Journal of Biogeography

SUPPORTING INFORMATION

Assessing Relative Variable Importance across Different Spatial Scales: A Two-Dimensional Wavelet Analysis

Gudrun Carl, Daniel Doktor, Oliver Schweiger, Ingolf Kühn

Appendix S2 R code for calculating scale-specific regressions.

For the purpose to assess the relationship between a spatial process and environmental variables as a function of spatial scale, i.e. to measure scale-dependent interactions, the method includes the following three quality criteria:
(1) 2-D wavelet variance,
(2) 2-D wavelet covariance, and
(3) scale-specific regression combined with an estimation of the relative variable importance.
This scale-specific regression allows a regression at each scale separately, because it is based on data decomposed into scale-specific subcomponents by means of wavelets (Carl & Kühn, 2010). Subsequently, the models are ranked using multi-model inference, and the importance of a variable in the regressions is indicated by the sum of Akaike weights over the models that include the variable (Burnham & Anderson, 2002).

Description: package WMRR
The package consists of functions to perform a spatial multilevel regression analysis based on a scale-specific regression and an automatically generated model selection. It allows assessing the impact of scale, i.e. spatial resolution, on the results of multiple regression and model selection. Therefore, it reaches a decision which predictors are relevant at which scale.

Details
The collection of functions includes:

Function name	Description
acfft	computes the autocorrelation.
aic.culc	computes Akaike information criterion.
gennormal	creates data containing a normally distributed response vector.
genbinary	creates data containing a binary distributed response vector.
genpoisson	creates data containing a poisson distributed response vector.
mmi.scale.wrm	performs automatically generated model selection and creates a model selection table according to the approach of multi-model inference (Burnham & Anderson, 2002). The analysis is carried out for scale-specific regression, i.e. where ScaleWRM and ScaleWRM.fit as modelling functions are supported. AICc is used to obtain model selection weights and to rank the models.
plot.covar	plots a figure for wavelet variance or covariance versus scale.
plot.rvi	plots a figure for relative variable importance
ScaleWRM	performs a scale-specific regression based on a wavelet-revised model using package waveslim (Whitcher, 2005). It can be used to fit generalized linear models taking the two-dimensional grid structure of

	datasets into account. The following error distributions (in conjunction with appropriate link functions) are allowed: "gaussian", "binomial"(binary) or "poisson". The model provides scale-specific results for data sampled on a contiguous geographical area. The dataset is assumed to be regular gridded and the grid cells are assumed to be square. Moreover, the model is a residual autocorrelation correcting method.
ScaleWRM.fit	does the same as ScaleWRM, but allows modifying iteration settings.
upscale	plots a figure for spatial upscaling.
wavecovar	performs a two-dimensional wavelet covariance analysis.
wavevar	performs a two-dimensional wavelet variance analysis.

References

Burnham, K.P. & Anderson, D.R. (2002) *Model selection and multimodel inference*. Springer, New York.

Carl, G. & Kühn, I. (2010) A wavelet-based extension of generalized linear models to remove the effect of spatial autocorrelation. *Geographical Analysis*, **42**, 323-337.

Whitcher, B. (2005) Waveslim: basic wavelet routines for one-, two- and three-dimensional signal processing. *R package version 1.5*.

See Also AIC, package MuMIn.

```
#############################################################################

# Packages

#############################################################################

library(waveslim)
library(rje)

#############################################################################

# Example

#############################################################################
# The generated data are just for checking the code.
# f       response variable (Poisson distribution)
# t1, t2  predictors
# t1      high autocorrelation
# t2      without autocorrelation
# (function: b*X = 2 +1*t1 -2.8*t2)

data<-genpoisson(32)
family<-"poisson"

attach(data)
formula<-formula(f~t1+t2)
coord<-cbind(x,y)

# t1 - high autocorrelation
ac<-acfft(x,y,t1)
ac
```

```
# GLM for comparison
m0<-glm(formula,family,data)
summary(m0)
# equivalent to ScaleWRM at scale=0
# ms0<-ScaleWRM(formula,family,data,coord,scale=0,plot=TRUE)

# wavelet variance
pv<-plot.covar(formula,data,coord,
                              wavelet="d4",wtrafo="modwt",plot="var")

# wavelet covariance
pc<-plot.covar(formula,data,coord,
                             wavelet="d4",wtrafo="modwt",plot="covar")

# scale-specific regressions for detail components at various levels
# ranked by multimodel inference
A<-array(NA,c(4,8,4))
level<-rep(NA,4)
for (i in 1:4) {
mmi<-   mmi.scale.wrm(formula,family,data,coord,scale=i,
                                                detail=TRUE,wavelet="d4")
A[,,i]<-mmi$result
level[i]<-mmi$level
}
plot.rvi(A,level)

# Upscaling for smooth components at various levels
upscale(t1,x,y,pad=median(t1))
upscale(t2,x,y,pad=median(t2))
upscale(f,x,y,pad=median(f))

##########################################################################




##########################################################################

# Functions of package ScaleWRM

##########################################################################


##########################################################################
ScaleWRM<-function(formula,family,data,coord,scale=0,detail=TRUE,
wavelet="haar",wtrafo="dwt",plot=FALSE,graph=FALSE){
##########################################################################
#
# scale-specific regression (wavelet-revised model)
#
# 2D analysis taking the grid structure of datasets into account
#
# Gudrun Carl, 2015
# Codes are written for R (www.r-project.org)
# using package waveslim
#
# autocorrelation correcting code
# for responses: "gaussian", "binomial"(binary) or "poisson"
# for spatial (2-dimensional) autocorrelation
# in macroecological data (regular gridded datasets,
```

```
# grid cells are assumed to be square)
#
#########################################################################
# Arguments:
# formula  with specified notation according to names in data frame
# family   "gaussian", "binomial"(binary) or "poisson"
# data     data frame
# coord    corresponding coordinates which have to be integer
# scale    0 or
#          higher integers possible (limit depends on sample size)
# detail   only detail components
# wavelet  type of wavelet: "haar", "d8"
# wtrafo   type of wavelet transform:  "dwt", "modwt"
# value:   ScaleWRM returns a list containing the following elements
#          b          estimate of regression parameters
#          s.e.       standard errors
#          z          z values (or corresponding values for statistics)
#          p          probabilities
#          fitted     fitted values
#          resid      Pearson residuals
#          if plot or graph is true:
#          ac.glm     autocorrelation of glm.residuals
#          ac         autocorrelation of wavelet.residuals
#
#########################################################################
require(waveslim)
n<-dim(data)[1]
l<-dim(data)[2]
x<-coord[,1]
y<-coord[,2]
if(length(x)!=n) stop("error in dimension")
logic1<-identical(as.numeric(x),round(x,0))
logic2<-identical(as.numeric(y),round(y,0))
if(!logic1 | !logic2) stop("coordinates not integer")

X<-model.matrix(formula,data)
nvar<-dim(X)[2]

if(is.vector(model.frame(formula,data)[[1]])){
yold<-model.frame(formula,data)[[1]]
ntr<-1
}
if(family=="binomial" & is.matrix(model.frame(formula,data)[[1]])){
yold<-model.frame(formula,data)[[1]][,1]
ntr<-model.frame(formula,data)[[1]][,1] +
                            model.frame(formula,data)[[1]][,2]
}

n.level<-scale
length.s<-3*n.level+1
s<-rep(0,length.s)
s[(length.s-3):(length.s-1)]<-1
if(!detail) s[length.s]<-1
 if(scale==0) {s<-c(1,1,1,1) ; n.level<-1}
# print(s)
beta<-matrix(NA,4,nvar)
resi<-matrix(NA,4,n)
ac<-matrix(NA,4,10)
se<-matrix(NA,4,nvar)

pdim<- max(max(y)-min(y),max(x)-min(x))*3/2
power<-0
while(2^power<pdim) power<-power+1
```

```
xmargin0<-as.integer((2^power-(max(x)-min(x)))/2)-min(x)+1
ymargin0<-as.integer((2^power-(max(y)-min(y)))/2)-min(y)+1

if(power<n.level) stop("scale is too high")

i4<-1
while(i4<5){     #.....................................................
if(i4==1){xmargin<-xmargin0 ; ymargin<-ymargin0}
if(i4==2){xmargin<-xmargin0+1 ; ymargin<-ymargin0}
if(i4==3){xmargin<-xmargin0+1 ; ymargin<-ymargin0+1}
if(i4==4){xmargin<-xmargin0 ; ymargin<-ymargin0+1}

# GLM for comparison
 m0<-glm(formula,family,data)
 res0<-resid(m0,type="pearson")
 beta0<-m0$coeff

#-------------------------------------------------------------------------

betaw<-rep(0,nvar)
lin<-X%*%betaw
if(family=="gaussian") pi<-lin
if(family=="binomial") pi<-exp(lin)/(1+exp(lin))
if(family=="poisson")  pi<-exp(lin)

#if(family=="gaussian") pi<-rep(0,n)
#if(family=="binomial") pi<-rep(.5,n)
#if(family=="poisson") pi<-rep(1,n)

it<-0

repeat{
it<-it+1
if(family=="gaussian") var<-rep(1,n)
if(family=="binomial") var<-ntr*pi*(1-pi)
if(family=="poisson")  var<-pi
sigma<-as.vector(sqrt(var))
W12<-diag(sigma)
Am12<-diag(1/sigma)
Xnew<-W12%*%X
ynew<-W12%*%X%*%betaw+Am12%*%(yold-ntr*pi)

F<-matrix(0,2^power,2^power)
T<-array(0,c(2^power,2^power,nvar))
for(ii in 1:n){
kx<-x[ii]+xmargin
ky<-y[ii]+ymargin
F[ky,kx]<-ynew[ii]
for (i3 in 1:nvar)
T[ky,kx,i3]<-Xnew[ii,i3]
}  # ii loop

p<-2^power*2^power
tt<-matrix(0,p,nvar)
if(is.na(max(abs(F)))) {mdwt$coeff<-rep(NA,nvar);break}
if(is.infinite(max(abs(F)))) {mdwt$coeff<-rep(NA,nvar);break}
FT<-mra.2d(F,wavelet,n.level,method=wtrafo)
FTS<-rep(0,length(FT[[1]]))
for(is in 1:length(s))
if(s[is]==1) FTS <- FTS + FT[[is]]
ft<-as.vector(FTS)
for (i3 in 1:nvar){
TT<-mra.2d(T[,,i3],wavelet,n.level,method=wtrafo)
```

```
TTS<-rep(0,length(TT[[1]]))
for(is in 1:length(s))
if(s[is]==1) TTS <- TTS + TT[[is]]
tt[,i3]<-as.vector(TTS)
}

xnam<-paste("tt[,",1:nvar,"]",sep="")
formula.dwt<-as.formula(paste("ft~",paste(xnam,collapse="+"),"-1"))
mdwt<-lm(formula.dwt)
if(sum(abs(tt[,1]))==0) mdwt$coeff[1]<-beta0[1]
if(max(abs(mdwt$coeff),na.rm=TRUE)>1e+10 ) {
                                    mdwt$coeff<-rep(NA,nvar);break}

lin<-X%*%mdwt$coeff
if(family=="gaussian") pi<-lin
if(family=="binomial") pi<-exp(lin)/(1+exp(lin))
if(family=="poisson") {pi<-exp(lin)
   if(min(pi)<1e-10 |max(pi)>1e+10) {mdwt$coeff<-rep(NA,nvar);break}
   }

if (identical(round(betaw,5),round(mdwt$coeff,5)) ) {i4<-4 ; break}
if (i4==4 & it > 200)  stop("too many iterations")
if (it > 200) break
betaw<-mdwt$coeff
}  # next step of iteration


if(sum(abs(tt[,1]))!=0 & !is.na(mdwt$coeff[1])){
Resmdwt<-matrix(resid(mdwt),2^power,2^power)
resmdwt<-rep(0,n)
for(i in 1:n) resmdwt[i]<-Resmdwt[y[i]+ymargin,x[i]+xmargin]
if(plot | graph) {
acw<-acfft(x,y,resmdwt)
}
if(!plot & !graph) {acw<-NA; acpw<-NA}
if(family=="binomial" | family=="poisson"){
if(scale==0) var.b<-solve(t(tt)%*%tt)
if(scale!=0) var.b<-solve(t(tt)%*%tt) * (4^(n.level-1))
}
if(family=="gaussian"){
var.b<-solve(t(tt)%*%tt)
if(scale==0) df<-n-nvar
if(scale!=0) df<-round(n/4^(n.level-1)) -nvar
sigma2<-sum(resmdwt^2)/df
var.b<-sigma2*var.b
}
s.e.<-rep(NA,nvar)
for(i in 1:nvar){
s.e.[i]<-sqrt(var.b[i,i])
}
}

if(sum(abs(tt[,1]))==0 | is.na(mdwt$coeff[1])) {
                                    acw<-NA; resmdwt<-NA; s.e.<-NA}
beta[i4,1:nvar]<-mdwt$coeff[1:nvar]
resi[i4,1:n ]<-resmdwt[1:n]
ac[i4,1:10]<-acw[1:10]
se[i4,1:nvar]<-s.e.[1:nvar]

   #-------------------------------------------------------------------
i4<-i4+1
} # i4 loop #.....................................................
```

```
glm.beta<-beta0
wavelet.beta<-apply(beta,2,mean,na.rm=TRUE)
if(plot | graph) ac0<-acfft(x,y,res0)
if(!plot & !graph) ac0<-NA
acw<-apply(ac,2,mean,na.rm=TRUE)
resw<-apply(resi,2,mean,na.rm=TRUE)
s.e.<-apply(se,2,mean,na.rm=TRUE)
lin<-X%*%wavelet.beta
if(family=="gaussian") pi<-lin
if(family=="binomial") pi<-exp(lin)/(1+exp(lin))
if(family=="poisson")  pi<-exp(lin)


# test statistics

z.value<-rep(NA,nvar)
pr<-rep(NA,nvar)
if(sum(abs(tt[,1]))!=0 & !is.na(wavelet.beta[1])) {
z.value<-wavelet.beta/s.e.
for(i in 1:nvar){
if(family=="gaussian"){
if(z.value[i]<=0) pr[i]<-2*pt(z.value[i],df)
if(z.value[i]>0)  pr[i]<-2*(1-pt(z.value[i],df))
}
if(family=="binomial" | family=="poisson"){
if(z.value[i]<=0) pr[i]<-2*pnorm(z.value[i])
if(z.value[i]>0)  pr[i]<-2*(1-pnorm(z.value[i]))
}
}
}

if(plot){
beta<-cbind(glm.beta,wavelet.beta,s.e.,z.value,pr)
beta<-beta[,2:5]
if(family=="gaussian")
colnames(beta) <- c("Estimate", "Std.Err", "t value", "Pr(>|t|)")
if(family=="binomial" | family=="poisson")
colnames(beta) <- c("Estimate", "Std.Err", "z value", "Pr(>|z|)")
cat("---","\n","Coefficients:","\n")
printCoefmat(beta)

cat("---","\n")
cat("Autocorrelation for glm.residuals","\n")
print(ac0)
cat("Autocorrelation for wavelet.residuals","\n")
print(acw)
}

if(graph){
y1<-min(min(ac0),min(acw))-.1
y2<-max(max(ac0),max(acw))+.1
plot(ac0,type="b",ylim=c(y1,y2),
    ylab="Autocorrelation of residuals", xlab="Lag distance",
    main=paste("Autocorrelation for scale = ", scale))
points(acw,pch=2,type="b")
v<-1:2
leg<-c("glm.residuals","wavelet.residuals")
legend(6,y2-.1,leg,pch=v)
}

coef<-as.vector(wavelet.beta)
names(coef)<-dimnames(X)[[2]]
```

```
call<-match.call()
fit<-list(call=call,b=coef,s.e.=s.e.,z=z.value,p=pr,fitted=pi,
          resid=resw,ac.glm=ac0,ac=acw)
class(fit)<-"wrm"
fit
}

#############################################################################


#############################################################################
# Function for autocorrelation of function reslm
acfft<-function(x,y,reslm,lim1=0,lim2=1,dmax=10){
#############################################################################
reslm<-reslm-mean(reslm)
mi<-max(x)-min(x)+1
mk<-max(y)-min(y)+1
n<-max(mi,mk)
n2<-n*n
Ares<-matrix(0,n,n)
mask<-matrix(0,n,n)
for(i in 1:length(x)){
kx<-x[i]-min(x)+1
ky<-y[i]-min(y)+1
Ares[ky,kx]<-reslm[i]
mask[ky,kx]<-1}
 filter1<-matrix(0,n,n)
 filter1[1,1]<-1
leng<-length(reslm)
ne<-convolve(convolve(Ares,filter1),Ares)[1,1]/leng
n3<-3*n
Ares0<-matrix(0,n3,n3)
Ares1<-matrix(0,n3,n3)
n1<-n+1
nn<-n+n
Ares0[n1:nn,n1:nn]<-Ares[1:n,1:n]
Ares1[1:n,1:n]<-Ares[1:n,1:n]
maske0<-matrix(0,n3,n3)
maske1<-matrix(0,n3,n3)
maske0[n1:nn,n1:nn]<-mask[1:n,1:n]
maske1[1:n,1:n]<-mask[1:n,1:n]
nx<-rep(1:n3,n3)
ny<-as.numeric(gl(n3,n3))

gr<-lim1
gr1<-lim2
h<-n*n3+n+1
corr<-rep(0,dmax)
kk<-0
while(kk<dmax){
kk<-kk+1
filter<-matrix(0,n3,n3)
for(i in 1:(n3*n3)) {
d<-sqrt((nx[h]-nx[i])^2+(ny[h]-ny[i])^2)

if(gr<d & d<=gr1) filter[ny[i],nx[i]]<-1}

sum<-convolve(convolve(maske0,filter),maske1)[1,1]
za<-convolve(convolve(Ares0,filter),Ares1)[1,1]/sum
corr[kk]<-za/ne
 gr<-gr+(lim2-lim1)
 gr1<-gr1+(lim2-lim1)
}
```

```
corr<-as.vector(corr)
}



#########################################################################
#########################################################################


#########################################################################
ScaleWRM.fit<-function(formula,family,data,coord,
scale=0,detail=TRUE,wavelet="haar",wtrafo="dwt",padzone=17/16,
b.ini=NULL,eps=0.001,denom.eps=1e-20,itmax=800,
plot=FALSE,graph=FALSE){
#########################################################################
#
# scale-specific regression (wavelet-revised model)
#
# 2D analysis taking the grid structure of datasets into account
#
# Gudrun Carl, 2015
# Codes are written for R (www.r-project.org)
# using package waveslim
#
# autocorrelation correcting code
# for responses: "gaussian", "binomial"(binary) or "poisson"
# for spatial (2-dimensional) autocorrelation
# in macroecological data (regular gridded datasets,
# grid cells are assumed to be square)
#
#########################################################################
# Arguments:
# formula  with specified notation according to names in data frame
# family   "gaussian", "binomial"(binary) or "poisson"
# data     data frame
# coord    corresponding coordinates which have to be integer
# scale    0 or
#          higher integers possible (limit depends on sample size)
# detail   only detail components
# wavelet  type of wavelet: "haar", "d8"
# wtrafo   type of wavelet transform:  "dwt", "modwt"
# value:   ScaleWRM returns a list containing the following elements
#          b          estimate of regression parameters
#          s.e.       standard errors
#          z          z values (or corresponding values for statistics)
#          p          probabilities
#          fitted     fitted values
#          resid      Pearson residuals
#          if plot or graph is true:
#          ac.glm     autocorrelation of glm.residuals
#          ac         autocorrelation of wavelet.residuals
#
#########################################################################
require(waveslim)
n<-dim(data)[1]
l<-dim(data)[2]
x<-coord[,1]
y<-coord[,2]
if(length(x)!=n) stop("error in dimension")
logic1<-identical(as.numeric(x),round(x,0))
logic2<-identical(as.numeric(y),round(y,0))
if(!logic1 | !logic2) stop("coordinates not integer")
converged.logi<-TRUE
```

```
X<-model.matrix(formula,data)
nvar<-dim(X)[2]

if(is.vector(model.frame(formula,data)[[1]])){
yold<-model.frame(formula,data)[[1]]
ntr<-1
}
if(family=="binomial" & is.matrix(model.frame(formula,data)[[1]])){
yold<-model.frame(formula,data)[[1]][,1]
ntr<-model.frame(formula,data)[[1]][,1] +
                            model.frame(formula,data)[[1]][,2]
}

n.level<-scale
length.s<-3*n.level+1
s<-rep(0,length.s)
s[(length.s-3):(length.s-1)]<-1
if(!detail) s[length.s]<-1
 if(scale==0) {s<-c(1,1,1,1) ; n.level<-1}

beta<-matrix(NA,4,nvar)
resi<-matrix(NA,4,n)
ac<-matrix(NA,4,10)
se<-matrix(NA,4,nvar)

pdim<- max(max(y)-min(y)+1,max(x)-min(x)+1)*padzone
power<-0
while(2^power<pdim) power<-power+1
xmargin0<-as.integer((2^power-(max(x)-min(x)))/2)-min(x)+1
ymargin0<-as.integer((2^power-(max(y)-min(y)))/2)-min(y)+1

if(power<n.level) stop("scale is too high")

i4<-1
if(padzone!=1) i4limit<-4
if(padzone==1) i4limit<-1
while(i4<(i4limit+1)) {     #....................................
if(i4==1){xmargin<-xmargin0 ; ymargin<-ymargin0}
if(i4==2){xmargin<-xmargin0+1 ; ymargin<-ymargin0}
if(i4==3){xmargin<-xmargin0+1 ; ymargin<-ymargin0+1}
if(i4==4){xmargin<-xmargin0 ; ymargin<-ymargin0+1}

# GLM for comparison
 m0<-glm(formula,family,data)
 res0<-resid(m0,type="pearson")
 beta0<-m0$coeff

#-------------------------------------------------------------------

if(is.null(b.ini)) { betaw<-rep(0,nvar)}
else {betaw<-b.ini}
lin<-X%*%betaw
if(family=="gaussian") pi<-lin
if(family=="binomial") pi<-exp(lin)/(1+exp(lin))
if(family=="poisson")  pi<-exp(lin)

#if(family=="gaussian") pi<-rep(0,n)
#if(family=="binomial") pi<-rep(.5,n)
#if(family=="poisson") pi<-rep(1,n)

it<-0
```

```
repeat{
it<-it+1
if(family=="gaussian") var<-rep(1,n)
if(family=="binomial") var<-ntr*pi*(1-pi)
if(family=="poisson")  var<-pi
sigma<-as.vector(sqrt(var))
W12<-diag(sigma)
Am12<-diag(1/sigma)
Xnew<-W12%*%X
ynew<-W12%*%X%*%betaw+Am12%*%(yold-ntr*pi)

F<-matrix(0,2^power,2^power)
T<-array(0,c(2^power,2^power,nvar))
for(ii in 1:n){
kx<-x[ii]+xmargin
ky<-y[ii]+ymargin
F[ky,kx]<-ynew[ii]
for (i3 in 1:nvar)
T[ky,kx,i3]<-Xnew[ii,i3]
}  # ii loop

p<-2^power*2^power
tt<-matrix(0,p,nvar)
if(is.na(max(abs(F)))) {mdwt$coeff<-rep(NA,nvar);break}
if(is.infinite(max(abs(F)))) {mdwt$coeff<-rep(NA,nvar);break}
FT<-mra.2d(F,wavelet,n.level,method=wtrafo)
FTS<-rep(0,length(FT[[1]]))
for(is in 1:length(s))
if(s[is]==1) FTS <- FTS + FT[[is]]
ft<-as.vector(FTS)
for (i3 in 1:nvar){
TT<-mra.2d(T[,,i3],wavelet,n.level,method=wtrafo)
TTS<-rep(0,length(TT[[1]]))
for(is in 1:length(s))
if(s[is]==1) TTS <- TTS + TT[[is]]
tt[,i3]<-as.vector(TTS)
}

xnam<-paste("tt[,",1:nvar,"]",sep="")
formula.dwt<-as.formula(paste("ft~",paste(xnam,collapse="+"),"-1"))
mdwt<-lm(formula.dwt)
if(sum(abs(tt[,1]))==0) mdwt$coeff[1]<-beta0[1]
if(max(abs(mdwt$coeff),na.rm=TRUE)>1e+10 ) {
                                  mdwt$coeff<-rep(NA,nvar);break}

lin<-X%*%mdwt$coeff
if(family=="gaussian") pi<-lin
if(family=="binomial") pi<-exp(lin)/(1+exp(lin))
if(family=="poisson") {pi<-exp(lin)
   if(min(pi)<1e-10 |max(pi)>1e+10) {mdwt$coeff<-rep(NA,nvar);break}
   }

if (max(abs(mdwt$coeff-betaw)/(abs(betaw)+denom.eps)) <= eps ) {
                                                 i4<-i4limit ; break}
if (i4==i4limit & it > itmax)  {converged.logi<-FALSE; break}
if (it > itmax) break
betaw<-mdwt$coeff
}  # next step of iteration


if(sum(abs(tt[,1]))!=0 & !is.na(mdwt$coeff[1])){
Resmdwt<-matrix(resid(mdwt),2^power,2^power)
resmdwt<-rep(0,n)
```

```
for(i in 1:n) resmdwt[i]<-Resmdwt[y[i]+ymargin,x[i]+xmargin]
if(plot | graph) {
acw<-acfft(x,y,resmdwt)
}
if(!plot & !graph) {acw<-NA; acpw<-NA}
if(scale==0) df<-n-nvar
if(scale!=0) df<-round(n/4^(n.level-1)) -nvar
if(family=="binomial" | family=="poisson"){
if(scale==0) var.b<-solve(t(tt)%*%tt)
if(scale!=0) var.b<-solve(t(tt)%*%tt) * (4^(n.level-1))
}
if(family=="gaussian"){
var.b<-solve(t(tt)%*%tt)
sigma2<-sum(resmdwt^2)/df
var.b<-sigma2*var.b
}
s.e.<-rep(NA,nvar)
for(i in 1:nvar){
s.e.[i]<-sqrt(var.b[i,i])
}
}

if(sum(abs(tt[,1]))==0 | is.na(mdwt$coeff[1])) {
                                        acw<-NA; resmdwt<-NA; s.e.<-NA}
beta[i4,1:nvar]<-mdwt$coeff[1:nvar]
resi[i4,1:n ]<-resmdwt[1:n]
ac[i4,1:10]<-acw[1:10]
se[i4,1:nvar]<-s.e.[1:nvar]

   #-----------------------------------------------------------------
i4<-i4+1
} # i4 loop #.....................................................

glm.beta<-beta0
wavelet.beta<-apply(beta,2,mean,na.rm=TRUE)
if(plot | graph) ac0<-acfft(x,y,res0)
if(!plot & !graph) ac0<-NA
acw<-apply(ac,2,mean,na.rm=TRUE)
resw<-apply(resi,2,mean,na.rm=TRUE)
s.e.<-apply(se,2,mean,na.rm=TRUE)
lin<-X%*%wavelet.beta
if(family=="gaussian") pi<-lin
if(family=="binomial") pi<-exp(lin)/(1+exp(lin))
if(family=="poisson")  pi<-exp(lin)


# test statistics

z.value<-rep(NA,nvar)
pr<-rep(NA,nvar)
if(sum(abs(tt[,1]))!=0 & !is.na(wavelet.beta[1])) {
z.value<-wavelet.beta/s.e.
for(i in 1:nvar){
if(family=="gaussian"){
if(z.value[i]<=0) pr[i]<-2*pt(z.value[i],df)
if(z.value[i]>0)  pr[i]<-2*(1-pt(z.value[i],df))
}
if(family=="binomial" | family=="poisson"){
if(z.value[i]<=0) pr[i]<-2*pnorm(z.value[i])
if(z.value[i]>0)  pr[i]<-2*(1-pnorm(z.value[i]))
}
}
}
```

```
if(plot){
beta<-cbind(glm.beta,wavelet.beta,s.e.,z.value,pr)
beta<-beta[,2:5]
if(family=="gaussian")
colnames(beta) <- c("Estimate", "Std.Err", "t value", "Pr(>|t|)")
if(family=="binomial" | family=="poisson")
colnames(beta) <- c("Estimate", "Std.Err", "z value", "Pr(>|z|)")
cat("---","\n","Selection of wavelet components: ",s,"\n")
cat("---","\n","Coefficients:","\n")
printCoefmat(beta)

cat("---","\n")
cat("Autocorrelation for glm.residuals","\n")
print(ac0)
cat("Autocorrelation for wavelet.residuals","\n")
print(acw)
}

if(graph){
y1<-min(min(ac0),min(acw))-.1
y2<-max(max(ac0),max(acw))+.1
plot(ac0,type="b",ylim=c(y1,y2),
    ylab="Autocorrelation of residuals", xlab="Lag distance",
    main=paste("Autocorrelation for scale = ", scale))
points(acw,pch=2,type="b")
v<-1:2
leg<-c("glm.residuals","wavelet.residuals")
legend(6,y2-.1,leg,pch=v)
}

coef<-as.vector(wavelet.beta)
names(coef)<-dimnames(X)[[2]]

call<-match.call()
fit<-list(call=call,b=coef,s.e.=s.e.,z=z.value,p=pr,df=df,
     fitted=pi,resid=resw,ac.glm=ac0,ac=acw ,converged=converged.logi)
class(fit)<-"wrm"
fit
}

##########################################################################


##########################################################################
mmi.scale.wrm<-function(formula,family,data,coord,scale,detail=TRUE,
wavelet="haar",wtrafo="dwt",n.eff=NULL){

##########################################################################
#
# Multi-model inference for scale-specific regression
#
# Gudrun Carl, 2015
# Code is written for R (www.r-project.org)
# using package rje
#
# Arguments:
# formula  with specified notation according to names in data frame
# family   "gaussian", "binomial"(binary) or "poisson"
# data     data frame
# coord    corresponding coordinates which have to be integer
# scale    0 or higher integers possible
#          (limit depends on sample size)
```

```
# detail   only detail components
# wavelet  type of wavelet: "haar", "d8"
# wtrafo   type of wavelet transform:  "dwt", "modwt"
# n.eff    effective sample size
#########################################################################
require(rje)

# Parameter: varnames, p
X<-model.matrix(formula,data)
if(dimnames(X)[[2]][1]!="(Intercept)") {
formula<-update(formula,  ~ . + 1)
X<-model.matrix(formula,data)
}
nvar<-dim(X)[2]
varnames<-dimnames(X)[[2]][-1]
p<-dim(X)[2]-1   # nvar-1 (without intercept)
rm(X)

# MultiModel Inference
# from formula,family,data,p,varnames
# Powerset
pset<-powerSetMat(p)
ip<-dim(pset)[1]
t<-terms(formula)

# Run every model and calculate AIC (multimodel inference)
coef.vec<-matrix(NA,ip,nvar)
df<-rep(NA,ip)
loglik<-rep(NA,ip)
AIC<-rep(NA,ip)

for (i in 1:ip) {

if(sum(pset[i,])!=0 & sum(pset[i,])!=p){
t1<-drop.terms(t,which(pset[i,]==0), keep.response = TRUE)
formula1<-reformulate(attr(t1, "term.labels"), formula[[2]])
formulae<-formula1
}

if(sum(pset[i,])==p) formulae<-formula
if(sum(pset[i,])==0) formulae<-as.formula(paste(formula[[2]],"~1"))

m0<-ScaleWRM.fit(formulae,family,data,coord,scale=scale,
detail=detail,wavelet=wavelet,wtrafo=wtrafo)

if(!m0$converged)
   m0<-ScaleWRM.fit(formulae,family,data,coord,scale=scale,
   detail=detail,wavelet=wavelet,wtrafo=wtrafo,
   b.ini=glm(formulae,family,data)$coef)

kv<-c(1,which(pset[i,]==1)+1)
coef.vec[i,kv]<-m0$b

mu<-m0$fitted
if(is.null(n.eff)) aic<-aic.culc(formulae,family,data,mu)
if(!is.null(n.eff)) aic<-aic.culc(formulae,family,data,mu,n.eff)
df[i]<-aic$df
loglik[i]<-aic$loglik
AIC[i]<-aic$AIC

}

result<-cbind(round(coef.vec,5),df,round(loglik,3),round(AIC,1))
```

```
# Calculate delta and weight (multimodel inference)
delta<-AIC-min(AIC)                          # = delta
weight<-exp(-delta/2)/sum(exp(-delta/2))     # = weight

# Print results
result<-cbind(result,round(delta,2),round(weight,3))
ord<-order(delta)
res<-result[ord,]
dimnames(res)[[1]]<-ord
dimnames(res)[[2]]<-c("(Int)",varnames,
                              "df","logLik","AIC","delta","weight")
if(!detail & scale>=1) scale<-scale-1
cat("---","\n","Level = ",scale,"\n")
 print(res,na.print = "")
fit<-list(result=res,level=scale)
fit
}
#############################################################################

#############################################################################
#                                    generate.txt                           #
#############################################################################
# Functions to generate spatial (2-dimensional) datasets
#
# sampled on square grid cells on a square map
# parameter n provides an n*n map
#
# Gudrun Carl, 2013
# Code is written for R (www.r-project.org)
#############################################################################


#############################################################################
gennormal<-function(n,seed=100) {
#############################################################################
# A function to generate a spatial (2-dimensional) dataset
# sampled on square grid cells on a square map
# where f is "gaussian", t1, t2 are predictors,
# x,y are cartesian coordinates,
# parameter n provides an n*n map
#############################################################################
if (!is.null(seed)) set.seed(seed)
n1<-n
n2<-n1*n1
x<-rep(1:n1,n1)
y<-as.numeric(gl(n1,n1))
coord<-cbind(x,y)
D<-as.matrix(dist(coord))
V<-0.8^D
W1<-t(chol(V))
var<-rep(0,1000)
for(i in 1:1000) {
e<-W1%*%rnorm(n2)
var[i]<-var(e)
}
phi2<-mean(var)

t1<-(1/sqrt(phi2))*W1%*%rnorm(n2,1,1)
t2<-rnorm(n2,1,1)
e<-rnorm(n2)
lin<- -8 +1*t1-2.8*t2
```

```
f<-lin + e

data<-data.frame(f,t1,t2,x,y)

}


#############################################################################
genbinary<-function(n,seed=100) {
#############################################################################
# A function to generate a spatial (2-dimensional) dataset
# sampled on square grid cells on a square map
# where f is "binary", t1, t2 are predictors,
# x,y are cartesian coordinates,
# parameter n provides an n*n map
#############################################################################
if (!is.null(seed)) set.seed(seed)
n1<-n
n2<-n1*n1
x<-rep(1:n1,n1)
y<-as.numeric(gl(n1,n1))
coord<-cbind(x,y)
D<-as.matrix(dist(coord))
V<-0.8^D
W1<-t(chol(V))
var<-rep(0,1000)
for(i in 1:1000) {
e<-W1%*%rnorm(n2)
var[i]<-var(e)
}
phi2<-mean(var)

t1<-(1/sqrt(phi2))*W1%*%rnorm(n2,1,1)
t2<-rnorm(n2,1,1)
e<-rnorm(n2)
lin<- -8 +1*t1-2.8*t2
pi<-exp(lin)/(1+exp(lin))

pi1<-lin+e/sqrt(0.3645)
pi1<-(pi1-mean(pi1))/sqrt(as.numeric(var(pi1)))
pi2<-pnorm(pi1)
f<-qbinom(pi2,1,mean(pi))

data<-data.frame(f,t1,t2,x,y)

}


#############################################################################
genpoisson<-function(n,seed=100) {
#############################################################################
# A function to generate a spatial (2-dimensional) dataset
# sampled on square grid cells on a square map
# where f is "poisson", t1, t2 are predictors,
# x,y are cartesian coordinates,
# parameter n provides an n*n map
#############################################################################
if (!is.null(seed)) set.seed(seed)
n1<-n
n2<-n1*n1
x<-rep(1:n1,n1)
y<-as.numeric(gl(n1,n1))
coord<-cbind(x,y)
```

```
D<-as.matrix(dist(coord))
V<-0.8^D
W1<-t(chol(V))
var<-rep(0,1000)
for(i in 1:1000) {
e<-W1%*%rnorm(n2)
var[i]<-var(e)
}
phi2<-mean(var)

t1<-(0.0777/sqrt(phi2))*W1%*%rnorm(n2,1,0.0777/sqrt(phi2))
t2<-rnorm(n2,1,0.0777)
e<-rnorm(n2)
lin<- 2 +1*t1-2.8*t2
pi<-exp(lin)

pi1<-lin+e/sqrt(1.38)
pi1<-(pi1-mean(pi1))/sqrt(as.numeric(var(pi1)))
pi2<-pnorm(pi1)
f<-qpois(pi2,mean(pi))

data<-data.frame(f,t1,t2,x,y)

}


#############################################################################
#############################################################################



#############################################################################
upscale<-function(f,x,y,wavelet="haar",wtrafo="dwt",pad=0){
#############################################################################
# Arguments:
# f        vector
# x        corresponding x-coordinates which have to be integer
# y        corresponding y-coordinates which have to be integer
# wavelet  type of wavelet
# wtrafo   type of wavelet transform
# pad      parameter for padding
#############################################################################

require(waveslim)

if(length(f)!=length(x) | length(f)!=length(y)) stop("error in dim")
logic1<-identical(as.numeric(x),round(x,0))
logic2<-identical(as.numeric(y),round(y,0))
if(!logic1 | !logic2) stop("coordinates are not integer")

n<-length(f)
pdim<- max(max(y)-min(y),max(x)-min(x))*5/4
power<-0
while(2^power<pdim) power<-power+1
xmargin<-as.integer((2^power-(max(x)-min(x)))/2)-min(x)+1
ymargin<-as.integer((2^power-(max(y)-min(y)))/2)-min(y)+1

F<-matrix(pad,2^power,2^power)
for(ii in 1:n){
kx<-x[ii]+xmargin
ky<-y[ii]+ymargin
F[kx,ky]<-f[ii]
}  # ii loop
```

```
## Plot
windows(8,8)
par(mfrow=c(2,2),
 mai=c(0,0,0.5,0),
omi=c(0,0,0,0),
 pty="s",cex.main=1)
colors<-gray((0:50)/50)

minvec<-rep(NA,4)
maxvec<-rep(NA,4)
for (i in 1:4){
if(i==1){FTS<-F
minvec[i]<-min(FTS)
maxvec[i]<-max(FTS)
}
if(i!=1){
level<-i-1
FT<-mra.2d(F,wavelet,level,method=wtrafo)
FTS<-FT[[3*level+1]]
minvec[i]<-min(FTS)
maxvec[i]<-max(FTS)
}
}
minFTS<-min(minvec,na.rm = TRUE)
maxFTS<-max(maxvec,na.rm = TRUE)

for (i in 1:4){
if(i==1){FTS<-F
}
if(i!=1){
level<-i-1
FT<-mra.2d(F,wavelet,level,method=wtrafo)
FTS<-FT[[3*level+1]]
}

FTS<-FTS-minFTS
FTS<-FTS/(maxFTS-minFTS)
image(FTS,zlim=c(0,1),axes=FALSE,col=colors,
                                          main=paste("level = ", i-1))
} # i-loop
}
#############################################################################



#############################################################################
plot.covar<-function(formula,data,coord,wavelet="haar",wtrafo="dwt",
plot="covar"){
#############################################################################
#
# Plot: Wavelet variance/covariance
#
# Gudrun Carl, 2015
# Code is written for R (www.r-project.org)
#
#############################################################################
# Arguments:
# formula  with specified notation according to names in data frame
# data     data frame
# coord    corresponding coordinates which have to be integer
# wavelet  type of wavelet: "haar", "d8"
# wtrafo   type of wavelet transform:  "dwt", "modwt"
```

```
#########################################################################

x<-coord[,1]
y<-coord[,2]
X<-model.matrix(formula,data)
namvar<-dimnames(X)[[2]]
if(namvar[1]!="(Intercept)") nvar1<-1
if(namvar[1]=="(Intercept)") nvar1<-2
nvar2<-dim(X)[2]
namresp<-as.character(formula[[2]])
resp<-model.frame(formula,data)[[1]]

wvar0<-wavevar(resp,x,y,wavelet=wavelet,wtrafo=wtrafo)
nscale<-length(wvar0)
if(plot=="var"){
# Variance
wvar<-matrix(NA,nvar2,nscale)
for (kk in nvar1:nvar2){
wvar[kk,]<-wavevar(X[,kk],x,y,wavelet=wavelet,wtrafo=wtrafo)
}
plot(wvar0[1:nscale],type="b",ylim=c(-.1,.9),pch=16,
    ylab="Variance", xlab="Level",
    main=paste(paste(wavelet,wtrafo)," - wavelet variance") )
for(kk in nvar1:nvar2){
points(wvar[kk,1:nscale],pch=kk,type="b")
}
leg<-c(namvar[nvar1:nvar2],namresp)
v<-nvar1:nvar2
v<-c(v,16)
legend(4.5,0.87,leg,pch=v)
}

if(plot=="covar"){
# Covariance
wcvar<-matrix(NA,nvar2,nscale)
for (kk in nvar1:nvar2){
wcvar[kk,]<-wavecovar(resp,X[,kk],x,y,wavelet=wavelet,wtrafo=wtrafo)
}
plot(wcvar[nvar1,1:nscale],type="b",ylim=c(-.1,.6),pch=nvar1,
    ylab="Covariance", xlab="Level",
    main=paste(paste(wavelet,wtrafo)," - wavelet covariance") )
for(kk in (nvar1+1):nvar2){
points(wcvar[kk,1:nscale],pch=kk,type="b")
}
leg<-rep(NA,nvar2-nvar1+1)
for (kk in nvar1:nvar2){
leg[kk]<-paste(namresp,namvar[kk],sep="---")
}
if(nvar1==2) leg<-leg[-1]

v<-nvar1:nvar2
legend(3.5,0.4,leg,pch=v)
}

if(plot=="var") res<-rbind(wvar0,wvar)
if(plot=="covar") res<-wcvar
fit<-list(result=res)
fit
}
#########################################################################
```

```
#########################################################################
aic.culc<-function(formula,family,data,mu,n.eff=NULL){
#########################################################################
# Gudrun Carl, 2015
# Code is written for R (www.r-project.org)
#
# Arguments:
# formula  with specified notation according to names in data frame
# family   "gaussian", "binomial"(binary) or "poisson"
# data     data frame
# mu       fitted values
# value:   loglik     log likelihood
#          df         degrees of freedom
#          AIC
#          AICc
#########################################################################

X<-model.matrix(formula,data)
if(is.vector(model.frame(formula,data)[[1]])){
y<-model.frame(formula,data)[[1]]
ntr<-1
}
if(family=="binomial" & is.matrix(model.frame(formula,data)[[1]])){
y<-model.frame(formula,data)[[1]][,1]
ntr<-model.frame(formula,data)[[1]][,1] +
                            model.frame(formula,data)[[1]][,2]
}

n <- dim(X)[1]
nvar<-dim(X)[2]
if(family=="gaussian"){
sos<-sum((y-mu)^2)                      # sum of squares
sigma2<- sos/n                          # variance
#loglik<- -n/2*log(2*pi) - n*log(sigma2^(1/2)) - 1/(2*sigma2)*sos
#loglik<- -n/2*log(2*pi) - n*log(sigma2^(1/2)) - n/2
#loglik<- -n/2*log(2*pi) - n/2*log(sigma2) - n/2
#loglik<- -n/2*(log(2*pi) + log(sigma2) +1)
loglik<- -n/2*(log(2*pi*sigma2) +1)    # log likelihood
if(!is.null(n.eff)) loglik<- -n.eff/2*(log(2*pi*sigma2) +1)
K<-nvar+1        # nvar (number of pred.+interc.) +1 (for variance)
}
if(family=="binomial"){                   # choose= binomial coeff.
loglik<- sum(y*log(mu/(1-mu)) + ntr*log(1-mu) + log(choose(ntr,y)) )
K<-nvar          # without variance (since var. is function of mean)
}
if(family=="poisson"){
# loglik<- sum(y*log(mu)-mu)      # useful for delta in mmi
loglik<- sum(y*log(mu)-mu) - sum(log(factorial(y))) # factorial
K<-nvar          # without variance (since var. is function of mean)

}

AIC  <- -2* loglik + 2*K              # AIC
AICc <- -2* loglik + 2*K + 2*K*(K+1)/(n-K-1)    # AICc

list(loglik=loglik,df=K,AIC=AIC,AICc=AICc)
}
#########################################################################



#########################################################################
wavevar<-function(f,x,y,wavelet="haar",wtrafo="dwt"){
```

```
#########################################################################
#
# Wavelet variance analysis
#
# 2D analysis taking the grid structure of datasets into account
#
# Gudrun Carl, 2015
# Code is written for R (www.r-project.org)
#
# using package waveslim
#########################################################################
# Arguments:
# f         function as vectors
# x,y       corresponding coordinates which have to be integer
# wavelet   type of wavelet
# wtrafo    type of wavelet transform
#########################################################################
library(waveslim)

n<-length(f)
pdim<- max(max(y)-min(y)+1,max(x)-min(x)+1)
power<-0
while(2^power<pdim) power<-power+1
xmargin<-as.integer((2^power-(max(x)-min(x)))/2)-min(x)+1
ymargin<-as.integer((2^power-(max(y)-min(y)))/2)-min(y)+1

 f<-scale(f)   # for scaling and centering

F<-matrix(0,2^power,2^power)
for(ii in 1:n){
kx<-x[ii]+xmargin
ky<-y[ii]+ymargin
F[kx,ky]<-f[ii]
}  # ii loop

level<-power

p<-2^power*2^power
if(wtrafo=="dwt") F.dwt<-dwt.2d(F,wavelet,level)
if(wtrafo=="modwt") F.dwt<-modwt.2d(F,wavelet,level)

# wavelet variance (1/n) * sum[abs(f.dwt)^2 ]
Var<-rep(NA,level)
for(ik in 1:level){
ii<-3*(ik-1)+1
FS1 <- (1/n) * sum(abs(F.dwt[[ii]])^2)
FS2 <- (1/n) * sum(abs(F.dwt[[ii+1]])^2)
FS3 <- (1/n) * sum(abs(F.dwt[[ii+2]])^2)
Var[ik]<-FS1+FS2+FS3    # all 3 components
}
# windows()
# plot(Var)
Var<-round(Var,4)
Var
}
#########################################################################


#########################################################################
wavecovar<-function(f1,f2,x,y,wavelet="haar",wtrafo="dwt"){
#########################################################################
#
# Wavelet covariance analysis
```

```
#
# 2D analysis taking the grid structure of datasets into account
#
# Gudrun Carl, 2015
# Code is written for R (www.r-project.org)
#
# using package waveslim
#########################################################################
# Arguments:
# f1,f2    functions as vectors
# x,y      corresponding coordinates which have to be integer
# wavelet  type of wavelet
# wtrafo   type of wavelet transform
#########################################################################

library(waveslim)

n<-length(f1)
pdim<- max(max(y)-min(y)+1,max(x)-min(x)+1)
power<-0
while(2^power<pdim) power<-power+1
xmargin<-as.integer((2^power-(max(x)-min(x)))/2)-min(x)+1
ymargin<-as.integer((2^power-(max(y)-min(y)))/2)-min(y)+1

 f1<-scale(f1)   # for scaling and centering
 f2<-scale(f2)   # for scaling and centering

F1<-matrix(0,2^power,2^power)
F2<-matrix(0,2^power,2^power)
for(ii in 1:n){
kx<-x[ii]+xmargin
ky<-y[ii]+ymargin
F1[kx,ky]<-f1[ii]
F2[kx,ky]<-f2[ii]
}  # ii loop

level<-power

p<-2^power*2^power
if(wtrafo=="dwt"){
F1.dwt<-dwt.2d(F1,wavelet,level)
F2.dwt<-dwt.2d(F2,wavelet,level)
}
if(wtrafo=="modwt"){
F1.dwt<-modwt.2d(F1,wavelet,level)
F2.dwt<-modwt.2d(F2,wavelet,level)
}

# wavelet covariance (1/n) * sum[abs(f1.dwt*f2.dwt) ]
CVar<-rep(NA,level)
for(ik in 1:level){
ii<-3*(ik-1)+1
FS1 <- (1/n) * sum(abs(F1.dwt[[ii]]*F2.dwt[[ii]]))
FS2 <- (1/n) * sum(abs(F1.dwt[[ii+1]]*F2.dwt[[ii+1]]))
FS3 <- (1/n) * sum(abs(F1.dwt[[ii+2]]*F2.dwt[[ii+2]]))
CVar[ik]<-FS1+FS2+FS3   # all 3 components
}
CVar
plot(CVar)

iiende<-level*3+1
CVarende <- (1/n) * sum(abs(F1.dwt[[iiende]]*F2.dwt[[iiende]]))
CVartotal<-sum(CVar)+CVarende
```

```
CVartotal

# windows()
# plot(CVar)
CVar<-round(CVar,4)
CVar
}
##########################################################################



##########################################################################
plot.rvi<-function(A,level){
##########################################################################

nvar<-dim(A)[2]-6
klimitscale<-dim(A)[3]
ip<-dim(A)[1]

WeightSums<-matrix(NA,nvar,klimitscale)
for (kscale in 1:klimitscale){
for(kvar in 2:(nvar+1)){
for (i in 1: ip){
if(!is.na(A[i,kvar,kscale])) A[i,kvar,kscale]<-A[i,(nvar+6),kscale]
}}
B<-A[1:ip,2:(nvar+1),kscale]
WeightSums[,kscale]<-colSums(B,na.rm=TRUE)
}  # kscale

windows()
vec<-1:nvar
plot(level,WeightSums[1,],type="b",ylim=c(0,2),
xlim=c(min(level),max(level)),ylab="Relative Variable Importance",
xlab="Level", pch=2,lty=vec[1],lwd=2)
for (kvar in 2:nvar) {
points(level,WeightSums[kvar,],type="b",pch=kvar+1,
                                                    lty=vec[kvar],lwd=2)
}
leg<-dimnames(mmi$res)[[2]][vec+1]
v<-2:(nvar+1)
legend(2,2,leg,pch=v,lty=vec,lwd=2)
}
##########################################################################

##########################################################################
```