Sampling and Resampling

Thomas Lumley

Biostatistics

2005-11-3

Basic Problem

(Frequentist) statistics is based on the sampling distribution of statistics:

Given a statistic T_n and a true data distribution, what is the distribution of T_n

- The mean of samples of size 10 from an Exponential
- The median of samples of size 20 from a Cauchy
- The difference in Pearson's coefficient of skewness $((\bar{X} - m)/s)$ between two samples of size 40 from a Weibull(3,7).

This is easy: you have written programs to do it. In 512-3 you learn how to do it analytically in some cases.

But...

We really want to know:

What is the distribution of T_n in sampling from the true data distribution

But we don't know the true data distribution.

Empirical distribution

On the other hand, we do have an estimate of the true data distribution. It should look like the sample data distribution. (we write $\mathbb{F}_n$ for the sample data distribution and $\mathbb{F}$ for the true data distribution). The Glivenko–Cantelli theorem says that the cumulative distribution function of the sample converges uniformly to the true cumulative distribution.

We can work out the sampling distribution of $T_n(\mathbb{F}_n)$ by simulation, and hope that this is close to that of $T_n(F)$.

Simulating from $\mathbb{F}_n$ just involves taking a sample, with replacement, from the observed data. This is called the bootstrap. We write $\mathbb{F}_n^*$ for the data distribution of a resample.

Too good to be true?

There are obviously some limits to this

- It requires large enough samples for $\mathbb{F}_n$ to be close to F.
- It works better for some statistics (eg mean, variance) than others (eg median, quantiles)
- It doesn't work at all for some statistics (eg min, max, number of unique values)

The reason for the difference between statistics is that $\mathbb{F}_n$ needs to be "close to" F in an appropriate sense of "close" for the statistic. Precise discussions of this take a lot of math ['Hadamard differentiable in the uniform norm']

Uses of bootstrap

There are two main uses

- When you know the distribution of T_n is normal with mean θ, you just don't know how to compute the variance

- With a well-behaved statistic where the sample size is a little small for the Normal approximation.

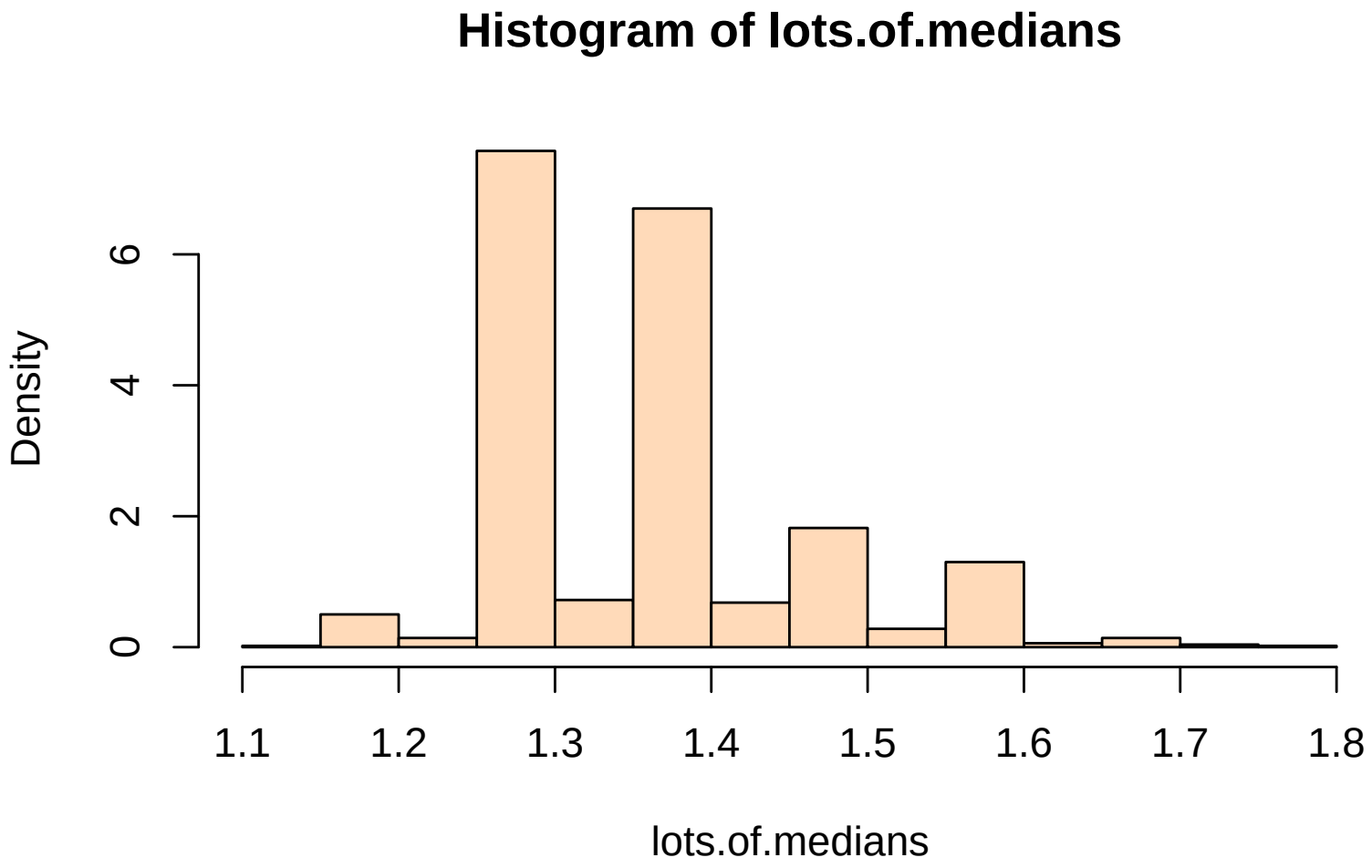
It can also be used when you don't know what the asymptotic distribution is, but then you do need quite a bit of analysis to be sure that the bootstrap works for this statistic.

Example

Median bilirubin in PBC data

```
data(pbc, package="survival")

resample.a.median<-function(x){
xstar<- sample(x, size=length(x), replace=TRUE)
median(xstar)
}

lots.of.medians<-replicate(1000, resample.a.median(pbc$bili))

hist(lots.of.medians, col="peachpuff",prob=TRUE)
```

Example


Histogram of lots.of.medians
Density
0
2
4
6
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
lots.of.medians

Notes

- sample() takes a sample from a given vector. This can be with or without replacement and with equal or unequal probabilities.

- replicate executes an expression many times and returns the results. It is tidier than a loop or apply.

- data() has a package argument for when you want the dataset but not the whole package.

- The histogram is fairly discrete, because the data are rounded to 2 decimal places: the true sampling distribution of the median is discrete. The true distribution of serum bilirubin isn't, but we have no data from that distribution.

How well does it work?

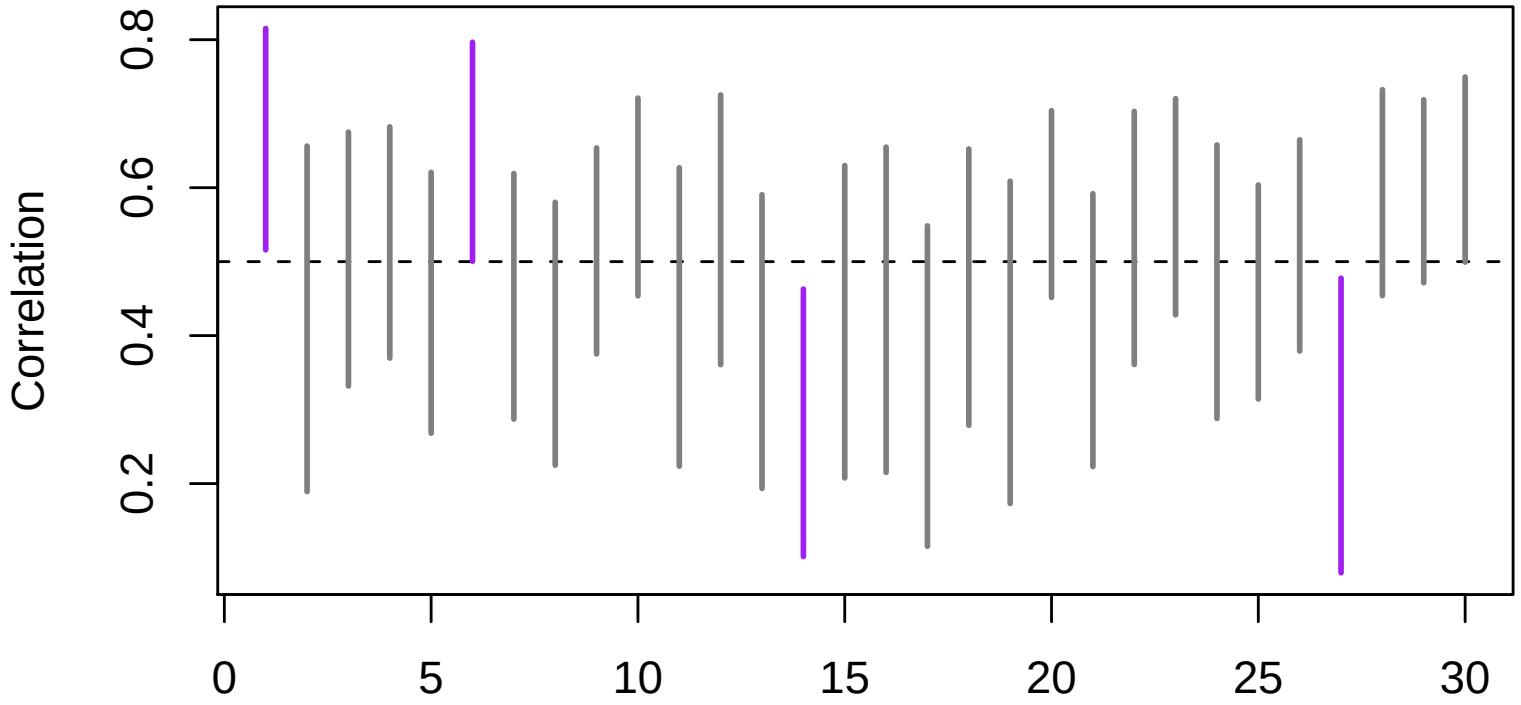
These graphs show the 5% and 95% points of the estimated sampling distribution. 90% of these should cover the true value. We need to use known distributions for this.

```
library(MASS)  ## Modern Applied Statistic in S (V&R)

resample.a.corr<-function(xy){
    index <- sample(nrow(xy),size=nrow(xy),replace=TRUE)
    cor(xy[index,1],xy[index,2])
}

lots.of.corr<-replicate(30, {
        dat<-mvrnorm(50,c(0,0), Sigma=matrix(c(1,.5,.5,1),2))
        replicate(400, resample.a.corr(dat))
       })
```

How well does it work?

```
qq<-apply(lots.of.corr,2,quantile, probs=c(0.05,0.95))
plot(1, 1, xlim=c(1,30), ylim=range(c(0.5,qq)),
       ylab="Correlation", xlab="")
abline(h=0.5,lty=2)
in.interval<-qq[1,]<0.5 & qq[2,]>0.5
segments(1:30,qq[1,],1:30,
    qq[2,],col=ifelse(in.interval,"grey50","purple"),lwd=2)
```

How well does it work?



Notes

- We need to simulate the entire bootstrap process — draw a real sample, take 400 resamples from it — thirty times

- We resample rows, by sampling from numbers `1...nrow(xy)` and then apply this as a subset index.

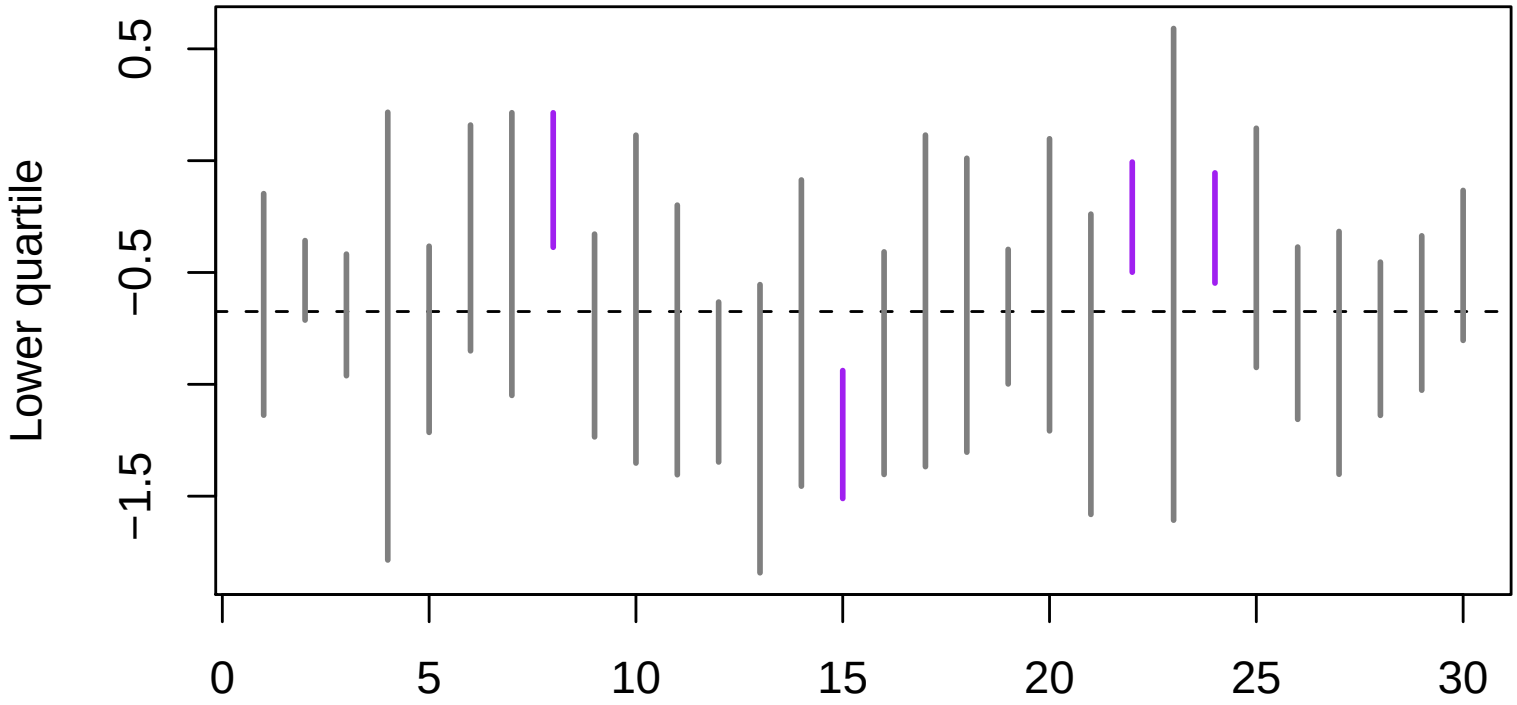
- 400 is a minimal reasonable number for boostraps and most simulations. The uncertainty in the 90% range is about 1.5%, in a 95% range would be about 3.5%. Usually between 1000 and 10,000 is a good number.

- The percentile bootstrap will always give estimates between -1 and 1 for correlation

Lower quartile

```
resample.a.q25<-function(x){
    x <- sample(x,length(x),replace=TRUE)
quantile(x, prob=0.25)
}

lots.of.q25<-replicate(30, {
        dat<-rnorm(20)
        replicate(400, resample.a.q25(dat))
       })

qq<-apply(lots.of.q25,2,quantile, probs=c(0.05,0.95))
plot(1,1,xlim=c(1,30),ylim=range(qq),ylab="Lower quartile",xlab="")
abline(h=qnorm(0.25),lty=2)
in.interval<-qq[1,]<qnorm(0.25) & qq[2,]>qnorm(0.25)
segments(1:30,qq[1,],1:30,
    qq[2,],col=ifelse(in.interval,"grey50","purple"),lwd=2)
```

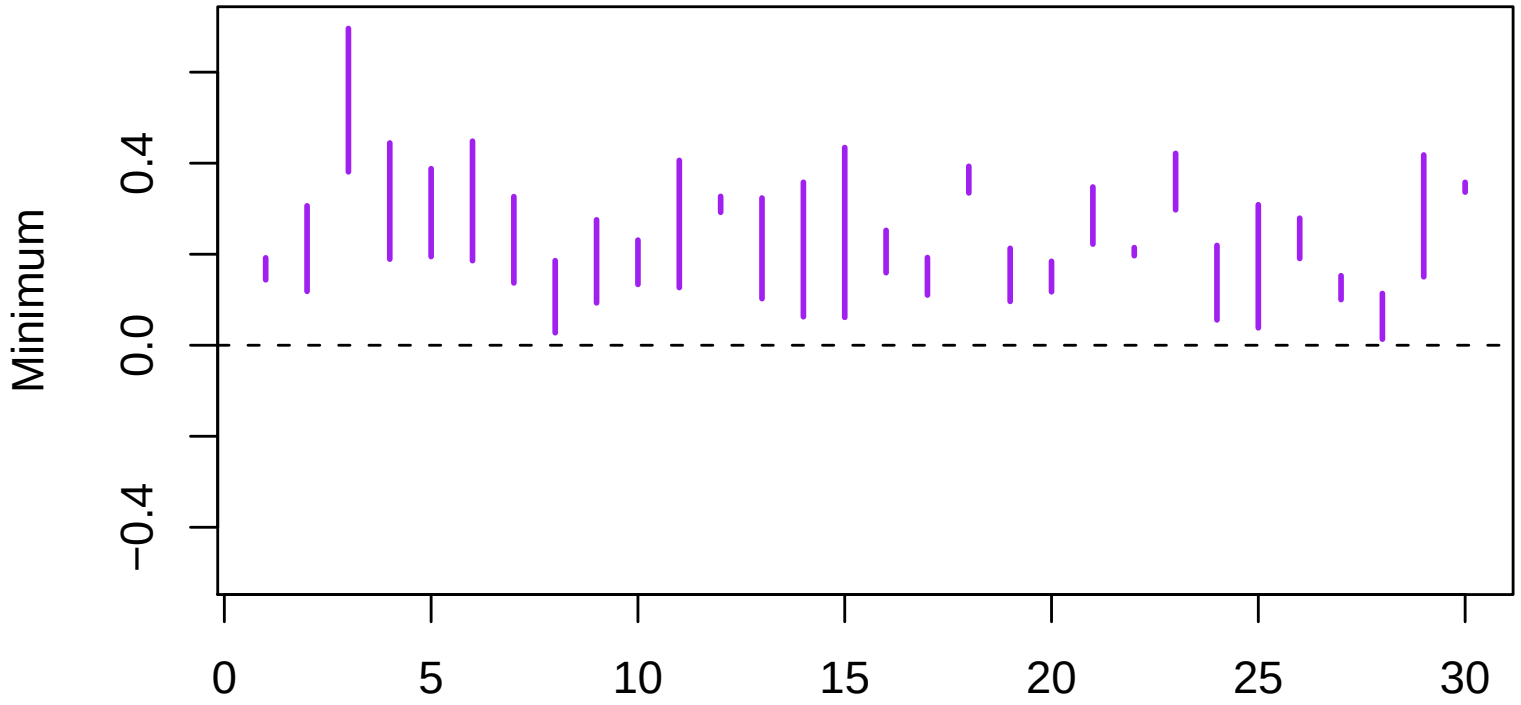
Lower quartile



Minimum

```
resample.a.min<-function(x){
    x <- sample(x,length(x),replace=TRUE)
    min(x)
}

lots.of.min<-replicate(30, {
        dat<-rgamma(20,2,2)
        replicate(400, resample.a.min(dat))
       })

qq<-apply(lots.of.min,2,quantile, probs=c(0.05,0.95))
plot(1,1,xlim=c(1,30),ylim=range(c(-0.5,qq)),ylab="Minimum",xlab="")
abline(h=0,lty=2)
in.interval <- qq[1,] < 0 & qq[2,]> 0
segments(1:30,qq[1,],1:30,
    qq[2,],col=ifelse(in.interval,"grey50","purple"),lwd=2)
```

Minimum



Bootstrap packages

You don't have to write your own bootstrap functions: there are two packages

- boot, associated with a book by Davison and Hinkley, and written by Angelo Canty

- bootstrap, associated with book by Efron and Tibshirani

The boot package comes with R and is more comprehensive. S-PLUS also has nice bootstrap functions written by Tim Hesterberg (at Insightful).

The boot package

As we did with resampling correlations, the boot function resamples row indices rather than data. You have to provide a function that takes a data set as its first argument and a set of row indices as the second argument.

We could redo the correlation example changing just a few lines

```
library(MASS)  ## Modern Applied Statistics in S (V&R)

resample.a.corr<-function(xy, index){
    cor(xy[index,1],xy[index,2])
}

lots.of.corr<-replicate(30, {
        dat<-mvrnorm(50,c(0,0), Sigma=matrix(c(1,.5,.5,1),2))
        boot(dat, resample.a.corr, R=400)$t       })
```

The boot package

```
qq<-apply(lots.of.corr,2,quantile, probs=c(0.05,0.95))
plot(1,1,xlim=c(1,30),ylim=range(c(0.5,qq)),ylab="Correlation",xlab="
abline(h=0.5,lty=2)
in.interval<-qq[1,]<0.5 & qq[2,]>0.5
segments(1:30,qq[1,],1:30,
    qq[2,],col=ifelse(in.interval,"grey50","purple"),lwd=2)
```

More usefully, the package provides a variety of bootstrap estimates. For one sample we get

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 400 bootstrap replicates
CALL :
boot.ci(boot.out = b)
Intervals :
Level      Normal              Basic
95%   ( 0.3008,  0.6914 )   ( 0.3069,  0.7266 )

Level     Percentile            BCa
95%   ( 0.2692,  0.6889 )   ( 0.2446,  0.6665 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

The BCa interval should be slightly more accurate, so if it agrees with the "Percentile" interval they are both likely to be trustworthy. If the BCa and Percentile intervals are importantly different then the Percentile interval is not reliable (and BCa may or may not be).

Stata has similar facilities. Programming the statistic to be bootstrapped was a bit of a pain, but things have improved in version 9.

Testing vs estimation

The bootstrap estimates the actual sampling distribution of a statistic (more accurately as sample size increases). This is useful for confidence intervals.

Sometimes we want a test of a precisely specified null hypothesis. This requires the sampling distribution that the statistic would have if the null hypothesis were true. We cannot just use resampling, since the null hypothesis may not be true in the real data.

Consider the randomized trial subset of the PBC data. If the treatment (D–penicillamine) has no effect whatsoever, then the data for everything except the treatment variable would stay the same if different people had been treated.

This is often called the strong null hypothesis: we are not just saying that survival is on average as good with treatment as without, we are saying that treatment has no effect at all on survival.

We can generate a sampling distribution where the strong null hypothesis is true (eg for ratio of median survival time) by just changing the values of the treatment variable. That is, we permute the vector `trt` of treatment values and reanalyze.

- If the strong null hypothesis is really true this will be the real sampling distribution.

- If the strong null hypothesis is not really true this will not be the real sampling distribution: we hope it is very different from the real sampling distribution.

- The p-value is the proportion of simulated differences that exceed the real observed difference

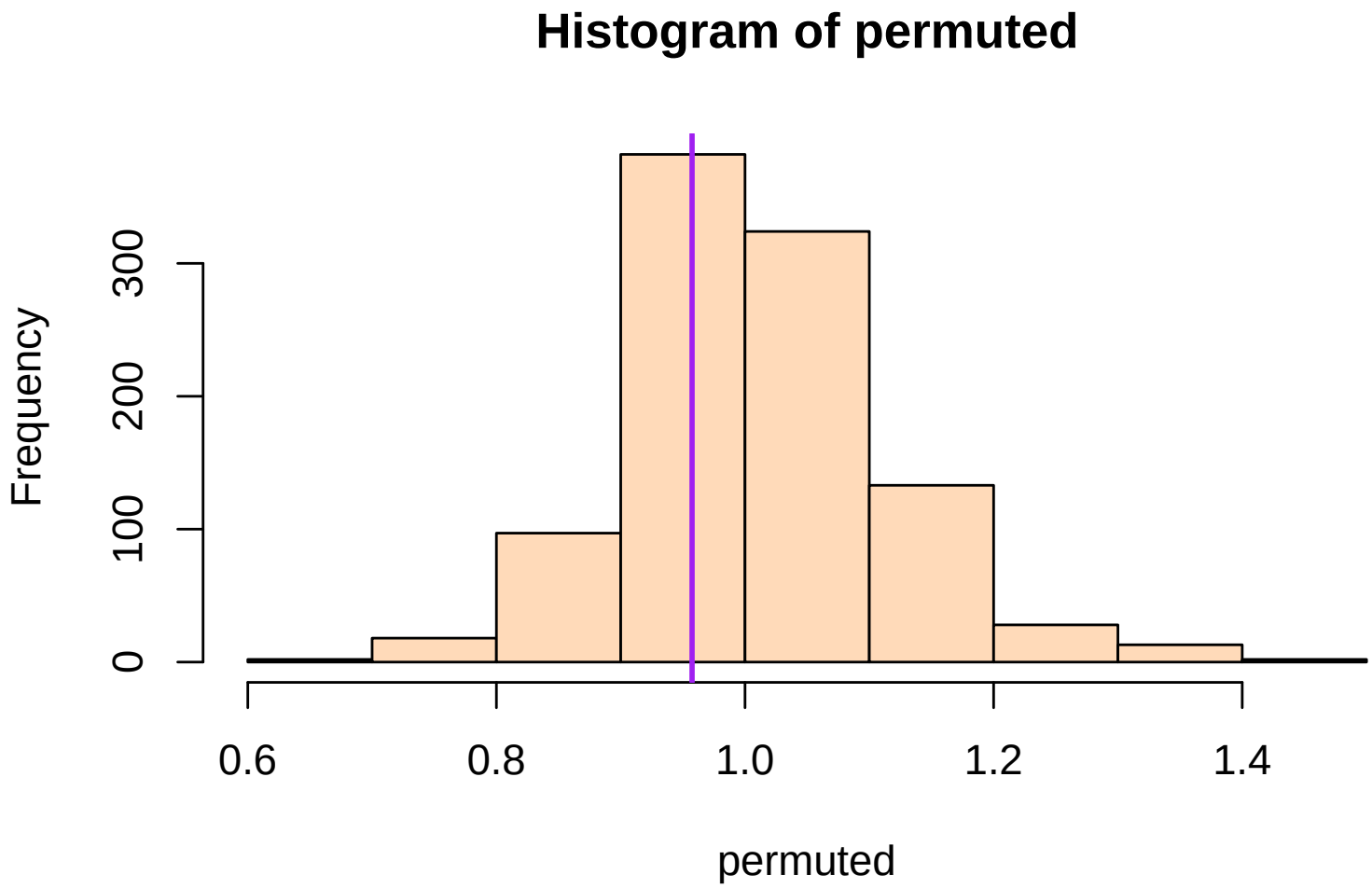
This is a permutation test, or when the data come from a randomized trial, a randomization test. Under the strong null hypothesis the p-value is exact.

Permuting survival

```
library(survival)
data(pbc)
pbcrand <- subset(pbc, trt %in% c(1,2))

medsurv<-function(km) km$time[min(which(km$surv<0.5))]

survratio<-function(treatment){
   km<-survfit(Surv(time,status)~treatment, data=pbcrand)
   medsurv(km[1])/medsurv(km[2])
}

observed.ratio <- survratio(pbcrand$trt)

permuted <- replicate(1000, survratio(sample(pbcrand$trt)))
```

We end up with 31% of permuted ratios less than the observed 0.9574 and 37% greater than 1/0.9574, so the two-sided p-value is 0.68.

Permuting survival



PBC trial

- The p-value of 0.68 says there is no evidence at all that the treatment affects survival.

- The data are consistent with the ratio of median survival being 1.

- We would also want a confidence interval, to ask whether the data are consistent with ratios of median survival interestingly different from 1. A bootstrap gives a 95% confidence interval (0.70, 1.26), so we can rule out large benefit or harm of the treatment.

Weak null hypothesis

A treatment may increase survival for people and decrease it for others, in a way that the ratio of median survival stays the same. The strong null hypothesis is false, but a weak null hypothesis is true.

The permutation test need not have the correct Type I error rate: it can be too high or too low. When groups are the same size the Type I error rate is often close to the nominal level.

Simulations are more complicated: we need many replications of a permutation test. To simplify computations we will switch to comparing means.

```
meandiff<-function(x,trt){
mean(x[trt==1])-mean(x[trt==2])
}
meanpermtest<-function(x,trt,n=1000){
observed<-meandiff(x,trt)
perms<-replicate(n, meandiff(x, sample(trt)))
mean(abs(observed)>abs(perms))
}

trt1<-rep(c(1,2),c(10,90))

perm.p<-replicate(1000, {
          x1<-rnorm(100, 0, s=trt1)
          meanpermtest(x1,trt1)})

table(cut(perm.p,c(0,.05,.1,.5,.9,.95,1)))
```

```
  (0,0.05] (0.05,0.1]  (0.1,0.5]  (0.5,0.9] (0.9,0.95]   (0.95,1]
        86         99        564        244           6           0
```

The p-values are too small, relative to a uniform distribution. If we reverse the standard errors we get

```
  (0,0.05] (0.05,0.1]  (0.1,0.5]  (0.5,0.9] (0.9,0.95]   (0.95,1]
        27         28        275        354          67         249
```

If the two groups each have 50 observations we get

```
  (0,0.05] (0.05,0.1]  (0.1,0.5]  (0.5,0.9] (0.9,0.95]   (0.95,1]
        50         45        403        407          52          43
```

which is much better.