```
occ.multiscale.fp.mcmc <- function(y,ctrl,groups,W,U,X,priors,start,tune,n.mcmc,adapt=TRUE){

        ###
        ###  Libraries and subroutines
        ###

        expit <- function(logit){
                exp(logit)/(1+exp(logit))
        }

        logit <- function(expit){
                log(expit/(1-expit))
        }

        get.tune <- function(tune,keep,k,target=0.44){  # adaptive tuning
                a <- min(0.01,1/sqrt(k))
                # a <- min(0.025,1/sqrt(k))
                exp(ifelse(keep<target,log(tune)-a,log(tune)+a))
        }

        y.lik <- function(y,p,phi,log=FALSE){
                tmp <- (1-phi)*(p^y)*((1-p)^(1-y))+(phi*y)
                if(log) tmp <- log(tmp)
                tmp
        }



        ###
        ###  Create variables
        ###
```

```
# browser()
        N <- nrow(X)  # number of sample units
        J <- tapply(groups$U[,2],groups$U[,1],function(x) length(x))  # number of subunits per sample
unit
        J.sum <- sum(J)  # total number of subunits in data set

        y.inv <- ifelse(y==1,0,1)
        qX <- ncol(X)
        qU <- ncol(U)
        qW <- ncol(W)

        # Create indicator variable that maps latent 'occupancy' state (z) to 'use' state (a)
        z.map <- match(groups$U$unit,groups$X$unit)

        # Create indicator variable that maps latent 'use' state (a) to observations (y)
        a.map <-
match(paste(groups$W$unit,groups$W$subunit),paste(groups$U$unit,groups$U$subunit))



        ###
        ###  Starting values
        ###

        z <- c(start$z)  # latent occupancy state
        beta <- as.vector(start$beta)  # coefficients for psi (occupancy probability)
        gamma <- as.vector(start$gamma)  # coefficients for theta (probability of use)
        alpha <- as.vector(start$alpha)  # coefficients for p (probability of detection)
        psi <- expit(X%*%beta)  # occupancy probability
        theta <- expit(U%*%gamma)  # probability of use
        p <- expit(W%*%alpha)  # detection probability
```

```
###
###  Priors
###

mu.beta <- matrix(priors$mu.beta,qX,1)  # prior mean for beta (coefficients for psi)
mu.gamma <- matrix(priors$mu.gamma,qU,1)  # prior mean for gamma (coefficients for theta)
mu.alpha <- matrix(priors$mu.alpha,qW,1)  # prior mean for alpha (coefficients for p)
sigma.beta <- priors$sigma.beta  # prior standard deviation for beta (coefficients for psi)
sigma.gamma <- priors$sigma.gamma  # prior standard deviation for gamma (coefficients for
theta)
sigma.alpha <- priors$sigma.alpha  # prior standard deviation for alpha (coefficients for p)



###
###  Create receptacles for output
###

beta.save <- matrix(0,n.mcmc,qX)
gamma.save <- matrix(0,n.mcmc,qU)
alpha.save <- matrix(0,n.mcmc,qW)
z.mean <- numeric(N)
a.mean <- z.map*0
phi.save <- numeric(n.mcmc)

keep <- list(beta=0,gamma=0,alpha=0)  # number of MH proposals accepted
keep.tmp <- keep  # for adaptive tuning
Tb <- 50  # frequency of adaptive tuning
```

```
	###
	###  Begin MCMC loop
	###

	for(k in 1:n.mcmc){
		if(k%%1000==0) cat(k," "); flush.console()

		###
		### Adaptive tuning
		###

		if(adapt==TRUE & k%%Tb==0) {  # Adaptive tuning
			keep.tmp <- lapply(keep.tmp,function(x) x/Tb)
			tune$beta <- get.tune(tune$beta,keep.tmp$beta,k)
			tune$gamma <- get.tune(tune$gamma,keep.tmp$gamma,k)
			tune$alpha <- get.tune(tune$alpha,keep.tmp$alpha,k)
			keep.tmp <- lapply(keep.tmp,function(x) x*0)
		}


		###
		###  Sample a
		###
# browser()
		z.tmp <- z[z.map]
		p1 <- z.tmp*theta*c(tapply(y.lik(y,p,phi),a.map,prod))
		p0 <- (1-z.tmp*theta)*c(tapply((phi^y)*((1-phi)^(1-y)),a.map,prod))
		# boxplot(p0~(tapply(y,a.map,sum)>0))
		theta.tmp <- p1/(p1+p0)
		# boxplot(theta.tmp~(tapply(y,a.map,sum)>0))
```

```
                a <- rbinom(J.sum,1,theta.tmp)



                ###
                ###  Sample z
                ###
# browser()
                a.inv <- ifelse(a==1,0,1)
                p1 <- psi*c(tapply((theta^a)*((1-theta)^(1-a)),z.map,prod))
                p0 <- (1-psi)*c(tapply(a.inv,z.map,prod))
                # boxplot(p0~(tapply(a,z.map,sum)>0))
                psi.tmp <- p1/(p1+p0)
                # boxplot(psi.tmp~(tapply(a,z.map,sum)>0))
                z <- rbinom(N,1,psi.tmp)



                ###
                ###  Sample beta (psi)
                ###

                beta.star <- rnorm(qX,beta,tune$beta)
                psi.star <- expit(X%*%beta.star)
                mh.star <- sum(dbinom(z,1,psi.star,log=TRUE))+
                        sum(dnorm(beta.star,mu.beta,sigma.beta,log=TRUE))
                mh.0 <- sum(dbinom(z,1,psi,log=TRUE))+
                        sum(dnorm(beta,mu.beta,sigma.beta,log=TRUE))
                if(exp(mh.star-mh.0) > runif(1)){
                        beta <- beta.star
                        psi <- psi.star
                        keep$beta <- keep$beta+1
```

```
                keep.tmp$beta <- keep.tmp$beta+1
        }



        ###
        ###  Sample gamma (theta)
        ###

        idx <- which(z[z.map]==1)  # Update depends only on subunits within occupied units
        gamma.star <- rnorm(qU,gamma,tune$gamma)
        theta.star <- expit(U%*%gamma.star)
        mh.star <- sum(dbinom(a[idx],1,theta.star[idx],log=TRUE))+
                sum(dnorm(gamma.star,mu.gamma,sigma.gamma,log=TRUE))
        mh.0 <- sum(dbinom(a[idx],1,theta[idx],log=TRUE))+
                sum(dnorm(gamma,mu.gamma,sigma.gamma,log=TRUE))
        if(exp(mh.star-mh.0) > runif(1)){
                gamma <- gamma.star
                theta <- theta.star
                keep$gamma <- keep$gamma+1
                keep.tmp$gamma <- keep.tmp$gamma+1
        }



        ###
        ###  Sample alpha (p)
        ###
# browser()
        idx <- which(a[a.map]==1)
        alpha.star <- rnorm(qW,alpha,tune$alpha)
        p.star <- expit(W%*%alpha.star)
```

```
                mh.star <- sum(y.lik(y[idx],p.star[idx],phi,log=TRUE))+
                        sum(dnorm(alpha.star,mu.alpha,sigma.alpha,log=TRUE))
                mh.0 <- sum(y.lik(y[idx],p[idx],phi,log=TRUE))+
                        sum(dnorm(alpha,mu.alpha,sigma.alpha,log=TRUE))
                if(exp(mh.star-mh.0) > runif(1)){
                        alpha <- alpha.star
                        p <- p.star
                        keep$alpha <- keep$alpha+1
                        keep.tmp$alpha <- keep.tmp$alpha+1
                }



                ###
                ### Sample phi (probability of a false positive)
                ###
# browser()
                phi <- rbeta(1,ctrl$v+priors$a,ctrl$M-ctrl$v+priors$b)



                ###
                ###  Save samples
                ###

                beta.save[k,] <- beta
                gamma.save[k,] <- gamma
                alpha.save[k,] <- alpha
                phi.save[k] <- phi
                a.mean <- a.mean+a
                z.mean <- z.mean+z
        }
```

```
        cat("\n")


        ###
        ###  Write output
        ###


        z.mean <- z.mean/n.mcmc
        a.mean <- a.mean/n.mcmc


        keep <- lapply(keep,function(x) x/n.mcmc)
        end <- list(beta=beta,gamma=gamma,alpha=alpha,z=z,a=a,phi=phi)  # ending values


        list(beta=beta.save,gamma=gamma.save,alpha=alpha.save,a.mean=a.mean,z.mean=z.mean,phi
=phi.save,

        keep=keep,end=end,y=y,X=X,U=U,W=W,priors=priors,start=start,tune=tune,n.mcmc=n.mcmc)
}
```