Bootstrap methods Part 2

Bootstrap:

Example

Can a moth remember what it learned as a caterpillar? (Blackiston et al., 2008, PLoS ONE)

We consider the odds ratio, where the odds (of success) in a group is the fraction of the proportion of success divided by the proportion of failure.

Air choice / Group	Treatment	Control
Clear air	32	25
Specific odor	9	21
Total	41	46

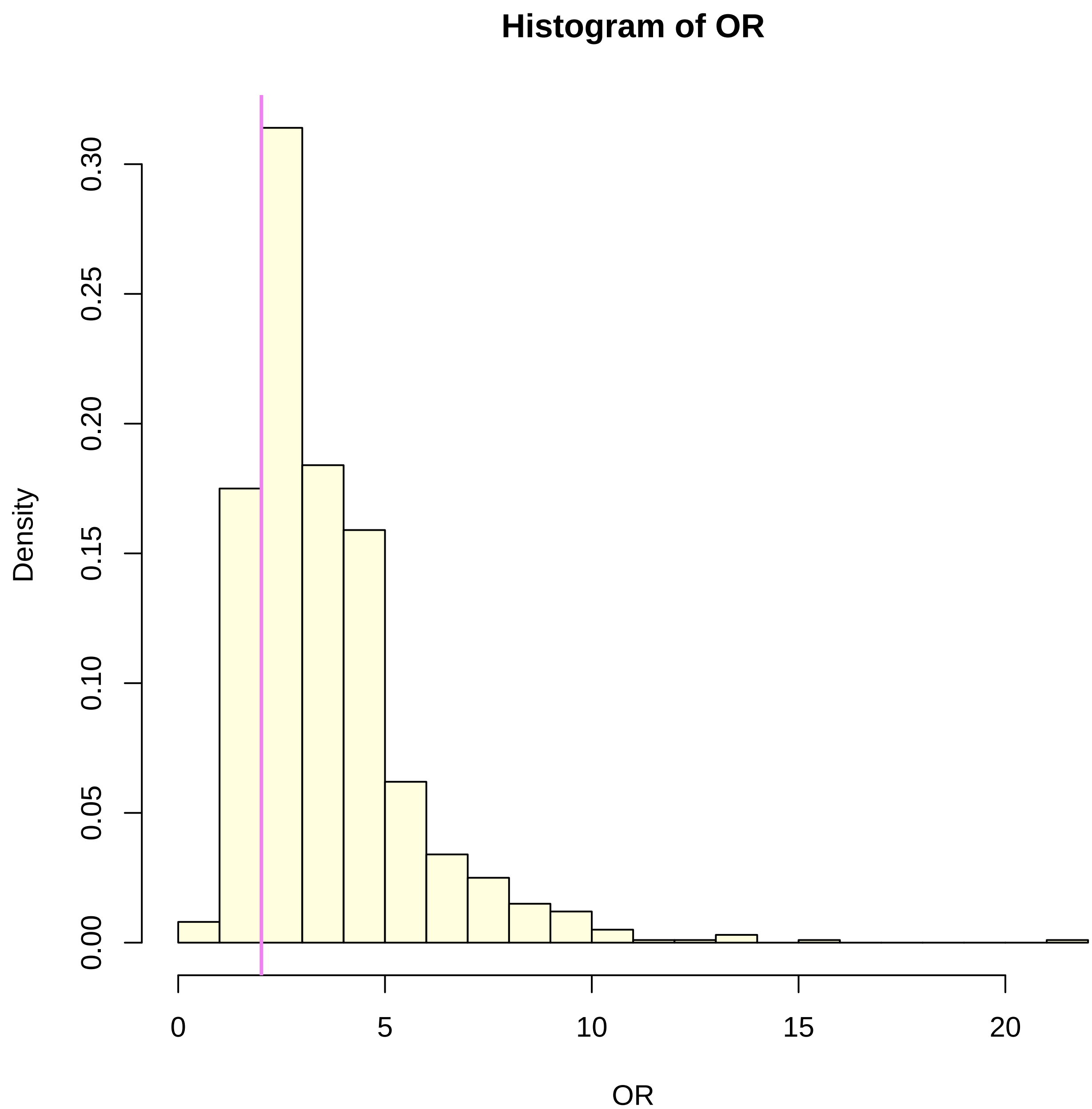
```
library(bootstrap)
B = 1000

treatment = c(rep(1,32), rep(0,9))
control = c(rep(1,25), rep(0,21))
set.seed(1)

pt = bootstrap(x=treatment, nboot=B, theta=mean)$thetastar
pc = bootstrap(x=control, nboot=B, theta=mean)$thetastar

OR = (pt/(1-pt))/(pc/(1-pc))
sd(OR)
```

```
[1] 2.01208
```

```
quantile(OR, c(0.025,0.975))
```

```
    2.5%    97.5% 
1.193182 8.913580 
```

```r
hist(OR, probability=T, breaks=20, col="lightyellow")
abline(v=sd(OR), col="violet", lwd=2)
```




```r
# Define the statistic
proportion = function(x) return(sum(x==1)/length(x))
set.seed(1)
```

```r
pt = bootstrap(x=treatment, nboot=B, theta=proportion)$thetastar
pc = bootstrap(x=control, nboot=B, theta=proportion)$thetastar
OR = (pt/(1-pt))/(pc/(1-pc))
sd(OR)
```

```
[1] 2.01208
```

Regression analysis

The Duncan data frame has 45 rows and 4 columns. Data on the prestige and other characteristics of 45 U.S. occupations in 1950.

```r
library(car)
library(MASS)
library(simpleboot)
data(Duncan)
attach(Duncan)

head(Duncan)
```

```
           type income education prestige
accountant prof     62        86       82
pilot      prof     72        76       83
architect  prof     75        92       90
author     prof     55        90       76
chemist    prof     64        86       90
minister   prof     21        84       87
```

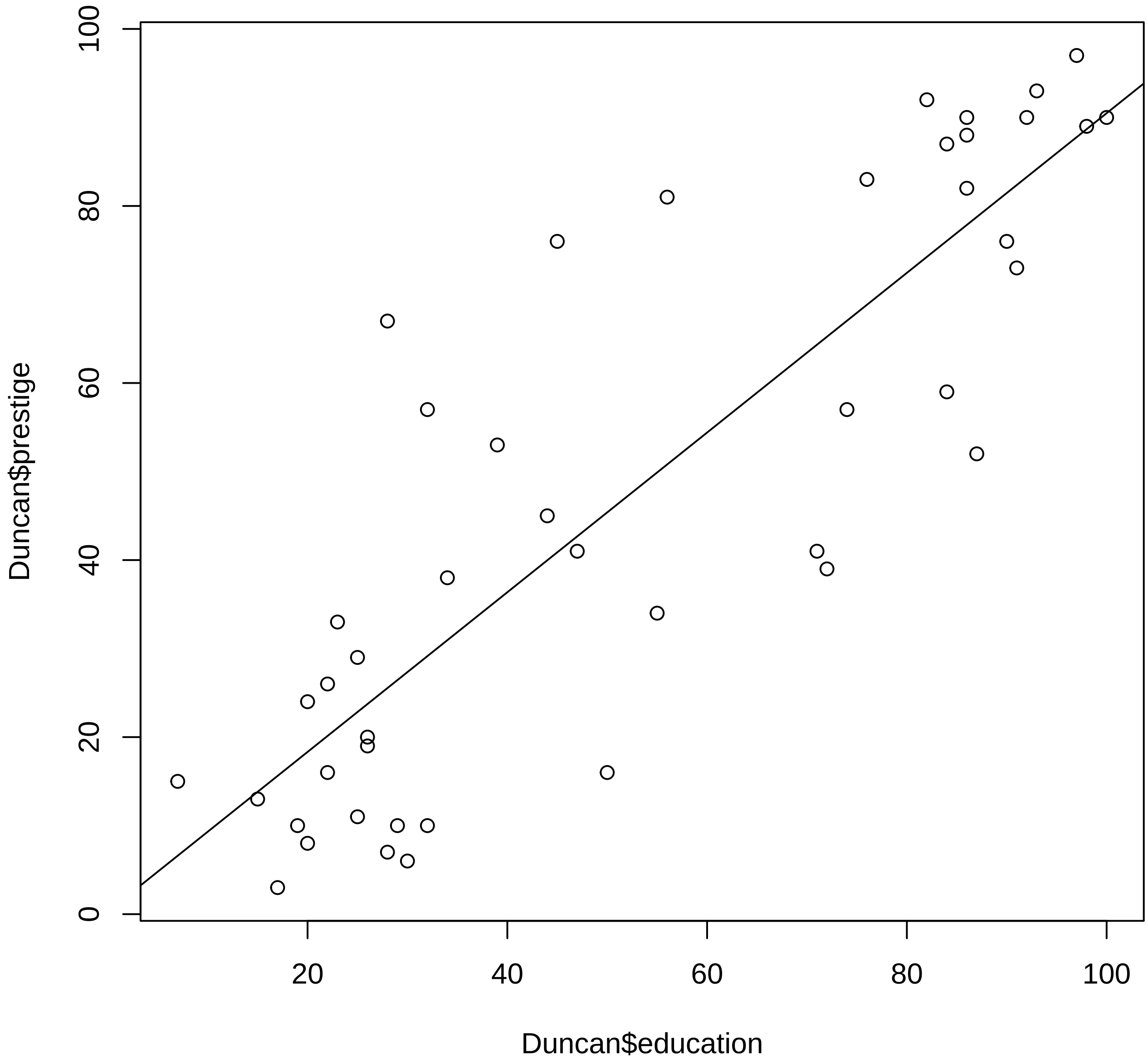
```r
model.dun1 = lm(prestige ~ education, data=Duncan)
coef(summary(model.dun1))
```

```
             Estimate Std. Error     t value     Pr(>|t|)
(Intercept) 0.2839995 5.09306471  0.05576201 9.557897e-01
education   0.9019958 0.08455492 10.66757246 1.170879e-13
```

```r
summary(model.dun1)$r.squared
```

```
[1] 0.7257602
```

```r
plot(Duncan$education, Duncan$prestige)
abline(model.dun1)
```




```r
model.dun2 = lm(prestige ~ income + education, data=Duncan)
coef(summary(model.dun2))
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) -6.0646629 4.27194117 -1.419650 1.630896e-01
income       0.5987328 0.11966735  5.003310 1.053184e-05
education    0.5458339 0.09825264  5.555412 1.727192e-06
```

```r
summary(model.dun2)$fstatistic
```

```
   value    numdf    dendf
101.2162   2.0000  42.0000
```

```r
summary(model.dun2)$adj.r.squared
```

```
[1] 0.8199912
```

Bootstrapping with package simpleboot

```r
library(simpleboot)
summary(lm.boot(lm(prestige ~ income + education), R=B, rows=T))
```

```
BOOTSTRAP OF LINEAR MODEL  (method = rows)

Original Model Fit
------------------
Call:
lm(formula = prestige ~ income + education)

Coefficients:
(Intercept)       income    education
    -6.0647       0.5987       0.5458

Bootstrap SD's:
(Intercept)       income    education
  3.0776095    0.1682817    0.1378040
```

A **robust** regression model:

```r
model.dun3 = rlm(prestige ~ income + education, data=Duncan)
coef(summary(model.dun3))
```

```
                 Value Std. Error   t value
(Intercept) -7.1107028 3.88131509 -1.832034
income       0.7014493 0.10872497  6.451593
education    0.4854390 0.08926842  5.437970
```

```r
B = 1000

rrpair = function(index, xdata){
rlm(prestige ~ income + education, data=xdata[index,])$coefficients
}
set.seed(1)
```

```r
rbet = bootstrap(x=1:45, B, theta=rrpair, xdata=Duncan)$thetastar

sd(rbet[1,])
```

```
[1] 3.046292
```

```r
sd(rbet[2,])
```

```
[1] 0.1787839
```

```r
sd(rbet[3,])
```

```
[1] 0.1392966
```

Bootstrapping residuals

```r
library(bootstrap)
data(Duncan)

regres = function(x, beta, xdata){
sprestige = beta[1] + beta[2]*xdata$income + beta[3]*xdata$education + x

return(coef(lm(sprestige ~ income + education, data=xdata)))
}

model = lm(prestige ~ income + education, data=Duncan)
res = model$residuals
beta.h = coef(model)

B = 1000
set.seed(1)
bet.res = bootstrap(x=res, B, regres, beta=beta.h, xdata=Duncan)$thetastar

sd(bet.res[1,])
```

```
[1] 4.100723
```

```r
sd(bet.res[2,])
```

```
[1] 0.1141664
```

```r
sd(bet.res[3,])
```

```
[1] 0.0979109
```

Bootstrapping residuals (with package simpleboot)

```r
library(simpleboot)

summary(lm.boot(lm(prestige ~ income + education), R=B, rows=F))
```

```
BOOTSTRAP OF LINEAR MODEL  (method = residuals)

Original Model Fit
------------------
Call:
lm(formula = prestige ~ income + education)

Coefficients:
(Intercept)       income    education
    -6.0647       0.5987       0.5458

Bootstrap SD's:
(Intercept)       income    education
 4.10436442   0.11725724   0.09643463
```

Bootstrap regression with library car

```r
N = 50
sd = 0.5
x = rnorm(N)
y = 10 * x + sd * rnorm(N)^2
datos = data.frame(y, x)
```

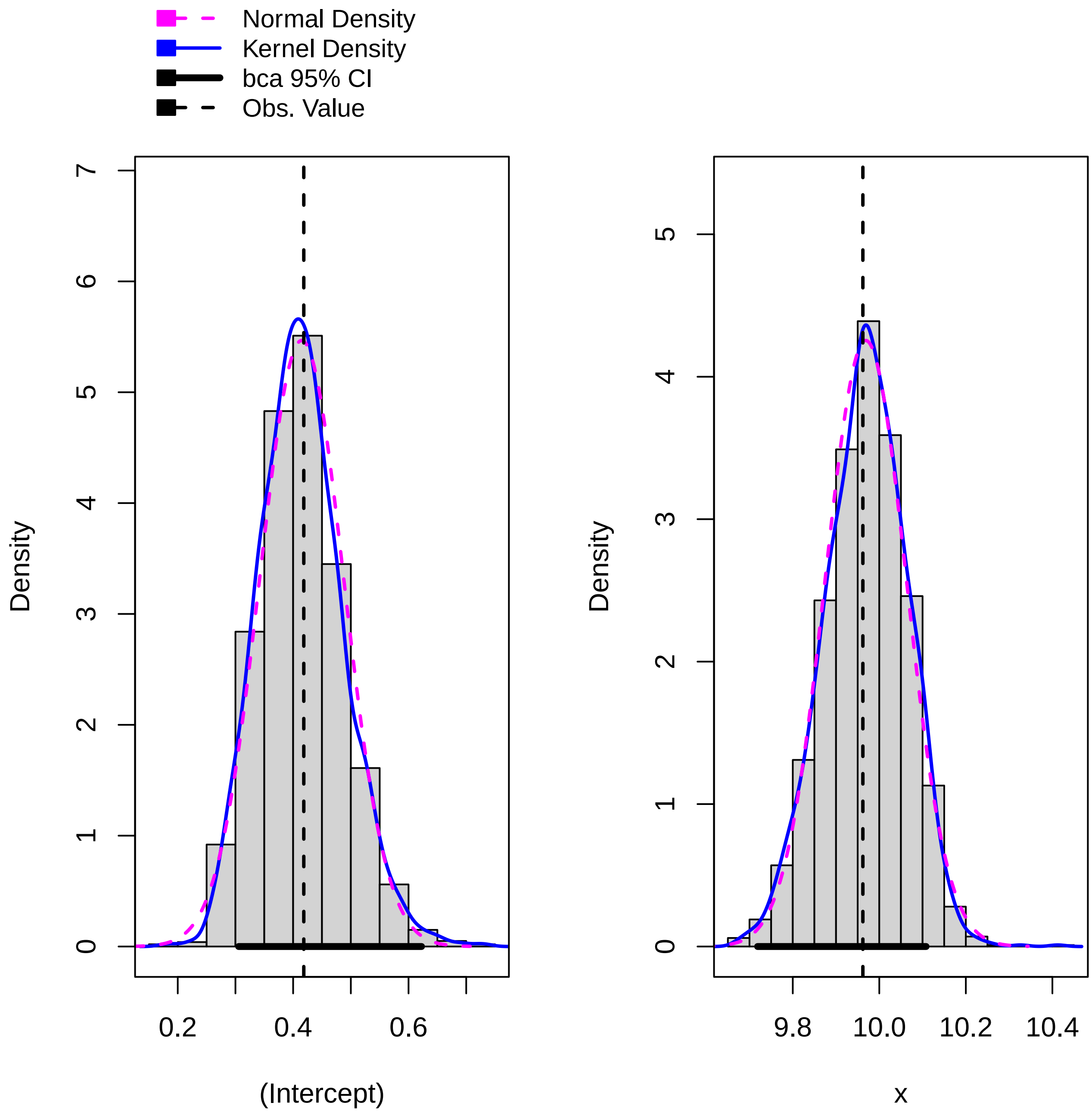
```r
library(car)

modeloB = lm(y ~ x, datos)
betahat.boot = Boot(modeloB, R=2000)
summary(betahat.boot)  # default summary
```

```
Number of bootstrap replications R = 2000
            original   bootBias   bootSE bootMed
(Intercept)  0.41846 -0.0036317 0.072978 0.41159
x            9.96203  0.0066940 0.093776 9.97080
```

```r
confint(betahat.boot)
```

```
Bootstrap bca confidence intervals

                2.5 %     97.5 %
(Intercept) 0.3055779  0.6225411
x           9.7187426 10.1081584
```

```r
hist(betahat.boot)
```



With **residuals**:

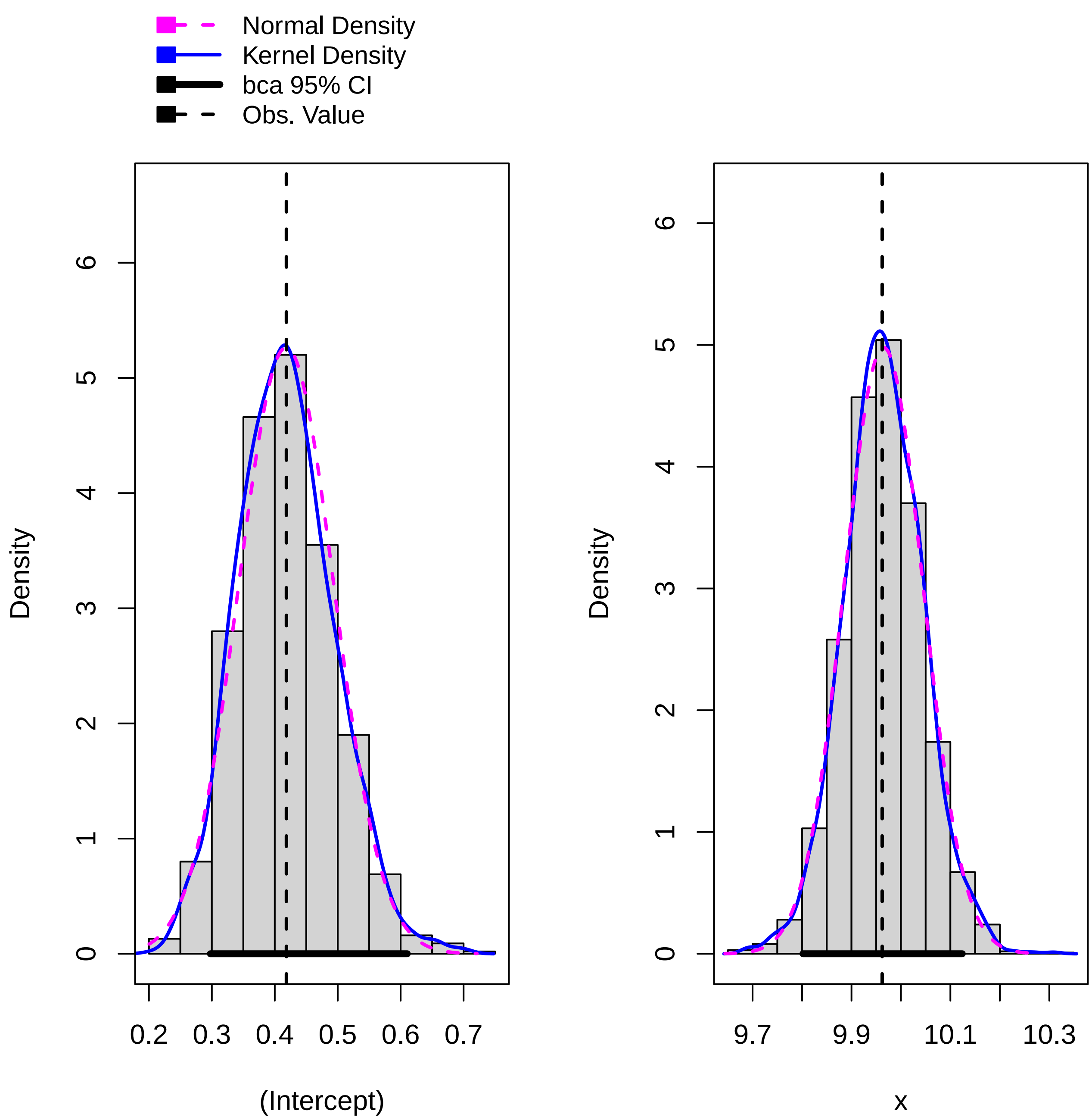
```
betahat.boot2 = Boot(modeloB, method = 'residual', R=2000)
summary(betahat.boot2)  # default summary
```

```
Number of bootstrap replications R = 2000
             original    bootBias   bootSE bootMed
(Intercept)  0.41846 -0.00023121 0.075772 0.41544
x            9.96203  0.00271319 0.080113 9.96321
```

```r
confint(betahat.boot2)
```

```
Bootstrap bca confidence intervals

                 2.5 %     97.5 %
(Intercept) 0.2976693  0.6105535
x           9.8018166 10.1241178
```

```r
hist(betahat.boot2)
```

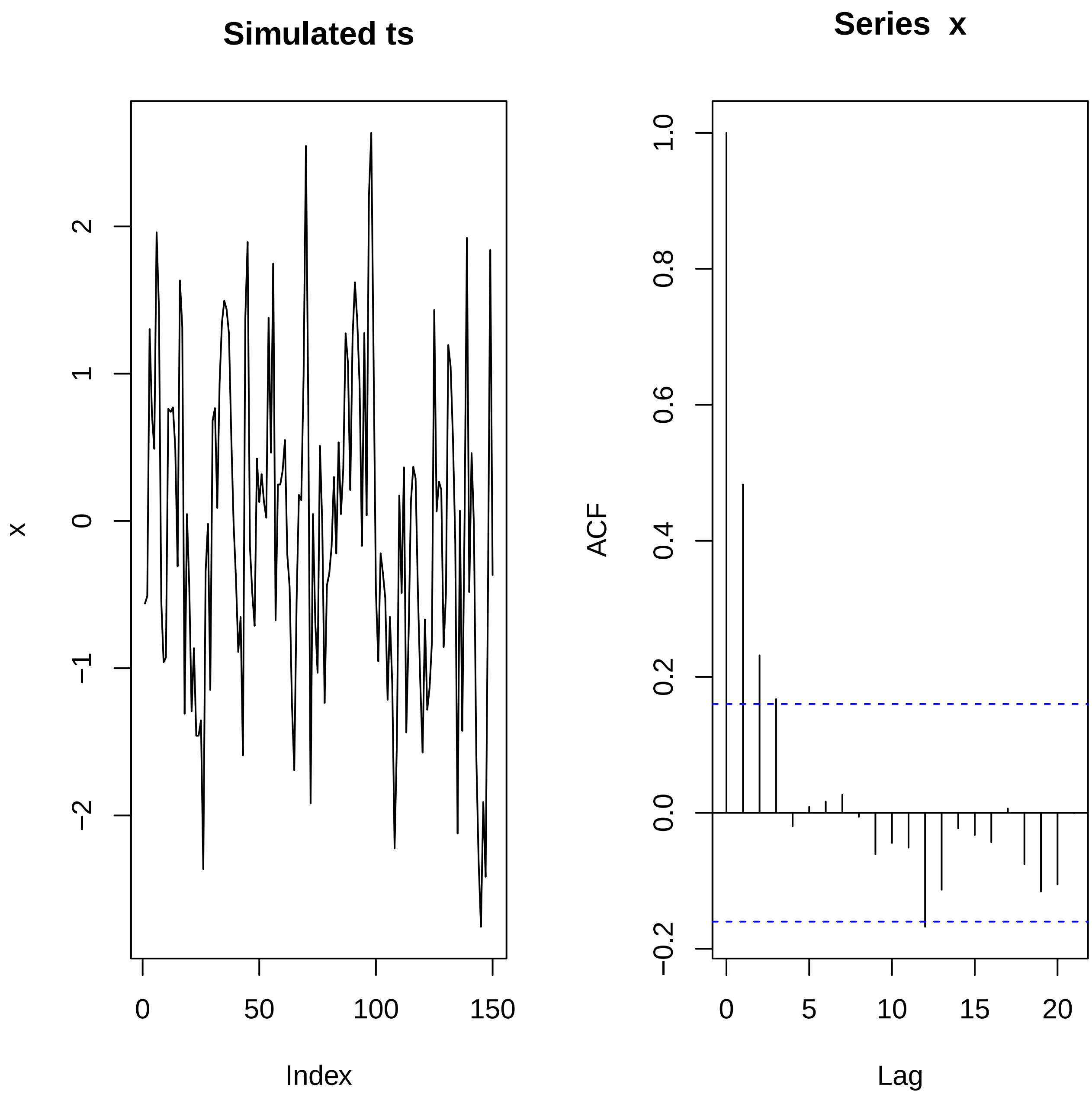


Time series

```r
beta.real = 0.5

set.seed(123)
x = c(rnorm(1))

for(i in 2:150) x = c(x, x[i-1]*beta.real + rnorm(1))

(model = arima(x, order=c(1,0,0), include.mean=F))
```

```

Call:
arima(x = x, order = c(1, 0, 0), include.mean = F)

Coefficients:
         ar1
      0.4816
s.e.  0.0711

sigma^2 estimated as 0.8958:  log likelihood = -204.72,  aic = 413.44
```

```r
par(mfrow=c(1,2))

plot(x, type="l", main="Simulated ts")
acf(x)
```

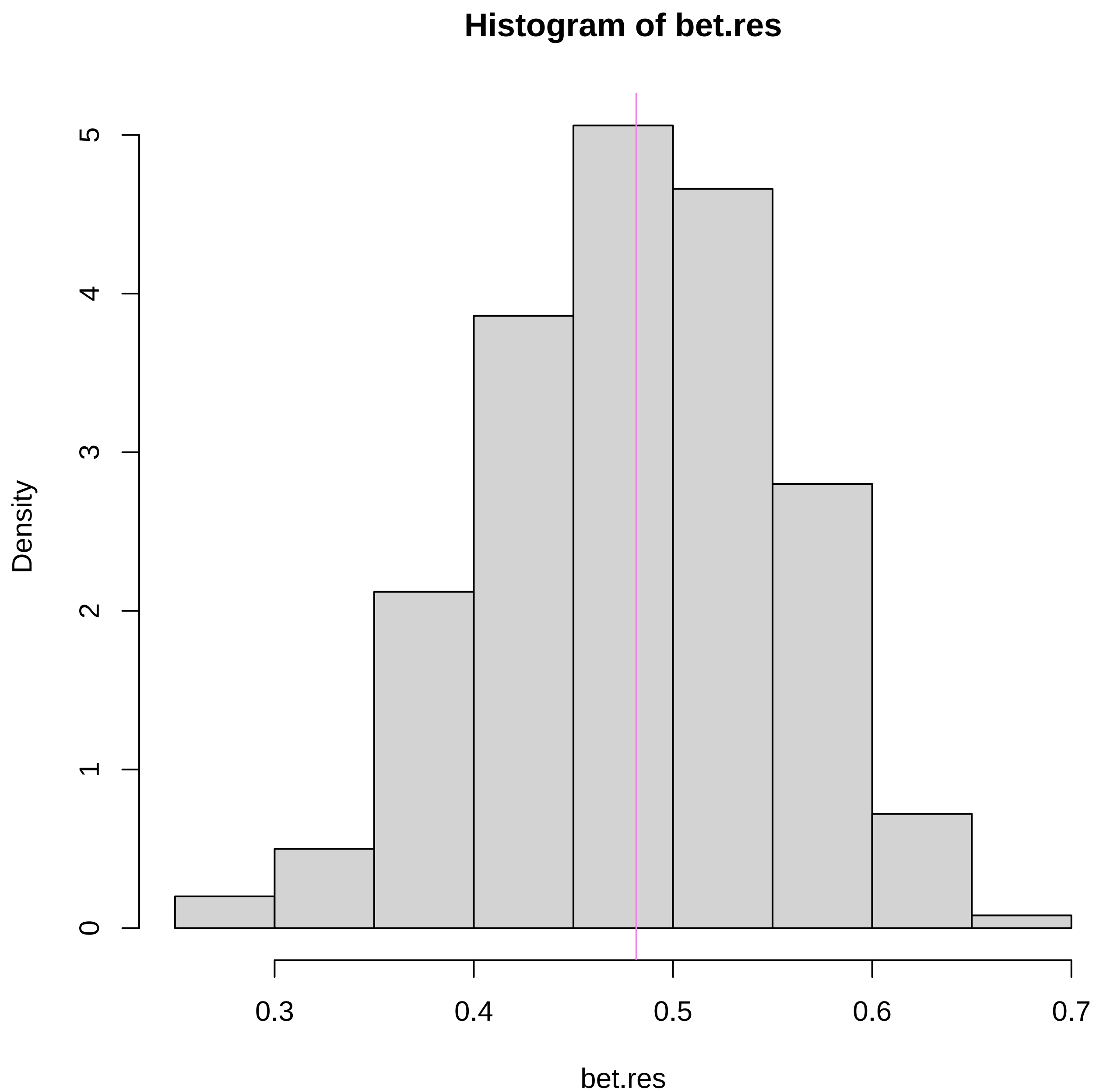



```
library(bootstrap)

ar1fit = function(x,beta){
y = arima.sim(n=150, list(ar=beta), innov=x)
return(coef(arima(y, order=c(1,0,0), include.mean=F)))
}

res = model$residuals
beta.h = coef(model)

set.seed(1)
```

```r
bet.res = bootstrap(x=res, 1000, ar1fit, beta=beta.h)$thetastar
sd(bet.res)
```

```
[1] 0.07260983
```

```r
hist(bet.res, probability=T)
abline(v=beta.h, col="violet")
```

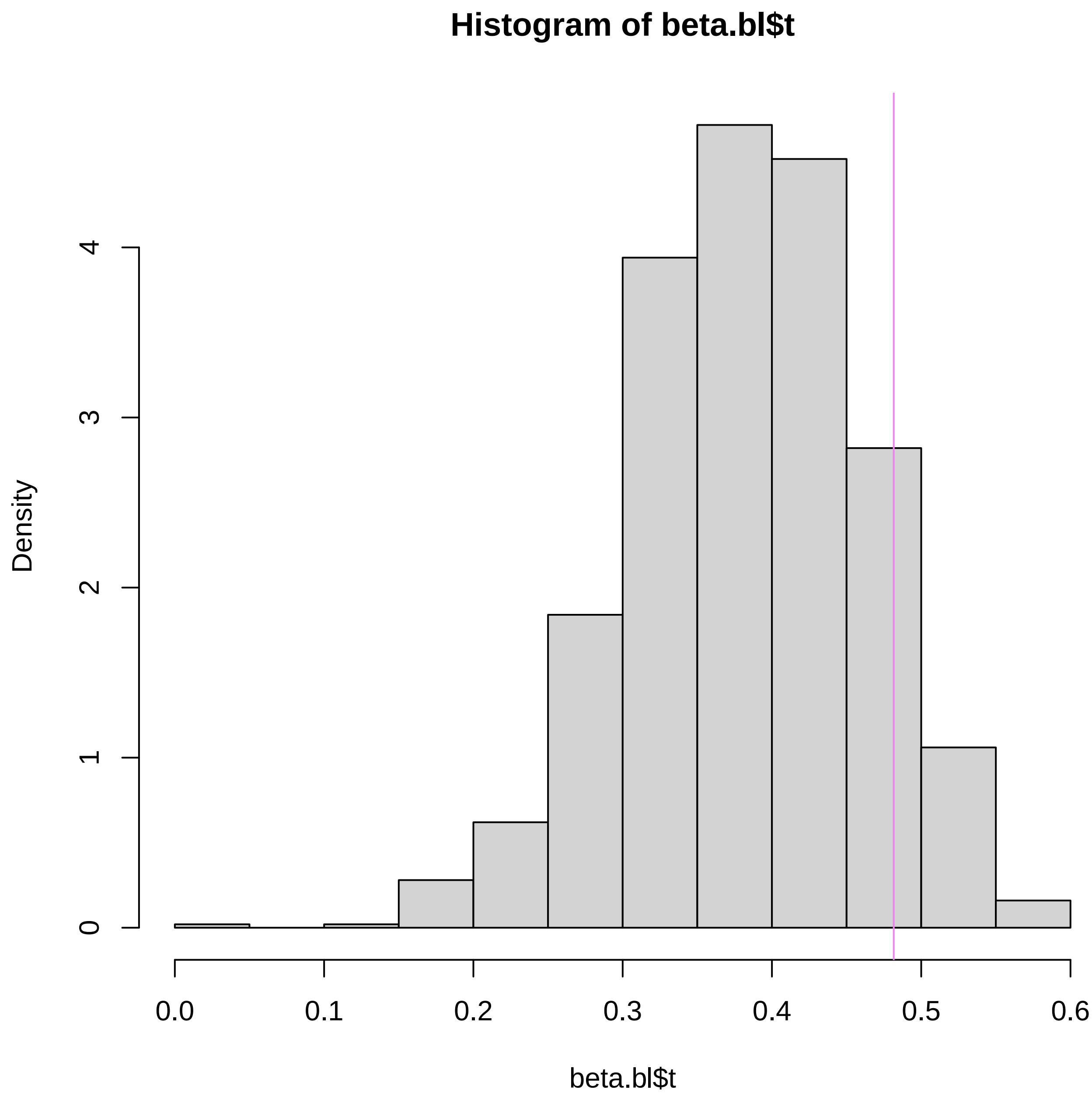



```r
library(boot)
```

```r
ar1 = function(ts) coef(arima(ts, c(1,0,0), include.mean=F))

(beta.bl = tsboot(x, statistic=ar1, R=1000, sim="fixed", l=5))
```

```
BLOCK BOOTSTRAP FOR TIME SERIES

Fixed Block Length of 5

Call:
tsboot(tseries = x, statistic = ar1, R = 1000, l = 5, sim = "fixed")


Bootstrap Statistics :
     original      bias    std. error
t1* 0.4815887 -0.09785202  0.07804321
```

```r
hist(beta.bl$t, probability=T)
abline(v=beta.h, col="violet")
```


Histogram of beta.bl$t
Density
0
1
2
3
4
0.0
0.1
0.2
0.3
0.4
0.5
0.6
beta.bl$t

Confidence intervals

By assuming an exponential distribution, the exact confidence interval for the parameter (or for the mean) can be calculated. See:

en.wikipedia.org/wiki/Exponential_distribution

$$IC_{\frac{1}{\lambda}} = \left[\frac{\bar{x}\cdot 2n}{\chi^2_{2n,1-\frac{\alpha}{2}}};\ \frac{\bar{x}\cdot 2n}{\chi^2_{2n,\frac{\alpha}{2}}}\right]$$

```
library(bootstrap)

data(aircondit,package="boot")
m = mean(aircondit$hours)
s = sd(aircondit$hours)
n = length(aircondit$hours)

# Exact CI (exponential)
c(m*2*n/qchisq(0.975, df=2*n), m*2*n/qchisq(0.025, df=2*n))
```

```
[1]  65.89765 209.17415
```

```
# Asymptotic CI
m + s/sqrt(n)*qnorm(0.975)*c(-1,1)
```

```
[1]  31.00421 185.16246
```

```
m + s/sqrt(n)*qt(0.975,df=n-1)*c(-1,1)
```

```
[1]  21.52561 194.64105
```

```
set.seed(1)

meanstar = bootstrap(x=aircondit$hours, theta = mean, nboot=1000)$thetastar
seB = sd(meanstar)

# CI with bootstrap estimate of se
m + seB*qnorm(0.975)*c(-1,1)
```

```
[1]  33.69673 182.46993
```

```
m + seB*qt(0.975,df=n-1)*c(-1,1)
```

```
[1]  24.54925 191.61742
```

```r
# Percetile bootstrap interval
quantile(meanstar, c(0.025,0.975))
```

```
     2.5%     97.5%
 48.48958 190.28958
```

```r
# Bootstrap BCa
set.seed(1)

bcanon(x=aircondit$hours, nboot=1000, theta=mean,
alpha=c(0.025,0.975))$confpoints
```

```
     alpha bca point
[1,] 0.025     57.25
[2,] 0.975    238.50
```

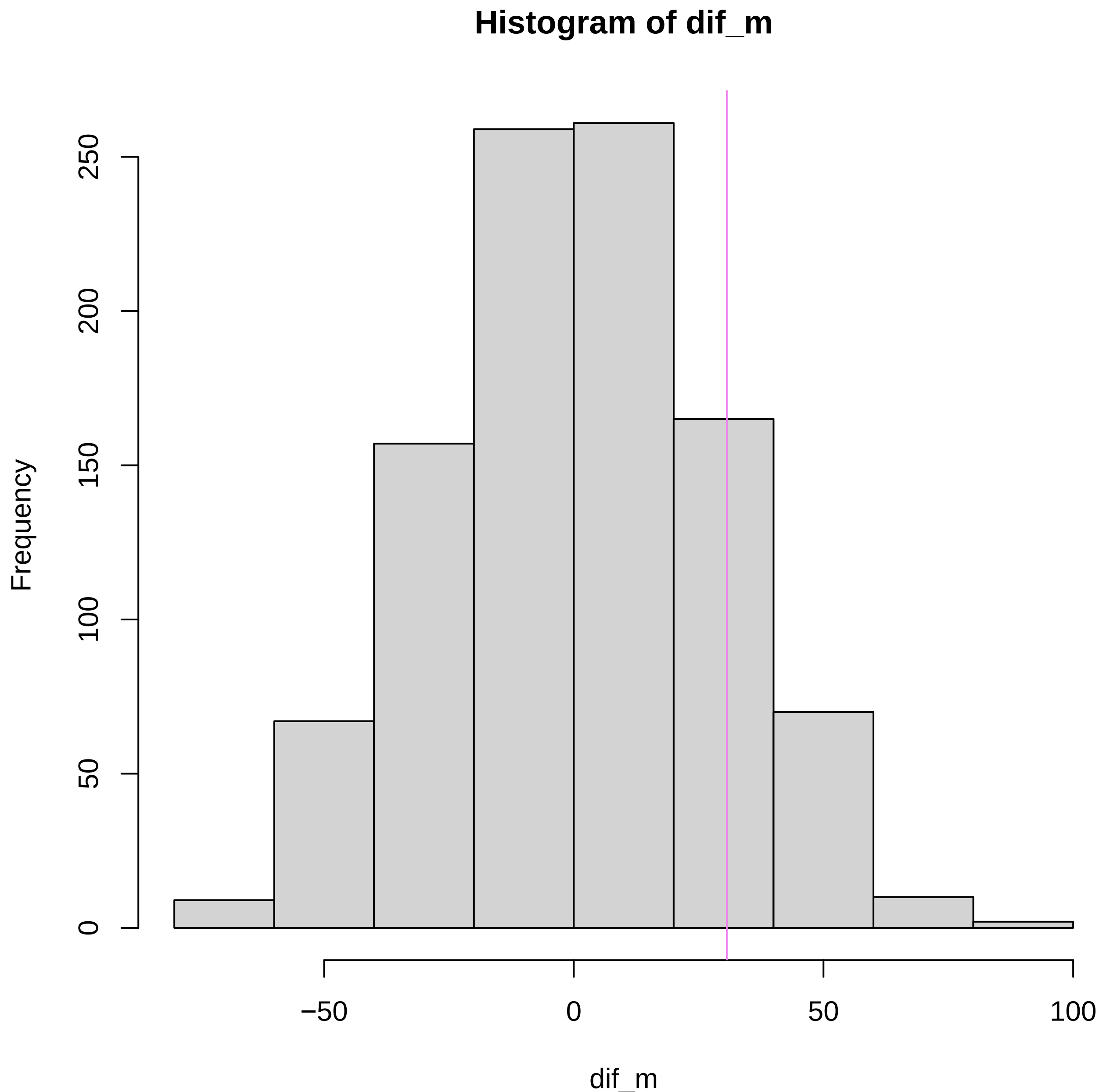
```r
library(bootstrap)

(dif_real = mean(mouse.t) - mean(mouse.c))
```

```
[1] 30.63492
```

```r
nt = length(mouse.t)
nc = length(mouse.c)
nperm = 1000
set.seed(123)

dif_m = vector(length=nperm)

for(i in 1:nperm){
  samp = sample(c(mouse.t, mouse.c), replace=F)
  dif_m[i] = mean(samp[1:nt]) - mean(samp[(nt+1):(nt+nc)])
}

(ASL_perm = sum(dif_m > dif_real)/nperm)
```

```
[1] 0.144
```

```r
hist(dif_m)
abline(v=dif_real, col="violet")
```




```
t.test(mouse.t,mouse.c, alternative="greater")$p.value
```

```
[1] 0.1576861
```

One-sample randomization test

Let us test whether the median of the air-conditioning dataset is $m_0 = 6$.

There are two observations below 6 out of a total of $n = 12$ observations.

Under H_0, the distribution of `X = number of observations below 6` is $Binomial(n = 12, p = 0.5)$ and $P(X \leq 2) = 0.019$.

The p-value for the above test (probability of taking a sample at random with less than 2, or more than 10, observations below the median) will be thus 0.038.

```
data(aircondit, package="boot")

n = length(aircondit$hours)
(real.6 = sum(aircondit$hours < 6))
```

```
[1] 2
```

```
counter = 0
set.seed(1)

for(i in 1:10000){
  counter = counter + (sum(sample(c(0,1), n, replace=T)) <= real.6)
}

2*counter/10000
```

```
[1] 0.0368
```

The two-sample bootstrap test statistic

```
library(bootstrap)

(dif.real = mean(mouse.t) - mean(mouse.c))
```

```
[1] 30.63492
```

```
nt = length(mouse.t)
nc = length(mouse.c)
nboot = 1000
set.seed(123)

dif.m = vector(length=nboot)

for(i in 1:nboot){
   samp = sample(c(mouse.t, mouse.c),replace=T)
   dif.m[i] = mean(samp[1:nt]) - mean(samp[(nt+1):(nt+nc)])
}

(ASLboot = sum(dif.m > dif.real)/nboot)
```

```
[1] 0.127
```

```
nboot = 1000
set.seed(123)

t.hat = dif.real/(sd(c(mouse.t,mouse.c))*sqrt(1/nc+1/nt))

vt = vector(length=nboot)

for(i in 1:nboot){
   samp = sample(c(mouse.t,mouse.c),replace=T)
   dif.b = mean(samp[1:nt])-mean(samp[(nt+1):(nt+nc)])
   vt[i] = dif.b/(sd(samp)*sqrt(1/nc+1/nt))
}

(ASLt = sum(vt > t.hat)/nboot)
```

```
[1] 0.139
```

One-sample tests

```
data(aircondit)

library(MKinfer)

# Test with Ho: mu = 100
boot.t.test(aircondit[["hours"]], mu=100)
```

```
    Bootstrap One Sample t-test

data:  aircondit[["hours"]]
number of bootstrap samples:  9999
bootstrap p-value = 0.7631
bootstrap mean of x (SE) = 107.7556 (34.9144)
95 percent bootstrap percentile confidence interval:
  46.58333 189.00000

Results without bootstrap:
t = 0.20554, df = 11, p-value = 0.8409
alternative hypothesis: true mean is not equal to 100
95 percent confidence interval:
  21.52561 194.64105
sample estimates:
mean of x
 108.0833
```

```r
unaMuestraBootpvalor = function(x, mu0, R = 1000){
# x: observed data
# mu0: Mean under null hypothesis

# statistic test
tstat = function(d, i, mu0) {
sqrt(length(i)) * abs(mean(d[i]) - mu0) / sd(d[i])
}

# test statistic for observed data x
t0 = tstat(x, 1:length(x), mu0)

# R resampled test statistics where
# mu0 = mean(x) passed to tstat function

bt = boot::boot(x, tstat, R = R, mu0 = mean(x))$t[,1]

# The p-value is obtained
c(pvalue = mean(bt > t0))
}
```

The function is applied to a set of data in which the null hypothesis is true:

```r
set.seed(666)

# H0 is correct
x = rexp(100, rate = 1/17)
unaMuestraBootpvalor(x, 17)
```

```
pvalue
  0.11
```

Two-sample tests

```r
DosMuestrasBootpvalor = function(x, y, alternativa = c("bilateral", "less_than", "greater_than"),
R = 2000){

# x: observed data (first sample)
# y: observed data (second sample)
# alternative - specifies the alternative hypothesis

alternativa = match.arg(alternativa)
n1 = length(x)
n2 = length(y)
```

```r
# test statistics
tstat = function(d, i){
boot.xy = d[i]
x = boot.xy[1:n1]
y = boot.xy[-(1:n1)]
s = sqrt( ((n1-1) * var(x) + (n2 - 1) * var(y)) / (n1 + n2 - 2) )
mu.x = mean(x)
mu.y = mean(y)
(mu.x - mu.y) / s / sqrt(1 / n1 + 1 / n2)
}

xy = c(x,y)

# Test statistics for the observed data
t0 = tstat(xy, 1:(n1 + n2))

# Resampling of the test statistic
bt = boot::boot(xy, tstat, R = R)$t[,1]

# p-value
if(alternativa == "greater_than") return(c(pvalue = mean(bt > t0)))
if(alternativa == "less_than") return(c(pvalue = mean(bt < t0)))
c(pvalue = mean(abs(bt) > abs(t0)))
}
```

The function is applied to a set of data in which the null hypothesis is true:

```r
# H0 is correct, mu_x is equal to mu_y
set.seed(666)
datos1 = rnorm(10, mean = 3, sd = 2)
datos2 = rnorm(20, mean = 3, sd = 2)

DosMuestrasBootpvalor(datos1, datos2, alternativa = "greater_than")
```

```
pvalue
 0.313
```

```r
# library(MKinfer)

boot.t.test(datos1, datos2)
```

```
    Bootstrap Welch Two Sample t-test

data:  datos1 and datos2
number of bootstrap samples:  9999
bootstrap p-value = 0.6675
bootstrap difference of means (SE) = 0.4797342 (1.008988)
95 percent bootstrap percentile confidence interval:
```

```
 -1.537767  2.465471

Results without bootstrap:
t = 0.44627, df = 14.379, p-value = 0.662
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.804340  2.755463
sample estimates:
mean of x mean of y
 2.807849  2.332287
```

It can be compared with a permutation test:

```r
# library(MKinfer)

perm.t.test(datos1, datos2)
```

```
    Permutation Welch Two Sample t-test

data:  datos1 and datos2
number of permutations:  9999
(Monte-Carlo) permutation p-value = 0.6545
permutation difference of means (SE) = 0.4712101 (0.9550028)
95 percent (Monte-Carlo) permutation percentile confidence interval:
 -1.349719  2.308100

Results without permutation:
t = 0.44627, df = 14.379, p-value = 0.662
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.804340  2.755463
sample estimates:
mean of x mean of y
 2.807849  2.332287
```