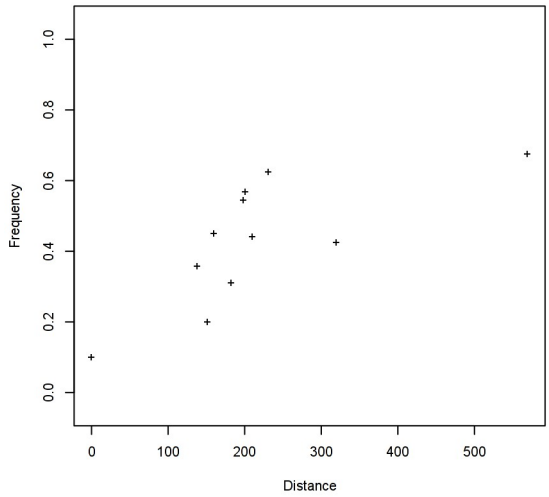
```
R version 2.14.2 (2012-02-29)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> ## Load the package
> library(hzar);
Loading required package: MCMCpack
Loading required package: coda
Loading required package: lattice
Loading required package: MASS
##
## Markov Chain Monte Carlo Package (MCMCpack)
## Copyright (C) 2003-2013 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
##
## Support provided by the U.S. National Science Foundation
## (Grants SES-0350646 and SES-0350613)
##
Loading required package: foreach
>
>
> ## Write out internal example data
> data(manakinMolecular);
> print(manakinMolecular);
   locationID distance ada.A ada.B ada.nSamples ak2.A ak2.B ak2.nSamples gsr.A
1           A     0.00 0.100 0.900           10 0.000 1.000           10 0.000
2           B   138.25 0.357 0.643           14 0.143 0.857           14 0.000
3           C   151.75 0.200 0.800           40 0.025 0.975           40 0.025
4           D   159.50 0.450 0.550           40 0.025 0.975           40 0.075
5           E   182.25 0.310 0.690           42 0.024 0.976           42 0.048
6           G   198.50 0.545 0.455           22 0.000 1.000           20 0.000
7           H   201.25 0.568 0.432           44 0.091 0.909           44 0.000
8           I   210.00 0.442 0.558           52 0.615 0.385           52 0.000
9           J   230.75 0.625 0.375           40 0.975 0.025           40 0.000
10          K   319.50 0.425 0.575           40 0.895 0.105           38 0.000
11          L   569.50 0.675 0.325           40 1.000 0.000           40 0.000
   gsr.B gsr.C gsr.D gsr.E gsr.nSamples pgm2.A pgm2.B pgm2.C pgm2.D
1  0.100 0.900 0.000 0.000           10  0.000  0.000  1.000  0.000
2  0.000 0.929 0.000 0.071           14  0.000  0.000  1.000  0.000
3  0.025 0.950 0.000 0.000           40  0.000  0.000  1.000  0.000
```

```
4  0.075 0.850 0.000 0.000           40  0.000  0.000  0.975  0.025
5  0.024 0.905 0.024 0.000           42  0.000  0.048  0.952  0.000
6  0.000 1.000 0.000 0.000           22  0.000  0.000  1.000  0.000
7  0.023 0.955 0.023 0.000           44  0.000  0.068  0.932  0.000
8  0.173 0.731 0.096 0.000           52  0.000  0.654  0.308  0.038
9  0.225 0.425 0.350 0.000           40  0.000  0.875  0.075  0.050
10 0.450 0.375 0.175 0.000           40  0.000  0.800  0.033  0.167
11 0.225 0.725 0.050 0.000           40  0.025  0.625  0.300  0.050
   pgm2.nSamples  l5.A  l5.B l5.nSamples pscn3.A pscn3.B pscn3.nSamples mtDNA.A
1             10 0.917 0.083          12   0.917   0.083             12   1.000
2             14 0.891 0.109          46   0.848   0.152             46   1.000
3             40 1.000 0.000          40   0.825   0.175             40   1.000
4             40 0.825 0.175          40   0.825   0.175             40   1.000
5             42 0.952 0.048          42   0.833   0.167             42   1.000
6             20 0.875 0.125          24   0.833   0.167             24   1.000
7             44 0.864 0.136          44   0.841   0.159             44   0.900
8             52 0.327 0.673          52   0.288   0.712             52   0.385
9             40 0.125 0.875          40   0.075   0.925             40   0.050
10            30 0.000 1.000          40   0.025   0.975             40   0.100
11            40 0.000 1.000          40   0.000   1.000             40   0.000
   mtDNA.B mtDNA.nSamples geneticHybridIndex.mu geneticHybridIndex.sigma
1    0.000             10                 0.984                     0.04
2    0.000             14                 0.908                     0.14
3    0.000             20                 0.973                     0.05
4    0.000             20                 0.912                     0.10
5    0.000             21                 0.936                     0.10
6    0.000             12                 0.923                     0.07
7    0.091             22                 0.867                     0.16
8    0.615             26                 0.325                     0.28
9    0.950             20                 0.056                     0.08
10   0.900             20                 0.043                     0.07
11   1.000             20                 0.005                     0.02
> write.table(manakinMolecular,     # The data we just loaded
+              file="mknExLoci.txt", # The file to overwrite
+              col.names=TRUE,       # The columns are named
+              row.names=FALSE,      # The rows are not named
+              sep="\t",             # The file will be tab-delimited
+              quote=TRUE)           # Use quotes as needed.
>
> ## As we no longer need the in-memory copy, drop the local reference
> manakinMolecular <- NULL
>
> ## Save all plots in a series of png files
> png(width=900, height=900, res=200, family="Arial", filename="mExPlot
    %03d.png",pointsize=8)
>
>
>
> ## A typical chain length.  This value is the default setting in the package.
> chainLength=1e5;
>
> ## Make each model run off a separate seed
> mainSeed=
+   list(A=c(596,528,124,978,544,99),
+        B=c(528,124,978,544,99,596),
```

```
+        C=c(124,978,544,99,596,528))
>
>
> if(require(doMC)){
+   ## If you have doMC, use foreach in parallel mode
+   ## to speed up computation.
+   registerDoMC()
+ } else {
+   ## Use foreach in sequential mode
+   registerDoSEQ();
+ }
Loading required package: doMC
Loading required package: iterators
Loading required package: multicore

Attaching package: ‘multicore’

The following object(s) are masked from ‘package:lattice’:

    parallel

>
>
> ## Molecular Analysis
>
> ## Load example Molecular data from the data table.
> manakinMolecular <- read.table("mknExLoci.txt",header=TRUE)
>
> ## Print sample data
> print(manakinMolecular)
   locationID distance ada.A ada.B ada.nSamples ak2.A ak2.B ak2.nSamples gsr.A
1           A     0.00 0.100 0.900           10 0.000 1.000           10 0.000
2           B   138.25 0.357 0.643           14 0.143 0.857           14 0.000
3           C   151.75 0.200 0.800           40 0.025 0.975           40 0.025
4           D   159.50 0.450 0.550           40 0.025 0.975           40 0.075
5           E   182.25 0.310 0.690           42 0.024 0.976           42 0.048
6           G   198.50 0.545 0.455           22 0.000 1.000           20 0.000
7           H   201.25 0.568 0.432           44 0.091 0.909           44 0.000
8           I   210.00 0.442 0.558           52 0.615 0.385           52 0.000
9           J   230.75 0.625 0.375           40 0.975 0.025           40 0.000
10          K   319.50 0.425 0.575           40 0.895 0.105           38 0.000
11          L   569.50 0.675 0.325           40 1.000 0.000           40 0.000
   gsr.B gsr.C gsr.D gsr.E gsr.nSamples pgm2.A pgm2.B pgm2.C pgm2.D
1  0.100 0.900 0.000 0.000           10  0.000  0.000  1.000  0.000
2  0.000 0.929 0.000 0.071           14  0.000  0.000  1.000  0.000
3  0.025 0.950 0.000 0.000           40  0.000  0.000  1.000  0.000
4  0.075 0.850 0.000 0.000           40  0.000  0.000  0.975  0.025
5  0.024 0.905 0.024 0.000           42  0.000  0.048  0.952  0.000
6  0.000 1.000 0.000 0.000           22  0.000  0.000  1.000  0.000
7  0.023 0.955 0.023 0.000           44  0.000  0.068  0.932  0.000
8  0.173 0.731 0.096 0.000           52  0.000  0.654  0.308  0.038
9  0.225 0.425 0.350 0.000           40  0.000  0.875  0.075  0.050
10 0.450 0.375 0.175 0.000           40  0.000  0.800  0.033  0.167
11 0.225 0.725 0.050 0.000           40  0.025  0.625  0.300  0.050
   pgm2.nSamples  l5.A  l5.B l5.nSamples pscn3.A pscn3.B pscn3.nSamples mtDNA.A
```

```
1             10 0.917 0.083          12   0.917   0.083             12   1.000
2             14 0.891 0.109          46   0.848   0.152             46   1.000
3             40 1.000 0.000          40   0.825   0.175             40   1.000
4             40 0.825 0.175          40   0.825   0.175             40   1.000
5             42 0.952 0.048          42   0.833   0.167             42   1.000
6             20 0.875 0.125          24   0.833   0.167             24   1.000
7             44 0.864 0.136          44   0.841   0.159             44   0.900
8             52 0.327 0.673          52   0.288   0.712             52   0.385
9             40 0.125 0.875          40   0.075   0.925             40   0.050
10            30 0.000 1.000          40   0.025   0.975             40   0.100
11            40 0.000 1.000          40   0.000   1.000             40   0.000
   mtDNA.B mtDNA.nSamples geneticHybridIndex.mu geneticHybridIndex.sigma
1    0.000             10                0.984                     0.04
2    0.000             14                0.908                     0.14
3    0.000             20                0.973                     0.05
4    0.000             20                0.912                     0.10
5    0.000             21                0.936                     0.10
6    0.000             12                0.923                     0.07
7    0.091             22                0.867                     0.16
8    0.615             26                0.325                     0.28
9    0.950             20                0.056                     0.08
10   0.900             20                0.043                     0.07
11   1.000             20                0.005                     0.02
> 
> ## ## Picking an allele for a locus
> ## useAlleles <- "ada.A";
> ## ua.nSamples <- "ada.nSamples"
> 
> ## Blank out space in memory to hold molecular analysis
> if(length(apropos("^mkn$",ignore.case=FALSE)) == 0 ||
+    !is.list(mkn) ) mkn <- list()
> ## We are doing just the one allele at one locus, but it is
> ## good to stay organized.
> mkn$AdaA <- list();
> ## Space to hold the observed data
> mkn$AdaA$obs <- list();
> ## Space to hold the models to fit
> mkn$AdaA$models <- list();
> ## Space to hold the compiled fit requests
> mkn$AdaA$fitRs <- list();
> ## Space to hold the output data chains
> mkn$AdaA$runs <- list();
> ## Space to hold the analysed data
> mkn$AdaA$analysis <- list();
> 
> 
> 
> ## Locus Ada, Allele A from Brumfield et al 2001
> mkn$AdaA$obs <-
+   hzar.doMolecularData1DPops(manakinMolecular$distance,
+                              manakinMolecular$ada.A,
+                              manakinMolecular$ada.nSamples);
> 
```

```
> ## Look at a graph of the observed data
> hzar.plot.obsData(mkn$AdaA$obs);
```




```
>
> ## Make a helper function
> mkn.loadAdaAmodel <- function(scaling,tails,
+                               id=paste(scaling,tails,sep="."))
+   mkn$AdaA$models[[id]] <<- hzar.makeCline1DFreq(mkn$AdaA$obs, scaling, tails)
>
> mkn.loadAdaAmodel("fixed","none","modelI");
> mkn.loadAdaAmodel("free" ,"none","modelII");
> mkn.loadAdaAmodel("free" ,"both","modelIII");
>
> ## Check the default settings
> print(mkn$AdaA$models)
$modelI
prior:
function (center, width, pMin, pMax)
{
    return(0)
}
<environment: namespace:hzar>
pExp:
pMin + (pMax - pMin) * (1/(1 + exp(-((x - center) * 4/width))))
req:
function (center, width, pMin, pMax)
return(pMin >= 0 & pMax <= 1 & pMin < pMax & width > 0)
<environment: namespace:hzar>
func:
function (center, width, pMin, pMax)
return(function(x) pMin + (pMax - pMin) * (1/(1 + exp(-((x -
    center) * 4/width)))))
Cline Parameters:
          init tune fixed lower upper
center 198.500  1.5 FALSE     0 569.5
width   64.750  1.5 FALSE     0 569.5
pMin     0.100  1.1  TRUE     0   1.0
pMax     0.675  1.1  TRUE     0   1.0

$modelII
prior:
function (center, width, pMin, pMax)
```

```
{
    return(0)
}
<environment: namespace:hzar>
pExp:
pMin + (pMax - pMin) * (1/(1 + exp(-((x - center) * 4/width))))
req:
function (center, width, pMin, pMax)
return(pMin >= 0 & pMax <= 1 & pMin < pMax & width > 0)
<environment: namespace:hzar>
func:
function (center, width, pMin, pMax)
return(function(x) pMin + (pMax - pMin) * (1/(1 + exp(-((x -
    center) * 4/width)))))
Cline Parameters:
          init tune fixed lower upper
center 198.500  1.5 FALSE     0 569.5
width   64.750  1.5 FALSE     0 569.5
pMin     0.100  1.1 FALSE     0   1.0
pMax     0.675  1.1 FALSE     0   1.0

$modelIII
req:
function (center, width, pMin, pMax, deltaL, tauL, deltaR, tauR)
return(width > 0 & deltaL >= 0 & deltaR >= 0 & (pMax - pMin) *
    (deltaL + deltaR) <= width * 50 & pMin >= 0 & pMax <= 1 &
    pMin < pMax & tauL >= 0 & tauL <= 1 & tauR >= 0 & tauR <=
    1)
<environment: namespace:hzar>
prior:
function (center, width, pMin, pMax, deltaL, tauL, deltaR, tauR)
{
    return(0)
}
<environment: namespace:hzar>
pExp:
pMin + (pMax - pMin) * ifelse(deltaL < -(x - center), 1/(1 +
    exp(4 * deltaL/width)) * exp(tauL/(1 + exp(-4 * deltaL/width)) *
    ((x - center) * 4/width + 4 * deltaL/width)), ifelse(deltaR <
    x - center, 1 - 1/(1 + exp(4 * deltaR/width)) * exp(-(tauR/(1 +
    exp(-4 * deltaR/width))) * ((x - center) * 4/width - 4 *
    deltaR/width)), 1/(1 + exp(-((x - center) * 4/width)))))
func:
function (center, width, pMin, pMax, deltaL, tauL, deltaR, tauR)
return(function(x) pMin + (pMax - pMin) * ifelse(deltaL < -(x -
    center), 1/(1 + exp(4 * deltaL/width)) * exp(tauL/(1 + exp(-4 *
    deltaL/width)) * ((x - center) * 4/width + 4 * deltaL/width)),
    ifelse(deltaR < x - center, 1 - 1/(1 + exp(4 * deltaR/width)) *
        exp(-(tauR/(1 + exp(-4 * deltaR/width))) * ((x - center) *
            4/width - 4 * deltaR/width)), 1/(1 + exp(-((x - center) *
        4/width))))))
Cline Parameters:
          init tune fixed lower upper
center 198.500  1.5 FALSE     0 569.5
width   64.750  1.5 FALSE     0 569.5
```

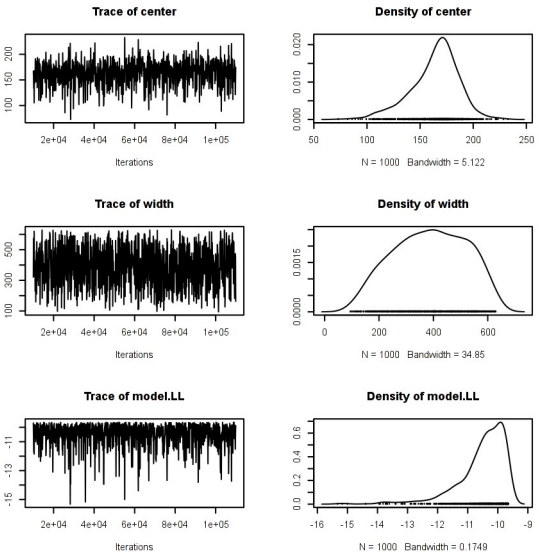
```
pMin     0.100  1.1 FALSE     0   1.0
pMax     0.675  1.1 FALSE     0   1.0
deltaL   1.000  1.5 FALSE     0 569.5
tauL     0.500  1.1 FALSE     0   1.0
deltaR   1.000  1.5 FALSE     0 569.5
tauR     0.500  1.1 FALSE     0   1.0

>
> ## Modify all models to focus on the region where the observed
> ## data were collected.
> ## Observations were between 0 and 570 km.
> mkn$AdaA$models <- sapply(mkn$AdaA$models,
+                           hzar.model.addBoxReq,
+                           -30 , 600,
+                           simplify=FALSE)
>
>
> ## Check the updated settings
> print(mkn$AdaA$models)
$modelI
prior:
function (center, width, pMin, pMax)
{
    return(0)
}
<environment: namespace:hzar>
pExp:
pMin + (pMax - pMin) * (1/(1 + exp(-((x - center) * 4/width))))
req:
function (center, width, pMin, pMax)
return(pMin >= 0 & pMax <= 1 & pMin < pMax & width < 630 & center >
    -30 & center < 600 & width > 0)
<environment: namespace:hzar>
func:
function (center, width, pMin, pMax)
return(function(x) pMin + (pMax - pMin) * (1/(1 + exp(-((x -
    center) * 4/width)))))
Cline Parameters:
          init tune fixed lower upper
center 198.500  1.5 FALSE   -30   600
width   64.750  1.5 FALSE     0   630
pMin     0.100  1.1  TRUE     0     1
pMax     0.675  1.1  TRUE     0     1

$modelII
prior:
function (center, width, pMin, pMax)
{
    return(0)
}
<environment: namespace:hzar>
pExp:
pMin + (pMax - pMin) * (1/(1 + exp(-((x - center) * 4/width))))
req:
function (center, width, pMin, pMax)
```

```
return(pMin >= 0 & pMax <= 1 & pMin < pMax & width < 630 & center >
    -30 & center < 600 & width > 0)
<environment: namespace:hzar>
func:
function (center, width, pMin, pMax)
return(function(x) pMin + (pMax - pMin) * (1/(1 + exp(-((x -
    center) * 4/width)))))
Cline Parameters:
          init tune fixed lower upper
center 198.500  1.5 FALSE   -30   600
width   64.750  1.5 FALSE     0   630
pMin     0.100  1.1 FALSE     0     1
pMax     0.675  1.1 FALSE     0     1

$modelIII
req:
function (center, width, pMin, pMax, deltaL, tauL, deltaR, tauR)
return(width > 0 & deltaL >= 0 & deltaR >= 0 & (pMax - pMin) *
    (deltaL + deltaR) <= width * 50 & pMin >= 0 & pMax <= 1 &
    pMin < pMax & tauL >= 0 & tauL <= 1 & tauR >= 0 & width <
    630 & deltaL < 630 & deltaR < 630 & center > -30 & center <
    600 & tauR <= 1)
<environment: namespace:hzar>
prior:
function (center, width, pMin, pMax, deltaL, tauL, deltaR, tauR)
{
    return(0)
}
<environment: namespace:hzar>
pExp:
pMin + (pMax - pMin) * ifelse(deltaL < -(x - center), 1/(1 +
    exp(4 * deltaL/width)) * exp(tauL/(1 + exp(-4 * deltaL/width)) *
    ((x - center) * 4/width + 4 * deltaL/width)), ifelse(deltaR <
    x - center, 1 - 1/(1 + exp(4 * deltaR/width)) * exp(-(tauR/(1 +
    exp(-4 * deltaR/width))) * ((x - center) * 4/width - 4 *
    deltaR/width)), 1/(1 + exp(-((x - center) * 4/width)))))
func:
function (center, width, pMin, pMax, deltaL, tauL, deltaR, tauR)
return(function(x) pMin + (pMax - pMin) * ifelse(deltaL < -(x -
    center), 1/(1 + exp(4 * deltaL/width)) * exp(tauL/(1 + exp(-4 *
    deltaL/width)) * ((x - center) * 4/width + 4 * deltaL/width)),
    ifelse(deltaR < x - center, 1 - 1/(1 + exp(4 * deltaR/width)) *
        exp(-(tauR/(1 + exp(-4 * deltaR/width))) * ((x - center) *
            4/width - 4 * deltaR/width)), 1/(1 + exp(-((x - center) *
        4/width))))))
Cline Parameters:
          init tune fixed lower upper
center 198.500  1.5 FALSE   -30   600
width   64.750  1.5 FALSE     0   630
pMin     0.100  1.1 FALSE     0     1
pMax     0.675  1.1 FALSE     0     1
deltaL   1.000  1.5 FALSE     0   630
tauL     0.500  1.1 FALSE     0     1
deltaR   1.000  1.5 FALSE     0   630
tauR     0.500  1.1 FALSE     0     1
```

```
>
> ## Compile each of the models to prepare for fitting
> mkn$AdaA$fitRs$init <- sapply(mkn$AdaA$models,
+                           hzar.first.fitRequest.old.ML,
+                           obsData=mkn$AdaA$obs,
+                           verbose=FALSE,
+                           simplify=FALSE)
0ac 4.0e+01dfg0ac 4.0e+01dfg0ac 1.6e+07dfg> ## Update the settings for the fitter
if desired.
> mkn$AdaA$fitRs$init$modelI$mcmcParam$chainLength <-
+   chainLength;                          #1e5
> mkn$AdaA$fitRs$init$modelI$mcmcParam$burnin <-
+   chainLength %/% 10;                   #1e4
> mkn$AdaA$fitRs$init$modelI$mcmcParam$seed[[1]] <-
+   mainSeed$A
>
> mkn$AdaA$fitRs$init$modelII$mcmcParam$chainLength <-
+   chainLength;                          #1e5
> mkn$AdaA$fitRs$init$modelII$mcmcParam$burnin <-
+   chainLength %/% 10;                   #1e4
> mkn$AdaA$fitRs$init$modelII$mcmcParam$seed[[1]] <-
+   mainSeed$B
>
> mkn$AdaA$fitRs$init$modelIII$mcmcParam$chainLength <-
+   chainLength;                          #1e5
> mkn$AdaA$fitRs$init$modelIII$mcmcParam$burnin <-
+   chainLength %/% 10;                   #1e4
> mkn$AdaA$fitRs$init$modelIII$mcmcParam$seed[[1]] <-
+   mainSeed$C
>
> ## Check fit request settings
> print(mkn$AdaA$fitRs$init)
$modelI
Fit request object:

Model Parameters:
         init tune lower upper
center 198.50  1.5   -30   600
width   64.75  1.5     0   630

Fixed Parameters:
pMin = 0.1
pMax = 0.675

MCMC properties:
chainLength = 1e+05
     burnin = 10000
  verbosity = 0
       thin = 100
  seed base = [ 596, 528, 124, 978, 544, 99 ]
    channel = 1

LL expression hidden
```

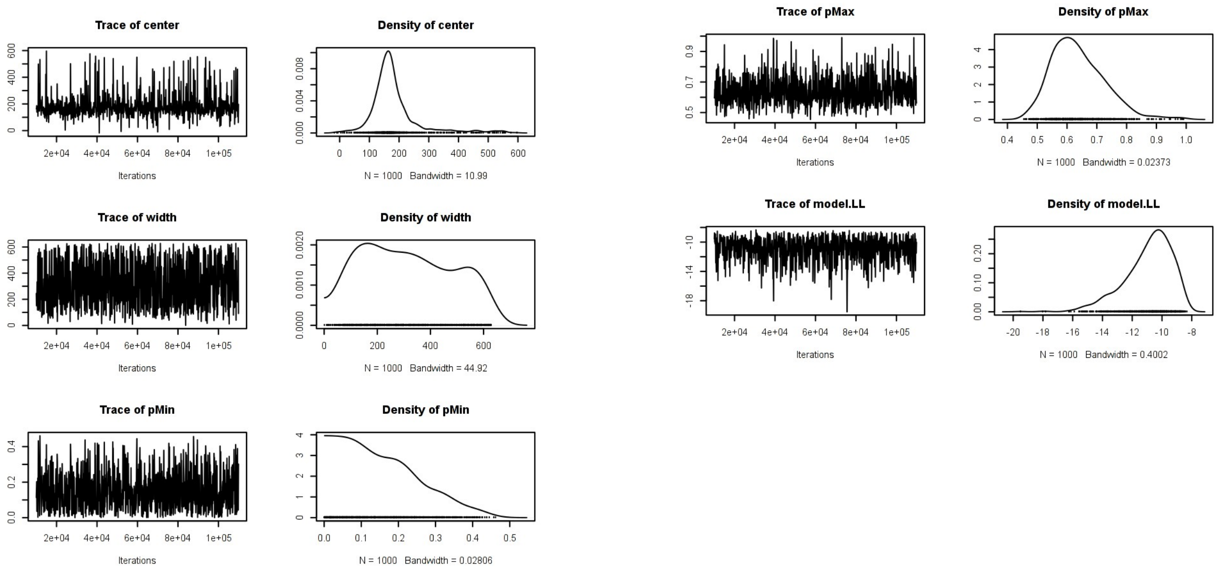
```
Covariance Matrix:
           center     width
center   491.8517 -1043.986
width  -1043.9862 19087.371

$modelII
Fit request object:

Model Parameters:
          init tune lower upper
center 198.500  1.5   -30   600
width   64.750  1.5     0   630
pMin     0.100  1.1     0     1
pMax     0.675  1.1     0     1

Fixed Parameters:
 =

MCMC properties:
chainLength = 1e+05
     burnin = 10000
  verbosity = 0
       thin = 100
  seed base = [ 528, 124, 978, 544, 99, 596 ]
    channel = 1

LL expression hidden

Covariance Matrix:
            center        width        pMin        pMax
center 5687.873199 2.613375e+03 5.504141418 3.404929680
width  2613.374982 2.958405e+04 0.311656513 8.087901735
pMin      5.504141 3.116565e-01 0.010328590 0.001219582
pMax      3.404930 8.087902e+00 0.001219582 0.005925997

$modelIII
Fit request object:

Model Parameters:
          init tune lower upper
center 198.500  1.5   -30   600
width   64.750  1.5     0   630
pMin     0.100  1.1     0     1
pMax     0.675  1.1     0     1
deltaL   1.000  1.5     0   630
tauL     0.500  1.1     0     1
deltaR   1.000  1.5     0   630
tauR     0.500  1.1     0     1

Fixed Parameters:
 =

MCMC properties:
chainLength = 1e+05
     burnin = 10000
```

```
  verbosity = 0
       thin = 100
  seed base = [ 124, 978, 544, 99, 596, 528 ]
    channel = 1

LL expression hidden

Covariance Matrix Diagonal:
      center        width         pMin         pMax       deltaL         tauL
5.695115e+03 2.884611e+04 1.001460e-02 1.757526e-02 3.259792e+04 1.074749e-01
      deltaR         tauR
5.003903e+04 9.535615e-02

>
>
>
>
> ## Run just one of the models for an initial chain
> mkn$AdaA$runs$init <- list()
> mkn$AdaA$runs$init$modelI <-
+   hzar.doFit(mkn$AdaA$fitRs$init$modelI)


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.35751
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
>
> ## Plot the trace
> plot(hzar.mcmc.bindLL(mkn$AdaA$runs$init$modelI))
```

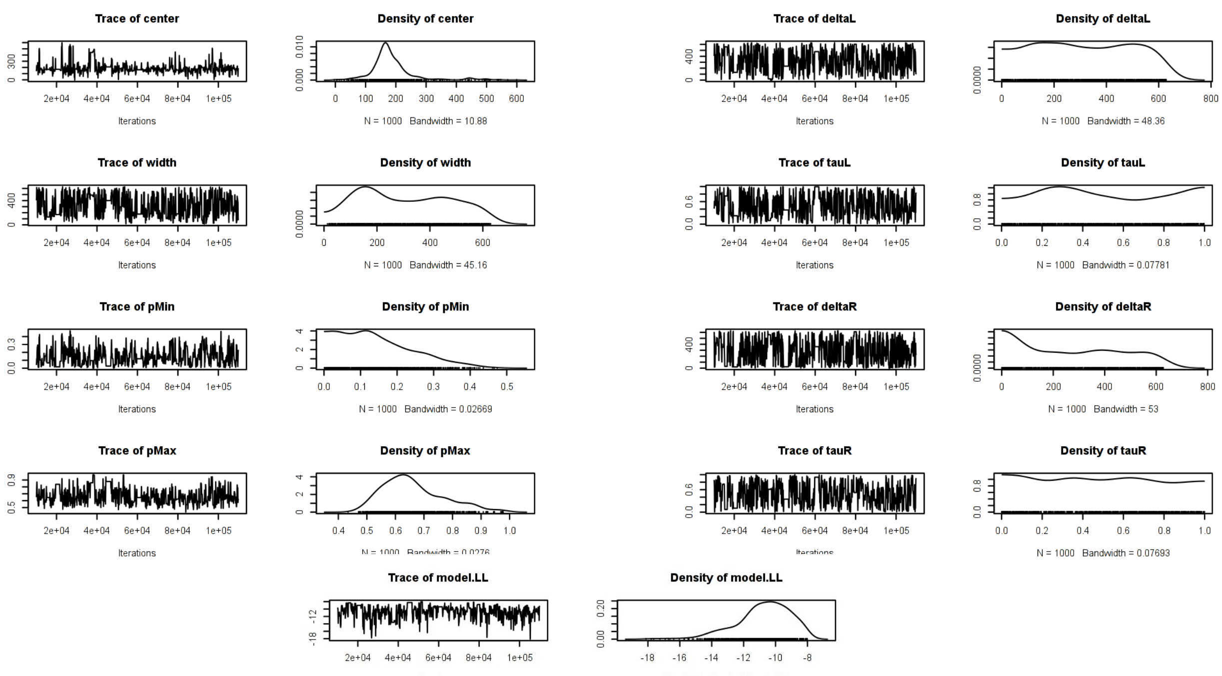



```
>
>
> ## Run another model for an initial chain
> mkn$AdaA$runs$init$modelII <-
+   hzar.doFit(mkn$AdaA$fitRs$init$modelII)

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.12500
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
>
```

```
> ## Plot the trace
> plot(hzar.mcmc.bindLL(mkn$AdaA$runs$init$modelII))
```




```
>
>
> ## Run another model for an initial chain
> mkn$AdaA$runs$init$modelIII <-
+   hzar.doFit(mkn$AdaA$fitRs$init$modelIII)


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.01811
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
>
> ## Plot the trace
> plot(hzar.mcmc.bindLL(mkn$AdaA$runs$init$modelIII))
```



```
>
>
> ## Compile a new set of fit requests using the initial chains
> mkn$AdaA$fitRs$chains <-
+   lapply(mkn$AdaA$runs$init,
+          hzar.next.fitRequest)
0ac 3.5e+00dfg0ac 1.3e+06dfg>
> ## Replicate each fit request 3 times, keeping the original
> ## seeds while switching to a new seed channel.
> mkn$AdaA$fitRs$chains <-
+   hzar.multiFitRequest(mkn$AdaA$fitRs$chains,
+                        each=3,
+                        baseSeed=NULL)
>
> ## Just to be thorough, randomize the initial value for each fit
>
> ## runif(9,-30,600) center for modelI, modelII, modelIII
> mkn$AdaA$fitRs$chains[[1]]$modelParam$init["center"]= 120.08256
> mkn$AdaA$fitRs$chains[[2]]$modelParam$init["center"]= 345.55072
> mkn$AdaA$fitRs$chains[[3]]$modelParam$init["center"]= 158.34218
> mkn$AdaA$fitRs$chains[[4]]$modelParam$init["center"]= 208.49748
> mkn$AdaA$fitRs$chains[[5]]$modelParam$init["center"]= 89.08333
> mkn$AdaA$fitRs$chains[[6]]$modelParam$init["center"]= 286.89100
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["center"]= 529.46032
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["center"]= 67.07035
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["center"]= 513.06504
>
> ## runif(9,0,630) width for modelI, modelII, modelIII
> mkn$AdaA$fitRs$chains[[1]]$modelParam$init["width"]= 283.09967
> mkn$AdaA$fitRs$chains[[2]]$modelParam$init["width"]= 436.41024
> mkn$AdaA$fitRs$chains[[3]]$modelParam$init["width"]= 84.82553
> mkn$AdaA$fitRs$chains[[4]]$modelParam$init["width"]= 385.90734
> mkn$AdaA$fitRs$chains[[5]]$modelParam$init["width"]= 279.79723
> mkn$AdaA$fitRs$chains[[6]]$modelParam$init["width"]= 38.92028
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["width"]= 42.87985
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["width"]=  98.21601
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["width"]= 230.45149
>
> ## runif(6,0,1) pMin for modelII, modelIII
> mkn$AdaA$fitRs$chains[[4]]$modelParam$init["pMin"]= 0.6778914
> mkn$AdaA$fitRs$chains[[5]]$modelParam$init["pMin"]= 0.4866557
> mkn$AdaA$fitRs$chains[[6]]$modelParam$init["pMin"]=  0.4565415
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["pMin"]= 0.8985962
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["pMin"]= 0.3901906
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["pMin"]= 0.1531809
>
> ## runif(6,0,1) pMax for modelII, modelIII
> mkn$AdaA$fitRs$chains[[4]]$modelParam$init["pMax"]= 0.4567925
> mkn$AdaA$fitRs$chains[[5]]$modelParam$init["pMax"]= 0.7801083
> mkn$AdaA$fitRs$chains[[6]]$modelParam$init["pMax"]= 0.3347732
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["pMax"]=  0.8740998
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["pMax"]=  0.4463791
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["pMax"]=  0.1640979
>
> ## runif(3,0,630) deltaL for modelIII
```

```
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["deltaL"]= 290.2459
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["deltaL"]= 242.7785
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["deltaL"]= 351.2703
>
> ## runif(3,0,1) tauL for modelIII
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["tauL"]= 0.3205238
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["tauL"]= 0.9736836
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["tauL"]= 0.6674259
>
> ## runif(3,0,630) deltaR for modelIII
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["deltaR"]=472.6632
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["deltaR"]=390.6439
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["deltaR"]=545.4608
>
> ## runif(3,0,1) tauR for modelIII
> mkn$AdaA$fitRs$chains[[7]]$modelParam$init["tauR"]=0.9146836
> mkn$AdaA$fitRs$chains[[8]]$modelParam$init["tauR"]=0.2064311
> mkn$AdaA$fitRs$chains[[9]]$modelParam$init["tauR"]= 0.2435238
>
> ## Go ahead and run a chain of 3 runs for every fit request
> mkn$AdaA$runs$chains <-  hzar.doChain.multi(mkn$AdaA$fitRs$chains,
+                                             doPar=TRUE,
+                                             inOrder=FALSE,
+                                             count=3)
[1] 1
[1] 1
[1] 1
[1] 1


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.37530
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] 2


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.37475
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] 2
           Length Class  Mode
modelParam 5      -none- list
cM         4      -none- numeric
```

```
llFunc     1      -none- function
mcmcParam  5      -none- list
mcmcRaw    0      -none- NULL
           Length Class  Mode
modelParam 5      -none- list
cM         4      -none- numeric
llFunc     1      -none- function
mcmcParam  5      -none- list
mcmcRaw    0      -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.37699
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] 2
           Length Class  Mode
modelParam 5      -none- list
cM         4      -none- numeric
llFunc     1      -none- function
mcmcParam  5      -none- list
mcmcRaw    0      -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.15365
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] 2
0ac 2.9e+00dfg           Length Class  Mode
modelParam  5     -none- list
cM         16     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.38121
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
```

```
mcmcRaw    2000   mcmc   numeric
[1] 3


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.37385
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] 3
           Length Class  Mode
modelParam 5      -none- list
cM         4      -none- numeric
llFunc     1      -none- function
mcmcParam  5      -none- list
mcmcRaw    0      -none- NULL
           Length Class  Mode
modelParam 5      -none- list
cM         4      -none- numeric
llFunc     1      -none- function
mcmcParam  5      -none- list
mcmcRaw    0      -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.38132
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] 3
           Length Class  Mode
modelParam 5      -none- list
cM         4      -none- numeric
llFunc     1      -none- function
mcmcParam  5      -none- list
mcmcRaw    0      -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.16031
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
```

```
[1] 3
0ac 2.8e+00

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.37911
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] "Chain Complete"
[1] 1


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.36229
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] "Chain Complete"
[1] 1


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.38418
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM            4   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    2000   mcmc   numeric
[1] "Chain Complete"
[1] 1
dfg           Length Class  Mode
modelParam  5     -none- list
cM         16     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.15223
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
```

```
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] 2
0ac 2.4e+00

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.14932
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] 2
0ac 2.8e+00

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.15753
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] "Chain Complete"
[1] 1
dfg           Length Class  Mode
modelParam  5     -none- list
cM         16     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL
dfg           Length Class  Mode
modelParam  5     -none- list
cM         16     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.16710
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] 3
0ac 3.4e+00

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
The Metropolis acceptance rate was 0.16297
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] 3
0ac 2.6e+00dfg           Length Class  Mode
modelParam  5     -none- list
cM         16     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL
dfg           Length Class  Mode
modelParam  5     -none- list
cM         16     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03635
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] 2
0ac 9.0e+05dfg           Length Class  Mode
modelParam  5     -none- list
cM         64     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.16192
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] "Chain Complete"
[1] 1


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
The Metropolis acceptance rate was 0.16899
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           16   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    4000   mcmc   numeric
[1] "Chain Complete"


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03231
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] 2
0ac 1.2e+06dfg           Length Class  Mode
modelParam  5     -none- list
cM         64     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03559
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] 3


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03563
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] 2
0ac 1.1e+060ac 1.2e+06dfg           Length Class  Mode
modelParam  5     -none- list
cM         64     -none- numeric
llFunc      1     -none- function
```

```
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL
dfg           Length Class  Mode
modelParam  5     -none- list
cM         64     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03895
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] 3
0ac 1.3e+06dfg           Length Class  Mode
modelParam  5     -none- list
cM         64     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03711
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] 3
0ac 9.1e+05

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03575
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] "Chain Complete"
dfg           Length Class  Mode
modelParam  5     -none- list
cM         64     -none- numeric
llFunc      1     -none- function
mcmcParam   5     -none- list
```

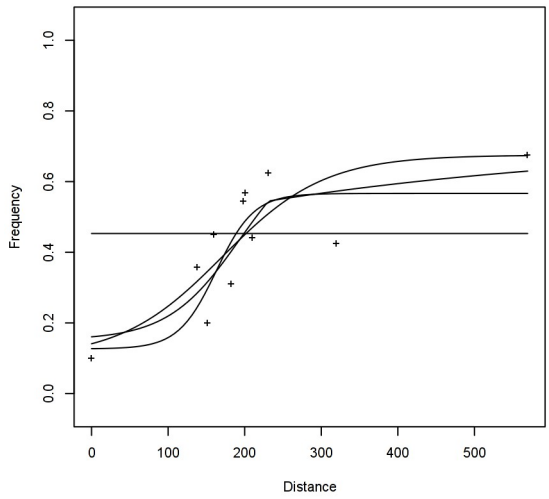
```
mcmcRaw     0     -none- NULL


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03550
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] "Chain Complete"


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.03742
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
           Length Class  Mode
modelParam    5   -none- list
cM           64   -none- numeric
llFunc        1   -none- function
mcmcParam     5   -none- list
mcmcRaw    8000   mcmc   numeric
[1] "Chain Complete"
>
> ## Did modelI converge?
> summary(do.call(mcmc.list,
+                 lapply(mkn$AdaA$runs$chains[1:3],
+                        function(x) hzar.mcmc.bindLL(x[[3]]) )) )

Iterations = 10001:109901
Thinning interval = 100
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean       SD Naive SE Time-series SE
center   163.34  21.7872  0.39778        0.42263
width    388.75 131.6276  2.40318        2.67828
model.LL -10.52   0.8205  0.01498        0.01751

2. Quantiles for each variable:

           2.5%   25%    50%     75%   97.5%
center   114.64 150.9 165.88 177.517 201.370
width    149.42 285.8 388.58 497.153 613.764
model.LL -12.75 -10.8 -10.33  -9.947  -9.675

>
> ## Yes it did.
>
> ## Did modelII converge?
```

```
> summary(do.call(mcmc.list,
+                 lapply(mkn$AdaA$runs$chains[4:6],
+                        function(x) hzar.mcmc.bindLL(x[[3]]) )) )

Iterations = 10001:109901
Thinning interval = 100
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean        SD Naive SE Time-series SE
center   182.5278  81.26891 1.483760       1.692733
width    321.6701 169.11635 3.087628       3.288645
pMin       0.1553   0.10407 0.001900       0.002026
pMax       0.6371   0.08606 0.001571       0.001816
model.LL -10.9710   1.58416 0.028923       0.030669

2. Quantiles for each variable:

               2.5%       25%      50%      75%    97.5%
center    68.241395 142.33162 167.2562 197.0288 443.0284
width     49.298202 175.67373 314.7566 462.9844 615.5000
pMin       0.008185   0.07049   0.1391   0.2231   0.3906
pMax       0.502902   0.57812   0.6249   0.6829   0.8436
model.LL -14.760536 -11.82871 -10.6966  -9.8053  -8.6388

>
> ## Yes it did.
>
> ## Did modelIII converge?
> summary(do.call(mcmc.list,
+                 lapply(mkn$AdaA$runs$chains[7:9],
+                        function(x) hzar.mcmc.bindLL(x[[3]]) )) )

Iterations = 10001:109901
Thinning interval = 100
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean        SD Naive SE Time-series SE
center   182.8795  70.02232 1.278427       2.617362
width    309.0127 166.61393 3.041940       4.188703
pMin       0.1467   0.10118 0.001847       0.003175
pMax       0.6519   0.09667 0.001765       0.003207
deltaL   326.1814 176.98222 3.231238       4.739738
tauL       0.5026   0.28499 0.005203       0.007675
deltaR   295.5249 190.41689 3.476521       4.866734
tauR       0.5016   0.28807 0.005259       0.007353
model.LL -10.7675   1.64926 0.030111       0.059465
```

```
2. Quantiles for each variable:

               2.5%      25%      50%      75%    97.5%
center    83.577579 147.5827 169.4396 198.5397 399.3444
width     50.527392 165.8643 288.7968 445.3457 609.0797
pMin       0.006851   0.0645   0.1290   0.2104   0.3807
pMax       0.502072   0.5802   0.6364   0.7103   0.8746
deltaL    27.384655 171.0231 328.6941 482.7162 617.6769
tauL       0.027661   0.2613   0.5103   0.7436   0.9783
deltaR    11.863388 115.9556 295.7309 462.6814 614.9925
tauR       0.024665   0.2556   0.5032   0.7520   0.9752
model.LL -14.665994 -11.7018 -10.5255  -9.5374  -8.4502

> 
> ## Yes it did.
> 
> ## All three models have three convergent chains, so additional runs
> ## are not needed.
> 
> ## Just to show how, an additonal run is requested for each chain of
> ## modelI.  These three runs will not be included in the analysis
> ## stage.
> 
> mkn$AdaA$fitRs$extra.modelI <-
+   lapply(lapply(mkn$AdaA$runs$chains[1:3],  # Work from modelI chains
+                 function(x) x[[3]]),        # Use third run from chain
+          hzar.next.fitRequest)
> 
> 
> ## Only do a single run.
> mkn$AdaA$runs$extra.modelI <-
+   hzar.doFit.multi(mkn$AdaA$fitRs$extra.modelI,
+                    doPar=TRUE,
+                    inOrder=FALSE)


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.37821
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.37472
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The Metropolis acceptance rate was 0.38117
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
> 
> ## Double check, are these three runs also convergent?
> summary(do.call(mcmc.list,
+                 lapply(mkn$AdaA$runs$extra.modelI,
+                        hzar.mcmc.bindLL )))
```

```
Iterations = 10001:109901
Thinning interval = 100
Number of chains = 3
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean       SD Naive SE Time-series SE
center   164.28  22.1253  0.40395        0.41875
width    387.84 133.6527  2.44015        2.29871
model.LL -10.52   0.8046  0.01469        0.01394

2. Quantiles for each variable:

           2.5%    25%    50%    75%   97.5%
center   114.66 151.48 166.63 178.79 202.344
width    144.45 283.44 387.57 497.51 614.282
model.LL -12.61 -10.81 -10.32  -9.95  -9.673

>
> ## Yes.
>
> ## Start aggregation of data for analysis
>
> ## Create a model data group for the null model (expected allele
> ## frequency independent of distance along cline) to include in
> ## analysis.
> mkn$AdaA$analysis$initDGs <- list(
+   nullModel =  hzar.dataGroup.null(mkn$AdaA$obs))
>
> ## Create a model data group (hzar.dataGroup object) for each
> ## model from the initial runs.
> mkn$AdaA$analysis$initDGs$modelI <-
+   hzar.dataGroup.add(mkn$AdaA$runs$init$modelI)
> mkn$AdaA$analysis$initDGs$modelII <-
+   hzar.dataGroup.add(mkn$AdaA$runs$init$modelII)
> mkn$AdaA$analysis$initDGs$modelIII <-
+   hzar.dataGroup.add(mkn$AdaA$runs$init$modelIII)
>
> ## Create a hzar.obsDataGroup object from the four hzar.dataGroup
> ## just created, copying the naming scheme (nullModel, modelI,
> ## modelII, modelIII).
> mkn$AdaA$analysis$oDG <-
+   hzar.make.obsDataGroup(mkn$AdaA$analysis$initDGs)
[1] 4
> mkn$AdaA$analysis$oDG <-
+     hzar.copyModelLabels(mkn$AdaA$analysis$initDGs,
+                          mkn$AdaA$analysis$oDG)
>
> ## Convert all 27 runs to hzar.dataGroup objects, adding them to
> ## the hzar.obsDataGroup object.
> mkn$AdaA$analysis$oDG <-
+   hzar.make.obsDataGroup(lapply(mkn$AdaA$runs$chains,
+                                 hzar.dataGroup.add),
```

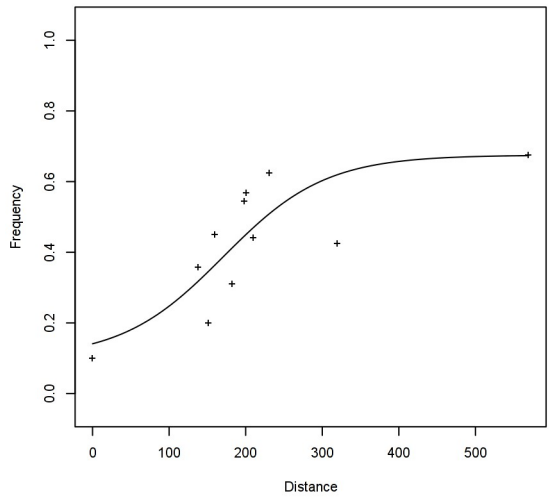
```
+                          mkn$AdaA$analysis$oDG);
>
> ## Check to make sure that there are only four hzar.dataGroup
> ## objects named nullModel, modelI, modelII, and modelIII in the
> ## hzar.obsDataGroup object.
> print(summary(mkn$AdaA$analysis$oDG$data.groups))
          Length Class          Mode
nullModel 6      hzar.dataGroup list
modelI    6      hzar.dataGroup list
modelII   6      hzar.dataGroup list
modelIII  6      hzar.dataGroup list
>
> ## Compare the 3 cline models to the null model graphically
> hzar.plot.cline(mkn$AdaA$analysis$oDG);
$nullModel
NULL

$modelI
NULL

$modelII
NULL

$modelIII
NULL
```




```
>
> ## Do model selection based on the AICc scores
> print(mkn$AdaA$analysis$AICcTable <-
+       hzar.AICc.hzar.obsDataGroup(mkn$AdaA$analysis$oDG));
              AICc
nullModel 39.31872
modelI    23.32204
modelII   24.65754
modelIII  31.69649
>
> ## Print out the model with the minimum AICc score
> print(mkn$AdaA$analysis$model.name <-
+   rownames(mkn$AdaA$analysis$AICcTable
+            )[[ which.min(mkn$AdaA$analysis$AICcTable$AICc )]])
[1] "modelI"
>
> ## Extract the hzar.dataGroup object for the selected model
> mkn$AdaA$analysis$model.selected <-
+   mkn$AdaA$analysis$oDG$data.groups[[mkn$AdaA$analysis$model.name]]
>
>
> ## Look at the variation in parameters for the selected model
> print(hzar.getLLCutParam(mkn$AdaA$analysis$model.selected,
+                          names(mkn$AdaA$analysis$model.selected$data.param)));
```

```
  center2LLLow center2LLHigh width2LLLow width2LLHigh
1     110.3029      201.2508    97.67553     629.8941
>
> ## Print the maximum likelihood cline for the selected model
> print(hzar.get.ML.cline(mkn$AdaA$analysis$model.selected))
$param.free
      center    width
360 170.8367 266.2181

$param.all
$param.all$center
[1] 170.8367

$param.all$width
[1] 266.2181

$param.all$pMin
[1] 0.1

$param.all$pMax
[1] 0.675


$clineFunc
function (x)
pMin + (pMax - pMin) * (1/(1 + exp(-((x - center) * 4/width))))
<environment: 0x3b58fb0>

$logLike
[1] -9.645271

$isValid
[1] TRUE

attr(,"class")
[1] "hzar.cline"
>
```

```
> ## Plot the maximum likelihood cline for the selected model
> hzar.plot.cline(mkn$AdaA$analysis$model.selected);
```




```
>
> ## Plot the 95% credible cline region for the selected model
> hzar.plot.fzCline(mkn$AdaA$analysis$model.selected);
```

```
>
> ## End Molecular Analysis
>
> dev.off()
null device
          1
>
```