Resampling

Etienne Low-Décarie

2018-01-07

Contents

Original produced by Tom Cameron

Randomization methods

We will try some randomization methods - bootstrapping, jackknifing and permutation.

Type all the code in a script file (saving it periodically) and then, when you want to rerun the analyses on different data, you can do so at the touch of a button

- Here is data from a population of black grouse with the size of the leks (mating grounds, in an area measurement):

```
Lek<-seq(1,16)
Size<-c(25,32,47,96,10,46,40,90,18,12,8,9,25,14,16,35)
Number<-c(2,0,8,0,0,5,2,6,4,1,0,1,3,0,0,0)
```

This simply means generate 3 columns of data called Lek, Size and Number; the first is a sequence of numbers from 1 to 16, the second two mean that I am entering by hand as a column of data. Now, to join those 3 columns together in a data.frame, type:

```
Grouse<-data.frame(cbind(Lek,Size,Number))
```

This means create a new data.frame called grouse by column-binding Lek, Size and Number. * Produce a plot showing how the number of black grouse on a lek changes with its size * Now we want to ask the following questions: * What is the strength of the correlation between lek size and number of birds? cor(Number, Size) * Is this correlation signficantly different from zero? cor.test(Number, Size) * This indicates that r = 0.3907 and P = 0.1346 * The problem we have is that we only have 16 leks and so we are not sure about the shape of the underlying distribution (i.e. Poisson, negative binomial or something else). Also, there are one or two potential outliers that could produce a spurious relationship or obscure a genuine one. These are exactly the sorts of problems that are randomization methods were developed to cope with.

Bootstrapping

- Bootstrapping will allow us to determine a more accurate estimate of r and its standard error, by taking account of any skew in the data
- To obtain bootstrapped estimates of the correlation between Lek Size and Number of birds. Use this code:

```
corr<-function(d,i){
cor(d[i,2],d[i,3]) # this takes pairs of data from columns 2 and 3 of
}           # the data frame and randomly chosen row i
```

```
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     corr
```

```
boot1<-boot(Grouse,corr,1000)
boot.ci(boot1,conf=0.95,type=c("norm","basic","perc","bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot1, conf = 0.95, type = c("norm", "basic",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.1738,  0.9063 )   (-0.0657,  0.9487 )
##
## Level     Percentile            BCa
## 95%   (-0.1673,  0.8471 )   (-0.3209,  0.7916 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

The first bit of code creates a function to take the data and correlate samples of it (as specified by randomly chosen indices – the "i"s coming from the boot function). The final bit performs the bootstrap and from it, calculates various types of confidence interval. * Now type: plot(boot1)

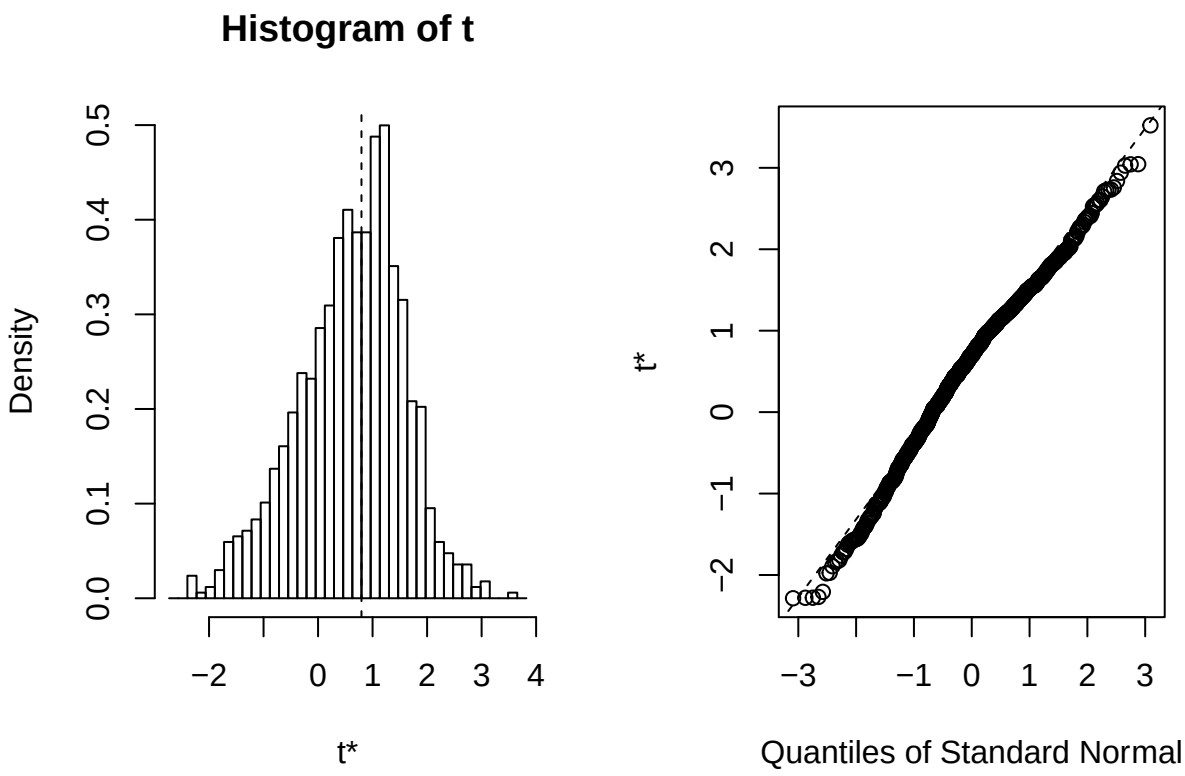
> Q: How do you interpret this output? Q: What is the best estimate of the correlation coefficient and its s.e.? Q: What is the bias-corrected 95% confidence interval? Note. The following code bootstraps the coefficients of a simple regression model (y~x), with the output passing the coefficients (for intercept and slope) which are stored in the boot.obj in t (the table of results) i.e. boot.obj$t[,1] is the intercept and t[,2] is the slope].

```
regs<-function(d,i){
  mod<-lm(d[i,3]~d[i,2])
  return(coefficients(mod))
}
```

```
b<-boot(data=Grouse,statistic=regs, R=1000)
```

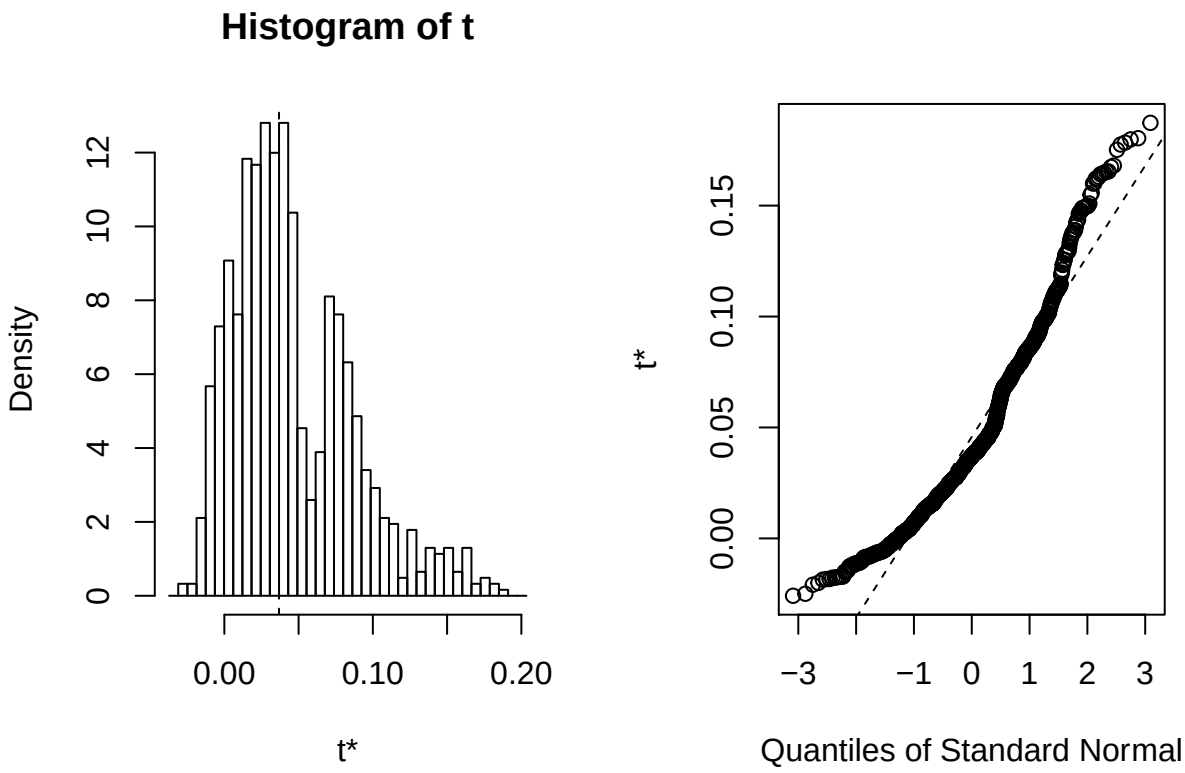
plots the intercept estimates

```
plot(b,index=1)
```



plots the slope estimates

```
plot(b,index=2)
```



gives the intercept CIs

```
boot.ci(b, type="all",index = 1, main="intercepts")
```

```
## Warning in boot.ci(b, type = "all", index = 1, main = "intercepts"):
## bootstrap variances needed for studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b, type = "all", index = 1, main = "intercepts")
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.8930,  2.8702 )   (-0.7685,  3.1337 )
##
## Level     Percentile            BCa
## 95%   (-1.5408,  2.3614 )   (-1.0448,  2.9016 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

gives the slope CIs

```
boot.ci(b, type="all",index = 2, main="slopes")
```

```
## Warning in boot.ci(b, type = "all", index = 2, main = "slopes"): bootstrap
## variances needed for studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b, type = "all", index = 2, main = "slopes")
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.0520,  0.1077 )   (-0.0757,  0.0847 )
##
## Level     Percentile            BCa
## 95%   (-0.0111,  0.1493 )   (-0.0179,  0.1287 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

Jackknifing

- Now let's check for outliers, using the jackknife-after-bootstrap: jack.after.boot(boot1)

  Q: How do you interpret this plot? ?jack.after.boot may help

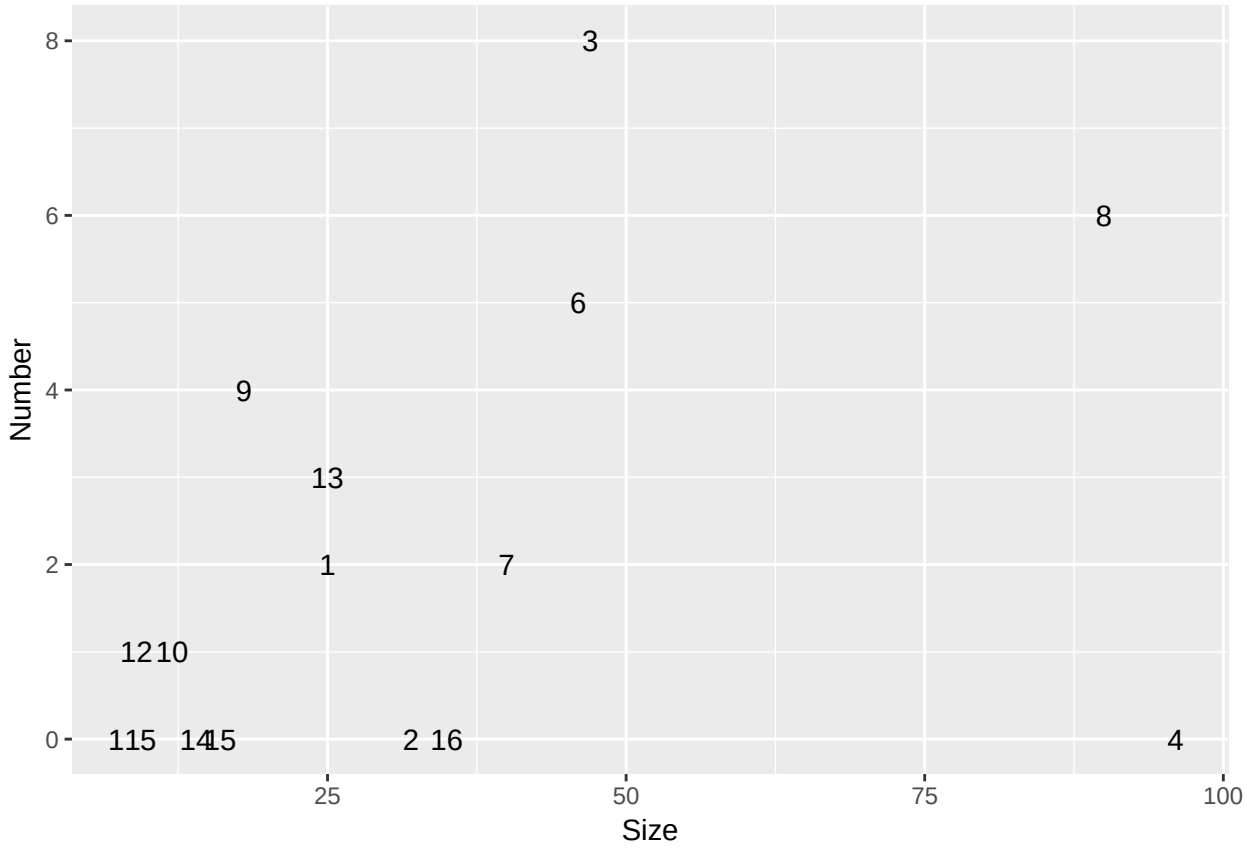
- Try plotting Number vs Size, with Lek number instead of symbols

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
p <- qplot(data=Grouse,
           y=Number,
```

```
          x=Size,
          label=Lek,
          geom="text")
print(p)
```



Q: Which lek is the outlier? (at this point, you would check for typos, etc)

Now let us jackknife the correlation from first principles: This function takes our data (in a frame `num_size<-data.frame(cbind(Number, Size))` and goes thru it line by line dropping each pair of data and recalculating the correlation, storing it in the array "jk_res".

```
num_size<-data.frame(cbind(Number, Size))
```

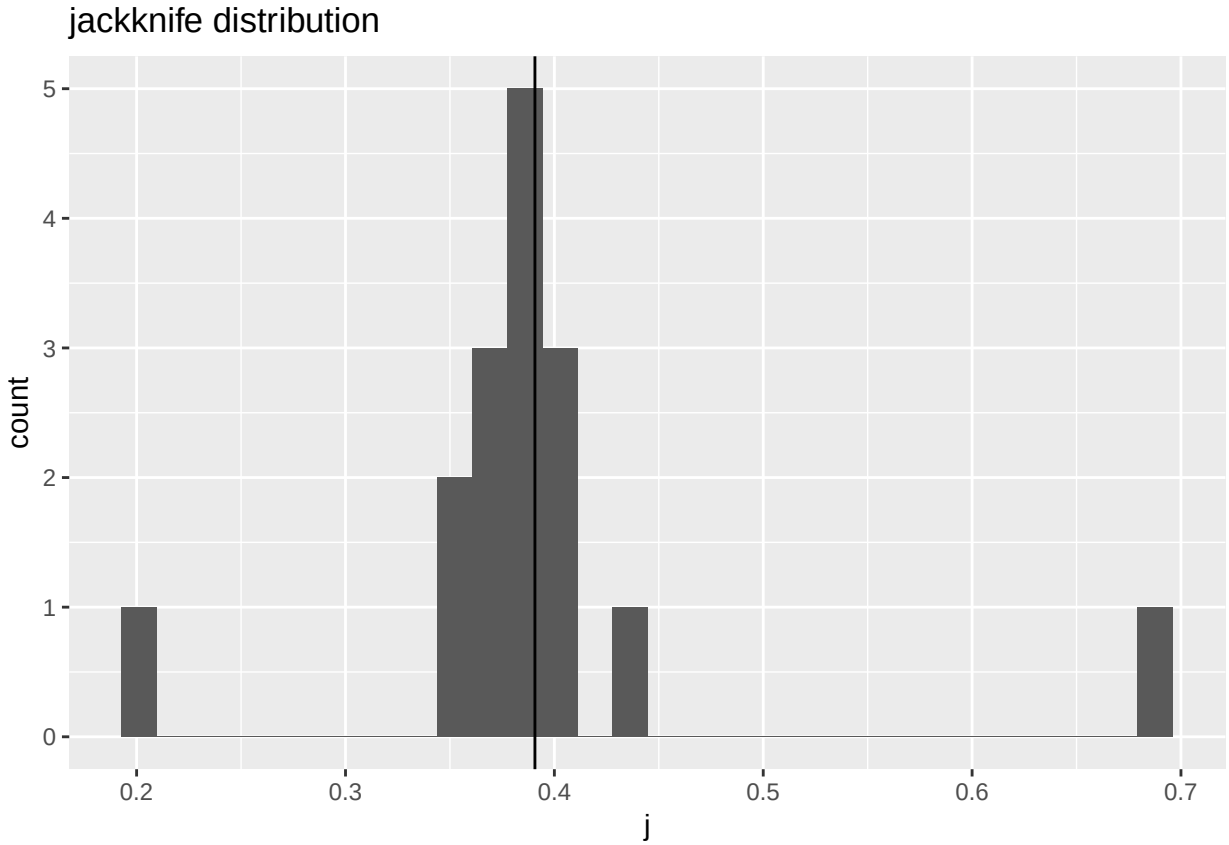
function to jack a correlation:

```
jkc<-function(d){
  n<-nrow(d)
  jk_res<-rep(0,n)
  tmp<-c(1:n)
  for(i in 1:n){
    jk_res[i]<-cor(d[eval(tmp!=i),1],d[eval(tmp!=i),2])
    }
  return(jk_res)
  }

j<-jkc(num_size)
```

Now plot the results to show observed correlations vs the distribution of jackknifed correlations:

```
p <- qplot(x=j,
           main="jackknife distribution")+
  geom_vline(xintercept = cor(num_size[,1],num_size[,2]))
print(p)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Look at j and see which pairs of data, when left out, make the biggest difference to the correlation. Are these outliers?

The descriptive statistics of the jackknife distribution can be calculated using code like this:

calculate parametric 95% CI based on mean and SD of distribution

```
error <- qt(0.975,df=(length(j)-1))*(sqrt(var(j))/sqrt(length(j)))
left <- mean(j)-error
right <- mean(j)+error
print(c(left,mean(j),right))
```

```
## [1] 0.3434076 0.3930863 0.4427650
```

Permutation

- Now let's determine whether the correlation coefficient is significantly different from zero for our Number vs Size correlation.

- To do this we'll randomly resample from our Number data column (1000 times) and ask the question: if our Numbers are randomly sampled from the population, what would be the expected value of the correlation coefficient? We can then compare our observed correlation with the Null distribution.

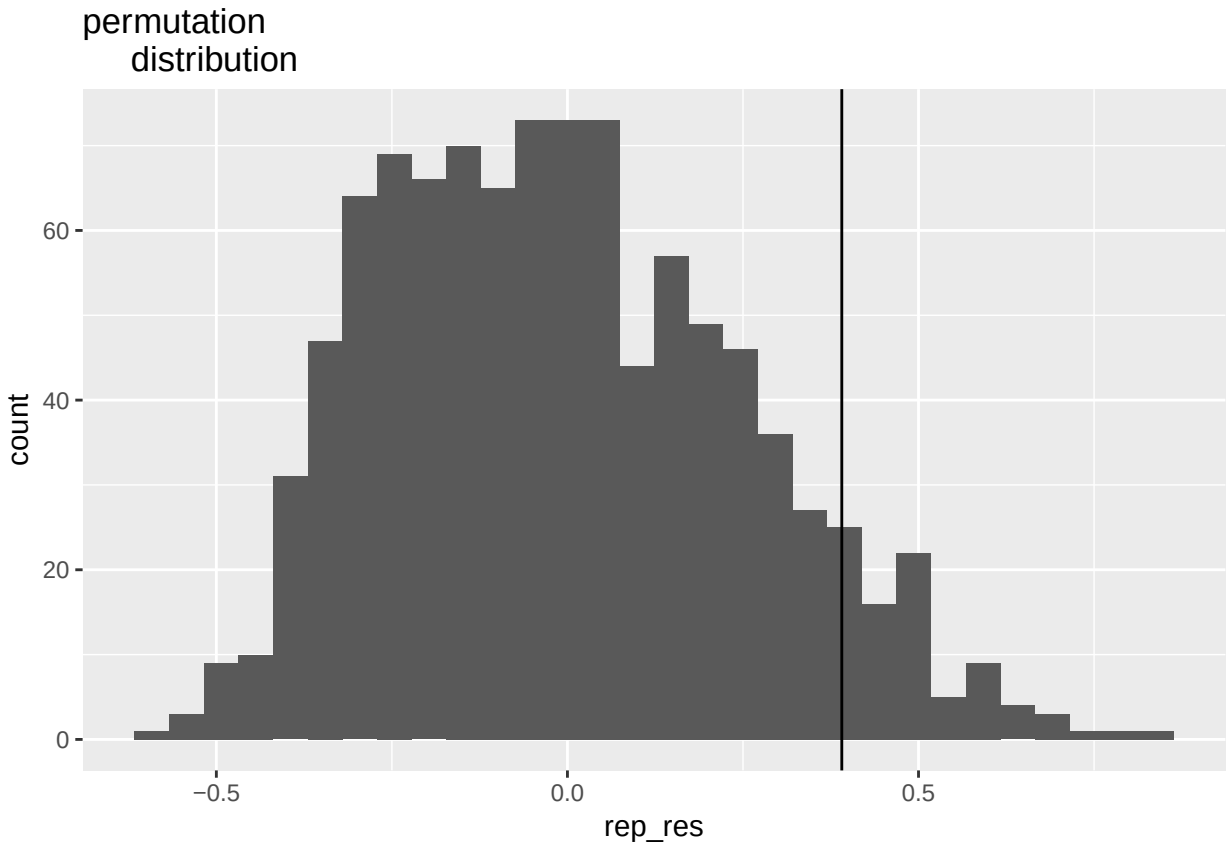
function to permute the data:

```
resample <- function(x, size){
  if(length(x) <= 1) {
    if(!missing(size) && size == 0) x[FALSE] else x
  }else sample(x, size, replace=FALSE)
}

x<-Number
y<-Size

ssize<-1000  #this sets our sample size
rep_res<-rep(0,ssize) #this sets up an array to put the results in
n<-length(x)

for(i in 1:ssize){  # this loops 1000 times collecting new samples
  samp<-resample(x, n)
  rep_res[i]<-cor(y,samp)       # and calcs the corr
}
```

this draws the graph

```
p <- qplot(x=rep_res,
main="permutation
     distribution" )+
  geom_vline(xintercept = cor(x, y))
print(p)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



NB: Check out the graph (this shows the null distribution of correlation coefficients and vertical line the observed)

Q: What is the significance of the correlation coefficient's departure from zero?

Q: How does this differ from bootstrap (in methods and results)?

Now you have been through the analyses with the practical data, try it again by running the scripts with new data (make up some, modify the data above, use your own data etc etc). Remember, to get the macros to run, you'll need to set up the dataframe called (data) with the x and y variables stacked on top of each other, and with a column containing the grouping variable.