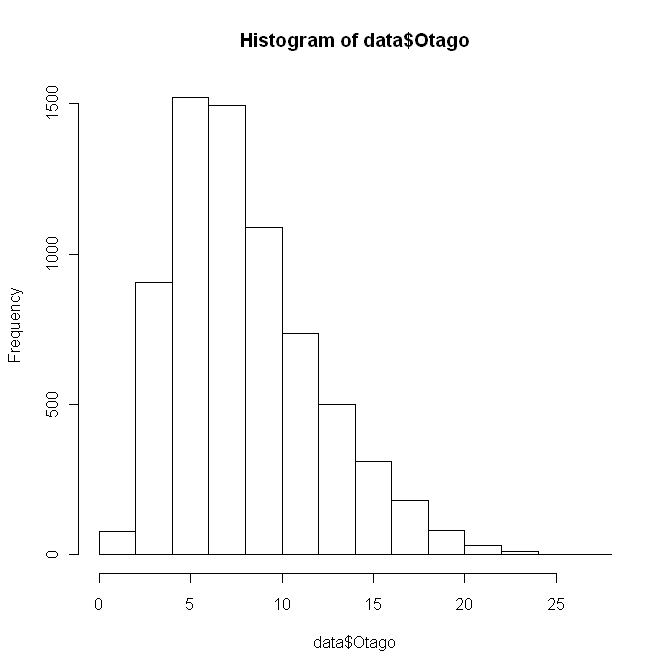
Amirhossein sadoghi

800519-2375

The data set ***wind.xls*** represents wind speed records for twelve Meridian sites distributed around the New Zealand. It has long been known that the wind speed matches well with the Weibull distribution. Your task is to make some inference about the parameter values for the city **Otago**

1. Load necessary information concerning Otago city to R. Plot the histogram of the wind speed and comment whether it looks like Weibull (you may also compare it with the shape of the related Rayleigh distribution)

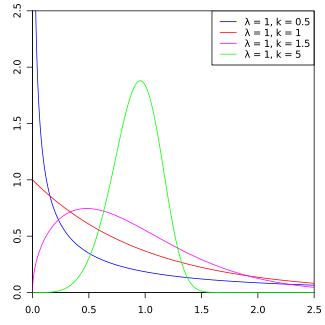


**\*0.5p**

In probability theory and statistics, the **Weibull distribution** is a continuousprobability distribution It gives the distribution of failures f(x;\lambda,k) =  \begin{cases}
\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-(x/\lambda)^{k}} & x\geq0\\
0 & x<0\end{cases}

where *k* > 0 is the *shape parameter* and λ > 0 is the *scale parameter* of the distribution

|  |  |
| --- | --- |
| parameters: | \lambda>0\, scale (real) k>0\, shape (real) |
| support: | x \in [0; +\infty)\, |
| [pdf](http://en.wikipedia.org/wiki/Probability_density_function): | f(x)=\begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-(x/\lambda)^{k}} & x\geq0\\ 0 & x<0\end{cases} |
| [cdf](http://en.wikipedia.org/wiki/Cumulative_distribution_function): | 1- e^{-(x/\lambda)^k} |
| mean: | \lambda \Gamma\left(1+\frac{1}{k}\right)\, |



as we can from pervious graphs this data can follow weibull distribution with adjusted scale and shape (the pink one) The Weibull distribution is a generalization of the Rayleigh distribution.with fixing shape parameter we can get Rayleigh distribution**\*0.5p**

1. Write your own function as a parameter of the data that finds the maximum log-likelihood estimates of the parameters *shape* and *scale* of the Weibull distribution by using an optimization method such as BFGS or CG. Compute the parameter estimates (if you get warnings, ignore them). What kind of problems may one encounter when these methods are used for the search of the maximum log-likelihood estimates? What other methods can you see as alternatives that do not have these problems?

Funloglikelihood<-function(vector){

landa<-vector[1];

k<-vector[2];

N<-length(data);

Result<- (N\*log(k)) - (N\*log(landa))-(N\*(k-1)\*sum(log(data[])))-(N\*(k-1)\*log(landa))-(sum(data[])/landa);

print(-Result);

}

Fungrad<-function(vector){

landa<-vector[1];

k<-vector[2];

N<-length(data);

Res<-c();

Res[1]<--(N/landa)-(k-1)\*N/landa - (log(landa)\*sum(data[]))

Res[2]<-(N/k)+sum(log(data[]))-N\*log(landa);

print(-Res);

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Optimization  Method | Initial value | value | parameter | function and gradient evaluations | Indicates  convergence |
| CG | 1 2 | 94174753 | 1 2 | 1701 101 | Maximum Iteration |
| CG | 5 2 | 94152677 | 5 2 | 1651 101 | Maximum Iteration |
| BFGS | 5 2 | 94152677 | 5 2 | 30 1 | success |
| BFGS | 1.0 1.5 | 47114728 | 1.0 1.5 | 31 1 | success |
| CG | 1.0 1.5 | 47114728 | 1.0 1.5 | 1701 101 | Maximum Iteration |
| CG | 2.0 1.5 | 47094184 | 2.0 1.5 | 1701 101 | Maximum Iteration |

optim(c(2,1.5), Funloglikelihood, Fungrad, method = "CG");

However log-likelihood function needs more evaluations for both gradient and main function in all methods. CG cannot converge in limited number of iteration but BFGS can have answer with less number of evaluations of main function and number of evaluations of gradients

rm(result)

Maxlikgamma<-function(vector){

scale<-vector[1];

shape<-vector[2];

result<-0;

result<-sum(log(dweibull(data[], shape=shape, scale = scale, log = FALSE)));

return (-result);

}

optim(c(2,1.5), Maxlikgamma, NULL, method = "CG");

1. Modify your function in such way that it returns only *shape* parameter. Consider this function as an estimator of *shape* and generate 100 bootstrap estimates of this

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Optimization  Method | Initial value | value | parameter | function and gradient evaluations | Indicates  convergence |
| CG | 2 1.5 | 9.063085 2.163650 | 18899.50 | 568 101 | Max Iteration |
| CG | 9 2 | 9.063085 2.163650 | 18899.50 | 660 101 |  |
| BFGS | 9 2 | 9.063085 2.163651 | 18899.50 | 28 4 | success |

**\*2p**

**\*0p No explanation of the problems with the methods and suggestions for alternatives**

1. Modify your function in such way that it returns only *shape* parameter. Consider this function as an estimator of *shape* and generate 100 bootstrap estimates of this parameter. Produce an appropriate plot and make comments. Compute 95% confidence bounds (**using percentile method only!**) for the parameter *shape* and answer whether the data can in principle come from Rayleigh distribution.
2. bounds (**using percentile method only!**) for the parameter *shape* and answer whether the data can in principle come from Rayleigh distribution.

Maxlikgamma<-function(vector,dat){

scale<-vector[1];

shape<-vector[2];

result<-0;

result<-sum(log(dweibull(dat[], shape=shape, scale = scale, log = FALSE)));

return (-result);

}

funcboot<-function(data,i){

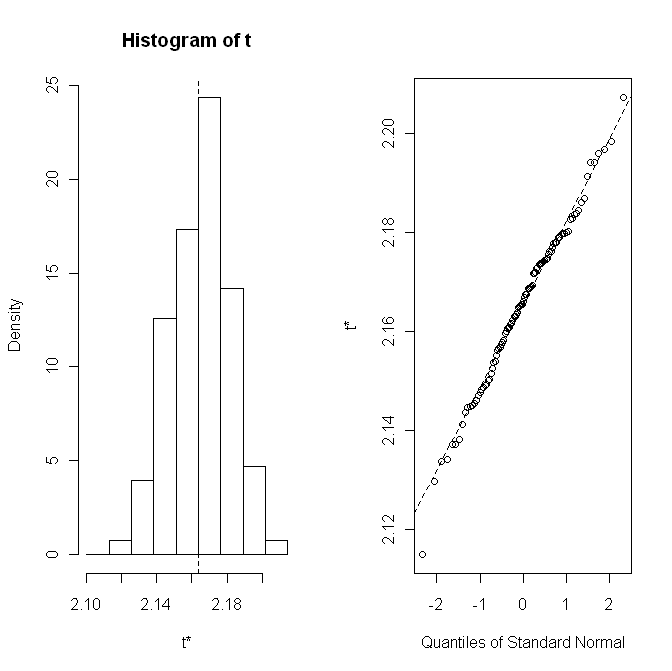
re <-optim(c(2,2), Maxlikgamma, NULL,dat = data[i],method = "BFGS");

return(re$par[2]);

}

result<-boot( data,funcboot, 100,sim="ordinary", stype="i");

CIconf<-boot.ci(result, conf=0.95, type="percent");**\*1.5p**



Bootstrap Statistics :

original bias std. error

t1\* 2.163650 -0.001139086 0.01843969

**\*0.5p**

Length Class Mode

R 1 -none- numeric

t0 1 -none- numeric

call 4 -none- call

***Assignment 2: Computing an integral***

Consider the distribution with density

1. Write your own function that can generate sample of size *n* from the distribution above using the acceptance-rejection method with uniform distribution as the majorizing density. The function should also compute what percent ***R***of the totally generated random numbers was rejected in this method. What was the value of the scaling constant *c* you used in deriving the majorizing function and why you have chosen this value?

rm(x);

mydis<- function (x)

{

result<- (pi\*(sin(pi\*x))/2);

return (result) ;

}

...............................................

rdexpacre<- function (n,c)

{

z<-c();

j<-0;

while(j<n)

{

u <-1;f<-1; g<-1;

x<- runif(1);

g<-dunif(x, min=0, max=1, log = FALSE)

u<-runif(1);

f <- mydis(x);

if (u**>** (f/(c\*g))) **\*WRONG sign**

{

j<-j+1;

z[j] <- x;

}

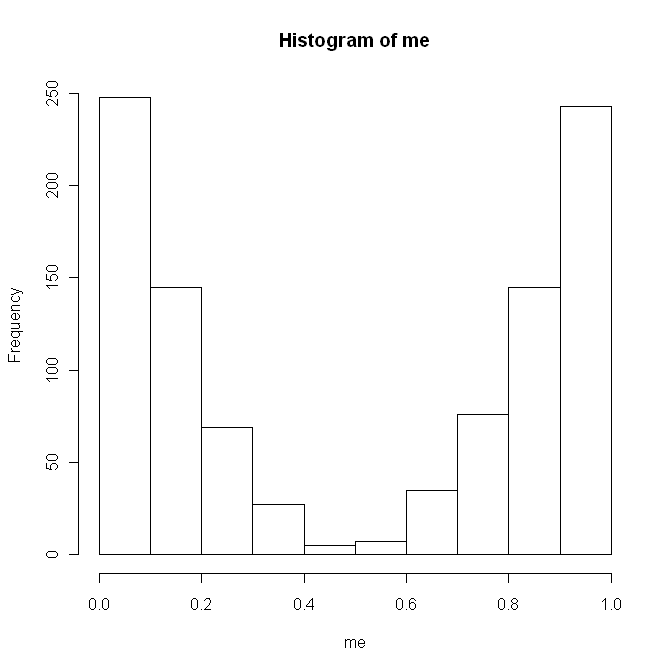
}

return(z)

}**\*1p**

me<-rdexpacre(1000,pi/2)

hist(me);



**\*0p Wrong histogram**

for calculation of maximum c:

C\*fy(x)>=fx(x) for 0<x<1

C\*(1-0)>=(pi\*sin(pi\*x)/2)

the maximum value of sin(pi\*x) is one so

C>=pi/2

**\*1p**

Generate a sample of size 1000 by using your function and plot the histogram. Does the sample look like it should?

1. Print out the computed rejection rate ***R***. Find out without using computer what would be the rejection rate ***R*** if the sample size would be very large (i.e., the expected value of ***R***).

rdexpacre<- function (n,c)

{

z<-c();

j<-0;

i<-0;

while(j<n)

{

u <-1;f<-1; g<-1;

x<- runif(1);

i<-i+1;

g<-dunif(x, min=0, max=1, log = FALSE)

u<-runif(1);

f <- mydis(x);

if (u> (f/(c\*g)))

{

j<-j+1;

z[j] <- x;

}

}

return(z,i)

}

me<-rdexpacre(1000,pi/2)

hist(me);

based on theory it should be equal C **\*0p WRONG**

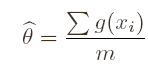
for generation 1000 it rejected 1519 that means our rejection rate is 1.5 **\*0.5p not rejected, but the total is 1519**

which is near our optimum value of C

with increasing C value we got more rejection rate

1. Use the obtained sample to compute the value of the integral to derive the value of the integral by applying the importance sampling. Why using the introduced above as importance function is better than using the density of the uniform distribution?

Estimate using the samples from steps 1

= 

rm(x);

mydiss<- function (x)

{

result<- sin(pi\*x)/(5+(x^2));

return (result) ;

}

rdexpacre<- function (n,c)

{

z<-c();

j<-0;

i<-0;

while(j<n)

{

u <-1;f<-1; g<-1;

x<- runif(1);

i<-i+1;

g<-dunif(x, min=0, max=1, log = FALSE)

u<-runif(1);

f <- mydiss(x);

if (u> (f/(c\*g)))

{

j<-j+1;

z[j] <- x;

}

}

return(z,i)

}

me<-rdexpacre(1000,3)

mean<-mean(me$z) **\*0p WRONG**

mean

[1] 0.5054762

rm(x);

mydiss<- function (x)

{

result<- sin(pi\*x)/(5+(x^2));

return (result) ;

}

integrate(mydiss, 0, 1) **\*0.5p**

1. Compute the same integral by using the function *integrate* in R and comment on the result.

0.1203855 with absolute error < 1.3e-15

Solution is different because of limitation of epsilon machin , As solution we can improve the accuracy of result by first sorting the numbers and summing them in order of increasing magnitude. If the numbers are all of the same sign and have roughly the same magnitude, a pair wise “fan-in” method may yield good accuracy (which is mention in detail in course literature )first numbers to be summed partially and then this partial sum are added until all sums are completed