Supplementary Appendix S1: Computer code

This appendix contains the R source code used for our MCMC method (the function "`evol.rate.mcmc()`" and several additional functions that are used to pre-process the posterior sample) as well as some very basic instructions on loading the source and using the method in R. Updated versions of this method will be distributed with the R phylogenetics package "phytools" (Revell 2011). The method calls functions from the R package "ape" (Paradis et al. 2004), so this package should be installed before use.

```
# these functions uses a Bayesian MCMC approach to estimate heterogeneity in the
# evolutionary rate for a continuous character
# based on Revell, Mahler, Peres-Neto, & Redelings. In revision.
# code written by Liam J. Revell 2010 (updated 2011)

# function for Bayesian MCMC
# written by Liam Revell 2010/2011
evol.rate.mcmc<-function(tree,x,ngen=10000,control=list()){
 # check for & load "ape"
 if(!require(ape)) stop("must first install 'ape' package.") # require ape
 # some minor error checking
 if(class(tree)!="phylo") stop("tree object must be of class 'phylo.'")
 if(is.matrix(x)) x<-x[,1]
 if(is.null(names(x))){
  if(length(x)==length(tree$tip)){
   message("x has no names; assuming x is in the same order as tree$tip.label")
   names(x)<-tree$tip.label
  } else
   stop("x has no names and is a different length than tree$tip.label")
 }
 if(any(is.na(match(tree$tip.label,names(x))))){
  message("some species in tree are missing from data, dropping missing taxa from the
tree")
  tree<-drop.tip(tree,tree$tip.label[-match(names(x),tree$tip.label)])
 }
 if(any(is.na(match(names(x),tree$tip.label)))){
  message("some species in data are missing from tree, dropping missing taxa from the
data")
  x<-x[tree$tip.label]
 }
 if(any(is.na(x))){
  message("some data given as 'NA', dropping corresponding species from tree")
  tree<-drop.tip(tree,names(which(is.na(x))))
 }
 # first, try and obtain reasonable estimates for control parameters
 # and starting values for the MCMC
 C<-vcv.phylo(tree); C<-C[names(x),names(x)]; n<-nrow(C); one<-matrix(1,n,1)
 a<-colSums(solve(C))%*%x/sum(solve(C)) # MLE ancestral value, used to start MCMC
 # MLE sigma-squared, used to start MCMC
 sig1<-as.numeric(t(x-one%*%a)%*%solve(C)%*%(x-one%*%a)/n)
 sig2<-sig1 # used to start MCMC
 flipped=FALSE # used to start MCMC
 # populate control list
 con=list(sig1=sig1,sig2=sig2,a=as.numeric(a),sd1=0.2*sig1,sd2=0.2*sig2,
sda=0.2*abs(as.numeric(a)),kloc=0.2*mean(diag(C)),sdlnr=1,rand.shift=0.05,print=100,
sample=100)
 # also might use: sig1mu=1000,sig2mu=1000
```

```
con[(namc <- names(control))] <- control
con<-con[!sapply(con,is.null)]
# print control parameters to screen
message("Control parameters (set by user or default):"); str(con)
# now detach the starting parameter values (to be compatible with downstream code)
sig1<-con$sig1; sig2<-con$sig1; a<-con$a
# all internal functions start here
# function to return the index of a random edge
random.node<-function(phy){
 # sum edges cumulatively\
 cum.edge<-vector(mode="numeric"); index<-vector(mode="numeric")
 for(i in 1:length(phy$edge.length)){
  if(i==1) cum.edge[i]<-phy$edge.length[1]
  else cum.edge[i]<-cum.edge[i-1]+phy$edge.length[i]
  index[i]<-phy$edge[i,2]
 }
 # pick random position
 pos<-runif(1)*cum.edge[length(phy$edge.length)]
 edge<-1
 while(pos>cum.edge[edge]) edge<-edge+1
 return (index[edge])
}
# return the indices of a vector that match a scalar
match.all<-function(s,v){
 result<-vector(mode="numeric")
 j=1
 for(i in 1:length(v)){
  if(s==v[i]){
   result[j]=i
   j=j+1
  }
 }
 if(j==1) result<-NA
 return(result)
}
# function to return a matrix of the descendant tips
# from each internal & terminal node
compute.descendant.species<-function(phy){
 D<-dist.nodes(phy)
 ntips<-length(phy$tip.label)
 Cii<-D[ntips+1,]
 C<-D; C[,]<-0
 counts<-vector()
 for(i in 1:nrow(D)) for(j in 1:ncol(D)) C[i,j]<-(Cii[i]+Cii[j]-D[i,j])/2
 tol<-1e-10; descendants<-matrix(0,nrow(D),ntips,dimnames=list(rownames(D)))
 for(i in 1:nrow(C)){
  k<-0
  for(j in 1:ntips){
   if(C[i,j]>=(C[i,i]-tol)){
    k<-k+1
    descendants[i,k]<-phy$tip.label[j]
   }
  }
  counts[i]<-k
 }
 names(counts)<-rownames(descendants)
 return(descendants=list(species=descendants,counts=counts))
}
# take a step on the tree (used in MCMC)
tree.step<-function(phy,node,bp,step,up=NA,flip=FALSE){
 if(step<0)
  step=-step # if user has given -step, positivize
 if(is.na(up))
```

```
    up=(runif(1)>0.5) # if up/down is not assigned in the middle of a branch
   # decide to go up (1) or down (0) with equal probability
   if(up){
    # pick a new position along the branch (or go to the end)
    new.bp<-min(bp+step,phy$edge.length[match(node,phy$edge[,2])])
    # adjust step length to remain step
    step=step-(new.bp-bp)
    # check to see if we're done
    if(step<1e-6){
     return(list(node=node,bp=new.bp,flip=flip))
    } else {
     # we're going up, so get the daughters
     daughters<-phy$edge[match.all(node,phy$edge[,1]),2]
     # pick a random daughter
     new.node<-daughters[ceiling(runif(1)*length(daughters))]
     if(is.na(new.node)){
      location<-tree.step(phy,node,phy$edge.length[match(node,phy$edge[,2])],
step,up=FALSE,flip) # we're at a tip
     } else {
      location<-tree.step(phy,new.node,0,step,up=TRUE,flip)
     }
    }
   } else {
    # pick a new position along the branch (or go to the start)
    new.bp<-max(bp-step,0)
    # adjust step length
    step=step-(bp-new.bp)
    # check to see if we're done
    if(step<1e-6){
     return(list(node=node,bp=new.bp,flip=flip))
    } else {
     # we're going down so find out who the parent is
     parent<-phy$edge[match(node,phy$edge[,2]),][1]
     # find the other daughter(s)
     daughters<-phy$edge[match.all(parent,phy$edge[,1]),2]
     # don't use parent if root
     if(parent==(length(phy$tip.label)+1)){
      parent=NULL # if at the base of the tree
     }
     # create a vector of the possible nodes: the parent, and sister(s)
     possible.nodes<-c(parent,daughters[-match(node,daughters)])
     # now pick randomly
     new.node<-possible.nodes[ceiling(runif(1)*length(possible.nodes))]
     # if parent
     if(is.null(parent)==FALSE&&new.node==parent){
      location<-tree.step(phy,new.node,phy$edge.length[match(new.node,phy$edge[,2])],
step,up=FALSE,flip)
     } else {
      location<-tree.step(phy,new.node,0,step,up=TRUE,flip=TRUE)
     }
    }
   }
  }
  # log-likelihood function
  likelihood<-function(y,phy,C,descendants,sig1,sig2,a,loc){
   C1<-matrix(0,nrow(C),ncol(C),dimnames=dimnames(C))
   C2<-matrix(0,nrow(C),ncol(C),dimnames=dimnames(C))
   n<-length(y)
   D<-matrix(1,n,1)
   if(loc$node>length(phy$tip.label)){
    tr1<-extract.clade(phy,loc$node)
    tr1$root.edge<-phy$edge.length[match(loc$node,phy$edge[,2])]-loc$bp
    temp<-vcv.phylo(tr1)+tr1$root.edge
```

```
   } else {
    temp<-matrix(phy$edge.length[match(loc$node,phy$edge[,2])]-loc$bp,
1,1,dimnames=list(c(phy$tip.label[loc$node]),c(phy$tip.label[loc$node])))
   }
   C2[rownames(temp),colnames(temp)]<-temp
   C1<-C-C2
   # tips<-phy$tip.label[-match(rownames(temp),phy$tip.label)]
   tips<-rownames(temp)
   V<-sig1*C1+sig2*C2
   logL<-as.numeric(-t(y-D%*%a)%*%solve(V)%*%(y-D%*%a)/2-n*log(2*pi)/2-
determinant(V)$modulus[1]/2)
   return(list(logL=logL,tips=tips))
 }
 # prior probability function
 log.prior<-function(s1,s2,a,location){
  # exponential prior
  # logpr<-dexp(s1,rate=1/con$sig1mu,log=TRUE)+dexp(s2,rate=1/con$sig2mu,log=TRUE)
  # log-normal
  logpr<-dnorm(log(s1)-log(s2),mean=0,sd=con$sdlnr,log=TRUE)-log(s1*s2)
  return(logpr)
 }
 # proposal on sig1 or sig2
 propose.sig<-function(sig,scale){
  # if normal
  sig.prime<-abs(sig+rnorm(n=1,sd=scale)) # normal proposal distribution
  # if cauchy
  # sig.prime<-abs(sig+rcauchy(n=1,scale=scale))
  return(sig.prime)
 }
 # proposal on a
 propose.a<-function(a,scale){
  # if normal
  a.prime<-a+rnorm(n=1,sd=scale) # normal proposal distribution
  return(a.prime)
 }
 # proposal on loc
 propose.loc<-function(phy,loc,k,r){
  loc.prime<-list()
  loc.prime$flip=FALSE
  if(runif(1)>r){
   # update node & bp by random walk: rexp()
   loc.prime<-tree.step(phy,loc$node,loc$bp,step=rexp(n=1,rate=1/k))
   # update node & bp by random walk: rnorm()
   # loc.prime<-tree.step(phy,loc$node,loc$bp,step=abs(rnorm(n=1,sd=sqrt(2)/k)))
  } else {
   loc.prime$node<-random.node(phy) # pick random branch
   loc.prime$bp<-runif(1)*phy$edge.length[match(loc.prime$node,phy$edge[,2])]
   if((runif(1)>0.5)) loc.prime$flip=TRUE
  }
  return(loc.prime)
 }

 # obtain remaining starting values for the MCMC
 location<-list()
 location$node<-random.node(tree)
 location$bp<-runif(1)*tree$edge.length[match(location$node,tree$edge[,2])]
 location$flip<-FALSE
 descendants<-compute.descendant.species(tree) # compute descendants
 logL<-likelihood(x,tree,C,descendants,sig1,sig2,a,location)$logL
 logpr<-log.prior(sig1,sig2,a,location)
 # create matrix for results
 results<-matrix(NA,floor(ngen/con$sample)+1,7,dimnames=list(c(0,1:(ngen/con$sample)),
c("state","sig1","sig2","a","node","bp","likelihood")))
```

```
 curr.gen<-matrix(NA,1,7,dimnames=list("curr",c("state","sig1","sig2","a","node",
"bp","likelihood")))
 results[1,]<-c(0,sig1,sig2,a,location$node,location$bp,logL) # populate the first row
 curr.gen[1,]<-results[1,]
 group.tips<-list(); tips<-list()
 group.tips[[1]]<-likelihood(x,tree,C,descendants,sig1,sig2,a,location)$tips
 tips[[1]]<-group.tips[[1]]
 message("Starting MCMC run:")
 print(results[1,])
 j<-2
 # now run Markov-chain
 for(i in 1:ngen){
  if(i%%4==1) sig1.prime<-propose.sig(sig1,scale=con$sd1) # update sig1
  else sig1.prime<-sig1
  if(i%%4==2) sig2.prime<-propose.sig(sig2,scale=con$sd2) # update sig2
  else sig2.prime<-sig2
  if(i%%4==3) a.prime<-propose.a(a,scale=con$sda) # update a
  else a.prime<-a
  if(i%%4==0){
   location.prime<-propose.loc(phy=tree,loc=location,k=con$kloc,r=con$rand.shift)
   if(location.prime$flip==TRUE){
    flipped.prime<-!flipped # flip the sigmas
   } else flipped.prime<-flipped
  } else {
   location.prime<-location
   flipped.prime<-flipped
  }
  if(!flipped.prime){
   temp<-likelihood(x,tree,C,descendants,sig1.prime,sig2.prime,a.prime,
location.prime)
   logpr.prime<-log.prior(sig1.prime,sig2.prime,a.prime,location.prime)
   if(exp(temp$logL+logpr.prime-curr.gen[1,"likelihood"]-logpr)>runif(1)){
    sig1<-sig1.prime; sig2<-sig2.prime; a<-a.prime; location<-location.prime
    logL<-temp$logL; logpr<-logpr.prime; flipped<-flipped.prime
    group.tips[[i+1]]<-temp$tips
   } else group.tips[[i+1]]<-group.tips[[i]]
  } else {
   temp<-likelihood(x,tree,C,descendants,sig2.prime,sig1.prime,a.prime,
location.prime)
   logpr.prime<-log.prior(sig2.prime,sig1.prime,a.prime,location.prime)
   if(exp(temp$logL+logpr.prime-curr.gen[1,"likelihood"]-logpr)>runif(1)){
    sig1<-sig1.prime; sig2<-sig2.prime; a<-a.prime; location<-location.prime
    logL<-temp$logL; logpr<-logpr.prime; flipped<-flipped.prime
    group.tips[[i+1]]<-setdiff(tree$tip.label,temp$tips)
   } else group.tips[[i+1]]<-group.tips[[i]]
  }
  rm(temp)
  curr.gen[1,]<-c(i,sig1,sig2,a,location$node,location$bp,logL)
  if(i%%con$print==0)
   print(curr.gen[1,])
  if(i%%con$sample==0){
   results[j,]<-curr.gen
   tips[[j]]<-group.tips[[i+1]]
   j<-j+1
  }
 }
 message("Done MCMC run.")
 # return results
 return(list(mcmc=results,tips=tips))
}

# this function finds the split with the minimum the distance
# to all the other splits in the sample
```

```
# written by Liam J. Revell 2010
minSplit<-function(tree,split.list,method="sum",printD=FALSE){
 # some minor error checking
 if(class(tree)!="phylo") stop("tree object must be of class 'phylo.'")
 # determine is split.list is a list from our mcmc run, or a matrix
 if(class(split.list)=="list") split.list<-split.list$mcmc[,c("node","bp")]
 else if(class(split.list)=="matrix") split.list<-split.list[,c("node","bp")]
 # start by creating a matrix of the nodes of the tree with their descendant nodes
 D<-dist.nodes(tree)
 ntips<-length(tree$tip.label)
 Cii<-D[ntips+1,]
 C<-D; C[,]<-0
 for(i in 1:nrow(D)) for(j in 1:ncol(D)) C[i,j]<-(Cii[i]+Cii[j]-D[i,j])/2
 tol<-1e-10; descendants<-matrix(0,nrow(D),ncol(D),dimnames=list(rownames(D)))
 for(i in 1:nrow(C)){
  k<-1
  for(j in 1:ncol(C)){
   if(C[i,j]>=(C[i,i]-tol)){
    descendants[i,k]<-j
    k<-k+1
   }
  }
 }
 distances<-matrix(0,nrow(split.list),nrow(split.list))
 for(i in 1:nrow(split.list)){
  for(j in i:nrow(split.list)){
   if(i!=j){
    # first, if on the same branch as current average
    # then just compute the difference
    if(split.list[i,1]==split.list[j,1]){
     distances[i,j]<-abs(split.list[i,2]-split.list[j,2])
    } else {
     distances[i,j]<-D[split.list[i,1],split.list[j,1]]
     # is the split j downstream
     if(split.list[j,1]%in%descendants[split.list[i,1],]){
      # downstream
      distances[i,j]<-distances[i,j]-(tree$edge.length[match(split.list[j,1],
tree$edge[,2])]-split.list[j,2])
      distances[i,j]<-distances[i,j]+(tree$edge.length[match(split.list[i,1],
tree$edge[,2])]-split.list[i,2])
     } else if(split.list[i,1]%in%descendants[split.list[j,1],]){
      # ancestral
      distances[i,j]<-distances[i,j]-
(tree$edge.length[match(split.list[i,1],tree$edge[,2])]-split.list[i,2])

      distances[i,j]<-distances[i,j]+(tree$edge.length[match(split.list[j,1],
tree$edge[,2])]-split.list[j,2])
     } else {
      # neither
      distances[i,j]<-distances[i,j]-(tree$edge.length[match(split.list[i,1],
tree$edge[,2])]-split.list[i,2])
      distances[i,j]<-distances[i,j]-(tree$edge.length[match(split.list[j,1],
tree$edge[,2])]-split.list[j,2])
     }
    }
    distances[j,i]<-distances[i,j]
   }
  }
 }
 if(method=="sumsq") distances<-distances^2
 if(method!="sumsq"&&method!="sum")
  message("allowable methods are 'sum' and 'sumsq' - using default method ('sum')")
 if(printD) print(distances)
```

```
 sum.dist<-colSums(distances)
 ind<-which.min(sum.dist) # this is the index of the minimum split
 # return minimum split
 return(list(node=split.list[ind,1],bp=split.list[ind,2]))
}

# function to analyze the posterior from evol.rate.mcmc()
# written by Liam Revell 2011
posterior.evolrate<-function(tree,ave.shift,mcmc,tips,showTree=FALSE){
 result<-matrix(NA,nrow(mcmc),7,dimnames=list(NULL,c("state","sig1","sig2","a","node",
"bp","likelihood")))
 tree$node.label<-NULL
 for(i in 1:nrow(mcmc)){
  shift=list(node=mcmc[i,"node"],bp=mcmc[i,"bp"])
  temp<-ave.rates(tree,shift,tips[[i]],mcmc[i,"sig1"],mcmc[i,"sig2"],ave.shift,
showTree=showTree)
  result[i,]<-c(mcmc[i,"state"],temp[[1]],temp[[2]],mcmc[i,"a"],mcmc[i,"node"],
mcmc[i,"bp"],mcmc[i,"likelihood"])
 }
 return(result)
}

# average the posterior rates
# written by Liam Revell 2011
ave.rates<-function(tree,shift,tips,sig1,sig2,ave.shift,showTree=TRUE){
 # first split and scale at shift
 unscaled<-splitTree(tree,shift)
 # now scale
 scaled<-unscaled
 if(length(setdiff(scaled[[1]]$tip.label,tips))!=1){
  scaled[[1]]$edge.length<-scaled[[1]]$edge.length*sig1
  scaled[[2]]$edge.length<-scaled[[2]]$edge.length*sig2
  if(!is.null(scaled[[2]]$root.edge))
   scaled[[2]]$root.edge<-scaled[[2]]$root.edge*sig2
 } else {
  scaled[[1]]$edge.length<-scaled[[1]]$edge.length*sig2
  scaled[[2]]$edge.length<-scaled[[2]]$edge.length*sig1
  if(!is.null(scaled[[2]]$root.edge))
   scaled[[2]]$root.edge<-scaled[[2]]$root.edge*sig1
 }
 # now bind
 tr.scaled<-paste.tree(scaled[[1]],scaled[[2]])
 if(showTree==TRUE) plot(tr.scaled)
 # now split tr.scaled and tree at ave.shift
 unscaled<-splitTree(tree,ave.shift)
 scaled<-splitTree(tr.scaled,ave.shift)
 # now compute the sig1 and sig2 to return
 sig1<-sum(scaled[[1]]$edge.length)/sum(unscaled[[1]]$edge.length)
 sig2<-sum(scaled[[2]]$edge.length)/sum(unscaled[[2]]$edge.length)
 return(list(sig1=sig1,sig2=sig2))
}

# function splits tree at split
# written by Liam Revell 2011
splitTree<-function(tree,split){
 if(split$node>length(tree$tip)){
  # first extract the clade given by shift$node
  tr2<-extract.clade(tree,node=split$node)
  tr2$root.edge<-tree$edge.length[which(tree$edge[,2]==split$node)]-split$bp
  #now remove tips in tr2 from tree
  tr1<-drop.clade(tree,tr2$tip.label)
  tr1$edge.length[match(which(tr1$tip.label=="NA"),tr1$edge[,2])]<-split$bp
 } else {
```

```
   # first extract the clade given by shift$node
   tr2<-list(edge=matrix(c(2L,1L),1,2),tip.label=tree$tip.label[split$node],
edge.length=tree$edge.length[which(tree$edge[,2]==split$node)]-split$bp,Nnode=1L)
   class(tr2)<-"phylo"
   # now remove tip in tr2 from tree
   tr1<-tree
   tr1$edge.length[match(which(tr1$tip.label==tr2$tip.label[1]),tr1$edge[,2])]<-
split$bp
   tr1$tip.label[which(tr1$tip.label==tr2$tip.label[1])]<-"NA"
  }
  trees<-list(tr1,tr2); class(trees)<-"multiPhylo"
  return(trees)
}

# function pastes subtree onto tip
# written by Liam Revell 2011
paste.tree<-function(tr1,tr2){
  if(length(tr2$tip)>1){
    temp<-tr2$root.edge; tr2$root.edge<-NULL
    tr1$edge.length[match(which(tr1$tip.label=="NA"),tr1$edge[,2])]<-
tr1$edge.length[match(which(tr1$tip.label=="NA"),tr1$edge[,2])]+temp
  }
  tr.bound<-bind.tree(tr1,tr2,where=which(tr1$tip.label=="NA"))
  return(tr.bound)
}

# function drops entire clade
# written by Liam Revell 2011
drop.clade<-function(tree,tip){
  tree<-drop.tip(tree,tip,trim.internal=FALSE)
  while(sum(tree$tip.label=="NA")>1){
    tree<-drop.tip(tree,"NA",trim.internal=FALSE)
  }
  return(tree)
}
```

The simplest way to run this is to simply copy the above function into a text file, save this source file, load the source in R along with a phylogenetic tree and dataset, and then run the analysis. This might look as follows:

```
> require(ape)
> source("evol.rate.mcmc.R")
> tree<-read.tree(file="treefile.tre") # treefile.tre is a tree in Newick format
> temp<-read.csv(file="datafile.csv",row.names=1)
> x<-as.vector(temp); names(x)<-rownames(temp) # convert to vector with names
> result<-evol.rate.mcmc(tree,x)
```

This will by default run for 10,000 generations and sample every 100 generations. "`result`" is a list containing two objects: "`$mcmc`" a matrix with the posterior sample; and "`$tips`" a vector containing a list of the set of tip species in σ_2^2 for each generation of the MCMC. We now need to pre-process the posterior sample. We can do this as follows, here excluding the first 2,000 generations (21 samples) as burn-in:

```
> min_split<-minSplit(tree,result$mcmc[21:101,c("node","bp")])
```

```
> mcmc<-posterior.evolrate(tree,min_split,result$mcmc[21:101,],result$tips[201:1001])
```

Here, the first line finds the split in posterior sample with the smallest summed distance to all the other samples; and the second line uses the split and our raw posterior to generate a pre-processed posterior sample as described in the main text.

In general, one should probably not use the default control parameters and settings (although some effort has been made to try and ensure that they are not totally preposterous in value). These can be adjusted as follows:

```
> result<-evol.rate.mcmc(tree,x,ngen=100000,control=list(sd2=2.0,sdlnr=3.0))
```

Here, the standard deviation of the proposal distribution for σ_2^2 has been set to 2.0 and the standard deviation of the log-normal distribution for the ratio σ_1^2/σ_2^2 has been increased to 3.0. We can also adjust the following controls: "`sd1`" is the standard deviation of the proposal distribution for σ_1^2; "`sda`" is the standard deviation of the proposal distribution for α; "`kloc`" is the scaling parameter ($1/\lambda$) for the symmetric exponential distribution of tree moves; "`rand.shift`" is the probability of moving to a random position in the tree; "`print`" controls the frequency with which results are printed to screen; and "`sample`" determines the sample frequency from the chain.

Once we have obtained our posterior sample, we can write it to file. The following command will create a tab-delimited output file that should be suitable for the MCMC diagnostics program, "Tracer."

```
> write.table(mcmc,file="mcmc.file.txt",sep="\t")
```

Alternatively, we can compute many of the diagnostics of "Tracer" in R using the "coda" package (Plummer et al. 2010). For instance:

```
> library(coda)
> burnin<-10000
> temp<-mcmc[,"sig2"]; class(temp)<-"mcmc"
> est.sig2<-mean(temp)
> ESS.sig2<-effectiveSize(temp)
> ci<-HPDinterval(temp,prob=0.95); sig2.ci<-c(ci[1,"lower"], ci[1,"upper"])
```

can be used to compute the estimate of σ_2^2 ("`est.sig2`") as the posterior mean; the effective

sample size for σ_2^2 ("`ESS.sig2`"); and 95% credible interval for σ_2^2 ("`sig2.ci`"). Other MCMC diagnostics are also available within "coda."

To visually inspect the trace of the log-likelihood we can create the plot of Figure 3A of the main text as follows:

```
> plot(result$mcmc[,"likelihood"],"l")
```

Another useful visualization is to plot the posterior probability for the shift point to each edge of the tree. This is a little trickier but can be accomplished as follows:

```
> temp<-hist(mcmc[,"node"],breaks=0.5:(length(tree$tip)+tree$Nnode+0.5))
> prob.node<-temp$density; names(prob.node)<-temp$mids
> prob.node<-prob.node[as.character(tree$edge[,2])]
> plot(tree); edgelabels(round(prob.node,digits=3))
```

Literature cited

Paradis, E., J. Claude, and K. Strimmer. 2004. APE: Analyses of phylogenetics and evolution in R language. *Bioinformatics* 20: 289-290.

Plummer, M., N. Best, K. Cowles, and K. Vines. 2010. Coda: Output analysis and diagnostics for MCMC. R package version 0.13-5.

Revell, L. J. 2011. phytools: Phylogenetic tools for comparative biology (and other things). R package version 0.0-4. Available: http://anolis.oeb.harvard.edu/~liam/R-phylogenetics

Supplementary Appendix S2: Proof of symmetry of the proposal distribution for randomly selecting a new point on an evolutionary tree

Introduction

Here we describe a proposal distribution with density $f(x, y)$ for randomly selecting a new point *y* on an evolutionary tree given a current point *x*, and show that the density $f(x, y)$ is symmetric in *x* and *y*. This symmetry allows us to set the Hastings ratio to 1 in the Metropolis-Hastings algorithm. The central intuition of this proof is that, given a specific distance traveled *l* and a specific route **r** from *x* to *y*, the probability of traveling the same route in the reverse direction must be the same. This follows from the fact that the same number of choices must be made for taking a left or right branch, and that the probability of such a choice is always $\frac{1}{2}$ regardless of the direction chosen. This is a generalization of the frequently used fact that modifying a Gaussian proposal distribution to reflect at the boundaries of a line segment preserves the symmetrical nature of the proposal.

Terminology

We can represent a route on the tree in terms of the leaf nodes or internal nodes that are visited on the way from the starting point *x* to the destination *y*. Thus a route **r** of length *n* has the form:

$r_1 \to r_2 \to ... \to r_{n-1} \to r_n$.

Here, we do not set $r_1 = x$ and $r_n = y$, but instead choose r_1 and r_n so that *x* is on the branch (r_1, r_2) and *y* is on the branch (r_{n-1}, r_n). This means that two paths on the tree with different starting and ending points can have the same route if they have the same start and end branches and the same nodes $r_2 ... r_{n-1}$ in between. We define the reverse route $\mathbf{r}^t$ as $r_n \to r_{n-1} \to ... \to r_2 \to r_1$.

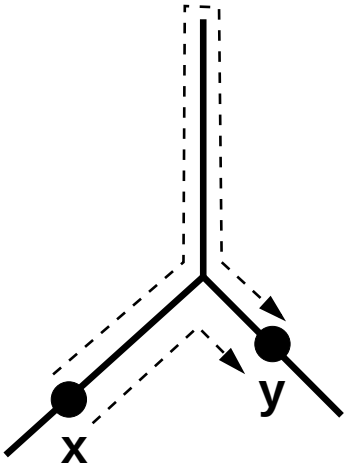

Figure 1. Two routes from *x* to *y* on a tree. The probability of each route is the same in both the forward and reverse directions.

We define the function $d(x, \mathbf{r}, y)$ to be the distance from *x* to *y* along the route **r**. If *x* is on the branch (r_1, r_2), then we say that **r** is "compatible" with *x*; if *y* is on the branch (r_{n-1}, r_n) then we

say that **r** is compatible with *y*. $d(x, \mathbf{r}, y)$ is only defined if **r** is compatible with *x* and *y*. The distance along a route is the same whether walking forwards or backwards, so $d(x, \mathbf{r}, y) = d(y, \mathbf{r}^t, x)$.

We define the function $W_{x,\mathbf{r}}(l)$ as the point obtained by starting from *x* and walking along route **r** for distance *l*. $W_{x,\mathbf{r}}(l)$ is only defined if *x* is compatible with **r**, and if the distance *l* is long enough to reach the final branch of **r**, but not so long that it overshoots and walks off the end of the branch.

Suppose we start from *x* and walk along **r** for a distance *l* to a point *y*, then the distance from *x* to *y* along **r** will be *l*. Stated mathematically, we have that $d(x, \mathbf{r}, W_{x,\mathbf{r}}(l)) = l$.

Proof

We seek to determine the density $f(x, y)$ such that $\Pr_x(Y \in [y, y + dy]) = f(x, y)dy$. We may ignore the possibility that *x* or *y* occur exactly on internal nodes of the tree, since this has zero probability of occurring. We assume that *L* is the random distance traveled along a random route **R**, and $Y = W_{x,\mathbf{R}}(L)$ is the random endpoint of this path.

Deriving the density $f(x, y)$ *of* $\Pr_x(Y)$

In order to determine the density $f(x, y)$, let us begin by considering the event:

$$\{Y \in [y, y + dy]\} = \{W_{x,\mathbf{R}}(L) \in [y, y + dy]\}.$$

We apply $d(x, \mathbf{R}, \cdot)$ to both $W_{x,\mathbf{R}}(L)$ and $[y, y + dy]$ to obtain:

$$\begin{aligned}\{Y \in [y, y + dy]\} &= \{L \in d(x, \mathbf{R}, [y, y + dy])\} \\ &= \{L \in [d(x, \mathbf{R}, y), d(x, \mathbf{R}, y + dy)]\}\end{aligned}$$

Note that this event implicitly includes the event that the route **R** is compatible with *x* and *y*. Now, changing *y* by an amount *dy* changes the distance $d(x, \mathbf{R}, y)$ by an amount *dy* or $-dy$. Therefore, we have

$$\{Y \in [y, y + dy]\} = \{L \in [d(x, \mathbf{R}, y), d(x, \mathbf{R}, y) \pm dy)]\}.$$

We may now write:

$$f(x, y)dy = \Pr_x(Y \in [y, y + dy])$$

$$= \sum_{\mathbf{r}} \int_{\mathbb{R}+} \Pr_x(Y \in [y, y+dy] \mid \mathbf{R} = \mathbf{r}, L = l) \times \Pr_x(\mathbf{R} = \mathbf{r} \mid L = l) \times \Pr_x(L = l) dl$$
$$= \sum_{\mathbf{r}} \int_{\mathbb{R}+} \Pr_x(L \in [d(x, \mathbf{R}, y), d(x, \mathbf{R}, y) \pm dy] \mid \mathbf{R} = \mathbf{r}, L = l) \times \Pr_x(\mathbf{R} = \mathbf{r} \mid L = l) \times \Pr_x(L = l) dl$$
$$= \sum_{\mathbf{r}} \int_{\mathbb{R}+} 1\{l \in [d(x, \mathbf{r}, y), d(x, \mathbf{r}, y) \pm dy]\} \times \Pr_x(\mathbf{R} = \mathbf{r} \mid L = l) \times \Pr_x(L = l) dl$$

Here we have applied Bayes rule and introduced the variables **R** and *L* by summing or integrating over all possible values. We note that the probability of $L = l$ refers to a density with respect to *dl*. The function $1\{l \in [d(x, \mathbf{r}, y), d(x, \mathbf{r}, y) \pm dy]\}$ is zero except on the infinitesimal set $\{l \in [d(x, \mathbf{r}, y), d(x, \mathbf{r}, y) \pm dy]\}$, and so we replace the domain of integration for *l* with that set, as follows:

$$f(x, y)dy = \sum_{\mathbf{r}} \int_{[d(x, \mathbf{R}, y), d(x, \mathbf{R}, y) \pm dy]} \Pr_x(\mathbf{R} = \mathbf{r} \mid L = l) \times \Pr(L = l) dl$$

The width *dl* of the infinitesimal set is *dy*, and the value of *l* on this set is $d(x, \mathbf{R}, y)$. We may thus evaluate the integral over *l* by substituting in these values for *dl* and *l*. Therefore:

$$f(x, y)dy = \left[\sum_{\mathbf{r}} \Pr_x(\mathbf{R} = \mathbf{r} \mid L = d(x, \mathbf{r}, y)) \times \Pr(L = d(x, \mathbf{r}, y)) \right] dy \qquad (1)$$

Thus, $f(x, y)$ is given by the term in square brackets. Intuitively, equation (1) represents the density of proposing *y* from *x* as a sum over all routes from *x* to *y* of the density of proposing *x* to *y* along route **r**. The density along route **r** is decomposed into the density of selecting the appropriate length *L* to *y* along the route **r**, multiplied by the probability of taking this particular route among all routes of length *L*.

Symmetry

In order to show that the density $f(x, y)$ is symmetric under the interchange of *x* and *y* we begin by defining the density $f(x, \mathbf{r}, y)$ from *x* to *y* along route **r** as:

$$f(x, \mathbf{r}, y) = \Pr_x(\mathbf{R} = \mathbf{r} \mid L = d(x, \mathbf{r}, y)) \times \Pr(L = d(x, \mathbf{r}, y)) . \qquad (2)$$

We seek to show that $f(x, \mathbf{r}, y) = f(y, \mathbf{r}^t, x)$. Now, the term $\Pr(L = d(x, \mathbf{r}, y))$ has this symmetry, since $d(x, \mathbf{r}, y) = d(y, \mathbf{r}^t, x)$ as noted above. The term $\Pr_x(\mathbf{R} = \mathbf{r} \mid L = d(x, \mathbf{r}, y))$ is given by the following expression, where $I(\mathbf{r})$ is the number of internal nodes among $r_2 ... r_{n-1}$:

$$\Pr(\mathbf{R} = \mathbf{r} \mid L = d(x, \mathbf{r}, y)) = \left(\frac{1}{2}\right)^{I(\mathbf{r})+1}. \quad (3)$$

This is because the choice of r_{i+1} is deterministic given r_i, unless r_i is an internal node. An additional choice is made as to which direction the path moves away from *x*. Thus, there are $I(\mathbf{r})+1$ choices made, and each choice has probability $\frac{1}{2}$. Since $I(\mathbf{r}) = I(\mathbf{r}^t)$ and $d(x, \mathbf{r}, y) = d(y, \mathbf{r}^t, x)$, we have $\Pr_x(\mathbf{R} = \mathbf{r} \mid L = d(x, \mathbf{r}, y)) = \Pr_x(\mathbf{R} = \mathbf{r}^t \mid L = d(y, \mathbf{r}^t, x))$. Therefore, $f(x, \mathbf{r}, y)$ is symmetric.

Finally, to show that $f(x, y)$ is symmetric in *x* and *y* we note that

$$f(x, y) = \sum_{\mathbf{r}} f(x, \mathbf{r}, y) = \sum_{\mathbf{r}^t} f(x, \mathbf{r}, y) = \sum_{\mathbf{r}^t} f(y, \mathbf{r}^t, x) = f(y, x).$$

QED.

Supplementary Appendix S3: Complete results from simulation test of the method

Table S1. Summary of results from simulation 1. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 0.1$, and $\alpha = 0.0$. Column headers are: "$\hat{\sigma}_1^2$ (*a*, *g*, *m*)" – arithmetic (*a*) and geometric (*g*) means of the posterior same for σ_1^2, and median (*m*); "ESS" – effective sample size; "95% CI" – 95% credible interval; "$\hat{\sigma}_2^2$ (*a*, *g*, *m*)" – same as "$\hat{\sigma}_1^2$ (*a*, *g*, *m*)," but for σ_2^2; "node" and "bp" – node and position along the edge of the simulated rate shift; "node*" and "bp*" estimated rate shift; and, finally, "sp_1" and "sp_2" – number of tip species in σ_1^2 and σ_2^2, respectively.

	$\hat{\sigma}_1^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\sigma}_2^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp_1	sp_2
1	0.86	0.85	0.85	800	0.60	1.13	0.14	0.13	0.13	923	0.06	0.23	-0.2	800	-0.8	0.37	102	0.35	102	0.18	0.17	80	20
2	1.13	1.11	1.11	800	0.78	1.50	0.21	0.20	0.18	562	0.10	0.45	-0.1	800	-0.5	0.40	173†	0.01	102†	0.02	0.03	72	28
3	0.97	0.95	0.95	316	0.67	1.33	0.19	0.17	0.15	150	0.08	0.52	-0.5	800	-1.3	0.34	112	0.02	112	0.04	0.02	70	30
4	0.85	0.83	0.83	605	0.54	1.20	0.12	0.12	0.12	800	0.08	0.17	-0.4	708	-0.9	0.13	102	0.15	102	0.05	0.09	44	56
5	1.07	1.05	1.04	800	0.70	1.46	0.10	0.09	0.09	553	0.06	0.15	0.2	911	-0.7	1.09	148	0.04	148	0.02	0.02	63	37
6	0.93	0.92	0.91	699	0.64	1.25	0.14	0.12	0.10	587	0.05	0.38	0.0	800	-0.8	0.98	133	0.03	133	0.08	0.04	75	25
7	1.06	1.04	1.04	800	0.72	1.43	0.08	0.08	0.08	800	0.04	0.12	-0.2	800	-0.8	0.48	155	0.01	155	0.07	0.07	60	40
8	0.98	0.96	0.97	658	0.63	1.33	0.10	0.10	0.10	712	0.07	0.15	0.1	800	-0.4	0.71	102	0.13	102	0.08	0.05	53	47
9	1.09	1.07	1.07	686	0.73	1.47	0.06	0.06	0.06	507	0.03	0.10	-0.4	706	-1.3	0.38	124	0.08	124	0.05	0.03	59	41
10	1.09	1.07	1.07	800	0.75	1.48	0.07	0.06	0.06	899	0.04	0.10	0.4	651	-0.3	0.99	105	0.03	105	0.08	0.04	65	35
11	1.02	1.01	1.00	800	0.76	1.37	0.09	0.09	0.09	807	0.05	0.15	0.0	800	-0.7	0.70	178	0.15	178	0.13	0.01	77	23
12	1.19	1.15	1.14	342	0.72	1.75	0.06	0.06	0.06	993	0.04	0.09	0.1	800	-0.7	0.72	137	0.28	137	0.11	0.17	36	64
13	1.44	1.42	1.42	210	0.95	1.98	0.31	0.27	0.25	44	0.10	0.78	-0.1	629	-0.7	0.58	104	0.03	103	0.02	0.05	77	23
14	1.28	1.26	1.25	689	0.85	1.73	0.07	0.07	0.07	710	0.04	0.12	-0.7	752	-1.5	0.07	165	0.02	165	0.03	0.01	65	35
15	1.09	1.08	1.08	701	0.74	1.42	0.11	0.10	0.10	694	0.05	0.18	0.0	800	-0.6	0.65	164	0.05	164	0.14	0.08	80	20
16	0.78	0.77	0.77	800	0.58	1.03	0.10	0.09	0.08	319	0.04	0.20	-0.8	800	-1.5	-0.1	149	0.10	149	0.07	0.04	80	20
17	1.19	1.17	1.17	800	0.85	1.60	0.18	0.17	0.16	800	0.07	0.42	-0.1	800	-1.0	0.61	142	0.15	142	0.09	0.05	75	25
18	0.76	0.75	0.75	800	0.55	1.02	0.17	0.15	0.14	386	0.05	0.38	-0.3	800	-1.0	0.24	179	0.11	179	0.11	0.00	80	20
19	1.25	1.23	1.24	687	0.86	1.65	0.10	0.09	0.09	800	0.05	0.14	0.4	800	-0.3	1.24	102	0.17	102	0.13	0.04	67	33
20	0.90	0.89	0.89	800	0.65	1.23	0.16	0.13	0.12	317	0.05	0.56	0.0	718	-0.5	0.53	181	0.02	181	0.06	0.04	80	20
Ave.	**1.05**	**1.03**	**1.03**	**680**	**0.71**	**1.42**	**0.13**	**0.12**	**0.11**	**618**	**0.06**	**0.27**	**-0.1**	**774**	**-0.8**	**0.56**	—	—	—	—	**0.05**	**68**	**32**
SD	**0.18**	**0.17**	**0.17**	**180**	**0.11**	**0.25**	**0.06**	**0.05**	**0.05**	**262**	**0.02**	**0.19**	**0.33**	**63.4**	**0.36**	**0.34**	—	—	—	—	**0.05**	**13**	**13**

†Edges subtend the root node and thus were scored as identical.

Table S2. Summary of results from simulation 2. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 1.0$ (i.e., no rate shift), and $\alpha = 0.0$. Column headers are as in Table S1.

	$\hat{\sigma}_1^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\sigma}_2^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp1	sp2
1	0.87	0.80	0.91	235	0.13	1.16	0.94	0.93	0.93	800	0.70	1.17	0.2	800	-0.7	1.02	107	0.07	104	0.18	0.30	67	33
2	1.00	0.98	0.98	660	0.74	1.30	0.97	0.96	0.96	739	0.70	1.30	-0.9	800	-1.8	-0.2	180	0.30	105	0.01	0.63	79	21
3	0.93	0.92	0.92	800	0.68	1.19	0.92	0.91	0.91	800	0.68	1.20	0.9	800	0.2	1.69	105	0.14	127	0.02	0.16	77	23
4	0.88	0.87	0.87	800	0.65	1.12	0.88	0.87	0.86	800	0.64	1.12	-0.1	800	-0.7	0.52	161	0.09	161	0.03	0.06	60	40
5	1.09	1.08	1.07	375	0.79	1.42	1.09	1.08	1.07	435	0.79	1.42	0.1	723	-0.5	0.75	138	0.06	145	0.01	0.13	37	63
6	1.23	1.22	1.22	800	0.91	1.60	1.25	1.24	1.23	701	0.91	1.61	0.3	757	-0.5	1.12	142	0.12	143	0.07	0.27	41	59
7	0.98	0.97	0.97	743	0.74	1.25	1.09	1.04	0.99	170	0.67	1.78	-0.5	800	-1.4	0.33	139	0.20	135	0.00	0.24	68	32
8	1.11	1.09	1.09	800	0.82	1.43	1.12	1.10	1.10	518	0.80	1.46	0.4	800	-0.2	1.18	134	0.04	103	0.01	0.06	47	53
9	1.37	1.36	1.35	769	0.98	1.78	1.40	1.38	1.38	659	0.98	1.81	-0.4	800	-1.4	0.50	137	0.05	103	0.03	0.08	50	50
10	1.20	1.19	1.18	800	0.85	1.53	1.19	1.18	1.17	800	0.91	1.59	0.1	670	-1.0	1.35	147	0.10	105	0.01	0.15	71	29
11	1.12	1.11	1.11	800	0.84	1.46	1.12	1.11	1.10	706	0.77	1.42	0.1	644	-0.7	0.97	116	0.10	144	0.05	0.09	43	57
12	1.03	1.02	1.01	800	0.73	1.32	1.06	1.04	1.03	706	0.79	1.44	0.2	800	-0.5	1.02	103	0.09	140	0.03	0.11	62	38
13	1.42	1.41	1.40	625	1.08	1.85	1.41	1.39	1.39	598	1.02	1.82	0.0	922	-0.8	0.79	164	0.02	165	0.02	0.06	63	37
14	1.04	1.03	1.02	549	0.76	1.32	1.04	1.03	1.02	531	0.77	1.35	-0.2	689	-1.1	0.63	103	0.25	103	0.04	0.21	65	35
15	0.86	0.85	0.85	915	0.65	1.13	0.86	0.85	0.85	800	0.66	1.15	-0.3	800	-1.0	0.45	126	0.10	119	0.08	0.18	80	20
16	1.29	1.27	1.27	635	0.95	1.67	1.27	1.25	1.26	422	0.94	1.68	0.5	800	-0.6	1.47	130	0.33	102	0.21	0.45	80	20
17	0.82	0.81	0.80	800	0.58	1.05	0.82	0.81	0.81	800	0.59	1.06	0.2	800	-0.6	0.91	103	0.04	145	0.05	0.08	57	43
18	0.98	0.97	0.97	800	0.69	1.24	1.00	0.98	0.97	800	0.70	1.31	-0.2	800	-1.0	0.45	171	0.20	171	0.00	0.19	77	23
19	1.01	1.00	1.00	398	0.66	1.37	1.25	1.21	1.15	239	0.71	2.08	0.0	800	-0.9	0.85	165	0.05	122	0.08	0.34	78	22
20	0.92	0.91	0.91	715	0.67	1.19	0.91	0.90	0.90	652	0.68	1.21	-0.2	800	-0.8	0.52	166	0.02	122	0.05	0.11	65	35
Ave.	**1.06**	**1.04**	**1.05**	**691**	**0.75**	**1.37**	**1.08**	**1.06**	**1.05**	**634**	**0.77**	**1.45**	**0.0**	**780**	**-0.8**	**0.82**	**—**	**—**	**—**	**—**	**0.19**	**63**	**37**
SD	**0.17**	**0.17**	**0.17**	**176**	**0.19**	**0.22**	**0.17**	**0.17**	**0.17**	**192**	**0.12**	**0.28**	**0.4**	**60**	**0.4**	**0.43**	**—**	**—**	**—**	**—**	**0.15**	**14**	**14**

Table S3. Summary of results from simulation 3. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 5.0$, and $\alpha = 0.0$. Column headers are as in Table S1.

	$\hat{\sigma}_1^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\sigma}_2^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp1	sp2
1	0.75	0.74	0.74	800	0.53	1.00	7.42	7.09	7.02	230	3.59	12.40	0.2	800	-0.5	0.75	104	0.04	103	0.33	0.10	80	20
2	0.82	0.81	0.80	800	0.54	1.16	3.71	3.63	3.64	621	2.40	5.32	-0.3	800	-1.0	0.50	112	0.16	112	0.17	0.02	55	45
3	1.39	1.36	1.33	114	0.86	2.01	3.73	3.44	3.58	176	1.34	6.45	-0.2	800	-1.0	0.69	142	0.11	142	0.20	0.09	79	21
4	1.16	1.14	1.13	784	0.77	1.57	5.25	5.06	4.99	331	2.79	8.24	-0.8	896	-1.7	0.20	135	0.00	135	0.04	0.04	73	27
5	2.20	2.15	2.22	206	1.25	3.05	2.63	2.56	2.47	186	1.65	4.04	0.4	800	-1.0	1.79	127	0.14	127	0.01	0.13	59	41
6	1.67	1.63	1.67	119	0.84	2.23	2.27	2.15	1.97	139	1.35	4.35	-0.3	800	-1.3	0.59	173	0.07	173	0.00	0.07	72	28
7	0.94	0.92	0.91	365	0.61	1.25	4.09	3.83	3.85	209	1.48	6.95	0.3	908	-0.3	1.03	166	0.17	165	0.09	0.20	80	20
8	0.96	0.91	0.86	342	0.48	1.70	3.83	3.76	3.75	564	2.55	5.36	0.4	797	-0.7	1.52	104	0.04	104	0.06	0.02	47	53
9	1.02	0.99	0.97	800	0.58	1.51	4.85	4.76	4.74	784	3.22	6.74	0.0	390	-1.0	0.90	130	0.05	130	0.12	0.07	38	62
10	1.11	1.09	1.09	960	0.77	1.50	5.15	4.86	4.69	648	2.30	8.32	1.3	1057	0.3	2.32	152	0.00	152	0.00	0.00	71	29
11	1.27	1.25	1.27	252	0.78	1.72	1.42	1.40	1.37	241	0.97	2.00	-0.5	800	-1.6	0.49	105	0.00	104	0.00	0.01	75	25
12	1.73	1.67	1.63	88	1.03	2.88	4.43	4.30	4.27	319	2.43	6.50	0.0	956	-1.3	1.08	156	0.22	156	0.01	0.20	55	45
13	2.00	1.97	1.98	349	1.27	2.64	2.27	2.21	2.15	240	1.41	3.38	0.7	721	-0.6	1.72	164	0.02	157	0.02	0.09	79	21
14	0.86	0.85	0.85	800	0.58	1.15	4.27	4.13	4.05	287	2.33	6.62	-0.3	800	-0.8	0.38	171	0.12	170	0.06	0.14	76	24
15	1.06	1.05	1.04	1094	0.73	1.43	5.79	5.56	5.56	203	2.97	9.08	0.0	800	-0.6	0.56	168	0.07	168	0.12	0.04	75	25
16	1.10	1.08	1.07	800	0.75	1.52	4.32	4.17	4.10	314	2.54	6.87	0.2	800	-0.6	0.97	168	0.12	168	0.09	0.03	67	33
17	1.33	1.29	1.30	613	0.80	1.93	4.81	4.72	4.68	643	3.23	6.84	0.3	800	-1.2	1.69	102†	0.01	154†	0.12	0.13	47	53
18	1.16	1.14	1.15	800	0.78	1.65	5.18	5.02	4.96	232	2.69	7.95	-0.3	800	-1.1	0.33	131	0.05	132	0.02	0.15	60	40
19	1.27	1.24	1.21	202	0.81	1.92	4.43	4.26	4.26	350	2.21	6.94	0.0	800	-1.0	0.95	133	0.11	133	0.06	0.05	64	36
20	1.02	1.00	0.99	800	0.65	1.42	6.64	6.49	6.46	529	3.90	9.47	-0.2	800	-1.1	0.73	148	0.10	148	0.18	0.07	56	44
Ave.	**1.24**	**1.21**	**1.21**	**554**	**0.77**	**1.76**	**4.33**	**4.17**	**4.13**	**362**	**2.37**	**6.69**	**0.0**	**806**	**-0.9**	**0.96**	—	—	—	—	—	**65**	**35**
SD	**0.39**	**0.38**	**0.39**	**324**	**0.22**	**0.57**	**1.46**	**1.41**	**1.41**	**369**	**2.31**	**6.40**	**0.0**	**807**	**-0.9**	**0.97**	—	—	—	—	—	**65**	**35**

†Edges subtend the root node and thus were scored as identical.

Table S4. Summary of results from simulation 3. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 10.0$, and $\alpha = 0.0$. Column headers are as in Table S1.

	$\hat{\sigma}_1^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\sigma}_2^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp1	sp2
1	1.21	1.17	1.15	377	0.73	1.74	9.32	9.09	9.03	405	5.59	13.5	-0.5	646	-1.2	0.20	112	0.12	112	0.13	0.01	61	39
2	1.26	1.24	1.23	800	0.83	1.67	6.65	6.43	6.36	286	3.65	10.2	0.0	800	-0.8	0.93	105	0.09	105	0.10	0.01	74	26
3	1.14	1.12	1.10	800	0.75	1.57	8.47	8.17	8.11	145	4.35	13.0	-0.9	703	-1.8	-0.1	161	0.17	151	0.11	0.23	80	20
4	1.28	1.24	1.23	697	0.73	1.94	8.19	8.04	8.00	671	5.41	11.3	0.0	800	-1.2	1.00	104	0.04	104	0.04	0.01	43	57
5	1.10	1.07	1.08	535	0.65	1.59	10.3	9.90	9.72	648	4.88	16.0	-0.2	800	-1.0	0.65	163	0.18	166	0.00	0.11	72	28
6	0.90	0.88	0.87	800	0.60	1.23	8.63	8.43	8.40	459	5.33	12.4	0.0	699	-0.7	0.80	108	0.07	108	0.08	0.00	58	42
7	1.07	1.06	1.05	800	0.70	1.47	9.80	9.57	9.49	534	6.03	14.2	-0.3	800	-2.3	1.29	102	0.01	102	0.10	0.09	59	41
8	0.77	0.76	0.74	647	0.48	1.08	9.17	8.84	8.65	142	5.20	13.8	0.7	679	-0.9	2.41	102	0.20	102	0.18	0.02	67	33
9	1.26	1.23	1.22	800	0.87	1.75	12.0	11.5	11.4	200	6.93	19.0	0.0	800	-0.8	0.96	160	0.08	160	0.08	0.00	70	30
10	1.01	0.99	1.00	887	0.68	1.40	21.1	20.4	20.0	179	11.3	32.2	0.3	800	-0.5	1.12	153	0.05	153	0.06	0.01	68	32
11	1.05	1.04	1.03	888	0.77	1.42	10.2	9.73	9.55	119	4.58	16.4	0.1	962	-0.7	0.88	120	0.21	120	0.09	0.12	76	24
12	1.17	1.12	1.13	623	0.58	1.76	16.7	16.1	16.1	112	8.86	25.2	0.1	800	-0.8	0.83	174	0.11	175	0.00	0.02	73	27
13	1.25	1.23	1.22	800	0.79	1.78	9.37	9.15	9.04	574	5.49	13.4	0.1	800	-1.5	1.70	153	0.13	153	0.02	0.11	52	48
14	0.88	0.87	0.87	800	0.62	1.16	8.53	8.23	8.18	318	4.59	13.3	0.7	800	-0.1	1.72	157	0.12	157	0.08	0.04	77	23
15	0.94	0.92	0.92	800	0.66	1.31	8.76	8.48	8.32	216	4.92	13.1	0.2	721	-0.6	1.00	133	0.08	133	0.04	0.04	69	31
16	1.31	1.28	1.24	893	0.81	1.92	13.2	12.8	12.7	258	8.27	19.5	0.3	800	-0.6	1.21	166	0.02	166	0.06	0.04	65	35
17	1.25	1.22	1.21	800	0.82	1.66	12.8	12.3	12.2	800	6.76	20.4	0.1	547	-0.9	1.04	142	0.10	142	0.09	0.02	77	23
18	1.24	1.22	1.22	800	0.89	1.60	11.9	11.4	11.2	721	5.85	19.2	-0.1	704	-1.1	0.80	123	0.13	123	0.08	0.05	77	23
19	0.98	0.97	0.96	681	0.71	1.30	5.51	5.23	5.19	800	2.48	8.88	-1.0	1263	-1.8	-0.3	124	0.22	124	0.20	0.03	79	21
20	1.25	1.22	1.21	699	0.85	1.72	12.5	12.2	12.0	800	7.64	18.1	-1.6	693	-3.8	0.26	102†	0.01	140†	0.10	0.10	61	39
Ave.	**1.12**	**1.09**	**1.08**	**746**	**0.73**	**1.55**	**10.7**	**10.3**	**10.2**	**419**	**5.91**	**16.2**	**-0.1**	**781**	**-1.2**	**0.92**	—	—	—	—	**0.05**	**68**	**32**
SD	**0.16**	**0.15**	**0.15**	**127**	**0.11**	**0.25**	**3.52**	**3.40**	**3.35**	**253**	**1.98**	**5.5**	**0.6**	**142**	**0.8**	**0.62**	—	—	—	—	**0.06**	**10**	**10**

†Edges subtend the root node and thus were scored as identical.

Table S5. Summary of results from simulations of trees with number of taxa $N = 30$. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 10.0$, and $\alpha = 0.0$. Column headers are as in Table S1.

	$\hat{\sigma}_1^2$ (a, g, m)			ESS	95% CI		$\hat{\sigma}_2^2$ (a, g, m)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp1	sp2
1	2.61	2.36	2.51	362	0.63	4.66	3.96	3.68	3.52	270	1.53	7.18	-0.3	747	-1.6	1.26	40	0.29	40	0.03	0.27	15	15
2	1.04	0.78	0.69	116	0.21	3.42	9.83	9.28	9.26	463	4.75	16.7	0.67	800	-1.7	2.89	32†	0.00	50†	0.19	0.19	11	19
3	2.69	2.53	2.54	578	1.08	4.48	4.99	4.06	3.55	149	1.48	13.6	-0.5	706	-2.0	0.86	46	0.13	46	0.00	0.13	93	7
4	3.22	2.62	2.42	129	0.77	7.98	13.5	11.7	11.7	221	3.54	28.6	-0.3	900	-1.6	1.13	43	0.02	43	0.11	0.08	87	13
5	2.37	2.07	2.02	204	0.70	4.82	6.34	5.75	5.62	219	2.06	12.3	-0.3	800	-1.6	0.84	34	0.02	37	0.03	0.12	15	15
6	1.63	1.42	1.28	695	0.52	3.89	15.3	14.2	13.9	151	5.34	26.9	-0.1	800	-1.7	1.24	50	0.04	50	0.21	0.18	19	11
7	1.02	0.88	0.85	900	0.29	2.12	5.89	5.24	5.07	709	1.84	12.0	0.35	900	-0.9	1.51	52	0.11	52	0.06	0.05	91	9
8	4.46	3.56	4.56	242	0.38	8.58	7.31	6.97	6.85	360	3.52	12.0	-0.2	800	-2.2	1.97	41	0.13	41	0.11	0.02	10	20
9	2.65	2.42	2.58	187	0.62	4.59	4.64	3.97	3.55	130	1.63	12.3	0.68	601	-0.9	2.28	51	0.02	51	0.07	0.05	20	10
10	1.22	1.11	1.06	543	0.43	2.36	10.8	9.85	9.59	769	3.34	20.8	0.57	950	-0.5	1.62	45	0.01	45	0.03	0.03	19	11
11	2.01	1.90	1.84	308	1.08	3.41	1.79	1.71	1.71	800	0.82	2.84	-0.2	800	-1.5	1.17	32	0.71	44	0.18	1.00	24	6
12	1.32	1.27	1.25	557	0.71	2.21	1.41	1.35	1.33	529	0.72	2.31	-0.1	800	-1.0	0.87	52	0.21	35	0.00	0.59	21	9
13	1.96	1.54	1.33	438	0.48	5.48	8.03	7.47	7.38	711	3.17	14.3	0.25	758	-1.5	1.56	45	0.24	45	0.17	0.07	84	16
14	1.71	1.51	1.39	900	0.60	3.59	9.37	7.91	7.58	350	1.81	20.9	0.58	900	-1.2	2.14	54	0.10	54	0.21	0.11	93	7
15	0.86	0.80	0.78	900	0.41	1.45	14.1	12.5	12.2	452	4.23	28.8	0.18	812	-0.7	1.03	44	0.06	44	0.09	0.03	93	7
16	0.99	0.91	0.88	404	0.41	1.70	12.6	11.0	10.7	290	3.14	26.9	0.08	810	-0.9	1.15	37	0.33	37	0.14	0.19	91	9
17	2.28	2.04	2.12	276	0.63	4.24	4.95	4.24	3.82	386	1.50	11.7	0.31	900	-1.1	1.86	48	0.09	47	0.02	0.12	93	7
18	2.20	1.95	1.97	354	0.73	4.26	6.32	5.11	4.68	216	1.40	16.4	0.75	716	-0.5	2.31	52	0.08	52	0.08	0.01	92	8
19	1.09	1.03	1.00	900	0.48	1.86	22.7	20.0	19.4	147	6.34	46.5	-0.4	900	-1.4	0.52	48	0.02	48	0.09	0.07	93	7
20	1.36	1.23	1.17	679	0.55	2.76	12.1	10.8	10.6	539	2.86	25.4	0.0	900	-1.1	1.07	47	0.26	47	0.10	0.16	91	9
Ave.	**1.93**	**1.70**	**1.71**	**484**	**0.59**	**3.89**	**8.80**	**7.84**	**7.60**	**393**	**2.75**	**17.9**	**0.10**	**815**	**-1.3**	**1.46**	---	---	---	---	**0.17**	**58**	**11**
SD	**0.91**	**0.75**	**0.93**	**271**	**0.22**	**1.90**	**5.20**	**4.64**	**4.58**	**221**	**1.55**	**10.5**	**0.40**	**85**	**0.47**	**0.61**	---	---	---	---	**0.23**	**38**	**4**

†Edges subtend the root node and thus were scored as identical.

Table S6. Summary of results from simulations of trees with number of taxa $N = 50$. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 10.0$, and $\alpha = 0.0$. Column headers are as in Table S1.

	$\hat{\sigma}_1^2$ (a, g, m)			ESS	95% CI		$\hat{\sigma}_2^2$ (a, g, m)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp1	sp2
1	1.58	1.49	1.48	440	0.84	2.72	3.20	2.88	2.68	482	1.26	6.56	0.37	771	-1.0	1.58	54	0.09	52	0.04	0.48	40	10
2	0.63	0.61	0.61	800	0.32	1.02	6.84	6.50	6.34	123	3.24	11.6	-0.2	699	-1.6	1.38	54†	0.10	72†	0.01	0.39	31	19
3	1.45	1.38	1.34	139	0.80	2.48	7.02	6.43	6.55	139	2.18	12.8	0.26	800	-0.6	1.18	67	0.10	67	0.10	0.00	37	13
4	1.51	1.39	1.30	274	0.70	3.43	11.7	11.1	11.0	151	5.25	19.7	-0.3	933	-1.3	0.93	82	0.14	82	0.07	0.06	35	15
5	1.31	1.24	1.21	719	0.62	2.07	14.0	13.2	13.0	127	6.45	23.7	-0.1	630	-2.2	2.11	52	0.40	52	0.15	0.25	34	16
6	1.28	1.20	1.19	712	0.63	2.02	18.1	17.4	17.3	325	9.28	29.7	-0.3	800	-1.4	0.67	53	0.30	53	0.15	0.15	26	24
7	1.86	1.72	1.65	205	0.85	3.27	14.4	13.8	13.7	404	7.56	22.6	0.12	800	-1.3	1.85	76	0.04	76	0.01	0.03	25	25
8	0.65	0.61	0.60	800	0.31	1.08	7.40	7.10	6.92	285	3.98	11.8	-0.2	800	-0.9	0.41	67	0.03	67	0.02	0.01	22	28
9	1.64	1.53	1.45	621	0.74	2.99	20.8	15.1	14.2	104	3.16	62.1	-0.2	950	-1.2	0.86	54	0.18	59	0.02	0.18	39	11
10	1.15	1.09	1.06	389	0.54	1.97	4.79	4.25	4.13	115	1.38	10.3	-0.8	950	-1.8	0.0	85	0.31	85	0.34	0.03	34	16
11	0.80	0.78	0.78	811	0.48	1.19	10.8	9.88	9.72	263	4.09	20.58	0.34	900	-0.3	1.04	74	0.11	74	0.18	0.07	39	11
12	1.44	1.38	1.37	950	0.76	2.23	19.6	18.4	18.2	718	8.50	33.82	0.00	950	-1.0	1.05	81	0.23	81	0.15	0.08	33	17
13	1.26	1.20	1.17	278	0.64	2.16	10.8	9.64	9.76	135	2.85	21.0	-0.3	713	-1.4	0.93	69	0.03	69	0.05	0.02	37	13
14	1.19	1.13	1.10	207	0.64	1.84	9.82	9.13	9.00	132	3.98	18.1	0.11	800	-0.9	1.19	68	0.02	68	0.04	0.01	35	15
15	2.21	1.76	1.61	240	0.56	6.29	8.12	7.91	7.89	704	4.96	11.9	-0.7	800	-3.2	1.65	52	0.43	52	0.02	0.42	12	38
16	1.84	1.76	1.74	466	0.97	2.98	4.03	3.41	2.94	299	1.34	9.8	-0.7	950	-1.7	0.12	62	0.17	62	0.21	0.05	39	11
17	1.70	1.61	1.66	303	0.85	2.73	3.83	3.23	2.82	272	1.22	9.2	0.08	900	-0.9	1.02	88	0.21	88	0.06	0.14	40	10
18	1.09	1.03	1.00	800	0.49	1.75	10.6	10.2	10.1	421	5.94	16.9	0.52	800	-0.4	1.45	56	0.04	56	0.15	0.11	26	24
19	0.84	0.81	0.80	800	0.48	1.27	8.34	7.97	7.93	223	4.25	13.1	-1.0	1416	-1.6	-0.3	63	0.01	63	0.12	0.11	33	17
20	1.26	1.22	1.22	581	0.76	1.87	9.29	8.34	8.19	126	2.76	18.2	-0.2	800	-1.2	1.07	79	0.05	79	0.12	0.06	40	10
Ave.	**1.33**	**1.25**	**1.22**	**527**	**0.65**	**2.37**	**10.2**	**9.29**	**9.12**	**277**	**4.18**	**19.2**	**-0.2**	**858**	**-1.3**	**1.01**	---	---	---	---	**0.13**	**33**	**17**
SD	**0.41**	**0.36**	**0.34**	**260**	**0.18**	**1.16**	**5.09**	**4.51**	**4.49**	**187**	**2.40**	**12.3**	**0.40**	**160**	**0.64**	**0.61**	---	---	---	---	**0.14**	**7**	**7**

†Edges subtend the root node and thus were scored as identical.

Table S7. Summary of results from simulations of trees with number of taxa $N = 70$. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 10.0$, and $\alpha = 0.0$. Column headers are as in Table S1.

	$\hat{\sigma}_1^2$ (a, g, m)			ESS	95% CI		$\hat{\sigma}_2^2$ (a, g, m)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp1	sp2
1	0.80	0.78	0.76	800	0.47	1.16	16.5	15.8	15.6	268	8.55	27.6	0.14	716	-0.5	0.75	87	0.12	87	0.11	0.01	46	24
2	1.70	1.66	1.63	992	1.11	2.43	11.4	10.5	10.4	227	3.69	21.4	-0.1	900	-1.2	0.96	127	0.20	127	0.19	0.01	56	14
3	1.05	1.03	1.04	622	0.66	1.50	3.59	3.17	3.44	206	0.89	6.61	0.07	619	-0.8	1.00	113	0.11	113	0.06	0.04	53	17
4	1.09	1.06	1.06	800	0.68	1.57	9.24	8.81	8.66	160	4.06	15.3	0.00	800	-0.9	1.01	100	0.00	100	0.13	0.12	45	25
5	0.77	0.72	0.68	482	0.40	1.58	14.2	13.4	13.2	249	5.83	24.7	-0.3	900	-1.1	0.49	127	0.01	127	0.01	0.00	56	14
6	1.07	1.03	1.03	874	0.52	1.58	10.6	10.3	10.3	464	6.32	15.0	0.90	800	-1.1	2.80	72†	0.01	110†	0.00	0.02	31	39
7	1.21	1.18	1.17	800	0.75	1.75	9.28	8.93	8.78	285	4.78	14.4	-1.0	924	-1.9	-0.1	94	0.04	94	0.04	0.00	43	27
8	1.03	1.01	1.01	652	0.63	1.41	11.9	11.3	11.3	160	6.43	19.8	-0.3	1155	-1.1	0.60	122	0.05	122	0.07	0.02	51	19
9	0.91	0.86	0.82	900	0.47	1.63	11.6	10.9	10.7	230	4.74	20.1	-0.5	900	-1.3	0.21	113	0.04	113	0.03	0.02	53	17
10	0.92	0.90	0.89	800	0.61	1.34	9.11	8.77	8.76	221	4.98	14.4	0.5	697	-0.5	1.28	108	0.03	108	0.10	0.07	48	22
11	1.11	1.09	1.07	363	0.65	1.60	5.20	4.64	4.59	535	1.49	10.4	0.73	900	-0.2	1.53	83	0.08	85	0.04	0.11	56	14
12	0.80	0.79	0.79	800	0.52	1.11	6.55	6.18	6.09	119	2.87	11.1	0.02	891	-0.8	0.84	74	0.10	74	0.08	0.03	54	16
13	2.00	1.70	1.54	118	0.53	5.54	7.79	7.64	7.62	900	4.70	10.7	-0.1	1070	-2.2	1.70	72†	0.03	122†	0.16	0.19	19	51
14	0.91	0.89	0.89	900	0.58	1.28	11.4	10.7	10.5	312	5.01	20.7	-0.7	900	-1.7	0.12	124	0.32	124	0.19	0.12	53	17
15	0.96	0.94	0.93	714	0.62	1.36	14.3	13.6	13.2	111	7.29	24.8	0.0	800	-0.8	0.87	102	0.00	102	0.11	0.11	50	20
16	1.58	1.48	1.38	558	0.80	3.24	11.6	10.9	10.9	404	5.31	19.6	-0.5	882	-1.5	0.65	74	0.10	74	0.14	0.04	51	19
17	1.68	1.63	1.62	699	0.93	2.42	8.78	8.30	8.21	145	4.04	14.9	0.30	800	-0.9	1.35	101	0.05	101	0.03	0.02	48	22
18	0.99	0.96	0.95	900	0.60	1.38	8.99	8.45	8.35	463	3.66	15.5	0.09	730	-0.6	0.71	94	0.08	94	0.27	0.19	53	17
19	1.60	1.50	1.40	561	0.81	2.96	12.2	11.3	11.2	184	4.55	21.4	0.28	900	-0.9	1.58	127	0.08	127	0.03	0.05	56	14
20	1.28	1.24	1.21	340	0.71	1.85	6.39	6.16	6.19	647	3.26	9.87	0.11	898	-1.2	1.41	75	0.01	75	0.08	0.07	40	30
Ave.	**1.17**	**1.12**	**1.09**	**684**	**0.65**	**1.93**	**10.0**	**9.49**	**9.40**	**315**	**4.62**	**16.9**	**0.0**	**859**	**-1.1**	**0.99**	---	---	---	---	**0.06**	**48**	**22**
SD	**0.35**	**0.31**	**0.29**	**224**	**0.17**	**1.03**	**3.20**	**3.06**	**2.98**	**201**	**1.81**	**5.7**	**0.47**	**122**	**0.50**	**0.65**	---	---	---	---	**0.06**	**9**	**9**

†Edges subtend the root node and thus were scored as identical.

Table S8. Summary of results from simulations of trees with number of taxa $N = 200$. Generating values were $\sigma_1^2 = 1.0$, $\sigma_2^2 = 10.0$, and $\alpha = 0.0$. Column headers are as in Table S1.

	$\hat{\sigma}_1^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\sigma}_2^2$ (*a*, *g*, *m*)			ESS	95% CI		$\hat{\alpha}$	ESS	95% CI		node	bp	node*	bp*	dist.	sp1	sp2
1	1.29	1.28	1.27	800	0.99	1.63	9.18	9.00	8.99	413	6.19	13.0	0.07	800	-0.8	0.88	224	0.05	224	0.08	0.03	143	57
2	1.45	1.38	1.31	208	0.76	2.90	8.95	8.89	8.88	901	7.10	11.2	-0.8	800	-2.0	0.36	248	0.02	247	0.01	0.02	49	151
3	1.03	1.02	1.02	707	0.79	1.26	7.35	7.23	7.23	591	4.96	10.2	0.17	800	-0.7	1.06	228	0.19	228	0.13	0.06	146	54
4	1.08	1.07	1.08	800	0.84	1.32	10.1	9.89	9.79	232	6.24	14.4	-0.1	901	-1.2	0.83	355	0.28	355	0.14	0.14	154	46
5	0.93	0.92	0.92	800	0.71	1.23	9.44	9.34	9.31	654	6.90	12.3	0.15	672	-0.7	0.95	273	0.26	273	0.09	0.17	108	92
6	0.94	0.93	0.93	800	0.75	1.20	9.24	9.10	9.06	651	6.02	12.3	0.34	676	-0.6	1.33	209	0.00	207	0.11	0.09	137	63
7	1.18	1.17	1.16	688	0.87	1.54	9.61	9.51	9.55	800	7.11	12.6	0.56	718	-0.5	1.61	209	0.00	209	0.05	0.04	98	102
8	1.09	1.09	1.09	800	0.88	1.35	12.0	11.8	11.7	400	8.03	17.0	0.0	1057	-0.9	0.82	242	0.04	242	0.11	0.07	143	57
9	1.00	0.98	0.98	800	0.63	1.41	10.8	10.7	10.7	800	8.42	13.0	1.07	800	-0.1	2.19	239	0.13	239	0.15	0.03	51	149
10	0.90	0.90	0.89	800	0.66	1.13	10.7	10.6	10.5	692	7.71	13.8	0.68	800	-0.2	1.70	231	0.01	231	0.08	0.07	113	87
11	1.06	1.05	1.04	800	0.78	1.41	10.4	10.3	10.3	800	7.68	13.3	-0.5	800	-1.3	0.28	285	0.03	285	0.03	0.01	93	107
12	1.01	1.01	1.00	800	0.75	1.30	8.25	8.14	8.12	590	5.58	10.8	-0.1	800	-0.9	0.94	323	0.12	323	0.02	0.09	122	78
13	1.09	1.09	1.08	800	0.82	1.36	8.47	8.37	8.37	621	5.96	11.0	0.1	954	-0.6	1.03	269	0.05	269	0.03	0.02	121	79
14	0.94	0.94	0.93	858	0.74	1.18	8.95	8.74	8.63	265	5.59	13.0	-0.5	800	-1.2	0.32	343	0.07	343	0.04	0.03	156	44
15	1.08	1.07	1.06	800	0.81	1.35	8.32	8.14	8.02	307	5.47	12.2	-0.3	800	-1.5	0.76	357	0.12	357	0.09	0.03	156	44
16	1.04	1.02	1.02	800	0.66	1.42	10.9	10.8	10.8	800	8.30	13.4	-0.6	800	-1.4	0.13	241	0.08	241	0.06	0.02	62	138
17	1.15	1.14	1.11	652	0.82	1.60	10.1	9.94	9.87	800	7.26	12.8	0.80	1187	0.1	1.47	270	0.09	270	0.07	0.02	121	79
18	0.78	0.77	0.77	800	0.59	0.96	8.53	8.35	8.39	318	5.23	12.0	0.13	800	-0.5	0.73	294	0.04	294	0.08	0.04	150	50
19	1.19	1.18	1.17	947	0.93	1.46	7.46	7.29	7.19	340	4.66	10.8	-0.4	800	-1.2	0.45	204	0.00	204	0.15	0.15	154	46
20	1.06	1.05	1.03	800	0.78	1.39	12.1	11.8	11.7	192	7.09	17.6	-0.4	800	-1.2	0.42	317	0.03	317	0.05	0.01	158	42
Ave.	**1.07**	**1.05**	**1.04**	**763**	**0.78**	**1.42**	**9.54**	**9.40**	**9.35**	**558**	**6.57**	**12.8**	**0.0**	**828**	**-0.9**	**0.91**	---	---	---	---	**0.06**	**122**	**78**
SD	**0.15**	**0.14**	**0.13**	**143**	**0.10**	**0.38**	**1.34**	**1.33**	**1.32**	**229**	**1.14**	**1.9**	**0.49**	**120**	**0.51**	**0.54**	---	---	---	---	**0.05**	**35**	**35**