Christopher Vajdic

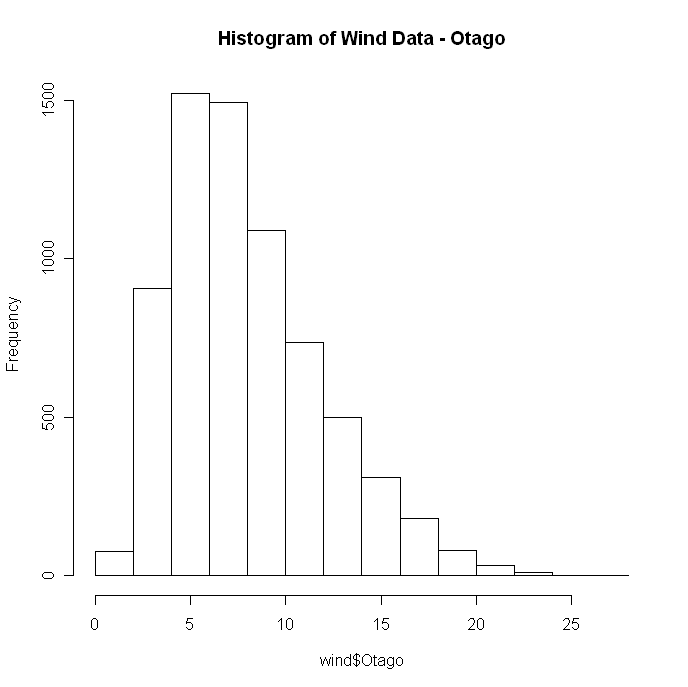
770527-8799

732A38 – 100524\_Exam

**Assignment 1**

A sample of wind speed measurements was gathered from the location of Otago, New Zealand. The following histogram, generated from the data, can support the assumption that the data originates from a Weibull distribution**\*0.5p**

(a common choice for wind data).



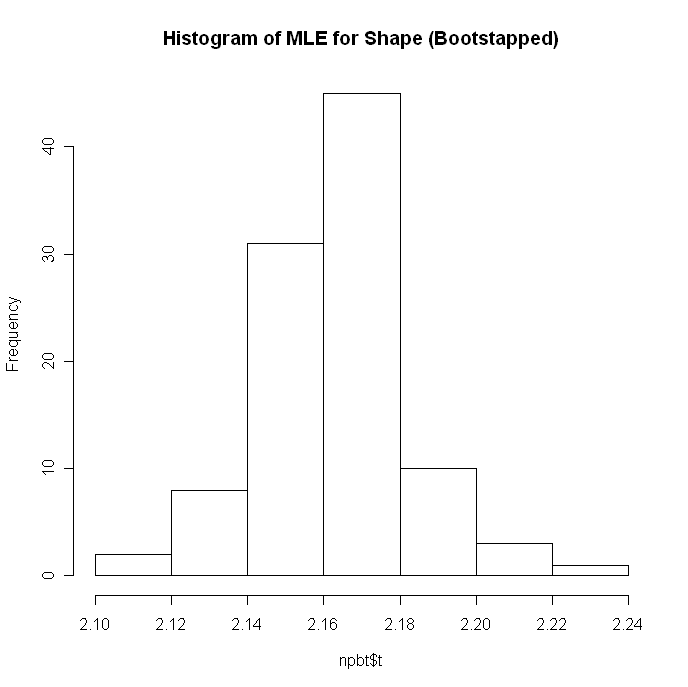
**\*0.5p**

The BFGS method for optimization was used to find the maximum likelihood estimators for the distribution parameters shape and scale. The objective function, *LLweibull*, was the log-likehood function given the sample data. The function was maximized with parameters Θ = (2.163648, 9.063065) for a given value of LL(x=X) = -18899.50. **\*2p**

The optimization strategy for finding distribution parameters is only useful when either an analytic form of the log-likelihood function is available or if a summation based on individual observation density values ( our strategy) can be done. Furthermore, difficulties can be encountered in optimization if the initial starting point is not close by to the global maximizer. **\*1p**

An alternative strategy to finding the distribution parameters involves using the genetic algorithm to maximize the objective function as opposed to a traditional optimization method like BFGS or CG. **\*0.5p**

A function *shape\_func* was implemented which was used to generate 100 estimates of the shape parameter for the Weibull distribution given the sample data. The estimates were generated using the optimization on a resampling (with replacement) of the original wind data. The following histogram reveals the distribution of the bootstrapped data. **\*1.5p**



**\*0.5p** The 95% confidence interval for the generated shape parameter was ( 2.121, 2.209). **\*0.5p** Therefore, with given confidence, one can reject a NULL hypothesis that the data was generated from a Rayleigh distribution (shape k = 2).

**\*1p**

**R Code**

> ## Exam Assignment I

> ## file import

> setwd("//neptunus/chrva439/pc/My Desktop/732A38");

> wind <- read.csv(file="wind.csv", sep=";", header=TRUE);

> win.graph()

> hist(wind$Otago,main="Histogram of Wind Data - Otago");

>

> ## log-likelihood func. is sum of Weibull pdf

> LLweibull <- function(theta){

+ sum(dweibull(wind$Otago, shape = theta[1], scale = theta[2], log=TRUE));

+ }

> control <- list();

> control <- c(-1);

> names(control) <- c("fnscale");

> init <- c(2,1); ## initial value of theta

> opt <- optim(init, LLweibull, method = "BFGS", control=control);

There were 26 warnings (use warnings() to see them)

> opt

$par

[1] 2.163648 9.063065

$value

[1] -18899.50

$counts

function gradient

59 16

$convergence

[1] 0

$message

NULL

>

>

> ## Non-Parametric Bootstrap

> ## shape func. on bootstrapped index

> shape\_func <- function(x,i){

+ LLweib <- function(theta){

+ sum(dweibull(wind$Otago[i], shape = theta[1], scale = theta[2], log=TRUE));

+ }

+ control <- list();

+ control <- c(-1);

+ names(control) <- c("fnscale");

+ init <- c(2,1); ## initial value of theta

+ opt <- optim(init, LLweib, method = "BFGS", control=control);

+ opt$par[1]; ## RETURNS SHAPE

+ }

>

> library(boot);

> ##Create type BOOT (Non-Parametric)

> npbt <- boot(wind$Otago,shape\_func,R=100,sim="ordinary",stype="i");

There were 50 or more warnings (use warnings() to see the first 50)

> print(npbt);

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = wind$Otago, statistic = shape\_func, R = 100, sim = "ordinary",

stype = "i")

Bootstrap Statistics :

original bias std. error

t1\* 2.163648 -0.0009266582 0.01944083

> win.graph()

> hist(npbt$t,main="Histogram of MLE for Shape (Bootstapped)");

> ## Create type ENVELOPE

> shape.ci <- boot.ci(npbt,conf=0.95,type="perc");

> shape.ci;

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

Based on 100 bootstrap replicates

CALL :

boot.ci(boot.out = npbt, conf = 0.95, type = "perc")

Intervals :

Level Percentile

95% ( 2.121, 2.209 )

Calculations and Intervals on Original Scale

Some percentile intervals may be unstable

>

**Assignment 2**

The function *ARgen2* was implemented in the R-language to generate random variables distributed according to the density function,

fx(x) = π/2 sin(πx), for 0 ≤ x ≤ 1.



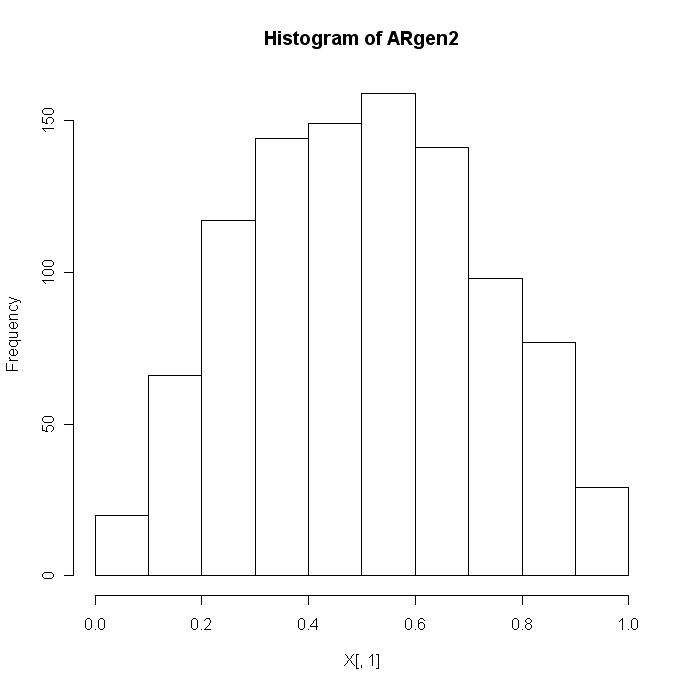
The implementation used the Accept/Reject Method with the uniform distribution as the majorizing function, fy(y). According to this method a constant c is chosen such that,

c fy(y) ≥ fx(x) for all x.

This constant, which boosts the majorizer to “cover” the target distribution should be as small as possible. In this case, solving for δfx/δx = 0, reveals the maximum value over the interval occurs at x = ½ , which when plugged back into the original equation fx(½) = π/2 = c, **\*1p**

the desired value of the scaling constant. Therefore, the ratio fx(y)/cfy(y) = sin(πy).

The function was used to generate 1000 randon variables. The rejection rate was 61.74%. The following histogram reveals the distribution of the results. The generated data appears to accurately reflect the density curve for the function provided from MATLAB above. **\*0.5p**

**\*0.5p**

An estimate of the integral was made using the sample generated previously from the given density function, now relabeled p(x). The integrand can be estimated by noting the decomposition of f(x) = g(x) p(x). This well-known formula provides the basis for the fact that a function of a random variable from a given distribution has an expected value over a domain equal to the integral over the domain of that function times the density function.

Taking the earlier density function as our p(x), the function g(x) = (2/π)/(5 + x^2). Therefore, the integral is equal to E[g(x)] = Σ g(xi)/m with resulting calculation of 0.1203352. **\*1p** The R-language has a built-in function *integrate*() which returned a value of 0.1203855 with absolute error < 1.3e-15 over the given intregrand domain revealing that the estimate generated by the bootstrapped parameter was surprisingly accurate. **\*1p**

Attempting to use the uniform distribution as the density function in the above estimation would not of eliminated the trigonometric function in the integrand, a source of consider complication in the resulting calculations. **\*0p WRONG**

**R Code**

> ## Exam Assignment II

>

> ## ARgen2 returns X ~ f(x) = pi/2\*sin(pi\*x)

>

> ARgen2 <- function (n) { ## Input n is number of iterations

+ z <- vector("numeric",0);

+ count <- vector("numeric",0);

+ for (i in 1:n){

+ u <- 1;

+ x <- 0;

+ counter <- 0;

+ while (u > sin(pi\*x)) { ## 1st iteration always TRUE

+ x <- runif(1); ## majorizer

+ u <- runif(1);

+ counter <- counter + 1;

+ }

+ z[i] <- x;

+ count[i]<- counter;

+ }

+ return(cbind(z,count));

+ }**\*1.5p**

>

> X <- ARgen2(1000);

> R <- sum(X[,2])/(sum(X[,2])+ 1000); ## Percentage Rejection

**\*0.5p since first iteration is always TRUE, counter gives you more than it should, points for this part are not reduced**

> win.graph()

> hist(X[,1],main="Histogram of ARgen2");

> R

[1] 0.6174445

> g <- function (x) { ## E[g(x)] = theta\_hat

+ (2/pi)/(5 + x^2);

+ }

> sum(g(X[,1]))/1000;

[1] 0.1203352

> f <- function (x) {

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

+ }

> integrate(f,0,1);

0.1203855 with absolute error < 1.3e-15

>