Resampling methods — 2019

Theodor Hosemann

Oliver Kirchkamp

Contents

1 Introduction

Homepage: https://www.kirchkamp.de/

Literature:

- Bradley Efron and Robert J. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, 1994.
- A. C. Davison, D. V. Hinkley, Bootstrap Methods and their Application, Cambridge University Press, 1997.

Aim of the course In this course we will discuss resampling methods like the bootstrap, jackknife, or permutation tests. Resampling methods provide solutions to a couple of interesting problems without making specific distributional assumptions.

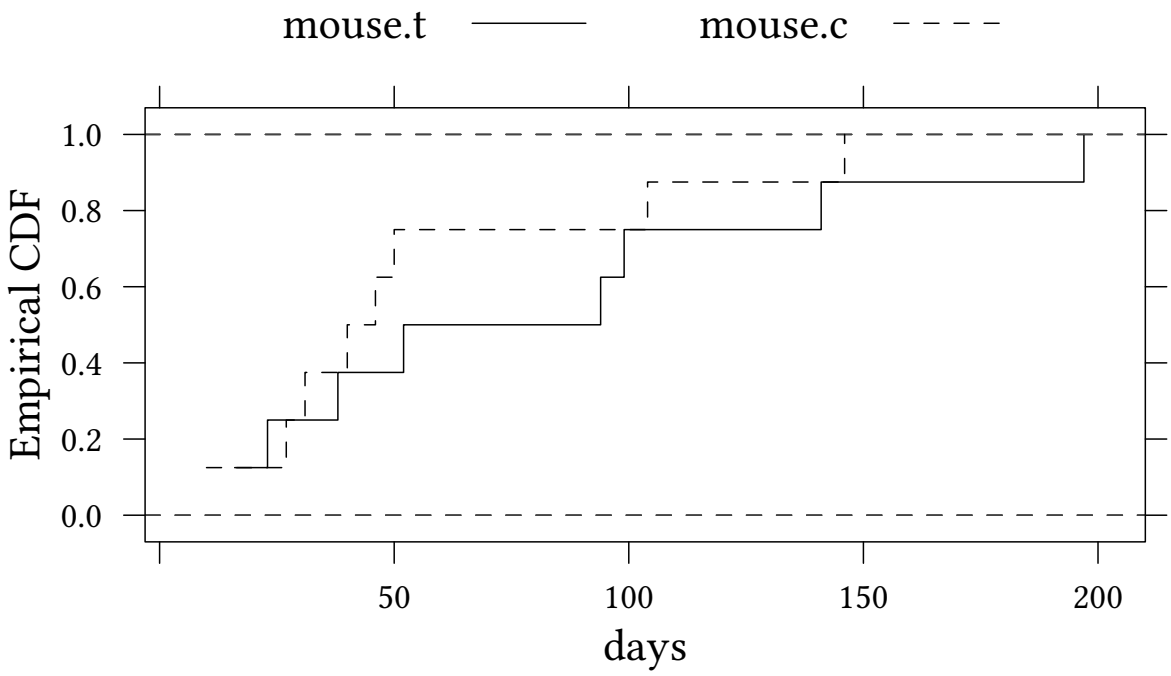
Purpose of this handout In this handout you find all slides from the lecture (in a more printer friendly version). You also find (most of) the examples in R I plan to use in the lecture.

1.1 Motivation

Let us start with a simple example: a comparison of two treatments:

```
library(latticeExtra)
library(bootstrap)
library(boot)
mouse.c

[1]  52 104 146  10  50  31  40  27  46

mouse.t

[1]  94 197  16  38  99 141  23
```

```
ecdfplot( ~ mouse.t + mouse.c,auto.key=list(columns=2),xlab="days")
```



Treated mice have a larger mean survival time:

```
mean(mouse.t)

[1] 86.85714

mean(mouse.c)

[1] 56.22222
```

Question: Is the difference significant?
Traditionally we estimate

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

and hence

$$\hat{\sigma}_{\bar{x}} = \sqrt{\frac{\hat{s}_x^2}{n}}$$

The estimated standard error of the difference between the two means $\bar{x}$ and $\bar{y}$ is

$$\hat{\sigma}_{\bar{x}-\bar{y}} = \sqrt{\frac{\hat{s}_x^2}{n} + \frac{\hat{s}_y^2}{m}}$$

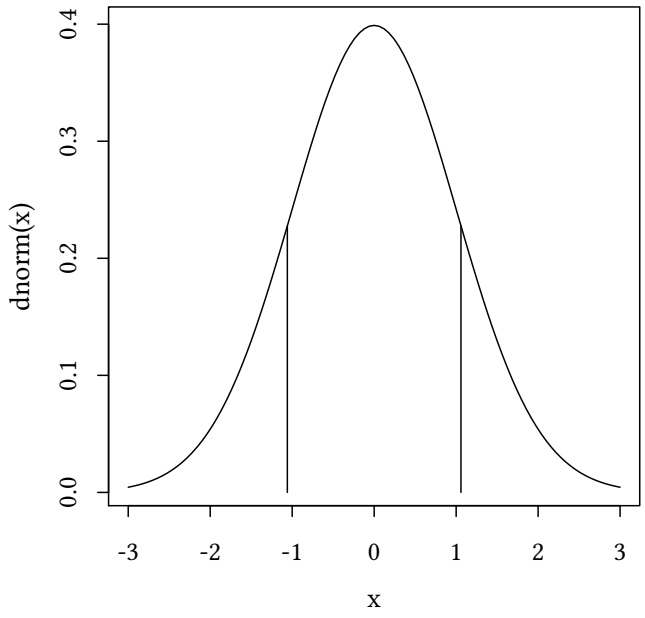
```
(sdDiff <- sqrt(var(mouse.t)/length(mouse.t)+var(mouse.c)/length(mouse.c)))

[1] 28.92647
```

Hence $t = \frac{\bar{x}-\bar{y}}{\sigma_{\bar{x}-\bar{y}}}$ is

```
(t <- (mean(mouse.t) - mean(mouse.c)) / sdDiff )

[1] 1.059062
```

Is this is large number?

```
t

[1] 1.059062

pnorm(t)

[1] 0.8552142

1-pnorm(t)

[1] 0.1447858

2*(1-pnorm(t))

[1] 0.2895715
```



Why are we allowed to do this?

Central limit theorem

Be $X_1, \ldots, X_n$ a sequence of independently and identically (i.i.d.) distributed random variables, then with expected value $E(X)$ and variance σ_X^2, then

$$\lim_{n\to\infty} \frac{\sum_{i=1}^n X_i - nE(X)}{\sigma_X \sqrt{n}} = \lim_{n\to\infty} \left(\frac{\sum_{i=1}^n X_i}{n} - E(X) \right) \cdot \frac{\sqrt{n}}{\sigma_X} \sim N(0,1)\,.$$

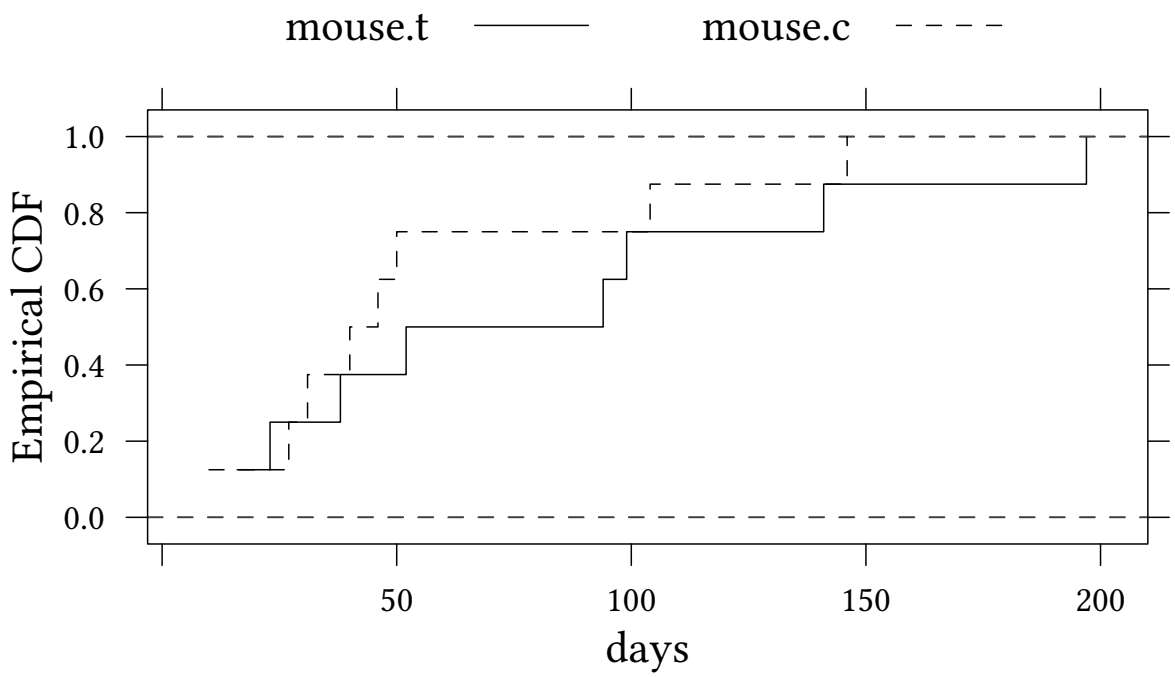
When do we have a problem?

- When n is small.
- When we are interested in a statistic other than the *mean*.

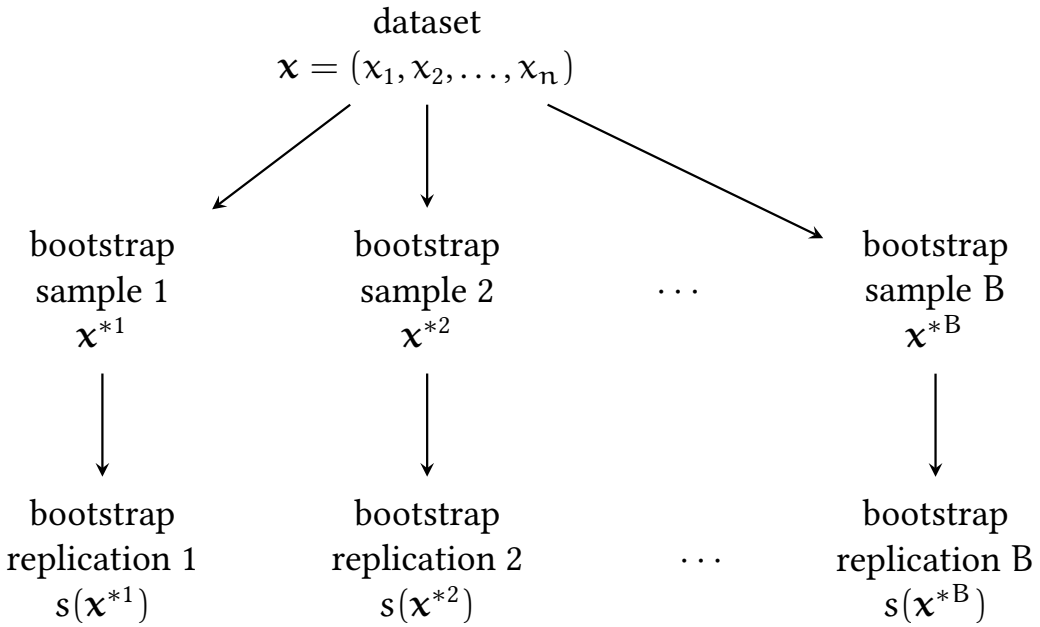
Example: Let us assume we are interested in the median:

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

[1] 30.63492

median(mouse.t) - median(mouse.c)

[1] 48
```

- In this example the difference in medians is larger than the difference in means. But what is the standard error of the difference in medians?
- Formulae for the standard error of medians require a specific distribution of X.
- Do we know the distribution of X? — No, but we can estimate it!
- Above we have estimated $E(X)$ by using our sample to calculate the statistic: $\bar{x} = \widehat{E(X)}$
- Now we use our sample as an estimation of the distribution of X.



Let us assume that we are interested in the distribution of the statistic s. Our *estimation of the distribution* of X is given by the *distribution of the dataset* $\mathbf{x}$.



Then our *estimate of the distribution of* s is given by the *distribution of the bootstrap replications* $s(\mathbf{x}^{*1}), s(\mathbf{x}^{*2}), \ldots, s(\mathbf{x}^{*B})$.

Our *estimate of the standard error of* $s(\mathbf{x})$ is given by the *standard deviation of the boostrap replications* $s(\mathbf{x}^{*1}), s(\mathbf{x}^{*2}), \ldots, s(\mathbf{x}^{*B})$.

The standard formula for $\hat{\sigma}_X$:

$$\hat{\sigma}_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

The BS estimate of the standard error of $s(\mathbf{x})$

$$\hat{\sigma}_{s,BS} = se^{B}_{b=1}(s(x^{*b})) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(s(x^{*b}) - \frac{1}{B}\sum_{b=1}^{B} s(x^{*b})\right)^2}$$

Remember:

$$\hat{\sigma}_{\bar{x}-\bar{y}} = \sqrt{\frac{\hat{s}_x^2}{n} + \frac{\hat{s}_y^2}{m}} = 28.93$$

```
set.seed(123)
```

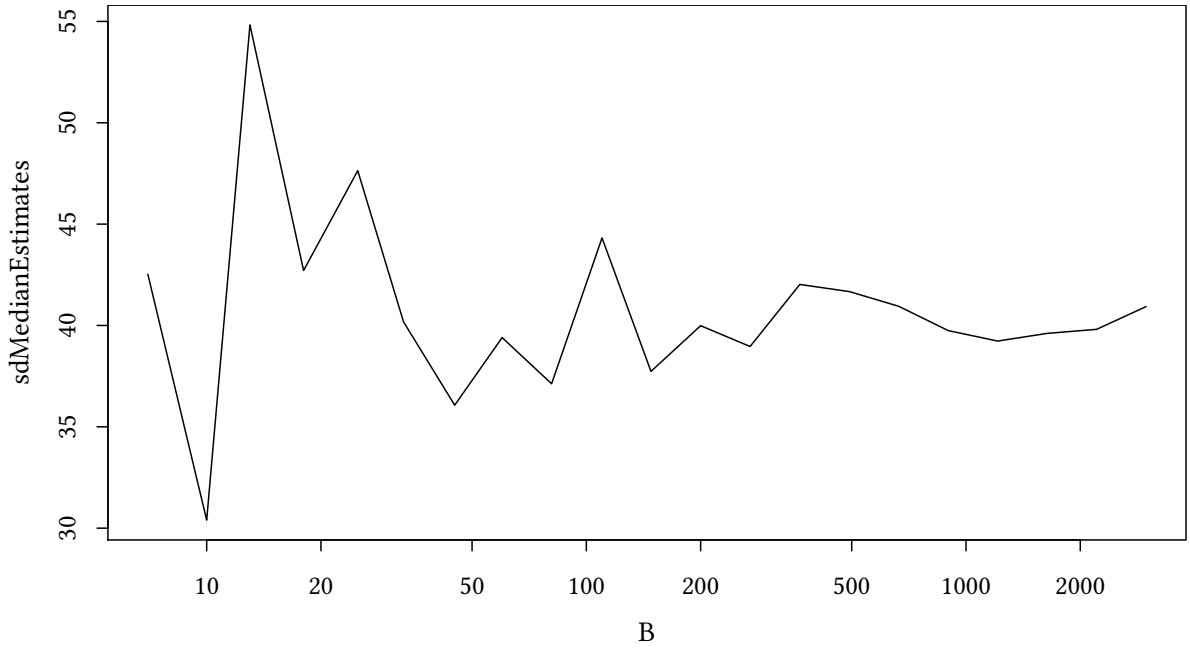
```
(sdDiff2 <- sd(replicate(100,mean(sample(mouse.t,replace=TRUE))-
                             mean(sample(mouse.c,replace=TRUE)))))

[1] 27.57955

(sdMedianDiff2 <- sd(replicate(100,median(sample(mouse.t,replace=TRUE))-
                                 median(sample(mouse.c,replace=TRUE)))))

[1] 39.22875
```

Is 100 large enough?

Try different sizes of the bootstrap:

```
sdMedian <- function(B) sd(replicate(B,median(sample(mouse.t,replace=TRUE))-
                                       median(sample(mouse.c,replace=TRUE))))
set.seed(123)
(B<-round(exp(seq(2,8,.3))))

 [1]    7   10   13   18   25   33   45   60   81  110  148  200  270  365  493
[16]  665  898 1212 1636 2208 2981

sdMedianEstimates <- sapply(B,sdMedian)
```

```
plot(sdMedianEstimates ~ B,log="x",t="l")
```



- A number of 50…200 bootstraps is usually sufficient to estimate standard errors.

→ we increase computational complexity by a factor of 50…200.

→ we do not have to worry about "unusual" statistics or distributions.

We replace complicated derivations based on specific distributions with computing power.

Exercise 1.1 *The dataset* `x1.csv` *contains two variables,* `x1` *and* `x2`*.*

- *What is the interquartile range of* `x1`*?*
- *What is the standard deviation of your estimate?*
- *Draw a graph with the number of bootstrap replications on the horizontal axis and the estimated standard deviation on the vertical axis. How quick does your estimation converge?*

2 Parameters, distributions and the plug-in principle

universe $\mathcal{U}$	U_1	U_2	$\cdots$	U_N	$\leftarrow$ units
population $\mathcal{X}$ (measurements of $\mathcal{U}$)	X_1	X_2	$\cdots$	X_N	population distribution F
random sample $\mathbf{x}$		$x_1, x_2, \cdots, x_n$			empirical distribution $\hat{F}$

Note that with independence $\hat{F}$ is a *sufficient statistic* for F (we can use either $\mathbf{x}$ or $\hat{F}$ to learn something about F).

What can we learn from $\mathbf{x}$ about $\mathbf{X}$?

Parameters and distributions

F is a distribution of X iff

$$F(x,\theta,\ldots) \equiv P(X < x|\theta,\ldots)$$

We call θ a *parameter* of F.

Parameters and statistics

$$\theta = t(F)$$

$t(\cdot)$ is a *statistic* of F.

- What happens if we don't know F, but (only) $\hat{F}$?

→ If F is determined by a *small number* of parameters (i.e. we know *a lot* about F, we miss only θ, e.g. we know that F is normal) then we can use ML or MM.

? But what if F has *many* other parameters (about which we do not care)?

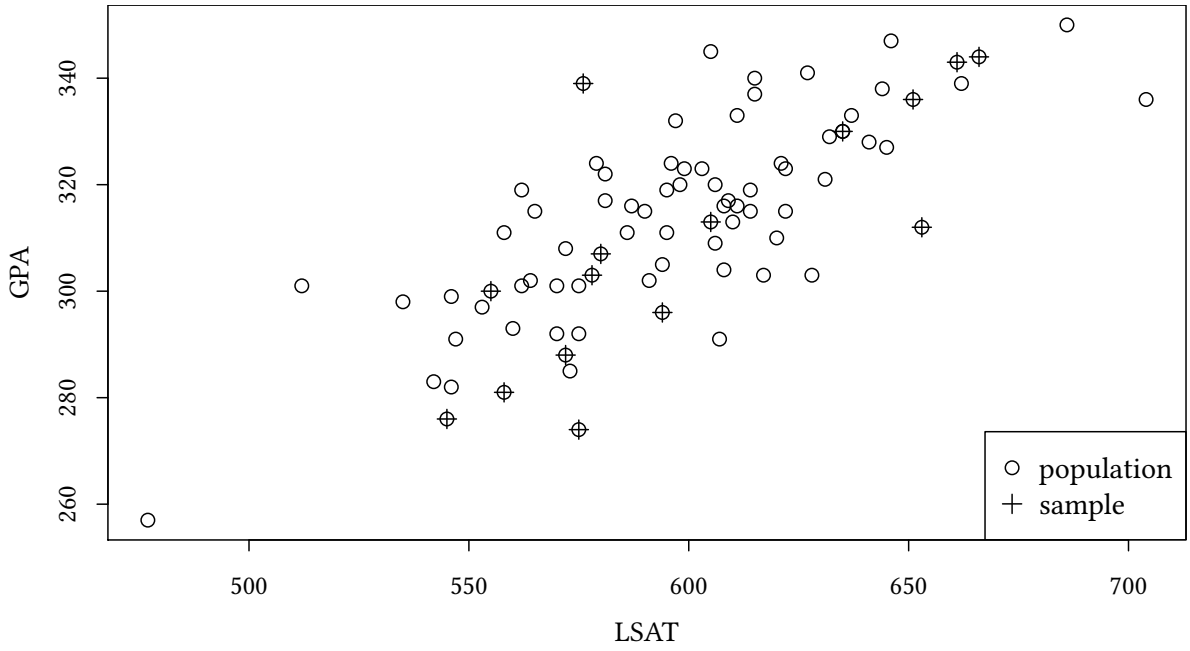
The plug-in principle

$$\hat{\theta} = t(\hat{F})$$

We (simply) apply the statistic $t(\cdot)$ to obtain an *estimator* for θ.

Example: the law school data

```
with(law82,plot(100*GPA ~ LSAT,ylab="GPA"))
points(within(law,GPA<-100*GPA),pch=3)
legend("bottomright",c("population","sample"),pch=c(1,3))
```



Our population are 82 law schools (dataset *law82*).

Our sample (`law`) only contains 15 observations.

We are interested in the *correlation* of GPA (Grade Point Average, undergraduate score) and LSAT (Law School Admission Test Score).

(i.e. the θ and the $t(\cdot)$ we are talking about is the correlation.)

The true score:

```
with(law82,cor(GPA,LSAT))

[1] 0.7599979
```

The plug-in estimate:

```
with(law,cor(GPA,LSAT))

[1] 0.7763745
```

Convergence Why does it make sense to use a plug-in estimator?

Glivenko-Cantelli's theorem (1933)

$$\sup_x |\hat{F}_n(x) - F(x)| \overset{n\to\infty}{\to} 0 \qquad \text{almost surely}$$

if $t(\hat{F})$ is continuous then

$$t(\hat{F}_n) \overset{n\to\infty}{\to} t(F) = \theta \qquad \text{almost surely}$$

We can use the bootstrap to measure…

- …the bias of the plug-in estimate
- …the standard error of the plug-in estimate

3 Estimating standard errors

3.1 The bootstrap algorithm for standard errors

- $x_1, x_2, \ldots, x_n$ are n *observed values.*
- $\hat{F}$ is the *empirical distribution* (with probability $1/n$ on each of $x_1, x_2, \ldots, x_n$).
- $\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ is a bootstrap sample, drawn from $\hat{F}$.

  ($\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ is a random sample of size n drawn with replacement from the observed values $x_1, x_2, \ldots, x_n$).
- $\hat{\theta}^* = s(\boldsymbol{x}^*)$ is a *bootstrap replication* of $\hat{\theta}$.
- $\mathrm{se}_F(\hat{\theta})$ is the standard error of $\hat{\theta}$ (we usually can not calculate $\mathrm{se}_F(\hat{\theta})$).
- $\mathrm{se}_{\hat{F}}(\hat{\theta}^*)$ is the (ideal) *bootstrap estimate* of $\mathrm{se}_F(\hat{\theta})$.

  $\mathrm{se}_{\hat{F}}(\hat{\theta}^*)$ can be hard to calculate.

  We can, however, get a good approximation of $\mathrm{se}_{\hat{F}}(\hat{\theta}^*)$.

The bootstrap algorithm for standard errors

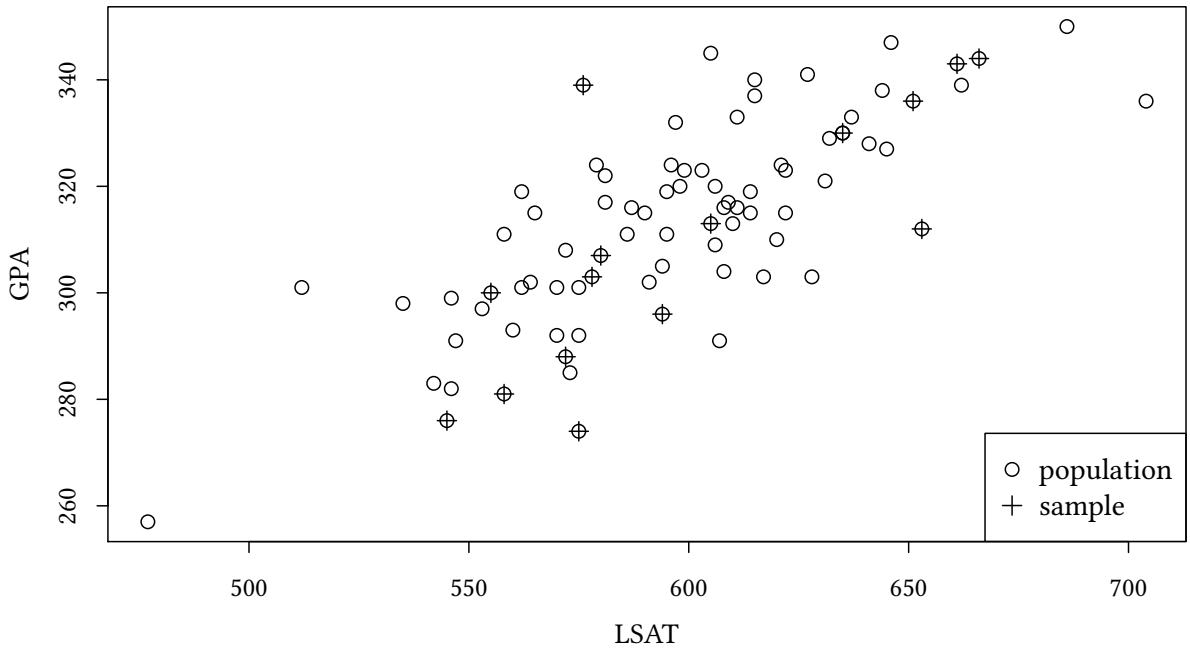
1. Draw B independent bootstrap samples $\boldsymbol{x}^{*1}, \boldsymbol{x}^{*2}, \ldots, \boldsymbol{x}^{*B}$.

   Each sample has size n and is drawn with replacement from $x_1, x_2, \ldots, x_n$.
2. For bootstrap sample $b \in \{1, 2, \ldots, B\}$ determine $\hat{\theta}^{*b} = s(\boldsymbol{x}^{*b})$.
3. $\hat{\sigma}_{s,BS} = \mathrm{se}^B_{b=1}(\hat{\theta}^{*b}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\theta}^{*b} - \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*b}\right)^2}$

$$\lim_{B\to\infty} \hat{\sigma}_{s,BS} = \underbrace{\mathrm{se}_{\hat{F}}(\hat{\theta}^*)}_{\text{ideal bootstrap}}$$

$$\hat{F} \to \left.\begin{array}{l} \boldsymbol{x}^{*1} \to s(\boldsymbol{x}^{*1}) = \hat{\theta}^{*1} \\ \boldsymbol{x}^{*2} \to s(\boldsymbol{x}^{*2}) = \hat{\theta}^{*2} \\ \vdots \\ \boldsymbol{x}^{*b} \to s(\boldsymbol{x}^{*b}) = \hat{\theta}^{*b} \\ \vdots \\ \boldsymbol{x}^{*B} \to s(\boldsymbol{x}^{*B}) = \hat{\theta}^{*B} \end{array}\right\} \to \mathrm{se}^B_{b=1}(\hat{\theta}^{*b}) = \hat{\sigma}_{s,BS}$$

3.2 The law school data again



We estimated the correlation between GPA and LSAT as follows:

```
(lawCor <- with(law,cor(GPA,LSAT)))

[1] 0.7763745
```

How precise is this estimate? If F is bivariate normal, then the estimated correlation coefficient $\hat{\rho}$ has a standard deviation of about

$$\hat{\sigma}_{\text{normal}} = \frac{1-\hat{\rho}^2}{\sqrt{n-3}} \approx 0.115$$

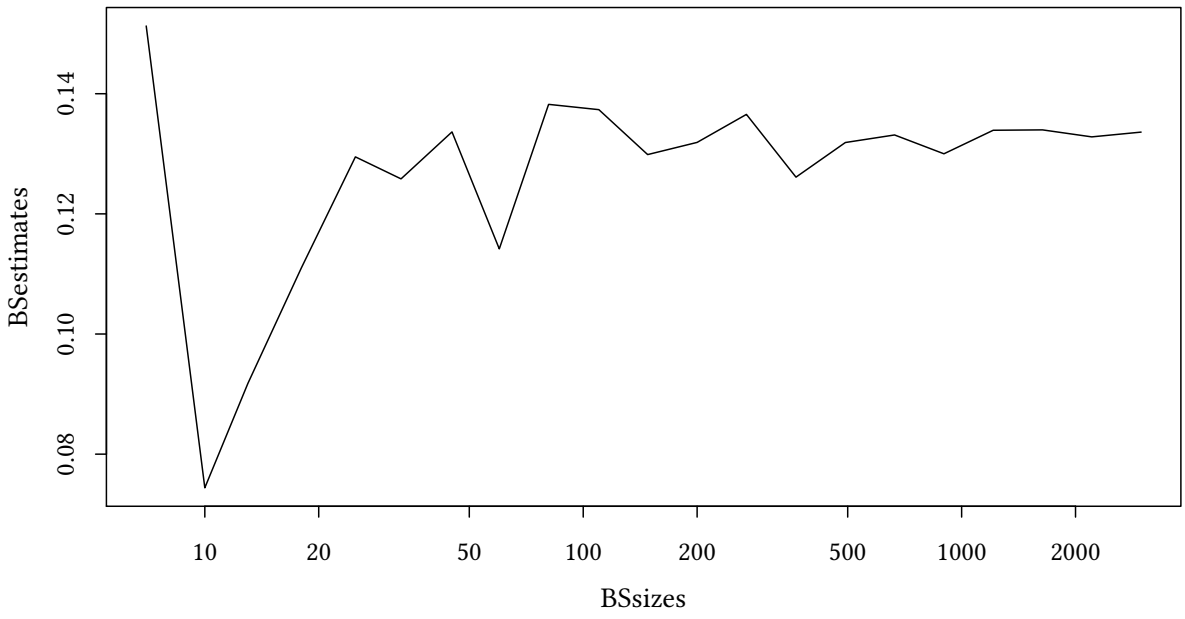
With a bootstrap we can live *without* the assumption that F is normal.

```
set.seed(123)
samplesize<-nrow(law)
ind<-1:samplesize
law.boot <- replicate(200,{i2 <- sample(ind,replace=TRUE);
                           with(law[i2,],cor(GPA,LSAT))})
sd(law.boot)

[1] 0.1200175
```

How quickly does the estimate of $\hat{\sigma}$ converge?

```
set.seed(123)
ind<-1:samplesize
lawBS <- function(B) sd(replicate(B,{i2 <- sample(ind,replace=TRUE);
                                     with(law[i2,],cor(GPA,LSAT))}))
BSsizes<-round(exp(seq(2,8,.3)))
BSestimates <- sapply(BSsizes,lawBS)
```

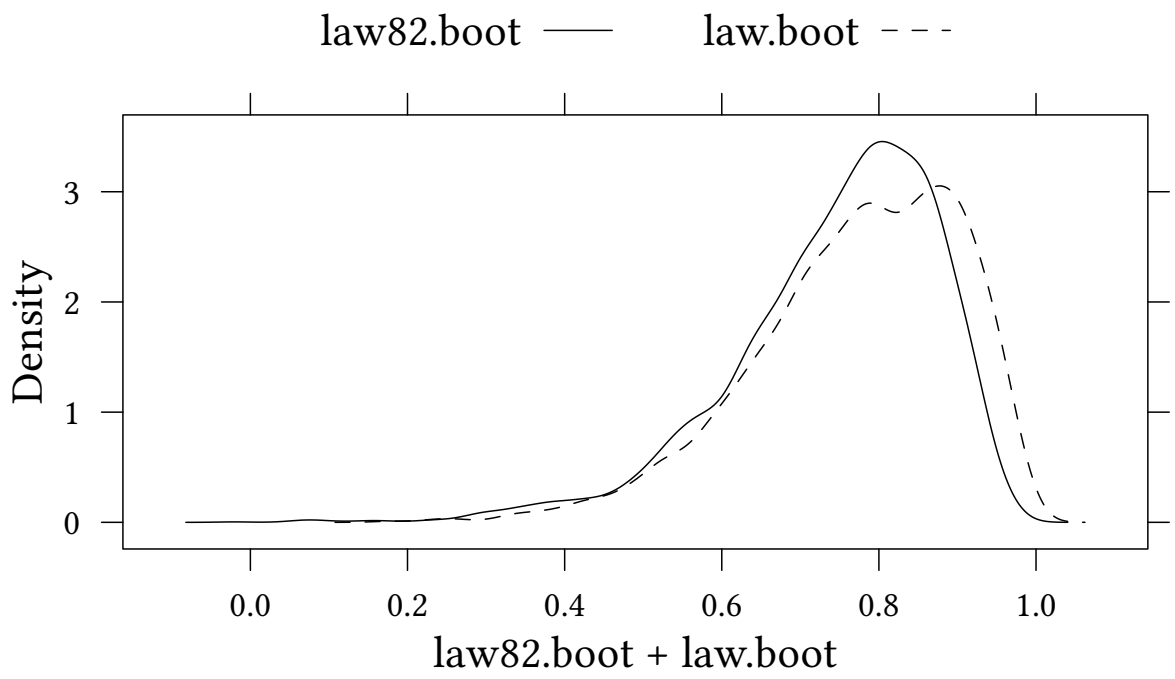
```
plot(BSestimates ~ BSsizes,log="x",t="l")
```



The distribution of $\hat{\theta}^*$ What is the distribution of the bootstrap replications $\hat{F}(\hat{\theta}^*)$ and how does this compare to $F(\hat{\theta}^*)$?

```
set.seed(123)
ind<-1:nrow(law)
law.boot <- replicate(5000,{i2 <- sample(ind,size=samplesize,replace=TRUE);
                            with(law[i2,],cor(GPA,LSAT))})
ind<-1:nrow(law82)
law82.boot <- replicate(5000,{i2 <- sample(ind,size=samplesize,replace=TRUE);
                              with(law82[i2,],cor(GPA,LSAT))})
```

```
densityplot(~ law82.boot + law.boot,plot.points=FALSE,
            auto.key=list(columns=2,size=3,between=1))
```



Comparison of the bootstrap estimate $\widehat{se}_{\hat{F}}(\hat{\sigma})$ with the standard error $se_F(\hat{\sigma})$:

```
sd(law.boot)

[1] 0.1323102

sd(law82.boot)

[1] 0.1317328
```

Exercise 3.1 *Look again at the dataset* `x1.csv`.

- *What is the average value of the ratio* `x1/x2`.
- *What is the standard deviation we would obtain with the standard formula?*
- *Use a bootstrap to estimate the standard deviation.*
- *How does the standard deviation converge? Show a graph!*

3.3 How many bootstrap samples do we need?

Coefficient of variation

$$cv_X := \frac{\sigma_X}{|\mu_X|}$$

$$cv(\widehat{se}_B) \doteq \sqrt{cv(\widehat{se}_\infty)^2 + \frac{1}{4B}(E(\Delta) + 2)}$$

- Δ is a measure for long-tailedness of the distribution of $\hat{\theta}^*$. For the normal distribution $\Delta = 0$ and usually $\Delta < 10$.

- The magnitude of $\text{cv}(\widehat{\text{se}}_\infty)$ depends on
  - $\hat{F}$,
  - the statistic $s(\cdot)$,
  - and the sample size.

3.4 The parametric bootstrap

Now we assume that we know a lot about F, e.g. that F follows a (multivariate) normal distribution.

- $x_1, x_2, \ldots, x_n$ are n *observed values.*

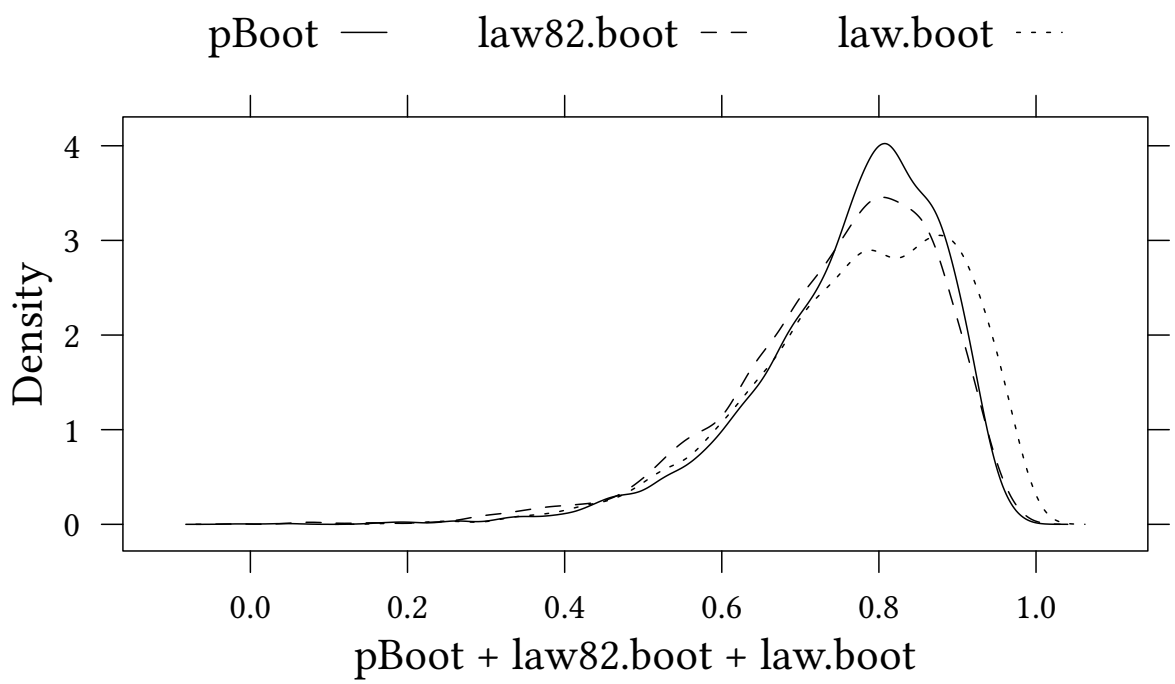
- estimate the parameters of F (e.g. mean, variance, covariance in the case of a normal distribution) and obtain F_{par}.

- Apply the bootstrap algorithm for standard errors to F_{par}.

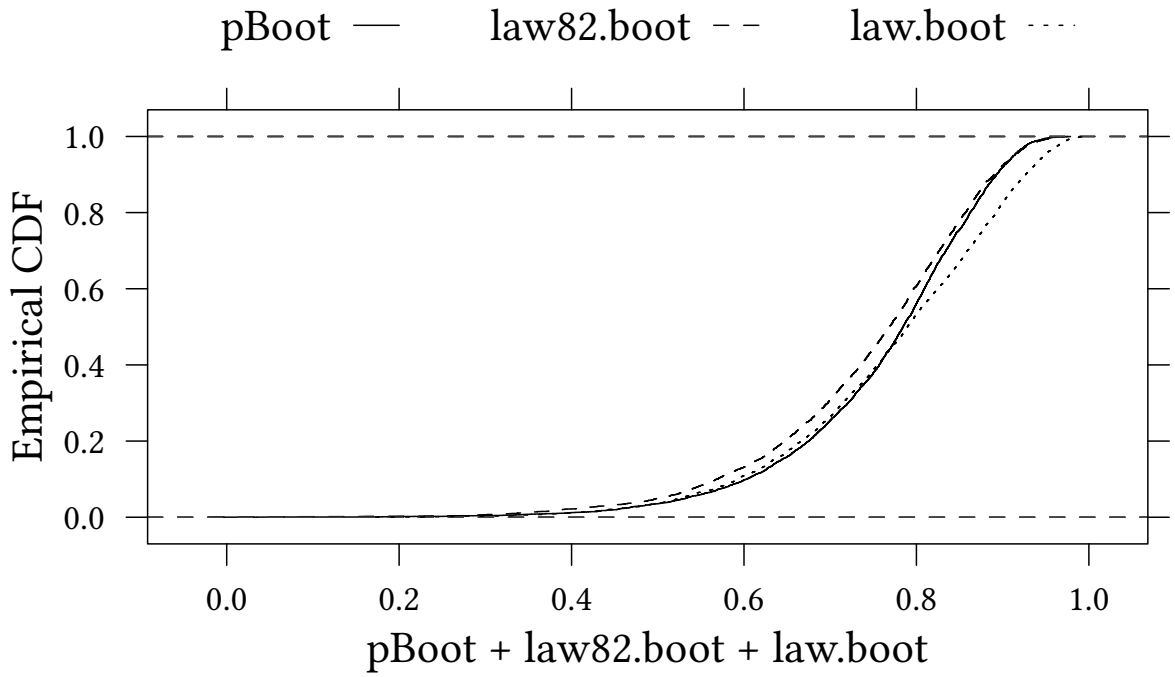
```
library(mvtnorm)
paraBoot <- function(XY,n=nrow(XY))  {
  sigma <- cov(XY)
  mu <- rapply(XY,mean)
  rmvnorm(n,mu,sigma)
}
set.seed(123)
pBoot <- replicate(5000,cor(paraBoot(law))[2,1])
```

Example: The law school data again

```
densityplot(~pBoot+law82.boot + law.boot,plot.points=FALSE,
            auto.key=list(columns=3,size=2,between=1))
```




```
ecdfplot(~pBoot+law82.boot + law.boot,plot.points=FALSE,
            auto.key=list(columns=3,size=2,between=1))
```



And what is the parametric bootstrap estimation of the standard deviation?

```
sd(pBoot)

[1] 0.1203052
```

Exercise 3.2 *Return to exercise 3.1. What result would you obtain with a parametric bootstrap?*

3.5 Eigenvalues und Eigenvectors

(more sampling from higher dimensional data)

- Variance-covariance matrices Σ are symmetric positive semidefinite.
- Eigenvalue decomposition of $\Sigma \rightarrow$ principal components of Σ
- Eigenvectors$\leftrightarrow$principal component
  Eigenvalue$\leftrightarrow$variance explained by the principal component
- similar to factor analysis

Let us construct a 100×4 matrix which we interpret as 100 observations of 4 different variables.

```
set.seed(123)
N <- 100
latent <- 10*runif(N)
latent2 <- 10*runif(N)
sd <- 1
x1 <- latent + sd * rnorm(N)
x2 <- latent + latent2 + sd * rnorm(N)
x3 <- - latent2 + sd * rnorm(N)
x4 <- latent2 - 2 * latent + sd * rnorm(N)
X  <- cbind(x1,x2,x3,x4)
```

Let us assume that we do not know the underlying structure and that we use eigenvalues to learn more about the structure:

```
eigen(cov(X))

eigen() decomposition
$values
[1] 53.212 18.772  0.956  0.853

$vectors
       [,1]   [,2]  [,3]   [,4]
[1,] -0.388  0.160 0.242  0.875
[2,] -0.292  0.737 0.465 -0.393
[3,] -0.107 -0.609 0.772 -0.150
[4,]  0.867  0.245 0.360  0.241
```

```
eigen(cov(X))[["values"]]

[1] 53.2123867 18.7720580  0.9558129  0.8527676
```

We find the first two eigenvalues fairly large (indeed, we constructed the data with two main latent variables in our mind), and the last two relatively small. Does that mean that they are not significant?

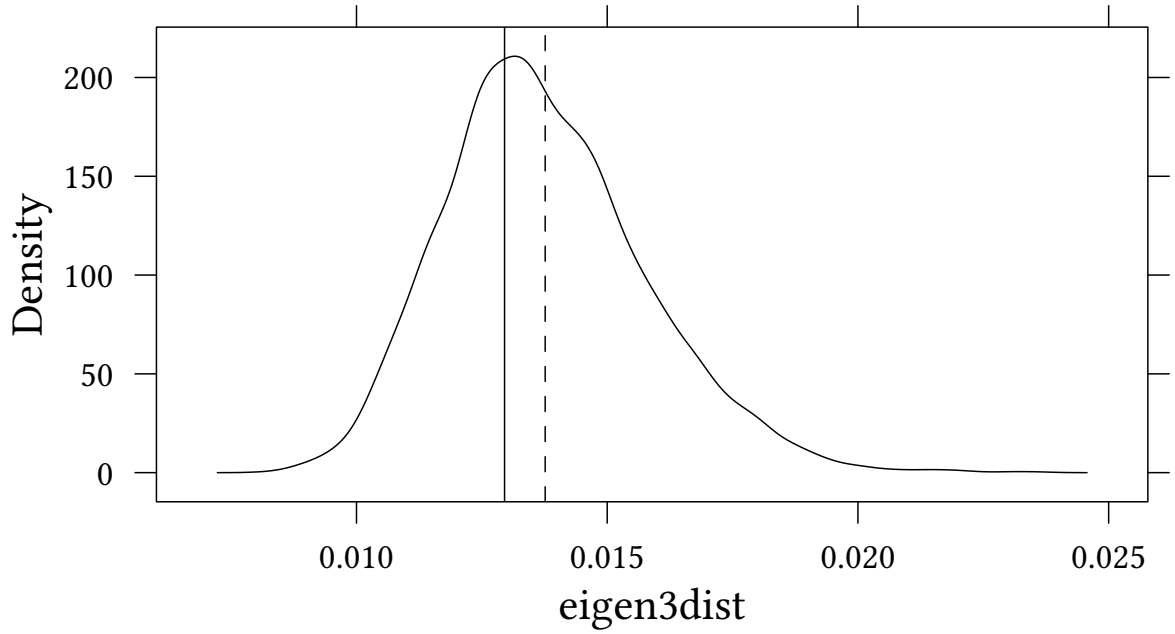
If $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are the eigenvalues, then our test statistic is

$$\hat{\sigma}^* = \hat{\lambda}_3^* / \sum_{i=1}^{4} \hat{\lambda}_i$$

```
ee <- eigen(cov(X))[["values"]]
ee[3]/sum(ee)

[1] 0.01295262
```

```
eigen3 <- function(X) {ee <- eigen(cov(X))[["values"]];ee[3]/sum(ee)}
ind <- 1:N; set.seed(123)
eigen3dist <- replicate(5000,eigen3(X[sample(ind,replace=TRUE),]))
```

```
densityplot(eigen3dist,plot.points=FALSE)+
    layer(panel.abline(v=c(eigen3(X),mean(eigen3dist)),lty=1:2))
```



Now let us compare our initial estimate…

```
eigen3(X)

[1] 0.01295262
```

…with the mean of the bootstrap replications:

```
mean(eigen3dist)

[1] 0.01376329
```

3.6 When bootstraps fail

Consider the following problem:

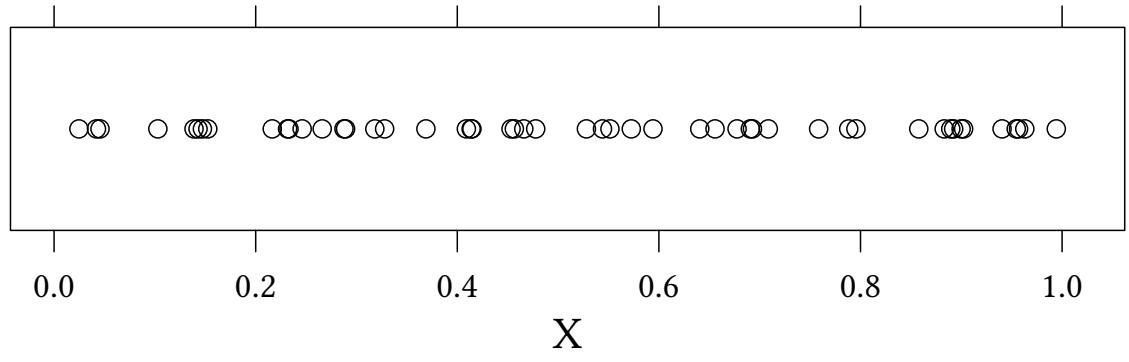
X follows a uniform distribution over $[0, \theta]$. The ML estimator for $\hat{\theta}$ is $\max(X_i)$. We have a sample of 20 observations.

We compare:

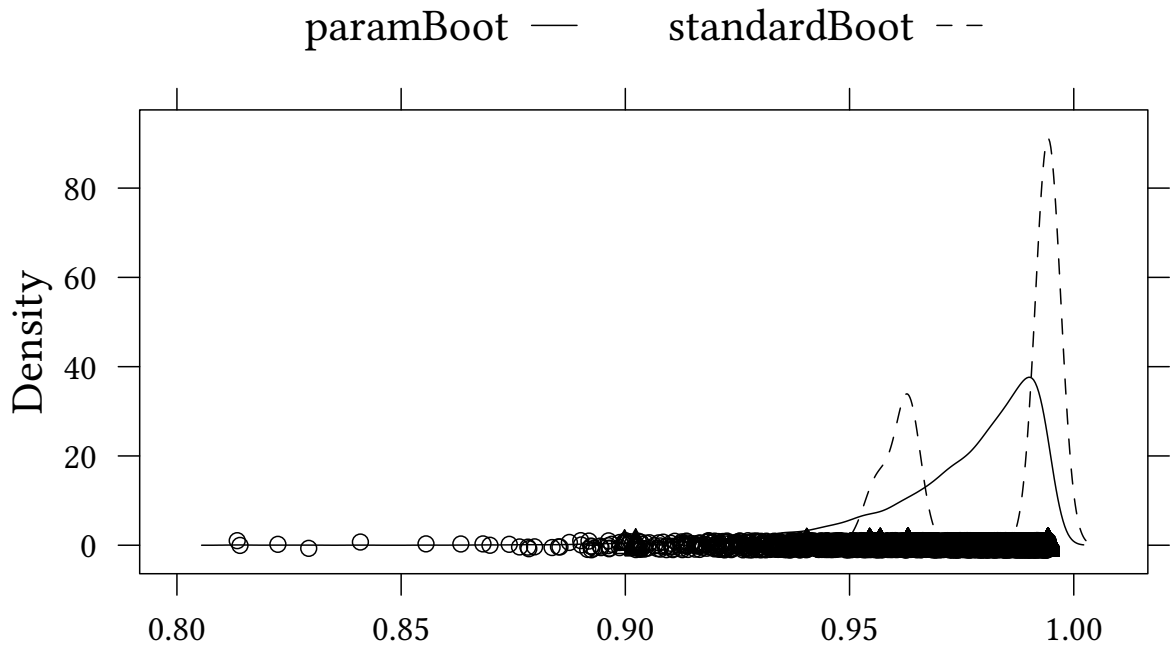
1. Standard bootstrap of $\hat{\theta}^*$.

2. Parametric bootstrap based on a uniform distribution $[0, \hat{\theta}]$.

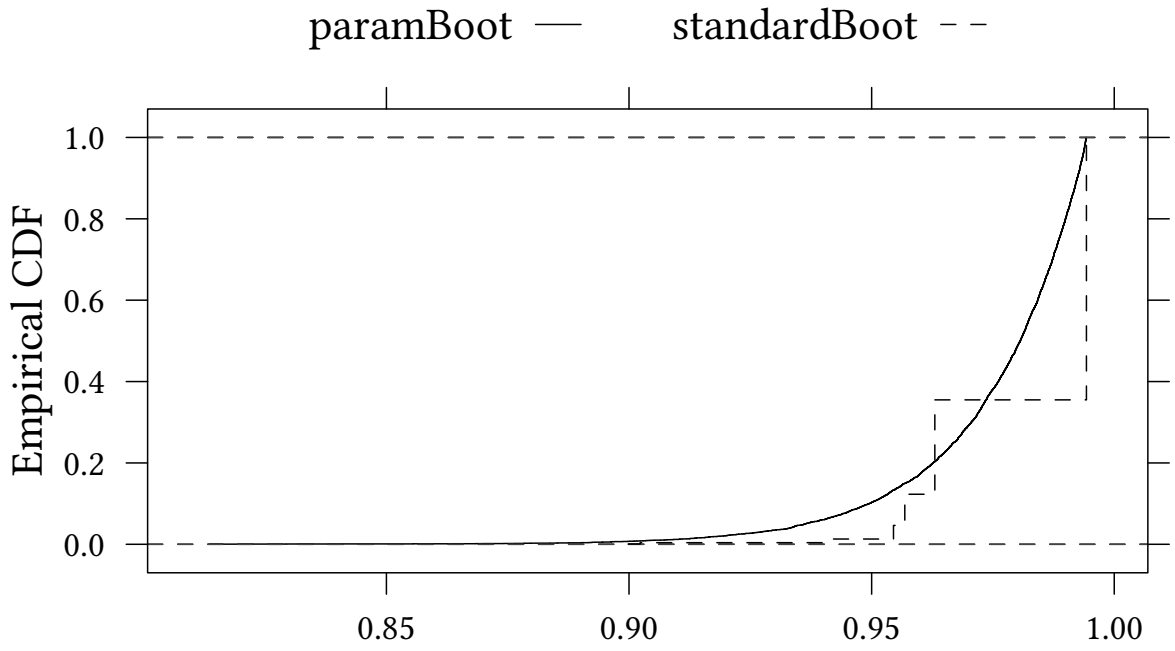
```
set.seed(123)
N <- 50
X <- runif(N)
(thetaHat <- max(X))

[1] 0.9942698

standardBoot <- replicate(5000,max(sample(X,N,replace=TRUE)))
paramBoot <- replicate(5000,max(runif(N,min=0,max=thetaHat)))
stripplot(~X)
```




```
densityplot(~paramBoot + standardBoot,xlab="",
            auto.key=list(columns=2,size=2,between=1))
```



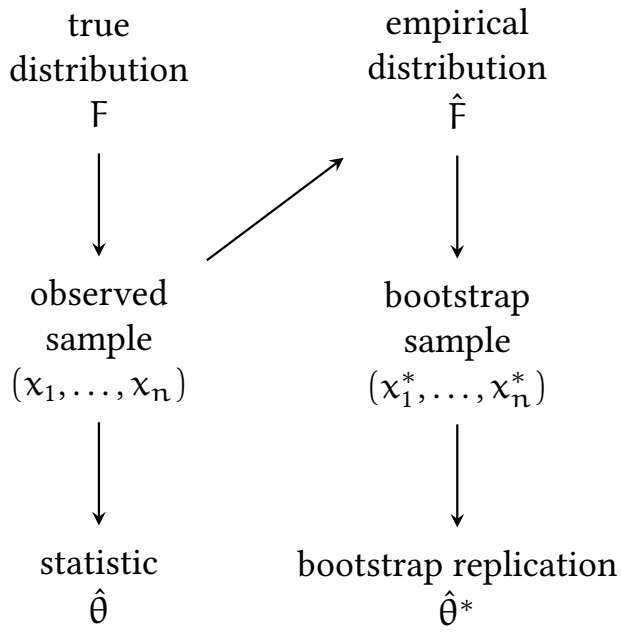

```
ecdfplot(~paramBoot + standardBoot,xlab="",
            auto.key=list(columns=2,size=2,between=1))
```



4 More complicated data structures

4.1 Motivation

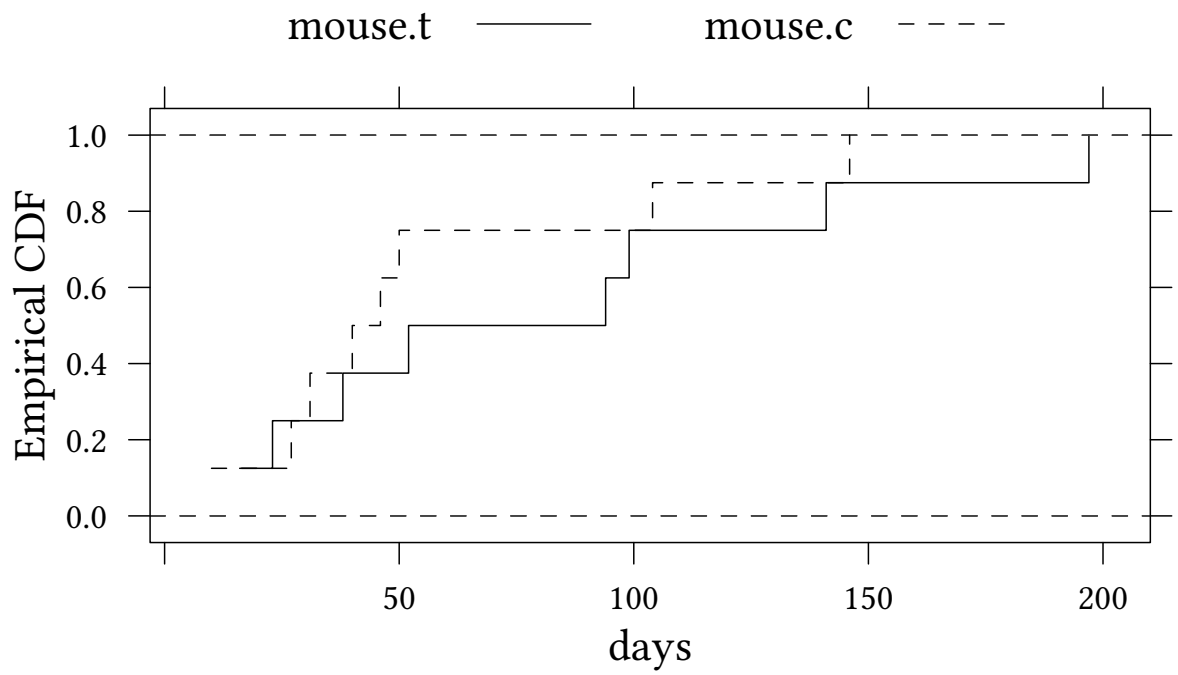
The bootstrap in one-sample problems



Crucial is the step from the observed sample $(x_1,\ldots,x_n)$ to the empirical distribution $\hat{F}$ in the bootstrap.

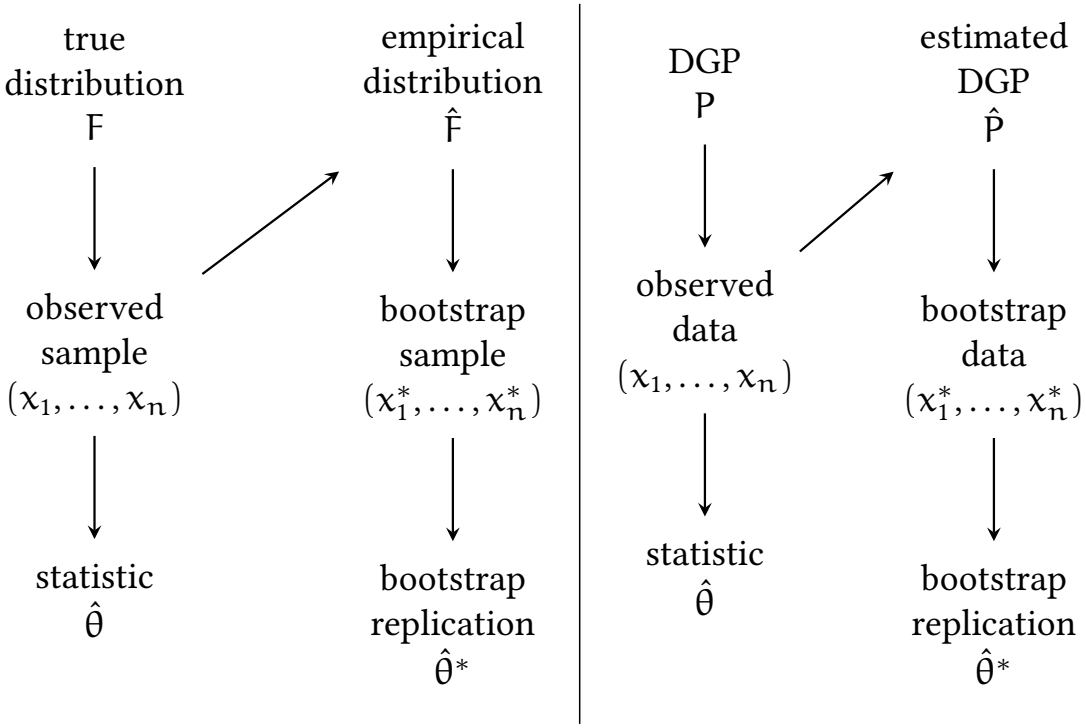
The above procedure does not work with more complicated data structures. Let us look again at the mice data:

```
ecdfplot(~mouse.t + mouse.c,auto.key=list(columns=2),xlab="days")
```



Here we have *two* samples. When we move from our observations to the bootstrap we use the distribution of *both* samples as a starting point for the bootstrap.

The bootstrap with more general data structures Let us call the data generating process P.



4.2 Regression

In a linear model the data structure is more complicated:

$$y_i = \beta x_i + \epsilon_i$$

- "bootstrapping pairs":
  we sample with replacement from the set

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

- "bootstrapping residuals":
  - We first estimate $\hat{\beta}$ from the model.
  - We then estimate the residuals $\hat{\epsilon}_i$.
  - We sample with replacement from the $\hat{\epsilon}_i$ and use the following dataset:

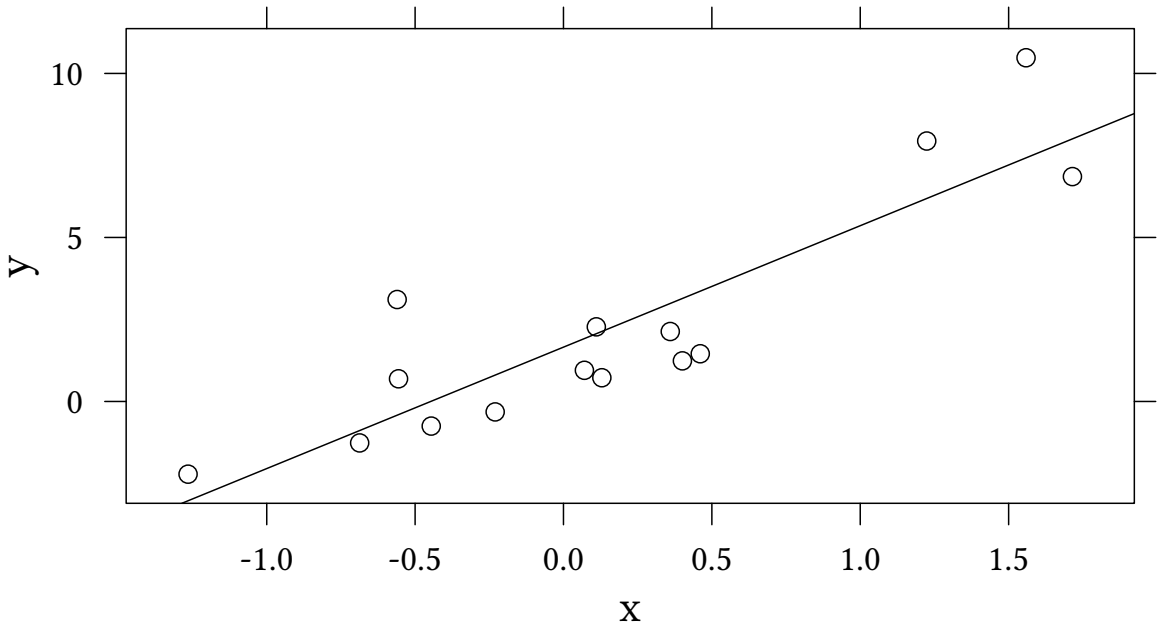
$$\{(x_1, \hat{\beta}x_1 + \hat{\epsilon}_{i_1}), (x_2, \hat{\beta}x_2 + \hat{\epsilon}_{i_2}), \dots (x_n, \hat{\beta}x_n + \hat{\epsilon}_{i_n})\}$$

4.2.1 Bootstrapping pairs

We first create some data points:

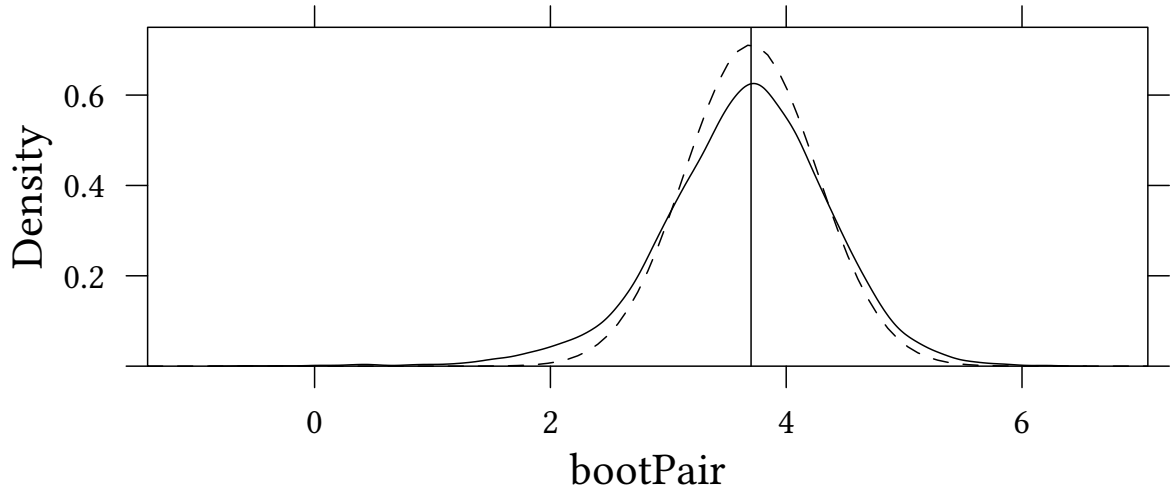
```
N <- 15
sd <- 1.5
x <- rnorm(N)
y <- 3 * x + sd * rnorm(N)^2
est <- lm(y ~ x)
```

```
xyplot (y ~ x) + layer(panel.abline(est))
```




```
set.seed(123)
bootPair <- replicate(5000,{
    ind <- sample(1:N,replace=TRUE);
    coef(lm(y[ind] ~ x[ind]))[2]
})
```

```
betaEst<-coef(est)[2]
sdBeta<-sqrt(vcov(est)[2,2])
densityplot(bootPair,plot.points=FALSE,ylim=c(0,.75))+layer(panel.abline(v=betaEst))+
    layer(panel.mathdensity(args=list(mean=betaEst,sd=sdBeta),col="black",n=100))
```



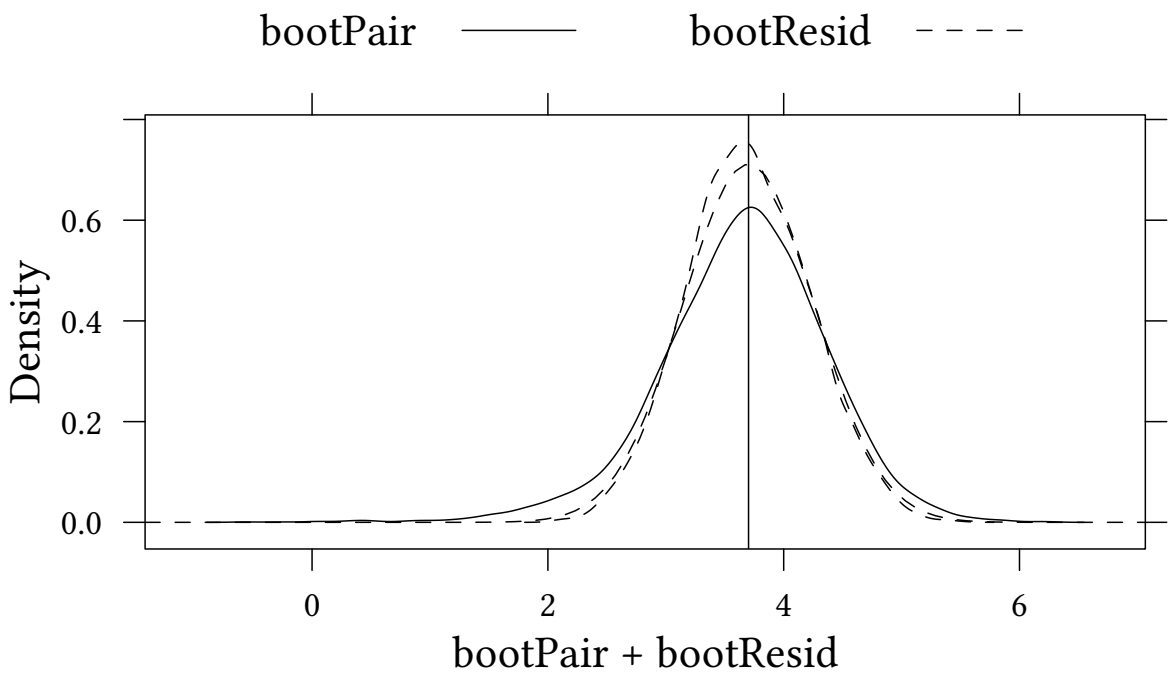
Compare the bootstrapped $\hat{\sigma}_\beta$ with the one from the regression model:

```
c(sd(bootPair),sqrt(vcov(est)[2,2]))

[1] 0.7113526 0.5610004
```

4.2.2 Bootstrapping residuals

```
rr <- residuals(est)
beta <- coef(est)
set.seed(123)
bootResid <- replicate(5000,{
  epsilon <- sample(rr,replace=TRUE)
  coef(lm ( (cbind(1,x) %*% beta + epsilon) ~ x))[2]
})
```

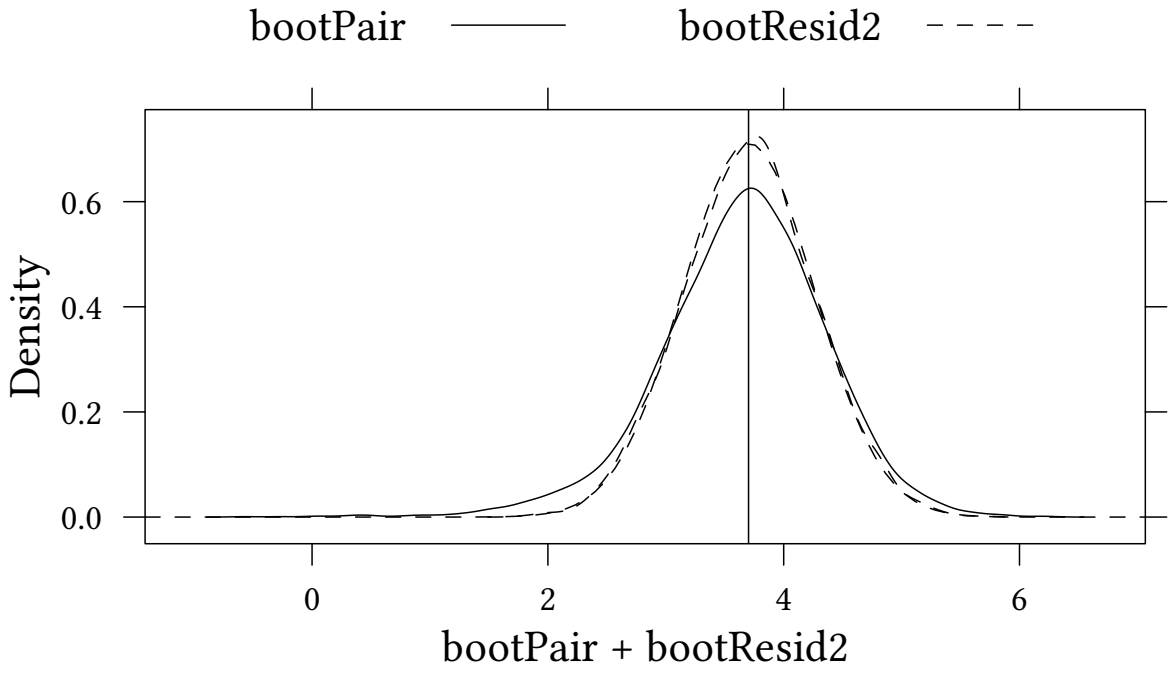
```
densityplot(~bootPair+bootResid,plot.points=FALSE,
            auto.key=list(columns=2))+layer(panel.abline(v=betaEst))+
    layer(panel.mathdensity(args=list(mean=betaEst,sd=sdBeta),
                            col="black",n=100))
```



simulate allows a simpler notation:

```
set.seed(123)
bootResid2<-replicate(5000,coef(lm(simulate(est)[,1] ~ x))[2])
```

```
densityplot(~bootPair+bootResid2,plot.points=FALSE,
            auto.key=list(columns=2))+layer(panel.abline(v=betaEst))+
    layer(panel.mathdensity(args=list(mean=betaEst,sd=sdBeta),
                            col="black",n=100))
```



Compare again the bootstrapped $\hat{\sigma}_\beta$ with the one from the regression model:

```
c(sd(bootPair),sd(bootResid),sqrt(vcov(est)[2,2]))

[1] 0.7113526 0.5208372 0.5610004
```

Exercise 4.1 *We use again the dataset* `data/x1.csv`. *Now we estimate the linear model*

$$x_2 = \beta_0 + \beta_1 x_1 + u$$

- *Estimate* β_0 *and* β_1
- *What is the standard deviation of the coefficients you obtain with the standard regression command?*
- *What is the standard deviation you obtain when you bootstrap pairs?*
- *What is the standard deviation you obtain when you bootstrap residuals?*

4.3 Timeseries — Example: A simple AR-1 process

Let us consider a simple AR-1 process:

$$y_t = \beta y_{t-1} + \epsilon_t$$

Let us first create an AR1 process by hand:

```
N <- 150
epsilon <- rnorm(N)
y <- epsilon
for (i in 2:N) y[i] <- y[i-1] * .7  + epsilon[i]
plot(y,t="l")
```

Of course, this can be done automatically. The same time series is generated by the following:

```
arima.sim(n=N,list(ar=.7),innov=epsilon,n.start=1,start.innov=0)

Time Series:
Start = 1
End = 150
Frequency = 1
 [1] -0.560475647 -0.622510442  1.122951005  0.856574095  0.728889601
 [6]  2.225287708  2.018617602  0.147971086 -0.583273091 -0.853953134
[11]  0.626314604  0.798234050  0.959535285  0.782357416 -0.008190944
[16]  1.781179476  1.744676112 -0.745343879  0.179615187 -0.347060777
[21] -1.310766250 -1.135511290 -1.820862351 -2.003494875 -2.027485680
[26] -3.105933287 -1.336366256 -0.782083262 -1.685595220  0.073898267
[31]  0.478193008  0.039663623  0.922890197  1.524156625  1.888490719
[36]  2.010583758  1.961326284  1.311016688  0.611749018  0.047753312
 [ reached getOption("max.print") -- omitted 110 entries ]
```

Let us try to estimate β:

```
(est.arima<-arima(y,order=c(1,0,0),include.mean=FALSE))


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

Coefficients:
         ar1
      0.6822
s.e.  0.0590

sigma^2 estimated as 0.8953:  log likelihood = -204.86,  aic = 413.72
```

What are possible approaches?

bootstrapping pairs: not possible here, would destroy the time structure

bootstrapping residuals: preserves the time structure, makes assumptions on independence of residuals

moving blocks: preserves parts of the time structure

Let us start with bootstrapping residuals:

4.3.1 Bootstrapping residuals

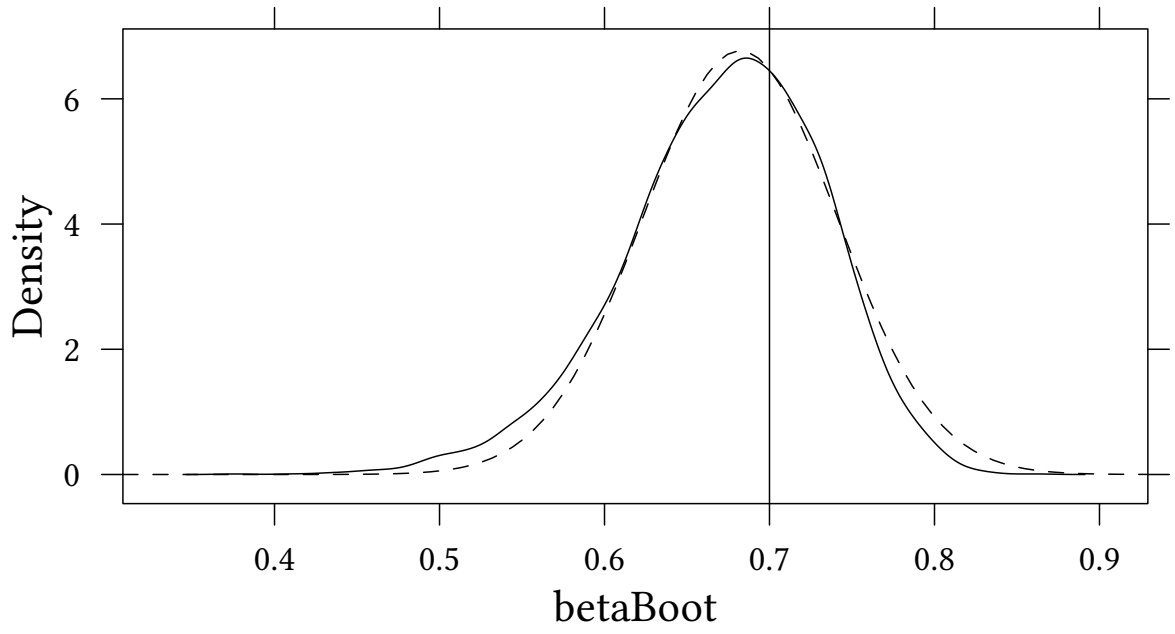
To construct the bootstrap sample we will use the estimated residuals:

```
set.seed(123)
rr <- residuals(est.arima)
betaB <- coef(est.arima)
betaBoot <- replicate(5000,{
    epsilon <- sample(rr,size=N,replace=TRUE)
    by <- arima.sim(n=N,list(ar=betaB),innov=epsilon,n.start=1,
                    start.innov=0)
    coef(arima(by,order=c(1,0,0),include.mean=FALSE))
})
```

Now let us compare the bootstrap estimate of the standard error with the regression estimate:

```
c(sd(betaBoot),sqrt(vcov(est.arima)))

[1] 0.06059008 0.05895473
```

```
sdBeta<-sqrt(vcov(est.arima))
densityplot(~betaBoot,plot.points=FALSE)+ layer(panel.abline(v=.7))+
        layer(panel.mathdensity(args=list(mean=betaB,sd=sdBeta),
                                col="black",n=100))
```



Now let us assume that we (wrongly) suspect the process to be AR-2:

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + \epsilon_t$$

We estimate

```
(est2.arima<-arima(y,order=c(2,0,0),include.mean=FALSE))


Call:
arima(x = y, order = c(2, 0, 0), include.mean = FALSE)

Coefficients:
         ar1     ar2
      0.6814  0.0011
s.e.  0.0817  0.0818

sigma^2 estimated as 0.8953:  log likelihood = -204.86,  aic = 415.72
```

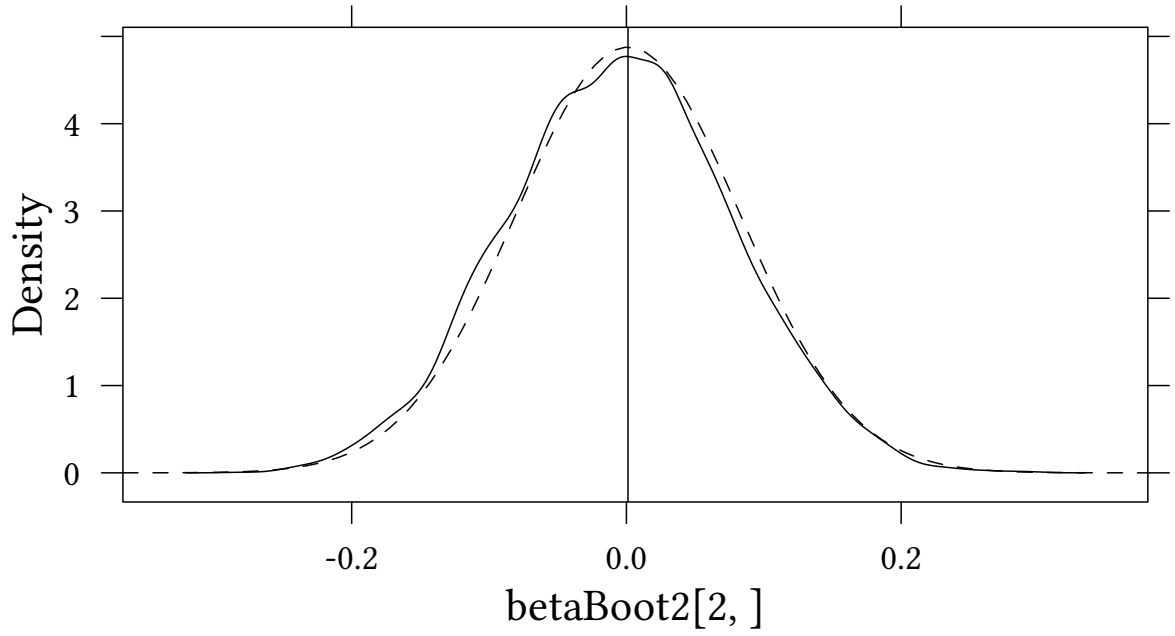
What can we say about the accuracy of $\hat{\beta}_2$?

```
set.seed(123)
rr <- residuals(est2.arima)
(betaB <- coef(est2.arima))

        ar1         ar2
0.681410377 0.001147245

seB<-sqrt(diag(vcov(est2.arima)))
betaBoot2 <- replicate(5000,{
  epsilon <- sample(rr,N,replace=TRUE)
  by <-arima.sim(n=N,list(ar=betaB),innov=epsilon)
  coef(arima(by,order=c(2,0,0),include.mean=FALSE))
})
```

Let us also here compare the estimated standard deviations of β_2:

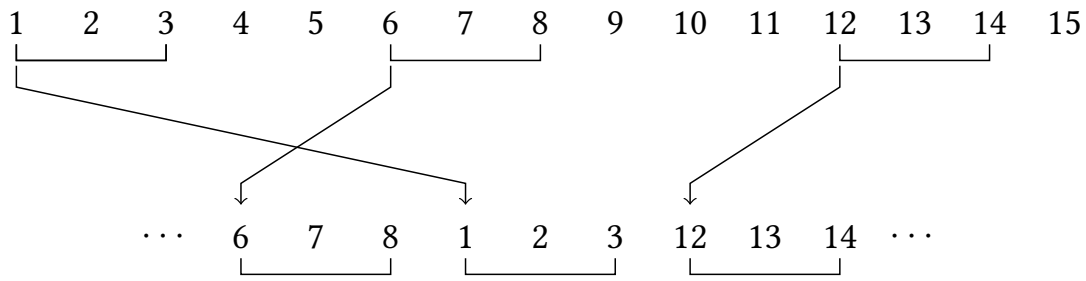
```
c(sd(betaBoot2[2,]),sqrt(vcov(est2.arima)[2,2]))

[1] 0.08177441 0.08179531
```

```
densityplot(betaBoot2[2,],plot.points=FALSE)+
    layer(panel.abline(v=betaB[2]))+layer(panel.mathdensity(
    args=list(mean=betaB[2],sd=seB[2]),col="black",n=100))
```



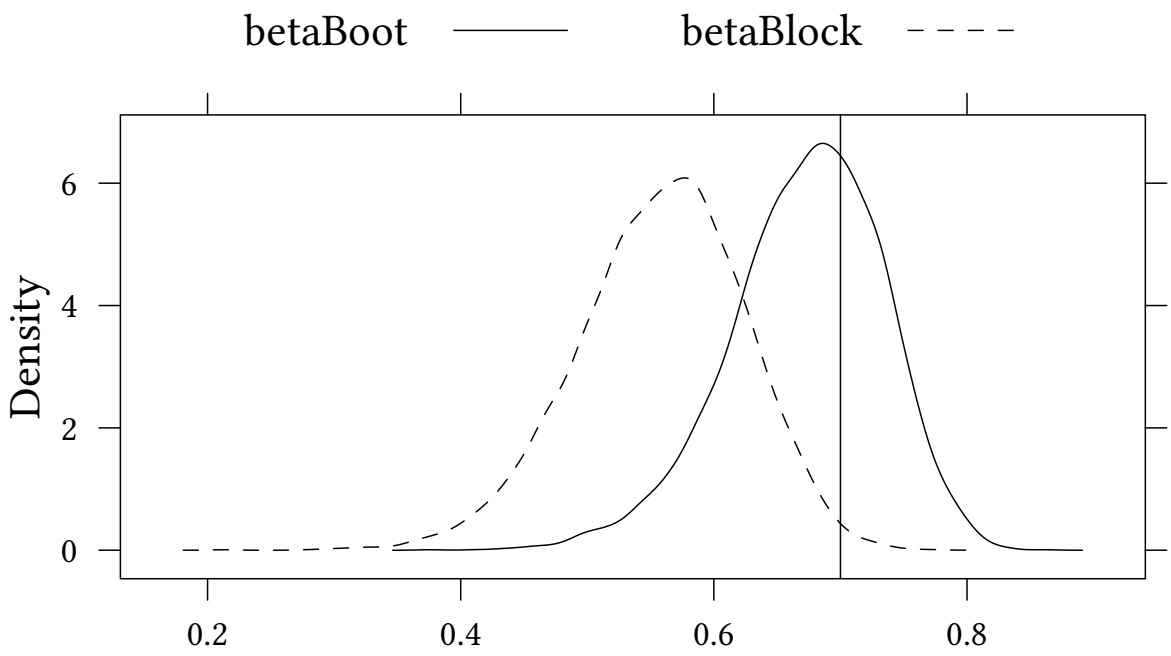
4.3.2 Moving blocks

The above regression model assumes that we know a lot about the data generating process. Here we just assume that a short block of data already contains a characteristic pattern.




```
blockLen <- 5
blockNum <- N/blockLen
betaBlock <- replicate(5000,{
  start <- sample(1:(N-blockLen+1),size=blockNum,replace=TRUE)
  blockedIndices <- c(sapply(start,function(x) seq(x,x+blockLen-1)))
  by <- y[blockedIndices]
  coef(arima(by,order=c(1,0,0),include.mean=FALSE))
})
```

```
densityplot(~betaBoot+betaBlock,xlab="",plot.points=FALSE,auto.key=list(columns=2))+
    layer(panel.abline(v=0.7))
```




```
c(sd(betaBlock),sqrt(vcov(est.arima)))

[1] 0.06614983 0.05895473
```

Exercise 4.2 *The dataset* `x2.csv` *contains a sequence of values for* x *and* y*. You estimate the following model:*

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 x_t + \epsilon_t$$

- *What estimates do you obtain with OLS?*
- *What are the standard deviations of your estimates if you use the traditional method?*
- *Use the bootstrapping residuals methods to obtain standard deviations.*
- *Use moving blocks to obtain standard deviations.*
- *How does the standard deviation depend on the block size? Provide a plot with the block size on the horizontal axis and the standard deviations on the vertical axis.*

Rules of Thumb (Hall et. al., Biometrica, 1995) The optimal block length:

- $n^{1/3}$ for bias or variance
- $n^{1/4}$ for a one-sided distribution function
- $n^{1/5}$ for a symmetrical distribution function

5 Bias

5.1 Estimating the bias

$$t(F) = \theta \quad \text{true parameter,} \qquad s(x) = \hat{\theta} \quad \text{estimator}$$

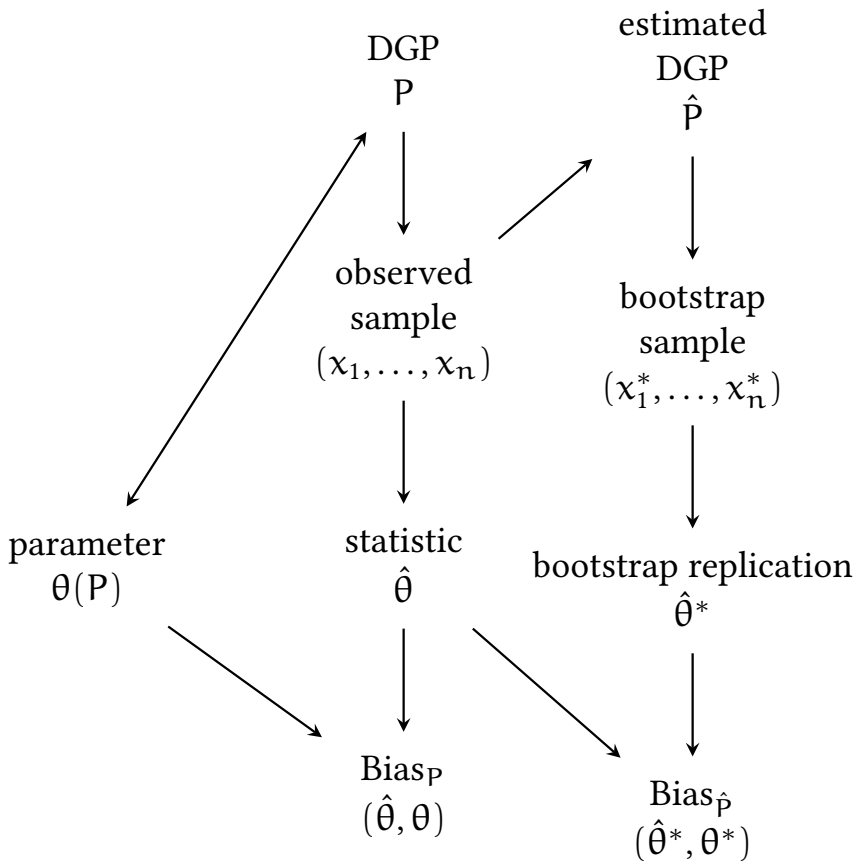
$$\text{Bias}_F = E_F[s(\boldsymbol{x})] - t(F)$$

as above we replace F by $\hat{F}$ to obtain the bootstrap estimate of the bias:

$$\text{Bias}_{\hat{F}} = E_{\hat{F}}[s(\boldsymbol{x}^*)] - s(\hat{F})$$

The bootstrap estimate of the bias

1. Draw B independent bootstrap samples $\boldsymbol{x}^{*1}, \boldsymbol{x}^{*2}, \ldots, \boldsymbol{x}^{*B}$.
   Each sample has size n and is drawn with replacement from $x_1, x_2, \ldots, x_n$.
2. For bootstrap sample $b = 1, 2, \ldots, B$ determine $\hat{\theta}^{*b} = s(\boldsymbol{x}^{*b})$.
3. $\text{Bias}_{\hat{F}} \approx \widehat{\text{Bias}}_B = \dfrac{1}{B} \sum_{b=1}^{B} \hat{\theta}^{*b} - s(\boldsymbol{x})$

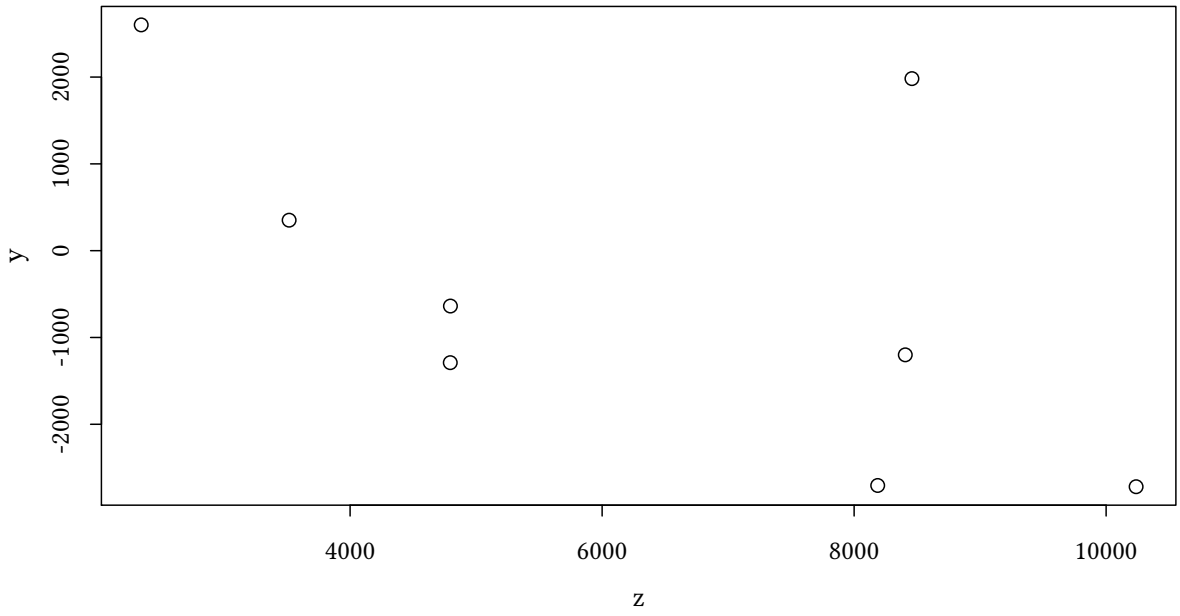


Example: The patch data

We are interested in "bioequivalence"

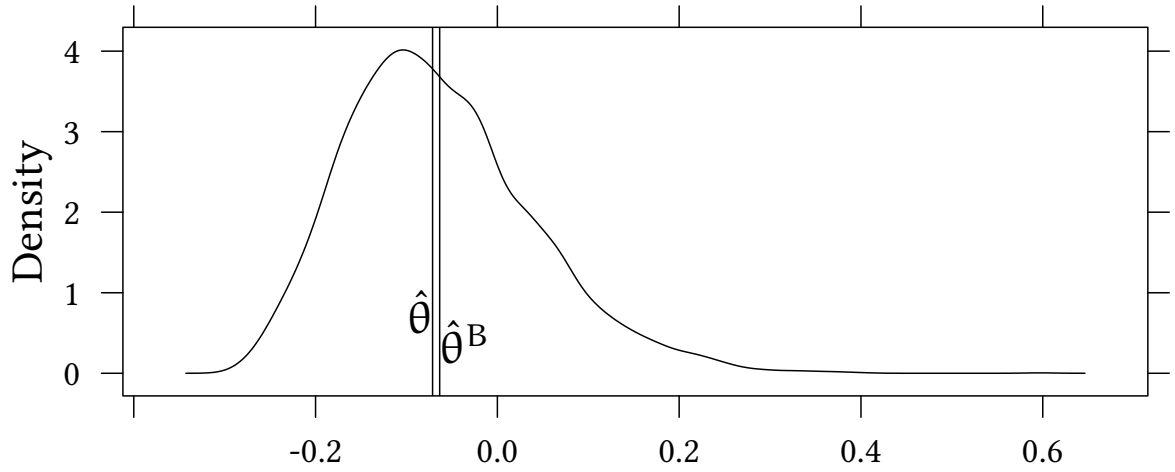
$$\frac{\mathrm{E}(\text{new patch})-\mathrm{E}(\text{old patch})}{\mathrm{E}(\text{old patch})-\mathrm{E}(\text{placebo patch})}=\frac{\mathrm{E}(y)}{\mathrm{E}(z)}\approx\frac{\bar{y}}{\bar{z}}=\frac{-452.25}{6342.375}\approx-0.0713$$

```
data(patch,package="bootstrap")
plot(y ~ z,data=patch)
```




```
N <- nrow(patch)
set.seed(123)
thetaHatS <- replicate(5000,{ind <- sample(1:N,replace=TRUE) ;
                             with(patch[ind,],mean(y)/mean(z))})
thetaHat <- with(patch,mean(y)/mean(z))
mean(thetaHatS)-thetaHat

[1] 0.007848273
```

```
densityplot(thetaHatS,plot.points=FALSE,xlab="")+
    layer(panel.abline(v=c(thetaHat,mean(thetaHatS))))+
    layer(panel.text(thetaHat,.5,"$\\hat{\\theta}$",adj=c(1,0)))+
    layer(panel.text(mean(thetaHatS),.5,"$\\hat{\\theta}^B$",adj=c(0,1)))
```



Estimated bias and standard deviation

```
bias <- mean(thetaHatS) - thetaHat
c(sd(thetaHatS),bias/sd(thetaHatS))

[1] 0.1042157 0.0753080
```

5.2 A better estimate for the bias

A problem in

$$\text{Bias}_{\hat{F}} \approx \widehat{\text{Bias}}_B = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*b} - s(\mathbf{x})$$

is that in $\frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*b}$ some observations are sampled more frequently than others, while in $s(\mathbf{x})$ all observations in $\mathbf{x}$ have the same weight. We should correct for the different weights.

- We can express the sampling process as a list of "sampled" indices.
- Alternative we can express the sampling process as frequencies:

$$P_i^* = \frac{1}{n}\cdot\#\{x_j^* = x_i\} \quad \text{for } i = 1,\dots,n$$

Hence, we define a *resampling vector*

$$\mathbf{P}^* = (P_1^*,\dots,P_n^*) \quad \text{with} \sum_{i=1}^{n} P_i^* = 1$$

In our case:

$$\hat{\theta}^* = \frac{\bar{y}^*}{\bar{z}^*} = \frac{\sum_{i=1}^{n} P_i^* y_i}{\sum_{i=1}^{n} P_i^* z_i}$$

More generally, similar to $\hat{\theta} = t(F)$ we write

$$\hat{\theta}^* = T(\mathbf{P}^*)$$

Let $\mathbf{P}^0 = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)$ be a vector of length n where all elements are $1/n$, i.e. all elements of $\mathbf{x}$ have the same weight.

$$T(\mathbf{P}^0) = \hat{\theta} = t(\hat{F})$$

Above we defined

$$\widehat{\text{Bias}}_B = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^{*b} - s(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T(\mathbf{P}_b^*) - T(\mathbf{P}^0)$$

Now we define

$$\bar{\mathbf{P}}^* = \frac{1}{B}\sum_{b=1}^{B} \mathbf{P}_b^*$$

Then a better estimate for the bias is

$$\overline{\text{Bias}}_B = \frac{1}{B}\sum_{b=1}^{B} T(\mathbf{P}_b^*) - T(\bar{\mathbf{P}}^*)$$

5.3 The jackknife

With the above "better" estimate we still have to calculate a large number of replications. Furthermore, all these replications are random. Is there a more systematic way?

(Maurice Quenouille, 1949)

The jackknife leaves out only one observation from n observations:

$$\mathbf{x} = (x_1, x_2, \cdots, x_n)$$

$$\mathbf{x}_{(i)} = (x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$$

For each jackknife replication we calculate

$$\hat{\theta}_{(i)} = s(\mathbf{x}_{(i)})$$

Then

$$\widehat{\text{Bias}}_{\text{jack}} = (n-1)\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)} - \hat{\theta}\right)$$

```
jacks <- sapply(1:N,function(i) with(patch[-i,],mean(y)/mean(z)))
(biasJack <- (N-1) * (mean(jacks) - thetaHat))

[1] 0.008002488
```

The jackknife works well if $t(\cdot)$ is a *smooth* statistic. (e.g. the median is not smooth) There is also a jackknife estimate of the standard error:

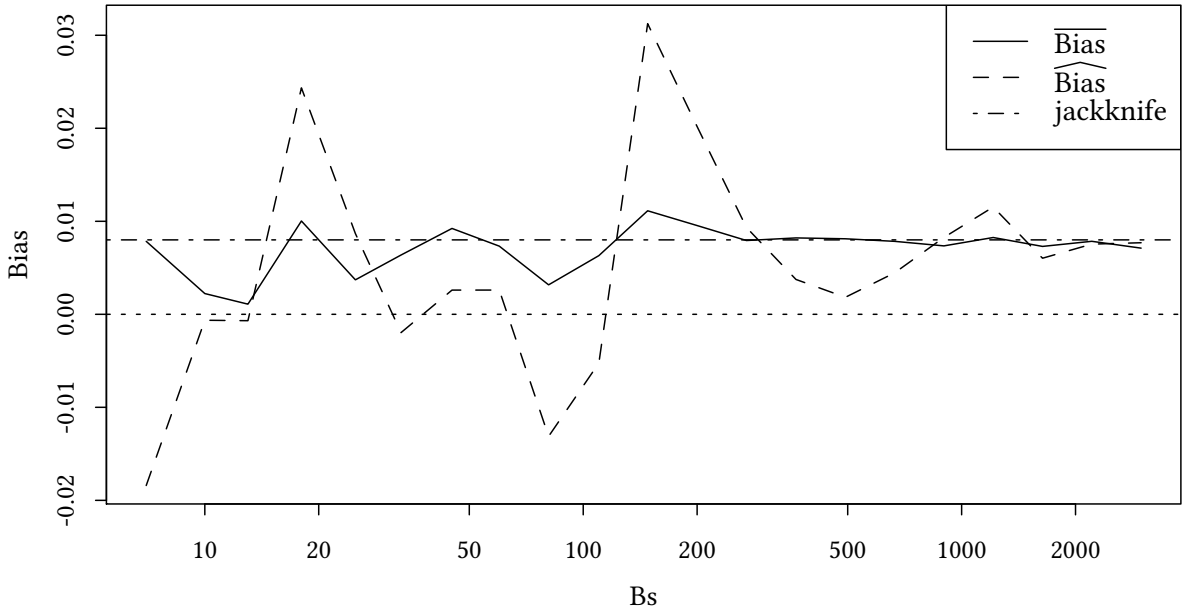
$$\widehat{\text{se}}_{\text{jack}} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{\theta}_{(i)} - \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)}\right)^2} = \frac{n-1}{\sqrt{n}}\,\text{se}(\hat{\theta}_{(i)})$$

```
c(sd(jacks) * (N-1)/sqrt(N),sd(thetaHatS))

[1] 0.1055278 0.1042157
```

5.4 Convergence of estimates of Bias

```
set.seed(123)
T <- function(P) with(patch,(y %*% P)/(z %*% P))
weights <- matrix(0,N,N)
diag(weights)<-1
Pzero <- rep(1,N)/N
Pvec <- function() apply(weights[sample(1:N,replace=TRUE),],2,sum)/N
estimate <- function(B) {
  Pbar <- rep(0,N)
  thetaStar <- mean(replicate(B,{PP <- Pvec();
                                 Pbar <<- Pbar + PP;
                                 T(PP)}))
  c(biasBar=thetaStar - T(Pbar/sum(Pbar)),biasHat=thetaStar - T(Pzero))
}
Bs <- round(exp(seq(2,8,.3)))
estimates <- sapply(Bs,estimate)
```

```
plot(estimates["biasHat",] ~ Bs,log="x",t="l",lty="dashed",ylab="Bias")
lines(estimates["biasBar",] ~ Bs,t="l")
abline(h=0,lty="dotted")
abline(h=biasJack,lty=4)
legend("topright",c(expression(bar(Bias)),expression(widehat(Bias)),"jackknife"),
       lty=c(1,2,4))
```



Exercise 5.1 *1. You suspect that your estimate for the coefficient of y_{t-1} in exercise 4.2 might be biased.*

- *What is the standard bootstrap estimate of the bias?*
- *What is the better estimate?*
- *What is the jackknife estimate?*

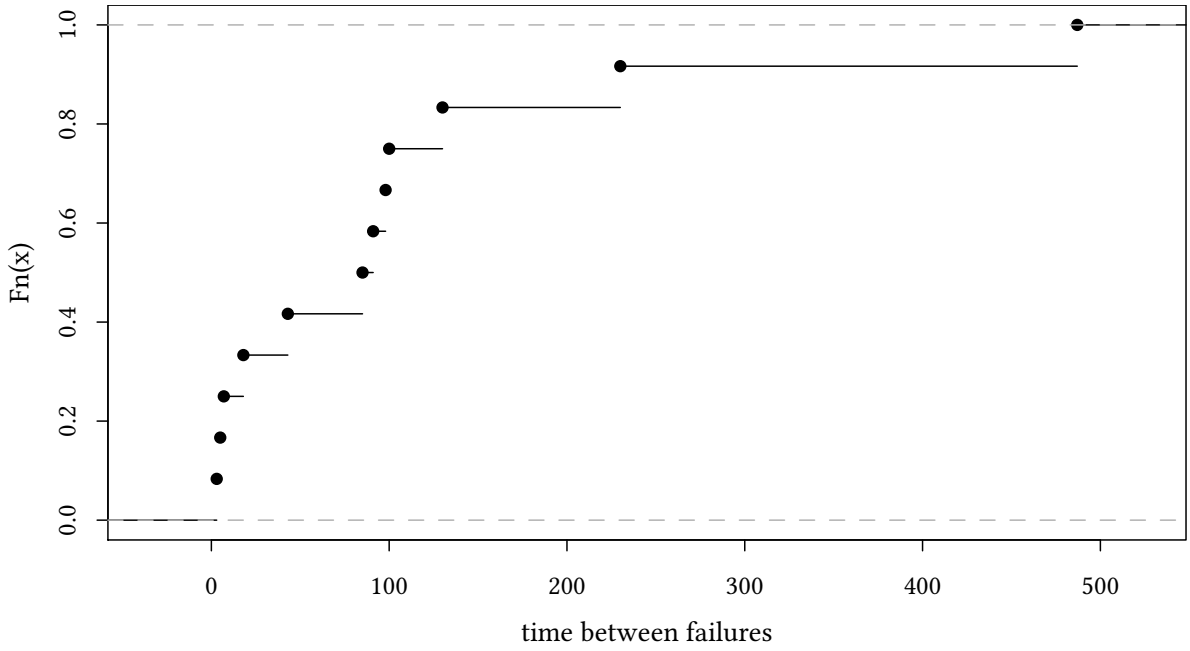
2. Now you use a median regression to estimate the relationship in exercise 4.2, i.e. you minimise the sum of absolute values of residuals, and not, as in OLS, the sum of squared residuals.

- *What coefficients do you estimate?*
- *What is the standard bootstrap estimate of the bias?*
- *What is the better estimate?*
- *What is the jackknife estimate?*

6 Confidence intervals

The `aircondit` data: Failure times of air-conditions.

```
data(aircondit,package="boot")
plot(ecdf(aircondit$hours),main="",xlab="time between failures")
```




```
airMean <- mean(aircondit$hours)
```

The distribution does not look normal.

6.1 The exact approach:

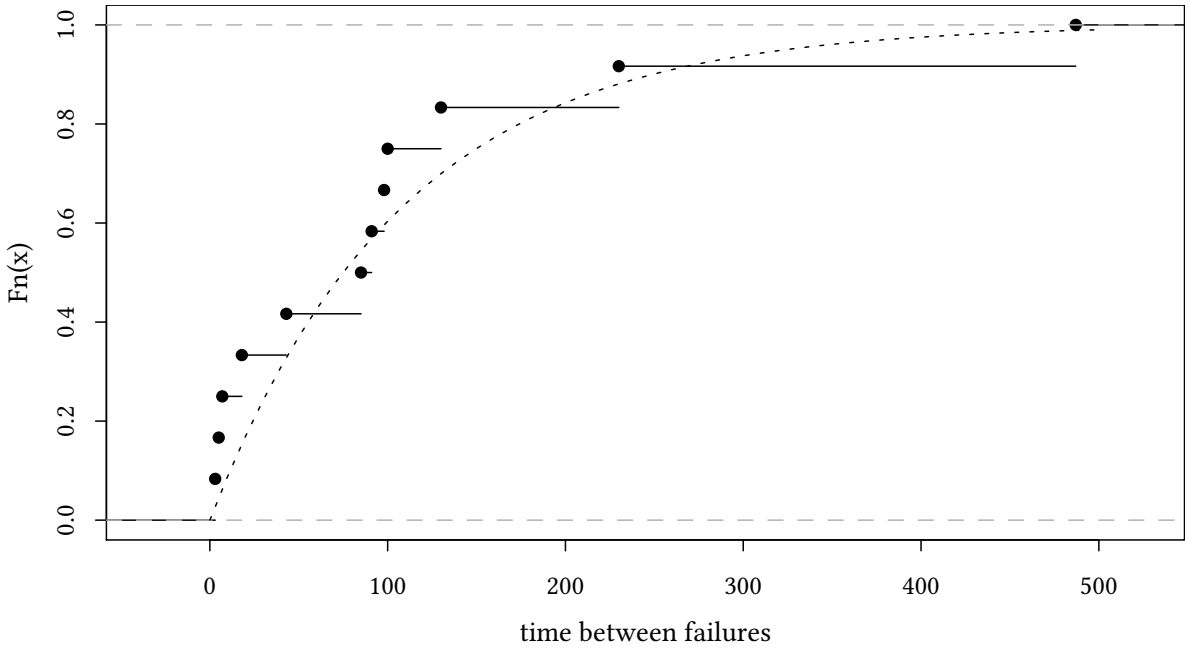
- Works even for small samples
- requires knowledge of the "true" distribution
- may require some serious transformations

If failure times follow a Poisson distribution, then the times between failure follow an exponential distribution.

$$f(x,\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geqslant 0 \\ 0 & x < 0 \end{cases}$$

To find λ we can use ML: $\hat{\lambda} = n/\sum x_i = 1/\bar{x}$.

```
 with(aircondit,{
   n <<- length(hours)
   lambda <<-length(hours)/sum(hours)
 })
plot(ecdf(aircondit$hours),main="",xlab="time between failures")
with(list(x=seq(0,500,10)),lines(1-exp(-lambda * x) ~ x,lty="dotted"))
```



With $\lambda = 0.00925$ and $\bar{x} = 108$ we can now try to work out the distribution of the mean of an exponentially distributed random variable.

$$\mathrm{CI}_{1/\lambda} = \left[\frac{1}{\bar{\lambda}}\frac{2n}{\chi^2_{2n,1-\alpha/2}}; \frac{1}{\bar{\lambda}}\frac{2n}{\chi^2_{2n,\alpha/2}}\right]$$

```
alpha<-5/100
sapply(c(1-alpha/2,alpha/2),
       function(alpha) 2*n/(lambda * qchisq(alpha,2*n)))

[1]  65.89765 209.17415
```

6.2 The normal approximation:

- Asymptotically the mean follows a normal distribution
- Hence: We can do this when the sample is “large”

6.2.1 The normal approximation based on parametric estimates of σ:

```
confint(lm(hours ~ 1,data=aircondit))

               2.5 %   97.5 %
(Intercept) 21.52561 194.6411
```

6.2.2 Normal approximation based on bootstrap estimates of σ:

```
set.seed(123)
xx <- with(aircondit,replicate(5000,{
    x<-sample(hours,replace=TRUE)
    c(mean=mean(x),sd=sd(x))}))
```

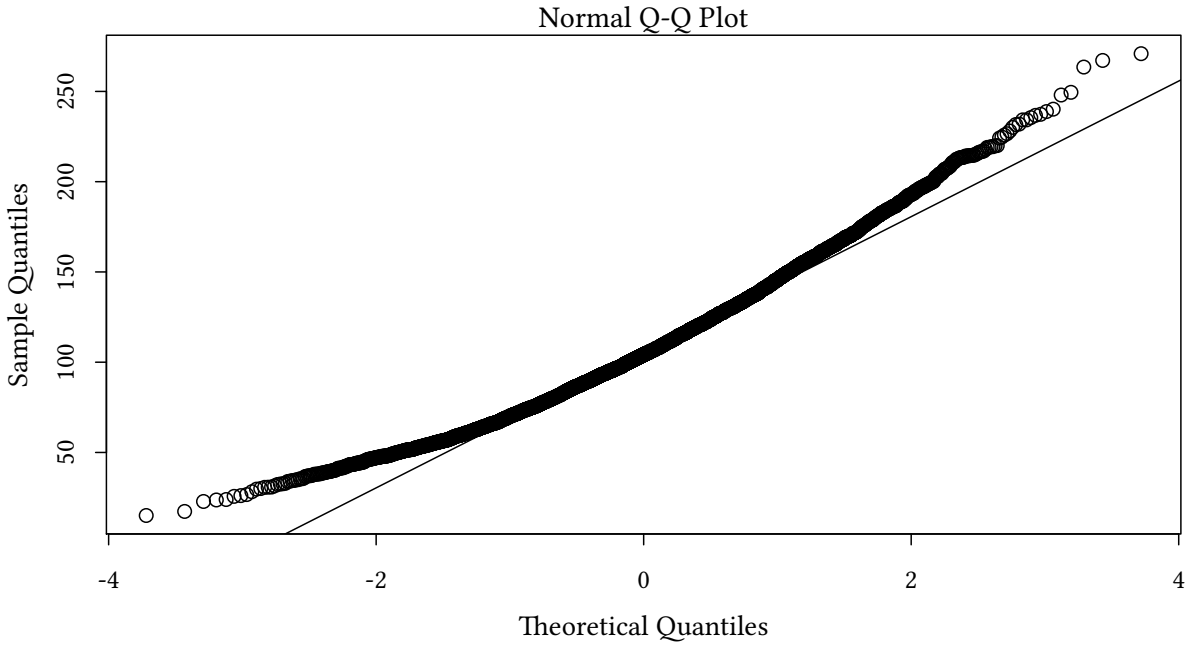
```
mean(xx["mean",]) + c(1,-1) * qnorm(alpha/2) * sd(xx["mean",])

[1]  34.4649 181.2955
```

Alternatively we use the built in function *boot*:

```
set.seed(123)
airBoot <- boot(aircondit$hours,function(x,i) mean(x[i]),R=5000)
boot.ci(airBoot,type="norm")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = airBoot, type = "norm")

Intervals :
Level      Normal
95%   ( 34.5, 182.0 )
Calculations and Intervals on Original Scale
```

Can we use a normal approximation for the distribution of the mean?

```
qqnorm(xx["mean",])
qqline(xx["mean",])
```



6.3 The bootstrap-t interval

$$\underset{(\hat{\theta})}{Q(\alpha/2)} - \theta - \underset{(\hat{\theta})}{Q(1-\alpha/2)}$$

Idea: If θ is the mean of a normally distributed random variable, then we have

$$\frac{\hat{\theta}-\theta}{\hat{\sigma}_{\hat{\theta}}} \sim t_{n-1} \quad \to \mathrm{CI} = \left[\hat{\theta} - t_{n-1}^{(1-\alpha/2)}\hat{\sigma}_{\hat{\theta}}, \hat{\theta} - t_{n-1}^{(\alpha/2)}\hat{\sigma}_{\hat{\theta}}\right]$$

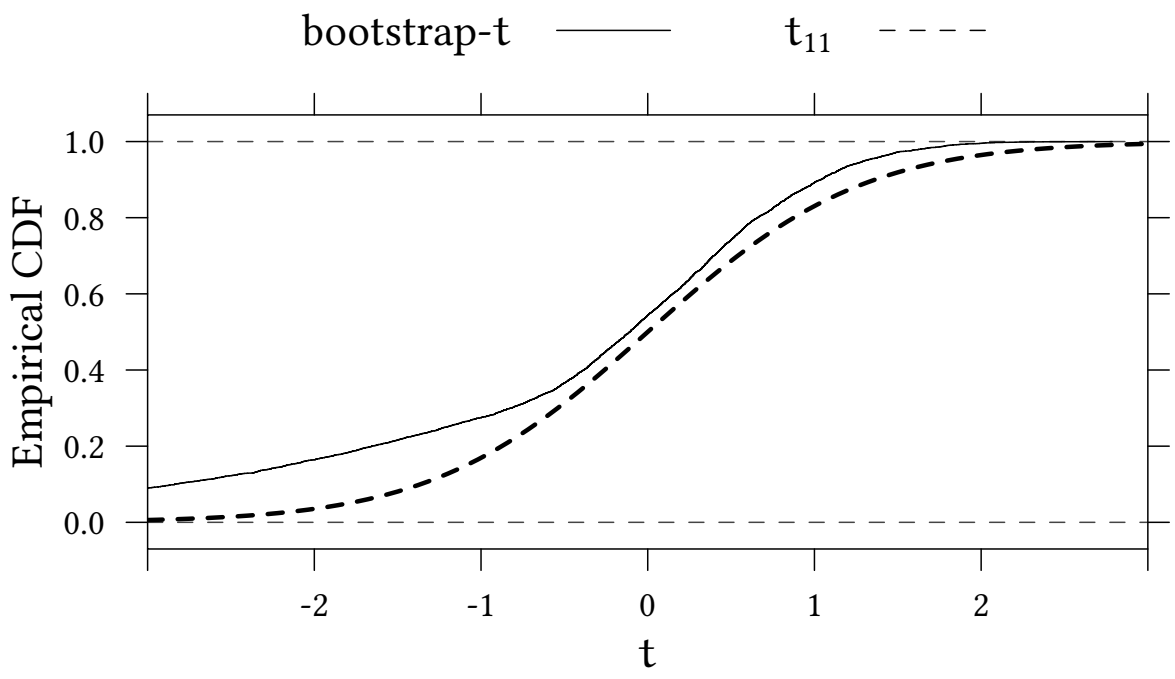
If θ is something else, we can bootstrap our own distribution of $\hat{t}$:
For each bootstrap sample b we calculate *approximate pivots*:

$$Z^*(b) = \frac{\hat{\theta}^*(b) - \hat{\theta}}{\hat{\sigma}^*_{\hat{\theta}}(b)}$$

(approximate since $Z^*(b)$ depends on $\hat{\theta}$)
The quantiles of the distribution of Z^* are the quantiles of $\hat{t}$.
This distribution can be *asymmetric*.

```
N <- nrow(aircondit)
thetaHat <- mean(aircondit[["hours"]])
bT <- (xx["mean",]-thetaHat)/(xx["sd",]/sqrt(N))
```

```
ecdfplot(bT,xlim=c(-3,3),xlab="$t$",key=simpleKey(c("bootstrap-$t$","$t_{11}$"),
                                        points=FALSE,lines=TRUE,columns=2))+
    layer(panel.curve(pt(x,N-1),col=mPal2[2],lty=2,lwd=3))
```



(note that we need many bootstrap samples to obtain reliable quantiles)
Now let us compare the quantiles of the orignal t-distribution with the bootstrap-t:

```
quantile(bT,c(alpha/2,1-alpha/2))

     2.5%     97.5%
-4.552214  1.549664

qt(c(alpha/2,1-alpha/2),N-1)

[1] -2.200985  2.200985
```

$$\text{CI} = \left[\hat{\theta} - \hat{\sigma}_{\hat{\theta}}\hat{t}_{n-1}^{(1-\alpha/2)}, \hat{\theta} - \hat{\sigma}_{\hat{\theta}}\hat{t}_{n-1}^{(\alpha/2)}\right]$$

```
mean(xx["mean",]) - sd(xx["mean",])*quantile(bT,c(1-alpha/2,alpha/2))

    97.5%      2.5%
 49.83369 278.39471

mean(xx["mean",]) - sd(xx["mean",])*qt(c(1-alpha/2,alpha/2),N-1)

[1]  25.43685 190.32358
```

```
set.seed(123)
airBoot <- boot(aircondit$hours,function(x,i)
    c(mean(x[i]),var(x[i])/length(x[i])),R=5000)
boot.ci(airBoot,type="stud")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = airBoot, type = "stud")

Intervals :
Level    Studentized
95%   ( 47.3, 293.8 )
Calculations and Intervals on Original Scale
```

6.3.1 Two problems with bootstrap-t

$\hat{\sigma}^*_{\hat{\theta}}(b)$: Above we did use a standard formula to obtain $\hat{\sigma}^*_{\hat{\theta}}(b)$. What, if such a formula does not exist? We can, of course, bootstrap $\hat{\sigma}^*_{\hat{\theta}}(b)$, but this requires to run a bootstrap for each of our bootstrap samples.

Transformation invariance: The bootstrap-t procedure is not invariant with respect to transformations of the variable we are studying.

Sometimes we want to transform x to get a better estimate for θ — but what if we do not know what transformation to apply? → use the bootstrap to find the "right" transformation → lose variance stability (which is more important) ↓

→ Percentile intervals.

Besides: What is a "good" transformation m of $\hat{\theta}$? Perhaps one where $m(\hat{\theta}) \sim N(\phi, c^2)$.

(First, note that the transformation applies to θ, not to X!)
Remember normal approximation of CI based on bootstrap estimates of σ:

```
mean(xx["mean",]) + c(1,-1) * qnorm(alpha/2) * sd(xx["mean",])

[1]  34.4649 181.2955
```

Now tranform *mean(hours)*

```
set.seed(123)
xxL <- with(aircondit,replicate(5000,{
    x<-sample(hours,replace=TRUE)
    c(mean=log(mean(x)),
      sd=jackknife(x,function(x) log(mean(x)))[["jack.se"]])}))
exp(mean(xxL["mean",]) + c(1,-1) * qnorm(alpha/2) * sd(xxL["mean",]) )

[1]  49.66404 206.64823
```

The same problem appears with bootstrap-t:

```
quantile(bT,c(alpha/2,1-alpha/2))

     2.5%     97.5%
-4.552214  1.549664

thetaHatL <- log(thetaHat)
bTL <- (xxL["mean",]-thetaHatL)/(xxL["sd",])
quantile(bTL,c(alpha/2,1-alpha/2))

    2.5%    97.5%
-2.97142  1.95040

exp(mean(xxL["mean",]) - quantile(bTL,c(1-alpha/2,alpha/2))
    * sd(xxL["mean",]))

    97.5%      2.5%
 49.83711 298.53880
```

We can do the same exercise using the *boot* and *boot.ci* functions.

```
set.seed(123)
airBootL <- boot(aircondit$hours,function(x,i)
    c(log(mean(x[i])),
      jackknife(x[i],function(x) log(mean(x)))[["jack.se"]]^2),
                R=5000)
exp(boot.ci(airBootL,type="stud")[["student"]][4:5])

[1]  48.3277 383.3665
```

6.4 Percentile intervals

$$\mathrm{CI}_{\%} = \left[\hat{\theta}^{*(\alpha/2)}, \hat{\theta}^{*(1-\alpha/2)}\right]$$

```
quantile(xx["mean",],c(alpha/2,1-alpha/2))

     2.5%     97.5%
 47.82917 190.58542
```

Btw.: This is transformation invariant:

```
exp(quantile(xxL["mean",],c(alpha/2,1-alpha/2)))

     2.5%     97.5%
 47.82916 190.58542
```

The same using *boot* and *boot.ci*:

```
set.seed(123)
airBoot <- boot(aircondit$hours,function(x,i) mean(x[i]),R=5000)
boot.ci(airBoot,type="perc")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = airBoot, type = "perc")

Intervals :
Level     Percentile
95%   ( 46.0, 190.1 )
Calculations and Intervals on Original Scale
```

Why is this an "interesting" confidence interval?

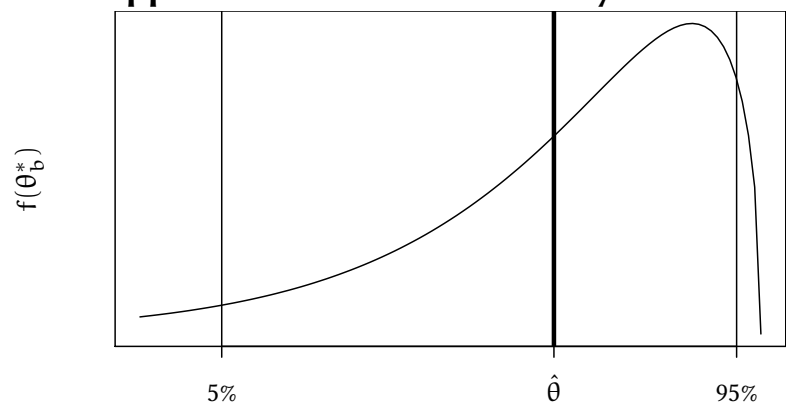
Percentile interval lemma
[Efron and Tibshirani] Suppose the transformation $\hat{\phi} = m(\hat{\theta})$ perfectly normalises the distribution of $\hat{\theta}$:

$$\hat{\phi} \sim N(\phi, c^2)$$

Then

$$CI_{\%} = \left[\hat{\theta}^{*(\alpha/2)}, \hat{\theta}^{*(1-\alpha/2)}\right] = \left[m^{-1}\left(\hat{\phi} - c \cdot z^{1-\alpha/2}\right), m^{-1}\left(\hat{\phi} - c \cdot z^{\alpha/2}\right)\right]$$

What happens if the intervals are asymmetric?



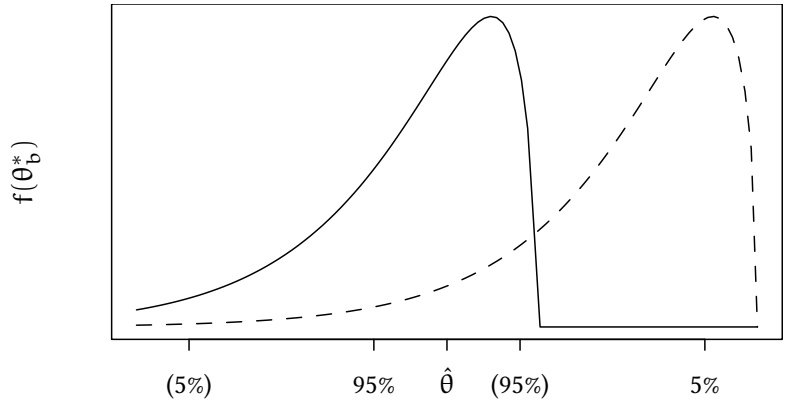
Note that the bootstrap-t and the percentile interval come to opposite conclusions. If θ_b^* has a long left tail, then the percentile interval will deviate from the mean more to the left than to the right.

The bootstrap-t will do the opposite.

Properties of percentile intervals:

+ Transformation-respecting (for monotone transformations).
+ Range-preserving (since the plug-in $\hat{\theta}$ obeys the same restriction as θ)
– Not necessarily unbiased (if $\hat{\theta}$ is biased)

6.5 Basic intervals



Let us assume that $\hat{\theta} - \theta$ follows the same distribution as $\hat{\theta}^B - \hat{\theta}$.

$$\text{CI}_{\text{basic}} = \left[2\hat{\theta} - \hat{\theta}^{*1-(\alpha/2)}, 2\hat{\theta} - \hat{\theta}^{*(\alpha/2)}\right]$$

```
2*thetaHat-quantile(xx["mean",],c(1-alpha/2,alpha/2))

    97.5%      2.5%
 25.58125 168.33750
```

The same with *boot* and *boot.ci*:

```
set.seed(123)
airBoot <- boot(aircondit$hours,function(x,i) mean(x[i]),R=5000)
boot.ci(airBoot,type="basic")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = airBoot, type = "basic")

Intervals :
Level      Basic
95%   ( 26.1, 170.2 )
Calculations and Intervals on Original Scale
```

Properties of basic intervals:

- Not transformation-respecting.
- Not range-preserving
+ Bias-correcting if $\hat{\theta} - \theta$ follows the same distribution as $\hat{\theta}^B - \hat{\theta}$

6.6 BC_a-intervals

BC_a=bias corrected, accelerated

Assume, there exists a monotone increasing transformation $g()$ and constants z_0 and a, such that:

$$g(\hat\theta) = g(\theta) + (1 + a \cdot g(\theta)) \cdot (Z - z_0) \quad \text{with } Z \sim N(0,1)$$

- z_0: bias correction
- a: acceleration (corrects for $se(\hat\theta) \neq$ const)

We do not know z_0 and a, but we can estimate them.

Plan: find (implicitely) the optimal transformation $g()$, apply it, determine CI, transform back.

Remember: Percentile intervals: $CI_{\%} = \left[\hat\theta^{*(\alpha/2)}, \hat\theta^{*(1-\alpha/2)}\right]$

Now we adjust the $\alpha/2$ and $1-\alpha/2$ levels: $BC_a = \left[\hat\theta^{*(\alpha_1)}, \hat\theta^{*(\alpha_2)}\right]$

with

$$\begin{aligned}\alpha_1 &= \Phi\left(\hat z_0 + \frac{\hat z_0 + z^{(\alpha/2)}}{1 - \hat a(\hat z_0 + z^{(\alpha/2)})}\right)\\ \alpha_2 &= \Phi\left(\hat z_0 + \frac{\hat z_0 + z^{(1-\alpha/2)}}{1 - \hat a(\hat z_0 + z^{(1-\alpha/2)})}\right)\end{aligned}$$

- $\hat z_0 = \Phi^{-1}\left(\frac{\#\{\hat\theta^*(b) < \hat\theta\}}{B}\right)$: bias correction
- $\hat a = \frac{\sum_{i=1}^n (\hat\theta_{(\cdot)} - \hat\theta_{(i)})^3}{6\left(\sum_{i=1}^n (\hat\theta_{(\cdot)} - \hat\theta_{(i)})^2\right)^{3/2}}$: acceleration (corrects for $se(\hat\theta) \neq$ const)

Note first that if $\hat z_0 = 0$ and $\hat a = 0$ then $\alpha_1 = \Phi(z^{(\alpha/2)}) = \alpha/2$ and $\alpha_2 = \Phi(z^{(1-\alpha/2)}) = 1 - \alpha/2$.

$\hat z_0$ shifts the interval to the left or to the right. $\hat a$ makes the interval wider or smaller.

- BC_a is *transformation respecting* (as percentile)
- BC_a converges faster than the standard and the percentile method.

```
jack <- function(x,FUN) sapply(1:length(x),function(i) FUN(x[-i]))
u <- jack(aircondit$hours,mean)
```

$$\hat a = \frac{\sum_{i=1}^n (\hat\theta_{(\cdot)} - \hat\theta_{(i)})^3}{6\left(\sum_{i=1}^n (\hat\theta_{(\cdot)} - \hat\theta_{(i)})^2\right)^{3/2}}$$

```
(acc <- sum ((mean(u) - u)^3) / (6 * sum((mean(u)-u)^2)^(3/2)))

[1] 0.09379807
```

$$\hat{z}_0 = \Phi^{-1}\left(\frac{\#\{\hat{\theta}^*(b)<\hat{\theta}\}}{B}\right)$$

```
(z0 <- qnorm(mean(xx["mean",]<thetaHat)))

[1] 0.1095075
```

$$\alpha_1 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z^{(\alpha/2)}}{1-\hat{a}(\hat{z}_0 + z^{(\alpha/2)})}\right)$$
$$\alpha_2 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z^{(1-\alpha/2)}}{1-\hat{a}(\hat{z}_0 + z^{(1-\alpha/2)})}\right)$$

```
alphaI <- function(alpha) {
    zA <- z0 + qnorm(alpha);
    pnorm(z0 + zA/(1-acc*zA))
}
```

$$\mathrm{BC}_a = \left[\hat{\theta}^{*(\alpha_1)}, \hat{\theta}^{*(\alpha_2)}\right]$$

```
quantile(xx["mean",],c(alphaI(alpha/2),alphaI(1-alpha/2)))

7.115143% 99.62907%
  57.0000  224.4335
```

Since people need BC_a frequently, the function is provided in the library *boot*.

```
airBoot <- boot(aircondit$hours,function(x,i) mean(x[i]),R=5000)
boot.ci(airBoot,type="bca")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = airBoot, type = "bca")

Intervals :
Level       BCa
95%   ( 55.1, 223.4 )
Calculations and Intervals on Original Scale
```

Comparison:

```
airBoot <- boot(aircondit$hours,
                function(x,i) c(mean(x[i]),var(x[i])/length(x)),R=5000)
boot.ci(airBoot,type="all")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 5000 bootstrap replicates

CALL :
boot.ci(boot.out = airBoot, type = "all")

Intervals :
Level      Normal              Basic             Studentized
95%   ( 34.5, 182.0 )   ( 26.1, 170.2 )   ( 47.3, 293.8 )

Level     Percentile            BCa
95%   ( 46.0, 190.1 )   ( 55.1, 223.4 )
Calculations and Intervals on Original Scale
```

```
abc.ci(aircondit$hours,function(x,w) weighted.mean(x,w))

[1]   0.95000  57.19276 226.71727
```

And, as a reminder, the values from the "exact" model:

```
[1]  65.89765 209.17415
```

…and the values from the normal (non-bootstrap) approximation:

```
               2.5 %   97.5 %
(Intercept) 21.52561 194.6411
```

6.7 Comparison

	BC_a	standard	percentile	bootstrap-t
transformation respecting	+		+	
speed	$1/n$	$1/\sqrt{n}$	$1/\sqrt{n}$	$1/n$

Exercise 6.1 *The file* `x3.csv` *contains a vector of numbers.*

1. *Determine the mean of these numbers.*
2. *You assume that these numbers follow a* χ^2 *distribution with* n *degrees of freedom. What is the distribution of the mean of these numbers?*
3. *Based on the* χ^2 *distribution, determine a* 95% *confidence interval for the mean.*

4. *Assume that the mean follows a normal distribution. Determine the confidence interval on the basis of the parametric estimate of the standard deviation.*
5. *Assume that the mean follows a normal distribution. Determine the confidence interval on the basis of the bootstrap estimate of the standard deviation.*
6. *Determine the bootstrap-t confidence interval.*
7. *Determine the percentile confidence interval.*
8. *Determine the BC_a confidence interval.*

7 Hypothesis Testing

7.1 Permutation tests

Null hypotheses Idea: we have two samples, $\mathbf{x}$ and $\mathbf{y}$, drawn from perhaps the same, perhaps different distributions:

$$\begin{aligned} F &\rightarrow \mathbf{x} = (x_1, x_2, \cdots, x_n) \\ G &\rightarrow \mathbf{y} = (y_1, y_2, \cdots, y_m) \end{aligned}$$

Our null hypothesis is

$$H_0 : F = G$$

Let us assume in this example that we do a one-sided test, i.e. our alternative hypothesis would assume that values of $\mathbf{x}$ are larger on average than those of $\mathbf{y}$.

Let us rephrase the problem in terms of a test statistic:

$$\hat{\theta} = \bar{\mathbf{x}} - \bar{\mathbf{y}}$$

What can we conclude from observing that $\hat{\theta}$ is large?
We expect $\hat{\theta}$ to be small as long as H_0 holds.
Call $\hat{\theta}^*_{H_0}$ the bootstrap of $\hat{\theta}$ under H_0.

We define the *achieved significance level* as follows:

$$\text{ASL} = \Pr\{\hat{\theta}^*_{H_0} \geqslant \hat{\theta}\}$$

ASL versus p-value:

- The p-value knows the distribution of $\hat{\theta}$ under H_0.
- The ASL only knows an estimate of this distribution.

$$\text{p-value} = \Pr\{\hat{\theta}_{H_0} \geqslant \hat{\theta}\}$$

- If we know that $F = N(\mu_x, \sigma^2)$ and $G = N(\mu_y, \sigma^2)$,
  then our null hypothesis H_0 is equivalent to $\mu_x = \mu_y$.
- If H_0 holds, $\hat\theta \sim N\left(0, \sigma^2 \cdot \left(\frac{1}{n} + \frac{1}{m}\right)\right)$.
  Then

$$\begin{aligned} \text{ASL} &= \Pr\{\hat\theta^*_{H_0} \geqslant \hat\theta\} \\ &= \Pr\left\{Z \geqslant \frac{\hat\theta}{\sigma\sqrt{1/n + 1/m}}\right\} \\ &= \Phi\left(-\frac{\hat\theta}{\sigma\sqrt{1/n + 1/m}}\right) \end{aligned}$$

In a Normal world $\hat\theta$ follows a "known" distribution.
We can bootstrap the distribution of $\hat\theta$ under H_0:

- We *join* the two samples $(\mathbf{x}, \mathbf{y})$ and *split* it randomly into two subsamples $\mathbf{x}'$ and $\mathbf{y}'$ of size n and m, respectively (i.e. we forget the original association to F and G).

The two-sample permutation test statistic

1. We choose B independent subsamples $\mathbf{x}'$ and $\mathbf{y}'$ taken (without replacement) from the joint sample $(\mathbf{x}, \mathbf{y})$.
2. We calculate $\hat\theta^*(b)$ for each sample.
3. $\widehat{\text{ASL}}_{\text{perm}} = \frac{1}{B} \cdot \#\{\hat\theta^*(b) > \hat\theta\}$

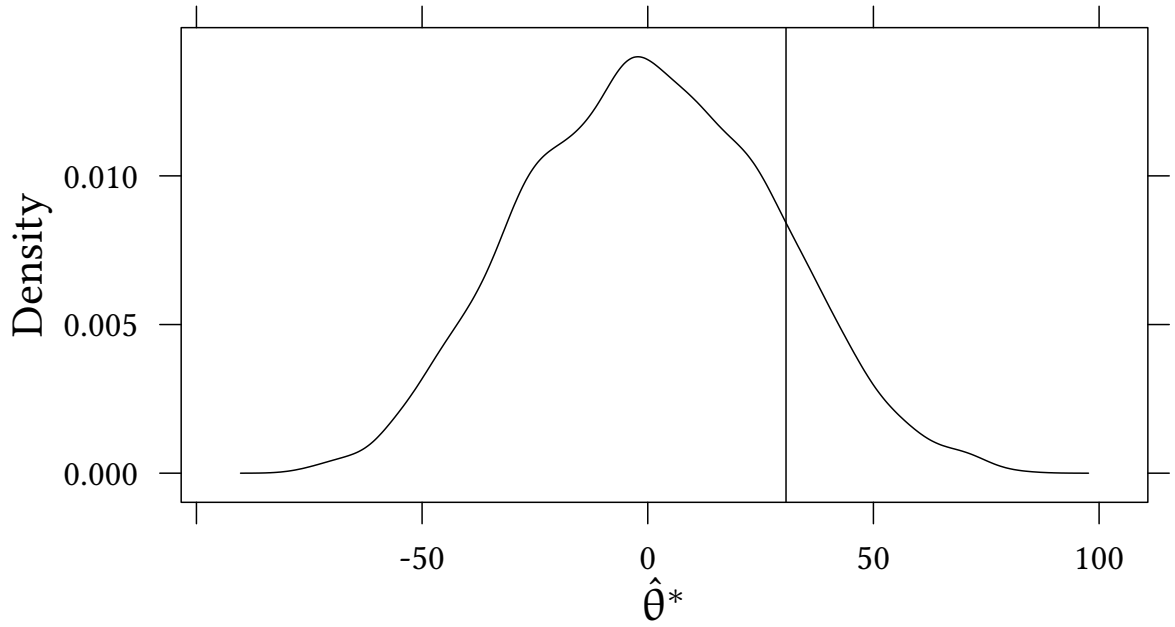
Ideally, we would take $B = \binom{N}{n}$ different subsamples to calculate the exact ASL:

$$\text{ASL}_{\text{perm}} = \#\{\hat\theta^*(b) > \hat\theta\} \Big/ \binom{N}{n}$$

```
t.test(mouse.t,mouse.c,alternative="greater",var.equal=TRUE)


Two Sample t-test

data:  mouse.t and mouse.c
t = 1.1214, df = 14, p-value = 0.1405
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -17.48178       Inf
sample estimates:
mean of x mean of y
 86.85714  56.22222
```

```
thetaHat <- mean(mouse.t) - mean(mouse.c)
xy <- c(mouse.c,mouse.t)
n <- length(mouse.t);N <- length(xy)
thetaStar <- replicate(5000,{indx <- sample(1:N,n);
                             mean(xy[indx])-mean(xy[-indx])})
mean(thetaStar>thetaHat)

[1] 0.1396
```

```
densityplot(thetaStar,plot.points=FALSE,xlab="$\\hat\\theta^*$")+layer(panel.abline(v=thetaHat
```



7.2 Bootstrap tests

The two-sample bootstrap test statistic

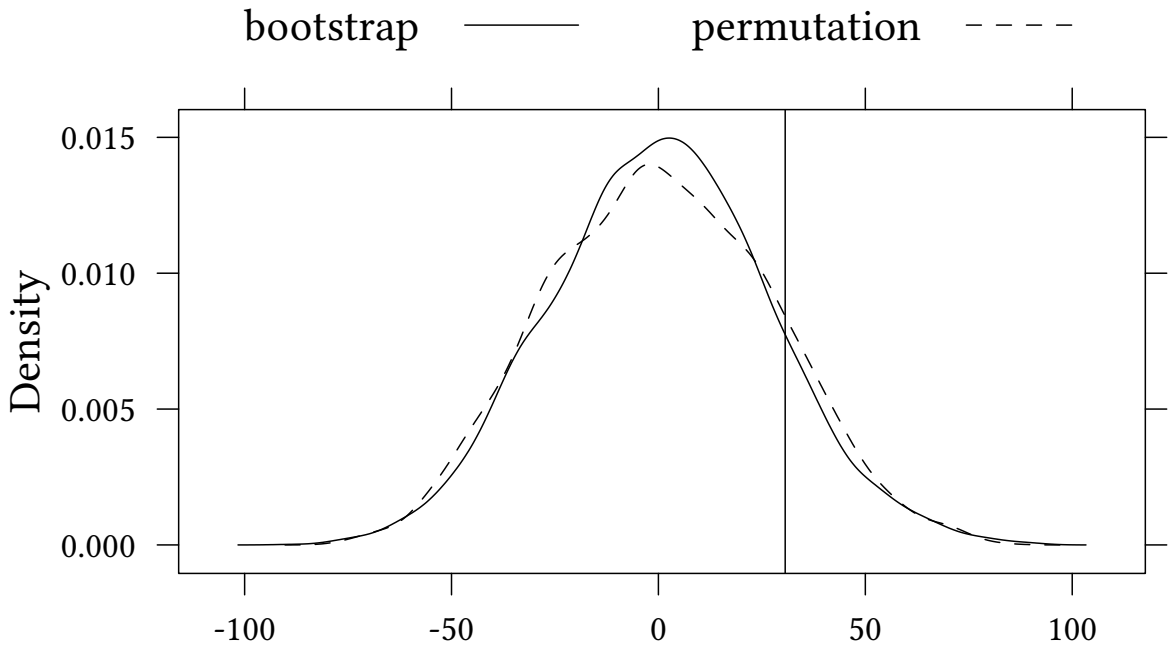
1. We choose B independent subsamples $\mathbf{x}'$ and $\mathbf{y}'$ taken (*with* replacement) from the joint sample $(\mathbf{x}, \mathbf{y})$.
2. We calculate $\hat{\theta}^*(b)$ for each sample.
3. $\widehat{\mathrm{ASL}}_{\mathrm{BS}} = \frac{1}{B} \cdot \#\{\hat{\theta}^*(b) > \hat{\theta}\}$

- Permutation test: sample *without* replacement
- Bootstrap test: sample *with* replacement

n_x	n_y	$\binom{n_x+x_y}{n_x}$	$\binom{n_x+x_y}{n_x}\binom{n_x+x_y}{n_y}$
3	2	10	100
10	10	184756	3.41348×10^{10}
30	20	4.71292×10^{13}	2.22116×10^{27}
100	100	9.05485×10^{58}	8.19903×10^{117}

```
xy <- c(mouse.c,mouse.t)
n <- length(mouse.t)
m <- length(mouse.c)
N <- length(xy)
thetaStar2 <- replicate(5000,mean(sample(xy,n,replace=TRUE))-
                             mean(sample(xy,m,replace=TRUE)))
mean(thetaStar>thetaHat)

[1] 0.1396

mean(thetaStar2>thetaHat)

[1] 0.1256
```

```
densityplot(~thetaStar2+thetaStar,xlab="",plot.points=FALSE,
         key=simpleKey(c("bootstrap","permutation"),points=FALSE,lines=TRUE,columns=2))+
    layer(panel.abline(v=thetaHat))
```



7.3 Bootstrap-t tests

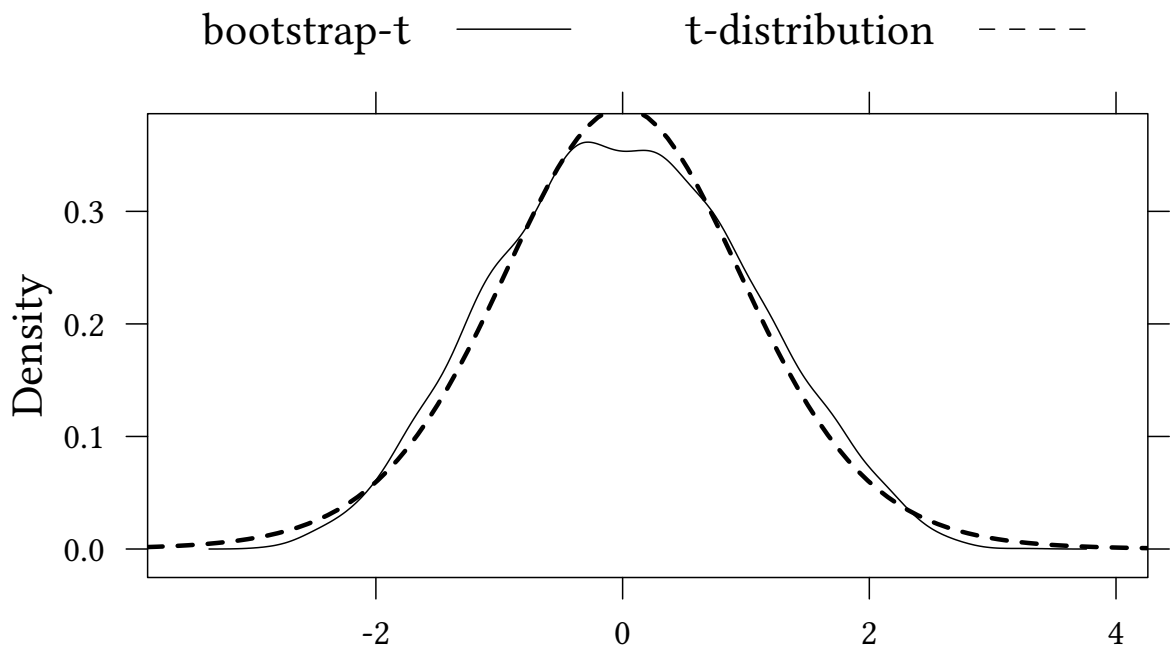
The two-sample bootstrap-t test statistic

1. We choose B independent subsamples $\mathbf{x}'$ and $\mathbf{y}'$ taken (*with* replacement) from the joint sample $(\mathbf{x}, \mathbf{y})$.
2. We calculate $\hat{\theta}^*(b)$ and $\hat{\sigma}^*_{\hat{\theta}}(b)$ for each sample.
3. $\widehat{\text{ASL}}_{\text{BS-t}} = \frac{1}{B} \cdot \# \left\{ \frac{\hat{\theta}^*(b)}{\hat{\sigma}^*_{\hat{\theta}}(b)} > \frac{\hat{\theta}}{\hat{\sigma}_{\hat{\theta}}} \right\}$

Let us first consider H_0 that the means and the standard deviations in both samples are the same:

```
tStat <- replicate(5000,{
    xx<-sample(xy,n,replace=TRUE);
    yy<-sample(xy,m,replace=TRUE);
    (mean(xx)-mean(yy))/(sd(c(xx,yy)*sqrt(1/n+1/m)))})
```

```
densityplot(tStat,plot.points=FALSE,
            key=simpleKey(c("bootstrap-$t$","$t$-distribution"),
                  points=FALSE,lines=TRUE,lty=c(1,2),columns=2),xlab="")+
        layer(panel.curve(dt(x,df=n+m-2),col=mPal2[2],lty=2,lwd=3))
```




```
(ASLt <- mean(tStat > thetaHat/(sd(xy)*sqrt(1/n+1/m))))
```

```
[1] 0.153
```

```
mean(thetaStar>thetaHat)
```

```
[1] 0.1396
```

```
mean(thetaStar2>thetaHat)
```

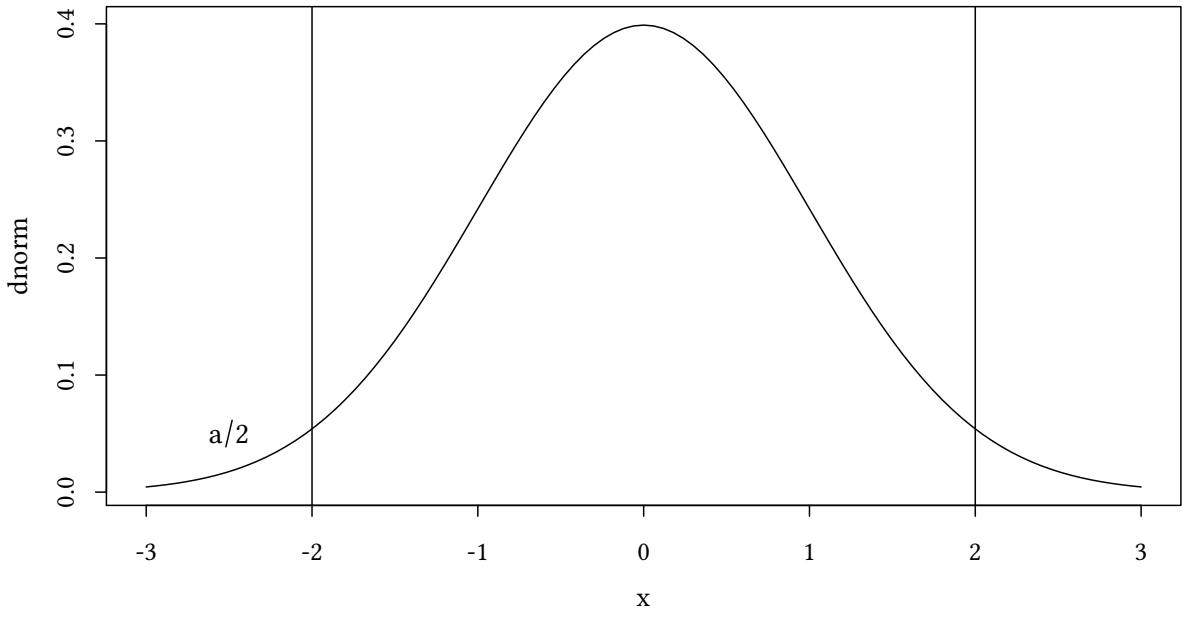
```
[1] 0.1256
```

Exercise 7.1 *At the beginning of this section we assumed identical standard deviations in both samples. Perform the same test without making this assumption.*

7.4 BC_a-tests

Above we have seen that the BC_a method can help us to determine confidence intervals more efficiently. Can we use this method for tests of hypotheses, too? Let us assume that all we have is a machine that calculates confidence intervals, but we have to calculate an ASL.

Idea:



We have to work out a level α such that the upper or lower end of the CI coincides with the value of θ under H_0.

Let us assume that we are interested in the lower end of the CI (i.e. H_1 states that $\theta > 0$).

Remember that

$$BC_a = \left[\hat{\theta}^{*(\alpha_1)}, \hat{\theta}^{*(\alpha_2)}\right]$$

with

$$\alpha_1 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z^{(\alpha/2)}}{1 - \hat{a}(\hat{z}_0 + z^{(\alpha/2)})}\right)$$

In our permutation test we observe α_1. We can use the above formula to work out $\alpha/2$:

$$\begin{aligned}\Phi^{-1}(\alpha_1) &= \hat{z}_0 + \frac{\hat{z}_0 + z^{(\alpha/2)}}{1-\hat{a}(\hat{z}_0 + z^{(\alpha/2)})}\\ \underbrace{\Phi^{-1}(\alpha_1) - \hat{z}_0}_{M} &= \frac{\hat{z}_0 + z^{(\alpha/2)}}{1-\hat{a}(\underbrace{\hat{z}_0 + z^{(\alpha/2)}}_{Q})}\end{aligned}$$

$$\begin{aligned}M &= \frac{Q}{1-\hat{a}Q}\\ M\cdot(1-\hat{a}Q) &= Q\\ M-\hat{a}QM &= Q\\ M &= Q\cdot(1+\hat{a}M)\\ \frac{M}{1+\hat{a}M} &= Q = \hat{z}_0 + z^{(\alpha/2)}\\ \frac{M}{1+\hat{a}M} - \hat{z}_0 &= z^{(\alpha/2)}\\ \Phi\left(\frac{M}{1+\hat{a}M} - \hat{z}_0\right) &= \alpha/2\\ \Phi\left(\frac{\Phi^{-1}(\alpha_1)-\hat{z}_0}{1+\hat{a}(\Phi^{-1}(\alpha_1)-\hat{z}_0)} - \hat{z}_0\right) &= \mathrm{ASL}_{\mathrm{BC}_a}\end{aligned}$$

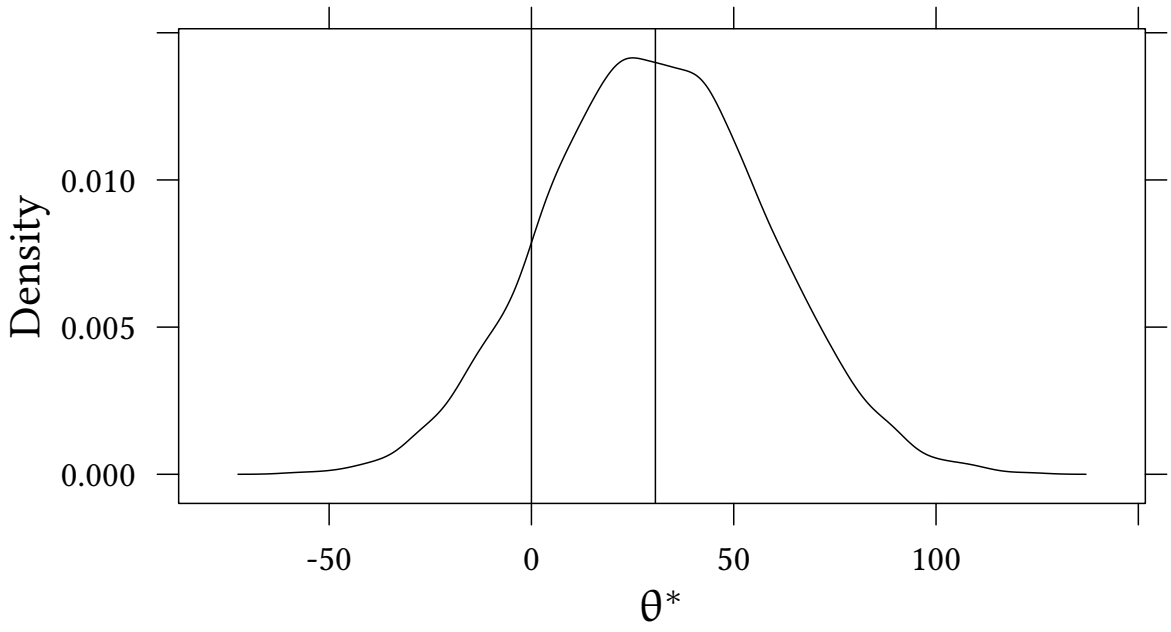
- We can use the same formula as above for $\hat{z}_0$:
  - $\hat{z}_0 = \Phi^{-1}\left(\frac{\#\{\hat{\theta}^*(b) < \hat{\theta}\}}{B}\right)$ (bias correction)
- The formula for $\hat{a}$ is slightly different with a two sample problem:
  We define $U_{x,i} = (n-1)(\hat{\theta}_{x,(\cdot)} - \hat{\theta}_{x,(i)})$ and $U_{y,i} = (m-1)(\hat{\theta}_{y,(\cdot)} - \hat{\theta}_{y,(i)})$

$$\hat{a} = \frac{1}{6}\frac{\frac{1}{n^3}\sum_{i=1}^{n} U_{x,i}^3 + \frac{1}{m^3}\sum_{i=1}^{m} U_{y,i}^3}{\left(\frac{1}{n^2}\sum_{i=1}^{n} U_{x,i}^2 + \frac{1}{m^2}\sum_{i=1}^{m} U_{y,i}^2\right)^{3/2}} \qquad \text{(acceleration)}$$

```
set.seed(123)
FUN <- function(x,y) mean(x)-mean(y)
thetaStar <- replicate(5000,
                       FUN(sample(mouse.t,replace=TRUE),
                           sample(mouse.c,replace=TRUE)))
thetaHat <- FUN(mouse.t,mouse.c)
z0 <- qnorm(mean(thetaStar<thetaHat))
jack2 <- function(x,y) sapply(1:length(x),function(i) FUN(x[-i],y))
U <- function(x,y) { u <- jack2(x,y) ; (length(u) - 1 ) * (mean(u) - u )}
Ux <- U(mouse.t,mouse.c)
Uy <- U(mouse.c,mouse.t)
n <- length(mouse.t)
m <- length(mouse.c)
ahat <- (sum(Ux^3)/n^3+sum(Uy^3)/m^3)/(sum(Ux^2)/n^2+sum(Uy^2)/m^2)^(3/2)/6
a1 <- mean(thetaStar<0)
ASLBCa <- pnorm((qnorm(a1) - z0)/(1+ahat*(qnorm(a1) - z0))-z0)
```

We obtain $z_0 = -0.00301$, $\hat{a} = 0.0272$, $a_1 = 0.122$, and $\text{ASL}_{\text{BC}_a} = 0.116$.

```
densityplot(thetaStar,plot.points=FALSE,xlab="$\\theta^*$")+
    layer(panel.abline(v=c(0,thetaHat)))
```



Exercise 7.2 *The dataset* `x4.csv` *contains data on* x *and* y*. You estimate a linear relationship*

$$y = \beta_0 + \beta_1 x + \epsilon$$

Your null hypothesis is that $\beta_1 = 0$*. The alternative hypothesis is that* $\beta_1 \neq 0$*.*

1. *What is the ASL of a parametric test?*

2. *What is the ASL of a permutation test?*
3. *What is the ASL of a bootstrap-*$\mathfrak{t}$ *test?*
4. *What is the ASL of a* $BC_{\mathfrak{a}}$ *test?*