R Code

Christopher Partlett and Prakash Patil

Calculating the test statistics T_1, T_2, and T_3

The following simple R function takes two arguments: `x`- a set of data, and `xcuts`- the number of points to use in the numerical integration (used to estimate the variance). The function returns the three standardised test statistics, T_1, T_2,and T_3.

```
uTnm_T <- function(x , xcuts = 512){

############
# Preamble functions
############


fisherz<-function(r)
{
Z = (1/2)*log( (1+r)/(1-r) )

return(Z)
}

AUC <- function(X,Y){

id <- order(X)

int = 0

if( length(Y) > 1){ int = sum(diff(X[id])*rollmean(Y[id],2)) }

return(int)
}

# # calculate integrals based on X at the points x

If2<-function(X,f2,x){

n = length(x)

if2 = rep(0,n)

for( i in 1:n){

# pick out all the Xs > x
```

```
Xx  =  X[ X > x[i] ]
f2x = f2[ X > x[i] ]

if2[i] = AUC(Xx,f2x)

}


return(if2)
}

# #

IfF<-function(X,f,F,x){

n = length(x)

ifF = rep(0,n)

fF = (F - 0.5)*f

for( i in 1:n){

Xx =   X[X > x[i] ]
fFx = fF[X > x[i] ]

ifF[i] = AUC(Xx,fFx)

}

return(ifF)
}

# #

###


remNaN<-function(x){

y <- x[!is.na(x)]

return(y)
}

#############
# Calls a C function to calculate f^(Xi)
#############

fden<-function(x){
```

```
    x = sort(x)
    nx = length(x)
    v = rep(0, nx)
    h = bw.nrd0(x)
    crun = .C("fden", as.double(x), out = as.double(v), as.integer(nx),
        as.double(h))
    fnh = (1/(nx - 1)) * (1/h) * (crun$out)
    return(fnh)
}

############
# Main function: Calculating eta-hat
############

x=remNaN(x)

x = sort(x)

n = length(x)

f = fden(x)
F = (ecdf(x))(x)

#
denf = density( x , kernel = c("epanechnikov") , n = xcuts )

xs = denf$x
fs = denf$y
Fs = (ecdf(x))(xs)
#

Tcov = -cov( f , F )

etah = -cor( f , F )

zetah = fisherz(etah)

############
# Estimating the variance
############

ix  = If2(xs,fs^2,x)
ix2 = IfF(xs,fs,Fs,x)

A = var(f)
B = var(F)

mu = mean(f)

Y0 =   ( 2 )*( f*F - 0.5*f ) + ix
```

```
Y = (   (2/sqrt(A*B))*( f*F - 0.5*f ) + (1/sqrt(A*B))*ix +
etah*( (F - 0.5)^2/(2*B) + (f-mu)^2/(2*A)
           + (1/B)*ix2 + (1/A)*(f - mu)*f )  )

sig20 = var(Y0)
sig2 = var(Y)
zsig2 = var(Y)*(1/( 1 - etah^2 )^2)

# #

T1 = sqrt(n)*etah/sqrt(sig2)
T2 = sqrt(n)*zetah/sqrt(zsig2)
T3 = sqrt(n)*Tcov/sqrt(sig20)

return( list( eta = T1 , zeta = T2 , cov = T3 ) )
}


##
```

C function for computing $\hat{f}(X_i)$

The following subroutine (called by `uTnm_T`) computes the kernel density estimate at the data point X_i.

```
#include "fden.h"
#include <math.h>

EXPORT void fden(double *x, double *v, int *nnx, double *hh)
{
//preamble

int nx = nnx[0];
int i;
int j;

double h = hh[0];

//function

float depan(float x)
{
float y=0.0;

if( (x>-1)&&(x<1) ){y= 0.75*(1-x*x) ;}

return(y);
}

//code


for(j=0 ; j<nx ;j++ ){

v[j] = 0 ;

for( i=0 ; i<nx ; i++ ){

if(i != j){ v[j] = v[j] + depan( (x[i] - x[j] )/h); }

}
}

                              // fnh = (1/(n-1))*(1/h)*(fnh)
}
```