.04$. The function `fdSperio` uses Reisen's regression on the log-periodogram smoothed with a lag Parzen window and obtains the same value but with better precision, $d = 0.10 \pm 0.01$. These values indicate a relatively weak long-memory process with noise power scaling roughly as $(1/f)^{0.2}$. In the time domain, a FARIMA model calculation using `fracdiff` gives a best-fit value of $d = 0.06$.

```
# IX. Estimates of the long-memory parameter d

install.packages("fracdiff") ; library(fracdiff)
d.FARIMA <- fracdiff(GX.ts, nar=27, nma=1, ar=ARmod$ar) ; d.FARIMA$d
d.GPH <- fdGPH(GX.ts) ; d.GPH$d ; d.GPH$sd.as
d.Reisen <- fdSperio(GX.ts) ; d.Reisen$d ; d.Reisen$sd.as
```

Several functions exist that compute H from the time-domain data. These results are noticeably inconsistent. We obtain $H = 0.12$ from an arc-hyperbolic-sine transformation of the autocorrelation function (`hurstACVF`) and 0.66–1.00 from various blockings of the time series (`hurstBlock`). Clearly, it is difficult to establish a consistent value for the $(1/f)^{\alpha}$-noise spectral shape. The likely cause of these discrepant results is that the red noise in GX 5-1 is not truly fractal, such as an $(1/f)^{\alpha}$ process where the scaling of the variance follows a power law.

```
# X. Estimates of the long-memory parameter H
```

```
install.packages("fractal") ; library(fractal)
H.ACVF <- hurstACVF(GX.ts) ; H.ACVF
H.block <- hurstBlock(GX.ts)  ; H.block
```

Wavelet analysis. While examining the GX 5-1 time series, we did not see any short-lived structures such as rapid increases or decreases in emission. Nonetheless, it is valuable to make a wavelet decomposition of the time series to assist in visualization of the data set at different timescales. Here we use CRAN's package `waveslim` to construct the discrete wavelet transform using Mallat's pyramid algorithm and Daubechies's orthonormal basis functions at ten different temporal scales. As expected, no obvious structure is seen.

```
# XI. Discrete wavelet transform

install.packages("waveslim") ; library(waveslim)
GX.wav <-  dwt(GX.ts,n.levels=10)
par(mfrow=c(3,1))
plot.ts(up.sample(GX.wav[[4]],2^4),type="h",axes=T,
    xlab="Time (1/128 sec)",ylab="")
abline(h=0)
plot.ts(up.sample(GX.wav[[7]],2^7),type="h",axes=T,
    xlab="Time (1/128 sec)",ylab="",lwd=2)
abline(h=0)
plot.ts(up.sample(GX.wav[[10]],2^{10}),type="h",axes=T,
    xlab="Time (1/128 sec)",ylab="",lwd=2)
abline(h=0)
par(mfrow=c(1,1))
```

4 R application: programming your own MCMC algorithm

Data and goal of the analysis. We will use a data set from the *Chandra* Orion Ultradeep Project (COUP). This is a time series of X-ray emission from a flaring young star in the Orion Nebula cluster. The raw data, which approximately obey a Poisson process, give the individual photon arrival times (in seconds) and their energies (in keV). The processed data considered here are obtained by grouping the events into evenly spaced 10 000-second bins.

We assume that the Poisson process is piecewise homogeneous with a single change-point. [Question: What is a change-point?] Our goal will be to identify the change-point and to estimate the intensities of the Poisson process before and after the change-point. We first read in the data. Note that this is just a convenient subset of the actual data set.

```
chptdat = read.table("https://www.dropbox.com/s/qbwiwknlsxdrhq2/
   COUP551_rates.dat?dl=0",skip=1)
```

Next we plot the time-series data. This plot suggests that the change-point occurs around bin 10.

```
Y=chptdat[,2] # store data in Y
ts.plot(Y,xlab="Bin number",main="Plot of time-series data")
```

Model description. A Bayesian change-point model with Gamma hyperpriors is adopted from the book by Carlin and Louis (2000). [Question: What is a hyperprior?] Let Y_i be the number of occurrences of some event during time interval i. The process is observed for n time intervals and we assume that there is a change-point at some intermediate time interval k (i.e., after time interval k the number of counts is significantly altered). Consider the following hierarchical change-point model:

$$\begin{aligned} Y_i|\theta, k &\sim \text{Poisson}(\theta) \quad \text{for } i = 1, \ldots, k\,; \\ Y_i|\lambda, k &\sim \text{Poisson}(\lambda) \quad \text{for } i = k+1, \ldots, n\,. \end{aligned} \tag{1}$$

We assume that the Y_i are mutually independent conditional on the parameters. [Question: What is conditional independence?] Also, consider the following prior distributions for the model parameters:

$$\begin{aligned} \theta|b_1 &\sim \text{Gamma}(0.5, b_1)\,; \\ \lambda|b_2 &\sim \text{Gamma}(0.5, b_2)\,; \\ b_1 &\sim \text{InvGamma}(a \to 0, 1)\,; \\ b_2 &\sim \text{InvGamma}(a \to 0, 1)\,; \\ k &\sim \text{UniformInt}(1, \ldots, n)\,. \end{aligned} \tag{2}$$

Here, we are assuming that θ, λ, and k are conditionally independent, and that b_1 and b_2 are independent.

Inference for this model is based on the five-dimensional posterior distribution $f(k, \theta, \lambda, b_1, b_2|\mathbf{Y})$, where $\mathbf{Y} = (Y_1, \ldots, Y_n)$. The posterior distribution is obtained up to a multiplicative constant by taking the product of the likelihood (i.e., the density of $\mathbf{Y}$ given the parameters) and the joint prior of the parameters. [Question: Why do we neglect the aforementioned multiplicative constant?] This gives

$$\begin{aligned} f(k, \theta, \lambda, b_1, b_2|\mathbf{Y}) &\propto \prod_{i=1}^{k} f_1(Y_i|\theta, k) \prod_{i=k+1}^{n} f_2(Y_i|\lambda, k) \\ &\quad \times g_1(\theta|b_1)\, g_2(\lambda|b_2)\, h_1(b_1)\, h_2(b_2)\, u(k) \\ &= \prod_{i=1}^{k} \frac{\theta^{Y_i} e^{-\theta}}{Y_i!} \prod_{i=k+1}^{n} \frac{\lambda^{Y_i} e^{-\lambda}}{Y_i!} \\ &\quad \times \frac{1}{\Gamma(0.5)\, b_1^{0.5}} \theta^{-0.5} e^{-\theta/b_1} \times \frac{1}{\Gamma(0.5)\, b_2^{0.5}} \lambda^{-0.5} e^{-\lambda/b_2} \\ &\quad \times \frac{e^{-1/b_1}}{\Gamma(a)\, b_1} \times \frac{e^{-1/b_2}}{\Gamma(a)\, b_2} \times \frac{1}{n}\,. \end{aligned} \tag{3}$$

We obtain full conditional distributions for each model parameter by ignoring all terms that are constant with respect to that parameter. Occasionally, these full conditional distributions are well-known distributions:

- Full conditional for θ:

$$\begin{aligned} f(\theta|k,\lambda,b_1,b_2,\mathbf{Y}) &\propto \prod_{i=1}^{k}\frac{\theta^{Y_i}e^{-\theta}}{Y_i!}\times\frac{1}{\Gamma(0.5)\,b_1^{0.5}}\theta^{-0.5}e^{-\theta/b_1} \\ &\propto \theta^{\sum_{i=1}^{k}Y_i-0.5}e^{-\theta(k+1/b_1)} \\ &\sim \text{Gamma}\left(\sum_{i=1}^{k}Y_i+0.5,\frac{b_1}{kb_1+1}\right). \end{aligned} \tag{4}$$

- Full conditional for λ:

$$\begin{aligned} f(\lambda|k,\theta,b_1,b_2,\mathbf{Y}) &\propto \prod_{i=k+1}^{n}\frac{\lambda^{Y_i}e^{-\lambda}}{Y_i!}\times\frac{1}{\Gamma(0.5)\,b_2^{0.5}}\lambda^{-0.5}e^{-\lambda/b_2} \\ &\sim \text{Gamma}\left(\sum_{i=k+1}^{n}Y_i+0.5,\frac{b_2}{(n-k)b_2+1}\right). \end{aligned} \tag{5}$$

- Full conditional for k:

$$\begin{aligned} f(k|\theta,\lambda,b_1,b_2,\mathbf{Y}) &\propto \prod_{i=1}^{k}\frac{\theta^{Y_i}e^{-\theta}}{Y_i!}\prod_{i=k+1}^{n}\frac{\lambda^{Y_i}e^{-\lambda}}{Y_i!} \\ &\propto \theta^{\sum_{i=1}^{k}Y_i}\lambda^{\sum_{i=k+1}^{n}Y_i}e^{-k\theta-(n-k)\lambda}. \end{aligned} \tag{6}$$

- Full conditional for b_1:

$$\begin{aligned} f(b_1|k,\theta,\lambda,b_2,\mathbf{Y}) &\propto \frac{1}{b_1^{0.5}}e^{-\theta/b_1}\times\frac{e^{-1/b_1}}{b_1} \quad\propto\quad b_1^{-1.5}e^{-(1+\theta)/b_1} \\ &\sim \text{InvGamma}(0.5,\theta+1). \end{aligned} \tag{7}$$

- Full conditional for b_2:

$$\begin{aligned} f(b_2|k,\theta,\lambda,b_1,\mathbf{Y}) &\propto \frac{1}{b_2^{0.5}}e^{-\lambda/b_2}\times\frac{e^{-1/b_2}}{b_2} \quad\propto\quad b_2^{-1.5}e^{-(1+\lambda)/b_2} \\ &\sim \text{InvGamma}(0.5,\lambda+1). \end{aligned} \tag{8}$$

Except for k, all other model parameters have standard full conditional distributions. Therefore, we can perform Gibbs updates on them, whereby samples are drawn from their full conditionals. The full conditional for k is not a standard distribution and so we need to use the more general Metropolis–Hastings update instead of a Gibbs update. The inverse-gamma distribution is said to be a conjugate prior for b_1 and b_2 in the present case, since it results in a posterior that is also an inverse-gamma distribution. This distribution has poorly behaved moments. For that reason, it may be better to adopt another prior density (such as the gamma distribution).

Setting up and running the MCMC algorithm. We use the Metropolis–Hastings algorithm to draw multiple samples from the posterior distribution (Eqs. 3 to 8). It works as follows:

1. Choose a starting value for the Markov chain, say, $(\theta_0, \lambda_0, k_0, b_{1,0}, b_{2,0}) = (1, 1, \text{floor}(n/2), 1, 1)$.

2. Update each parameter in turn at the j-th iteration, $j = 1, \ldots, N$:

   (a) Gibbs update of θ: Sample θ_j from the gamma distribution in Eq. (4) using the most up-to-date values of k and b_1.

   (b) Gibbs update of λ: Sample λ_j from the gamma distribution in Eq. (5) using the most up-to-date values of k and b_2.

   (c) Gibbs update of b_1: Sample $b_{1,j}$ from the inverse-gamma distribution in Eq. (7) using the most up-to-date value of θ.

   (d) Gibbs update of b_2: Sample $b_{2,j}$ from the inverse-gamma distribution in Eq. (8) using the most up-to-date value of λ.

   (e) Metropolis–Hastings update of k:

      i. Propose a new sample for k, k', according to a proposal distribution, $q(k|\theta, \lambda, b_1, b_2, \mathbf{Y})$. In the present example, we choose

$$q(k|\theta, \lambda, b_1, b_2, \mathbf{Y}) \sim \text{UniformInt}(2, \ldots, n-1)\,. \tag{9}$$

      ii. Compute the acceptance probability, $\alpha(k, k')$, using the most up-to-date values of k, θ, λ, b_1, and b_2:

$$\alpha(k, k') = \min\left[1, \frac{f(k'|\theta, \lambda, b_1, b_2, \mathbf{Y})}{f(k|\theta, \lambda, b_1, b_2, \mathbf{Y})} \frac{q(k|\theta, \lambda, b_1, b_2, \mathbf{Y})}{q(k'|\theta, \lambda, b_1, b_2, \mathbf{Y})}\right]. \tag{10}$$

      iii. Accept the proposed sample with probability $\alpha(k, k')$. If accepted then set $k_j = k'$, otherwise keep $k_j = k_{j-1}$.

   (f) You have now steered the Markov chain toward the new state $(\theta_j, \lambda_j, k_j, b_{1,j}, b_{2,j})$.

A file containing the R source code needed to run an MCMC algorithm for this particular application has been made available at `https://www.dropbox.com/s/87bx5ixy4eqs5jf/MCMCchpt.R?dl=0`. Save this file to your R working directory. Let us start by loading the source code.

```
source("MCMCchpt.R")
```

We can now run the MCMC algorithm.

```
mchain <- mhsampler(dat=Y, MCMCiterations=5000)
```

Congratulations! You have just completed your first ever MCMC run.

MCMC output analysis. We will now examine the output produced by the sampler and estimate quantities of interest. For instance, in order to estimate the expectation of a marginal distribution for a particular model parameter, we simply average all draws for that parameter.

```
mean(mchain[1,]) # obtain mean of first row (theta)
```

We may instead want to obtain estimates of the mean and median for all parameters at once.

```
apply(mchain,1,mean) # compute mean for all parameters at once
apply(mchain,1,median) # compute median for all parameters at once
```

Let us have a look at the marginal posterior distributions.

```
par(mfrow=c(3,2))
plot(density(mchain[1,]),main="smoothed density plot for theta posterior")
plot(density(mchain[2,]),main="smoothed density plot for lambda posterior")
plot(density(mchain[3,]),main="smoothed density plot for k posterior")
plot(density(log10(mchain[4,])),main="smoothed density plot for log10(b1) posterior")
plot(density(log10(mchain[5,])),main="smoothed density plot for log10(b2) posterior")
par(mfrow=c(1,1))
```

We compute the (posterior) probability that λ is greater than 10.

```
sum(mchain[2,]>10)/length(mchain[2,])
```

We can also examine how the estimate of the expectation of the marginal distribution for k changes with each iteration.

```
install.packages("batchmeans")
library(batchmeans)
estvssamp(mchain[3,])
```

We would like to assess whether or not the Markov chain is moving around quickly enough through parameter space so as to produce good parameter estimates. We call this property mixing. A way of doing this would be to inspect the sample autocorrelations.

```
par(mfrow=c(3,2))
acf(mchain[1,],main="acf plot for theta")
acf(mchain[2,],main="acf plot for lambda")
acf(mchain[3,],main="acf plot for k")
acf(mchain[4,],main="acf plot for b1")
acf(mchain[5,],main="acf plot for b2")
par(mfrow=c(1,1))
```

In case the samples are heavily autocorrelated, we should rethink our sampling scheme or, at the very least, substantially increase the number of iterations. Note that the sample autocorrelations are not negligible for θ, λ, and k. This can be easily overcome in the present application by simply running the chain for a longer period of time, since the sampler is fast. However, when facing complex high-dimensional problems, improving the sampler mixing can be critical. The acceptance rate for k proposals is seen to be smaller than 10 %, meaning that samples are stagnant more than 90 % of the time. A better proposal distribution for the Metropolis–Hastings update of a given parameter can help improve acceptance rates, often leading to a decrease in the sample autocorrelation. You are encouraged to experiment with different proposal distributions for k and inspect their effect on the sample autocorrelation. Suggestion: repeat the analysis after modifying the source code to draw sample proposals at random from, say, $q(k|\theta, \lambda, b_1, b_2, \mathbf{Y}) \sim \text{UniformInt}(k-4, \ldots, k+4)$. Just in case, a file containing the modified R source code is available at `https://www.dropbox.com/s/d8dfuhm2bwb4pad/MCMCchpt_altprop.R?dl=0`.

Assessing accuracy and determining chain length. There are two important issues to consider when running an MCMC algorithm. Fisrt, how do we assess the accuracy of the parameter estimates based on the available sample? In other words, how do we compute Monte Carlo standard errors? Second, for how long should we run a chain before we are confident that the results are reasonably accurate?

There is a vast literature on sophisticated methods for computing Monte Carlo standard errors for MCMC output. One method in particular, called Consistent Batch Means Estimation, is rather easy to implement and works reasonably well in practice. Suppose we are interested in estimating the expectation $\mu = E(g(X))$, where X is to be sampled. The batch means method works as follows:

1. Run the Markov chain for $N = bs$ iterations.
2. Let
$$Z_k = \frac{\sum_{j=(k-1)s+1}^{ks} g(X_j)}{s}\,. \tag{11}$$
If we think of the Markov chain as having been divided into b batches of size s each, then Z_k is the Monte Carlo estimate of μ based on the k-th batch. The batch size s should be large enough so that the Z_k are approximately independent. A heuristic approach is to use $s = \sqrt{N}$.
3. Let
$$\hat{\sigma}^2 = \frac{s}{b-1}\sum_{k=1}^{b}(Z_k - \hat{\mu})^2\,. \tag{12}$$
Then the batch means estimate of the Monte Carlo standard error is $\hat{\sigma}/\sqrt{N}$.

Using batch means, we compute standard errors for each of the five parameter estimates. Are these standard errors acceptable?

```
bm(mchain[1,])
bm(mchain[2,])
bm(mchain[3,])
bm(mchain[4,])
bm(mchain[5,])
```

Finally, a simple way of determining when to stop an MCMC run is suggested that is based on the batch means estimate of the standard errors. Run the MCMC algorithm and periodically compute Monte Carlo standard errors for the model parameters using batch means as described above. Once these standard errors attain a desired level of accuracy (specified by the user), stop the simulation. Try running the MCMC algorithm once again, but this time for a million iterations.

```
mchain2 <- mhsampler(dat=Y, MCMCiterations=1e+6)
```

Repeat the analysis and note whether or not the estimates and corresponding Monte Carlo standard errors have changed with respect to the previous sampler.

Making changes to the model. We replace the priors for b_1 and b_2 in Eq. (2) by

$$\begin{aligned} b_1 &\sim \text{Gamma}(0.01, 100)\,; \\ b_2 &\sim \text{Gamma}(0.01, 100)\,. \end{aligned} \tag{13}$$

The full conditionals for θ, λ, and k remain unaltered. However, we now obtain the following full conditionals for b_1 and b_2:

- Full conditional for b_1:

$$f(b_1|k,\theta,\lambda,b_2,\mathbf{Y}) \propto \frac{1}{b_1^{0.5}} e^{-\theta/b_1} \times b_1^{-0.99} e^{-b_1/100}\,. \tag{14}$$

- Full conditional for b_2:

$$f(b_2|k,\theta,\lambda,b_1,\mathbf{Y}) \propto \frac{1}{b_2^{0.5}} e^{-\lambda/b_2} \times b_2^{-0.99} e^{-b_2/100}\,. \tag{15}$$

These full conditionals are not standard distributions, thus we will need to perform Metropolis–Hastings updates on them. You should by now be able to modify the source code to accommodate these changes. Repeat the analysis and comment on its output relative to that of the previous sampler.

5 Rediscovering 51 Peg b with Systemic Live

Systemic Live is a web application running in your browser that allows you to visualize real radial-velocity data sets and to fit them using a planetary-system model. It can be accessed at `http://www.stefanom.org/systemic-live/`. No installation is needed, but you need to make sure that your browser supports the application.

A tutorial is available online that will guide you through the analysis of the radial-velocity data set of 51 Peg, hosting the first exoplanet (51 Peg b) to be discovered around a main-sequence star (Mayor and Queloz 1995; Marcy et al. 1997). How do the estimated planetary parameters compare with the ones in the original discovery paper?

You are also encouraged to analyze the data set of a star hosting a multiple-planet system. Suggestion: υ And A, the host of the first multiple-planet system to be discovered around a main-sequence star (Butler et al. 1997, 1999; Ligi et al. 2012).

6 Using Period04 in the analysis of astronomical time series

Period04 is a computer software especially dedicated to the statistical analysis of long astronomical time series containing gaps. Like its predecessor, Period98, the software provides the tools for extracting individual frequencies from the multiperiodic content of astronomical time series and offers a flexible interface for performing multiple-frequency fits. A user guide to Period04 is presented in Lenz and Breger (2005).

Visit the web page of Period04 at `http://www.univie.ac.at/tops/Period04/` and download the latest version of the software (make sure you read the system requirements beforehand).

Next, download the tutorial data files from the same web page. You will find two tutorials that you are supposed to undertake. In *Tutorial 1* you will use Period04 to examine a data set and determine its frequency content. You will be asked to first perform a Fourier analysis to obtain frequency guesses and then use the fit module to refine these frequencies. Note that the Fourier analysis cannot by itself solve the problem since it is a single-frequency method. *Tutorial 2* provides

a basic introduction on how to find a multiple-frequency solution to a data set taking into account a periodic time shift. Such a periodic time shift could be the result of orbital light-time effects (e.g., Mayer 1990).

References

Butler, R. P., Marcy, G. W., Fischer, D. A., Brown, T. M., Contos, A. R., Korzennik, S. G., Nisenson, P., and Noyes, R. W.: 1999, *ApJ* **526**, 916

Butler, R. P., Marcy, G. W., Williams, E., Hauser, H., and Shirts, P.: 1997, *ApJ* **474**, L115

Carlin, B. P. and Louis, T. A.: 2000, *Bayes and Empirical Bayes Methods for Data Analysis*, Chapman & Hall/CRC Texts in Statistical Science, Chapman & Hall/CRC, second edition

Lenz, P. and Breger, M.: 2005, *Communications in Asteroseismology* **146**, 53

Ligi, R., Mourard, D., Lagrange, A. M., Perraut, K., Boyajian, T., Bério, P., Nardetto, N., Tallon-Bosc, I., McAlister, H., ten Brummelaar, T., Ridgway, S., Sturmann, J., Sturmann, L., Turner, N., Farrington, C., and Goldfinger, P. J.: 2012, *A&A* **545**, A5

Marcy, G. W., Butler, R. P., Williams, E., Bildsten, L., Graham, J. R., Ghez, A. M., and Jernigan, J. G.: 1997, *ApJ* **481**, 926

Mayer, P.: 1990, *Bulletin of the Astronomical Institutes of Czechoslovakia* **41**, 231

Mayor, M. and Queloz, D.: 1995, *Nature* **378**, 355

Shumway, R. H. and Stoffer, D. S.: 2011, *Time Series Analysis and Its Applications*, Springer Texts in Statistics, Springer New York, third edition