APTS 2012–13 preliminary material

Computer-Intensive Statistics

Step-by-step Guide to Preliminary Exercises

This document gives expands on the hints given separately.

Bootstrapping

Efron (1979, 1982) gives the following data on admissions to 15 US Law Schools

```
LSAT:
576 635 558 578 666 580 555 661 651 605 653 575 545 571 594
GPA:
3.39 3.30 2.81 3.03 3.44 3.07 3.00 3.43 3.36 3.13 3.12 2.74 2.76 2.88 2.96
```

Ex 1 Enter these data into R as a data frame `law`, plot them and compute the correlation coefficient.

[There are many ways to enter the data. Here is one of the simplest that you can copy-and-paste to R. Do not omit the blank lines.

```
LSAT <- scan()
576 635 558 578 666 580 555 661 651 605 653 575 545 571 594

GPA <- scan()
3.39 3.30 2.81 3.03 3.44 3.07 3.00 3.43 3.36 3.13 3.12 2.74 2.76 2.88 2.96

law <- data.frame(LSAT, GPA)
law # print it for checking.
cor(law) # gives the full correlation matrix
cor(law)[1,2] # the element we want, or
cor(LSAT, GPA)
```

]

Use Fisher's theory to give a 95% confidence interval for ρ assuming normality.

[

```
n <- nrow(law)
r <- cor(LSAT, GPA)
## Confidence interval for atanh(rho)
ci <- qnorm(c(0.025, 0.975), atanh(r), sqrt(1/(n-3)))
ci
## and for rho
tanh(ci)
```

]

Ex 2 Create a bootstrap sample. Here is one simple way to do it:

```
law[sample(1:15, replace=TRUE), ]
```

Now compute its correlation coefficient

[

```
cor(law[sample(1:15, replace=TRUE), ])[1,2]
```

]

Repeat 1000 times and put in a vector, making use of the `replicate` function.

[

```
rhostar <- replicate(1000, cor(law[sample(1:15, replace=TRUE), ])[1,2])
```

]

Now take a look at the bootstrap distribution on both correlation and `atanh` scales.

[Here are some ways to look at the results using a histogram and a kernel density estimate. If you are unfamiliar with these, look them up in Venables & Ripley (2002).

```
hist(rhostar, prob=TRUE)
lines(density(rhostar, bw="SJ"))
hist(atanh(rhostar), prob=TRUE)
lines(density(atanh(rhostar), bw="SJ"))
```

]

Ex 3 [The solution is in the question itself.]

Ex 4 How about a confidence interval for ρ? We can use function `boot.ci` to produce several different confidence intervals – do so on both correlation and `atanh` scale.

[You need to use the function `boot.ci`. Here's a start

```
boot.ci(out, type=c("norm", "basic", "perc", "bca"))
```

You may well get a warning, and need to do more runs.

```
out <- boot(law, stat, R=10000)
boot.ci(out, type=c("norm", "basic", "perc", "bca"))
```

That is for confidence intervals on the original scale. For `atanh` scale, use

```
boot.ci(out, type=c("norm", "basic", "perc", "bca"), h=atanh, hinv=tanh)
```

]

Ex 5 Cox and Lewis (1966) reported 799 time intervals between pulses on a nerve fibre. The dataset can be downloaded from
`http://www.stat.cmu.edu/~larry/all-of-statistics/=data/nerve.dat`
and is used by Wasserman (2004, pp. 98, 111).

[To get the data into R use

```
nerve <-
scan("http://www.stat.cmu.edu/~larry/all-of-statistics/=data/nerve.dat")
summary(nerve) # a quick check
```

]

Use the bootstrap to get confidence intervals for the median and skewness of these data. You can either write your own R function to compute the skewness, or get one from contributed package `e1071` and about ten others. To see if you have one installed, use

```
help.search("skewness")
```

[If you do not, here is a simple version

```
skewness <- function (x) mean((x - mean(x))^3)/sd(x)^3
```

It is useful to jitter the data, e.g.

```
nerve2 <- jitter(nerve, amount=0.005)
```

You need to use functions `boot` and `boot.ci` as in the the previous exercises. E.g.

```
stat <- function(x, i) median(x[i])
out <- boot(nerve2, stat, R=1000)
plot(out)
boot.ci(out, type=c("norm", "basic", "perc", "bca"))

stat2 <- function(x, i) skewness(x[i])
out2 <- boot(nerve2, stat2, R=1000)
plot(out2)
boot.ci(out2, type=c("norm", "basic", "perc", "bca"))
```

To do better we could make use of a smoothed bootstrap (Davison & Hinkley, 1997, p. 531).

```
n <- length(nerve)
s <- 0.1 # the standard deviation of a normal kernel
ran.gen <- function(data, mle) {
    n <- length(data)
    rnorm(n, data[sample(n, n, replace=TRUE)], mle)
}
out3 <- boot(nerve, median, R=1000, sim="parametric",
             ran.gen = ran.gen, mle = s)
plot(out3)
boot.ci(out3, type=c("norm", "basic", "perc"))
```

Experiment with the amount of smoothing `s`. If you have enough patience you can try larger values of `R`.]

Spatial patterns and MCMC

Ex 6 The Strauss process (1) and ways to simulate it are contained in R packages `spatial` and `spatstat`.

Let us consider the Swedish pines data from Ripley (1981), described in Venables & Ripley (2002, §15.3). Retrieve and plot it in R by

```
library(MASS)
library(spatial)
pines <- ppinit("pines.dat")
eqscplot(pines, xlim = c(0, 10), ylim = c(0, 10), xlab = "", ylab = "")
```

How many points are there? How might you describe the pattern? By the way, the coordinates are in metres.

Venables & Ripley (2002, p. 443) suggest that $R = 0.7$ and $c = 0.15$ are reasonable estimates. Use function `Strauss` to simulate with these parameter values and compare a plot with the real data.

[Let's compare the data and 3 simulations:

```
par(mfrow=c(2,2), mar=c(3,3,1,1))
eqscplot(pines, xlim = c(0, 10), ylim = c(0, 10), xlab = "", ylab = "")
for (i in 1:3) {
    sp <- Strauss(n = 72, c = 0.15, r = 0.7)
    eqscplot(sp, xlim = c(0, 10), ylim = c(0, 10), xlab = "", ylab = "")
}
```

]

Ex 7 Wasserman (2004, p. 412) gives an example, and the following is based on his (incorrect) R code for his Figure 24.2.

```
metrop <- function(N, b)
{
    x <- numeric(N)
    for(i in 2:N){
        y <- rnorm(1, x[i-1], b)
        r <- (1+x[i-1]^2)/(1+y^2)
        u <- runif(1)
        x[i] <- ifelse(u < r, y, x[i-1])
    }
    x
}

par(mfrow=c(3,1))
N <- 1000

for(b in c(0.1, 1, 10)) {
  plot(metrop(N, b), type="l", xlab="", ylab="")
  abline(h=0, col="grey")
}
```

The aim here is to simulate from the Cauchy distribution as the stationary distribution of a Markov chain.

(a) The Markov chain being simulated here is a random walk in which moves are accepted with probability `r` and otherwise rejected. Write down a formal description of the stochastic process.

(b) What is the rôle of the parameter `b`? Why do you think that Wasserman chose to show the three values he did?

[`b` is the standard deviation of the steps in the random walk. If it is small, most Metropolis steps are accepted but progress is slow. If it is large, most steps are rejected.]

(c) Where did `N = 1000` come from? Is it a reasonable value? Experiment with changing it.

[It seems to be designed to be quick. It is too small, and `N <- 10000` would be better. In that case you would want to subsample to plot, e.g.

```
par(mfrow=c(3,1))
N <- 10000

for(b in c(0.1, 1, 10)) {
  x <- metrop(N, b)
  ind <- seq(1, N, 20) # every 20 steps
  plot(ind, x[ind], type="l", xlab="", ylab="")
  abline(h=0, col="grey")
}
```

]

(d) Compare the output with the intended Cauchy distribution.

[For example

```
x <- metrop(N, 1)[seq(1, N, 20)]
hist(x, prob = TRUE)
lines(density(x, bw="SJ"))
qqplot(qcauchy(ppoints(length(x)))[order(order(x))], x)
```

The tails are not long enough (at least when I did this).]

(e) (More ambitious.) If you are familiar with Markov chains, try to prove that there is a stationary distribution and that it is Cauchy.

(f) This is not a serious exercise in sampling from a Cauchy distribution – find out how it is done efficiently in real applications.

[Look at the help page for function `rcauchy`, or in the references in Appendix A.]