Fig.5

Devon DeRaad

3/18/2019

Install hzar

#install.packages("hzar")  
## Load the package  
library(hzar)

Set a few parameters

## A typical chain length  
chainLength=1e5  
## Increment skip by 15 or so for every trait  
rotateModelSeeds <- function(fiComp.1)  
 hzar.multiFitRequest(fiComp.1,  
 rotateSeed=TRUE, skip=500,  
 baseChannel=NULL, each=1,  
 baseSeed=c(596,528,124,978,544,99))

Read in the SiteLoc file, which should have two columns, one for the name of each locality (site) you chose, and a second with the distance of that site from the beginning of the cline.

#bring in data  
SiteLoc<-read.csv("~/Dropbox/scrub poster/manuscript/hzar.scrubbers/siteloc.csv")  
head(SiteLoc)

## Site Distance  
## 1 A 0  
## 2 B 100  
## 3 C 300  
## 4 D 430  
## 5 E 640  
## 6 F 760

Next we will read in the three data files, each with a separate variable we are hoping to model. The columns correspond to variables and the rows correspond to individuals. Site, Distance, and the variable of interest must be columns.

#body size for males  
malemorph<-read.csv("~/Dropbox/scrub poster/manuscript/hzar.scrubbers/malemorph.csv")  
head(malemorph)

## id Site Distance sex Comp.1  
## 1 18334 F 760 male -2.709351277  
## 2 23362 D 430 male -0.322080214  
## 3 27905 D 430 male -0.744350334  
## 4 27906 D 430 male -0.422455322  
## 5 27907 D 430 male 0.001468766  
## 6 27908 D 430 male 1.673770502

#body size for females  
femalemorph<-read.csv("~/Dropbox/scrub poster/manuscript/hzar.scrubbers/femalemorph.csv")  
head(femalemorph)

## id Site Distance sex Comp.1  
## 1 23358 D 430 female 2.544649  
## 2 23360 D 430 female 2.453736  
## 3 27065 C 300 female 1.575535  
## 4 27920 D 430 female 2.949411  
## 5 30749 D 430 female 1.925544  
## 6 32276 D 430 female 1.510762

#blue plumage of the back for all individuals  
S1blue<-read.csv("~/Dropbox/scrub poster/manuscript/hzar.scrubbers/s1.blue.csv")  
head(S1blue)

## id Site Distance S1.blue  
## 1 144750 A 0 0.3275797  
## 2 144751 A 0 0.3105317  
## 3 144752 A 0 0.3004815  
## 4 144749 A 0 0.2811858  
## 5 50598 A 0 0.3148386  
## 6 57751 B 100 0.2887933

These all look good

Open a pdf that will eventually save the final plot. I have commented this out here, but if you uncomment it and the ‘dev.off()’ after the plot is made, it will save a PDF for you.

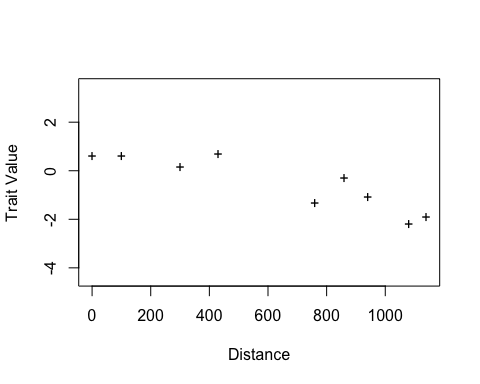
## ## Save all plots in a pdf file?  
#pdf(width=8, height=8, file="male.comp1.pdf")

Open lists

## Blank out space in memory to hold morphological analysis  
if(length(apropos("^mkn$",ignore.case=FALSE)) == 0 ||  
 !is.list(mkn) ) mkn <- list()  
## We are doing just the one quantitative trait, but it is  
## good to stay organized.  
mkn$Comp.1 <- list();  
## Space to hold the observed data  
mkn$Comp.1$obs <- list();  
## Space to hold the models to fit  
mkn$Comp.1$models <- list();  
## Space to hold the compiled fit requests  
mkn$Comp.1$fitRs <- list();  
## Space to hold the output data chains  
mkn$Comp.1$runs <- list();  
## Space to hold the analysed data  
mkn$Comp.1$analysis <- list()

This is where we set the character of interest as the focal character for the rest of the analysis.

## Beard Length Trait from Brumfield et al 2001  
mkn$Comp.1$obs <-  
 hzar.doNormalData1DRaw(hzar.mapSiteDist(SiteLoc$Site,  
 SiteLoc$Distance),  
 malemorph$Site,  
 malemorph$Comp.1)  
  
## Look at a graph of the observed data  
hzar.plot.obsData(mkn$Comp.1$obs)



## Make a helper function  
mkn.loadComp.1model <- function(scaling,tails,  
 id=paste(scaling,tails,sep=".")){  
 mkn$Comp.1$models[[id]] <<-  
 hzar.makeCline1DNormal(mkn$Comp.1$obs, tails)  
 ## As there is no quick option for "fixed" scaling, and the  
 ## combined sites "A", "B" and "J","K" have a fair number of samples (> 20),  
 ## fix the mean and variance of the left and right sides of  
 ## the cline to the values observed by the combined sites.  
 if (all(regexpr("fixed",scaling,ignore.case=TRUE) == 1 )){  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$muL <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$muR <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$varL <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$varR <<- TRUE  
 }  
 ## Helper function to work around low sample size  
 getCombo <- function(id1,id2,colName,frame=mkn$Comp.1$obs$frame)  
 (frame[id1,colName]\*frame[id1,"nEff"]+  
 frame[id2,colName]\*frame[id2,"nEff"])/  
 (frame[id1,"nEff"]+frame[id2,"nEff"])  
   
 ## Site A, B is the "left" side of the cline, so pull the  
 ## fixed values from there.  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$muL <<-  
 getCombo("A","B","mu")  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varL <<-  
 getCombo("A","B","var")  
 ## Site J, K is the "right" side of the cline, so pull the  
 ## fixed values from there.  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$muR <<-  
 getCombo("I","J","mu")  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varR <<-  
 getCombo("I","J","var")  
 ## Make a better estimate of varH using site D, E  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varH <<-  
 getCombo("E","F","var")-  
 (getCombo("I","J","var")+getCombo("A","B","var"))/2  
   
}  
## mkn.loadComp.1model("fixed","none","modelI");  
## mkn.loadComp.1model("free" ,"none","modelII");  
## mkn.loadComp.1model("free" ,"both","modelIII");  
mkn.loadComp.1model("fixed","none" ,"fixN");  
mkn.loadComp.1model("fixed","left" ,"fixL");  
mkn.loadComp.1model("fixed","right" ,"fixR");  
mkn.loadComp.1model("fixed","mirror","fixM");  
mkn.loadComp.1model("fixed","both" ,"fixB");  
mkn.loadComp.1model("free" ,"none" ,"optN");  
mkn.loadComp.1model("free" ,"left" ,"opComp.1");  
mkn.loadComp.1model("free" ,"right" ,"optR");  
mkn.loadComp.1model("free" ,"mirror","optM");  
mkn.loadComp.1model("free" ,"both" ,"optB");  
  
  
## Check the default settings  
##print(mkn$Comp.1$models)  
  
## Modify all models to focus on the region where the observed  
## data were collected.  
## Observations were between -10 and 130000 km.  
mkn$Comp.1$models <- sapply(mkn$Comp.1$models,  
 hzar.model.addBoxReq,  
 -10 , 1100,  
 simplify=FALSE)  
  
## Due to the large number of free variables, it is prudent to  
## reduce the tune setting of optB from 1.5 to 1.1  
hzar:::meta.tune(mkn$Comp.1$models$optB)<-1.1  
  
  
## Check the updated settings  
##print(mkn$Comp.1$models)  
  
## Compile each of the models to prepare for fitting  
## Note that we are using hzar.first.fitRequest.gC for fitting  
## guassian (aka "normal") clines.  
mkn$Comp.1$fitRs$init <- sapply(mkn$Comp.1$models,  
 hzar.first.fitRequest.gC,  
 obsData=mkn$Comp.1$obs,  
 verbose=TRUE,  
 simplify=FALSE)

## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauR  
## [1] 0.5  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 487.5  
##   
## $tauM  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 487.5  
##   
## $tauM  
## [1] 0.5  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauR  
## [1] 0.5  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 487.5  
##   
## $tauM  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 487.5  
##   
## $tauM  
## [1] 0.5  
##   
##   
## a$center  
## [1] 95  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 100  
##   
## $width  
## [1] 650  
##   
## $muL  
## [1] 0.6188497  
##   
## $muR  
## [1] -2.080915  
##   
## $varL  
## [1] 3.381186  
##   
## $varR  
## [1] 1.605924  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 487.5  
##   
## $deltaR  
## [1] 487.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5

## Update the settings for the fitter if desired.  
mkn$Comp.1$fitRs$init <- sapply(mkn$Comp.1$fitRs$init,  
 function(mdl) {  
 mdl$mcmcParam$chainLength <-  
 chainLength; #1e5  
 mdl$mcmcParam$burnin <-  
 chainLength %/% 10; #1e4  
 mdl },  
 simplify=FALSE)  
  
mkn$Comp.1$fitRs$init <-  
 rotateModelSeeds(mkn$Comp.1$fitRs$init)  
  
  
## Check fit request settings  
##print(mkn$Comp.1$fitRs$init)  
  
## Do just one run of the models for an initial chain  
mkn$Comp.1$runs$init <-  
 hzar.doFit.multi(mkn$Comp.1$fitRs$init)

## MCMCmetrop1R iteration 1 of 110000   
## function value = -139.44284  
## theta =   
## -9.99999  
## 830.10616  
## 25.43004  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.05071  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -128.53829  
## theta =   
## 1100.00000  
## 569.72138  
## 4.72696  
## 169.20806  
## 0.81688  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.01614  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -152.44515  
## theta =   
## 267.68821  
## 124.75131  
## 120.36510  
## 36.61011  
## 0.84614  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.08431  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -150.68276  
## theta =   
## 134.20500  
## 257.82719  
## 77.14874  
## 807.67523  
## 0.28079  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.01028  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -140.20258  
## theta =   
## 207.63695  
## 64.58821  
## 699.69840  
## 149.69282  
## 110.35546  
## 0.30329  
## 0.04652  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00277  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -152.50171  
## theta =   
## 368.78299  
## 823.83931  
## 2.76429  
## -4.13024  
## 0.14860  
## 18.95915  
## 0.14496  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00126  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -415.63319  
## theta =   
## 781.97923  
## 155.97887  
## -2.97329  
## -4.20225  
## 0.05167  
## 5538090.76082  
## 0.28666  
## 347.26092  
## 0.57975  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00021  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -180.53415  
## theta =   
## -9.99871  
## 444.05735  
## -2.83242  
## -4.26061  
## 0.38808  
## 21.44420  
## 14.45904  
## 917.70563  
## 0.24834  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00029  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -210.02493  
## theta =   
## 1100.00000  
## 769.24225  
## -2.99485  
## 0.39346  
## 0.00678  
## 642.43106  
## 1.87799  
## 1018.35391  
## 0.00807  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00032  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -236.59725  
## theta =   
## 1084.21374  
## 677.81466  
## -3.43624  
## -2.47532  
## 0.01918  
## 2728.53966  
## 0.00144  
## 784.17209  
## 0.00000  
## 0.87190  
## 0.00000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00014  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

names(mkn$Comp.1$runs$init) <- names(mkn$Comp.1$fitRs$init)  
  
  
  
## Compile a new set of fit requests using the initial chains   
mkn$Comp.1$fitRs$chains <-  
 lapply(mkn$Comp.1$runs$init,  
 hzar.next.fitRequest)

## 0ac 3.9e+04dfg0ac 3.4e+05dfg0ac 2.4e+08dfg0ac 5.0e+07dfg0ac 9.3e+05dfg0ac 3.4e+05dfg0ac 1.7e-01dfg0ac 7.1e+09dfg0ac 4.1e+03dfg0ac 6.2e-06dfg

## Replicate each fit request 3 times, keeping the original  
## seeds while switching to a new seed channel.  
mkn$Comp.1$fitRs$chains <-  
 hzar.multiFitRequest(mkn$Comp.1$fitRs$chains,  
 each=3,  
 baseSeed=NULL)  
  
## Just to be thorough, randomize the initial value for each fit  
  
  
## Go ahead and run a chain of 3 runs for every fit request  
mkn$Comp.1$runs$chains <- hzar.doChain.multi(mkn$Comp.1$fitRs$chains,  
 doPar=TRUE,  
 inOrder=FALSE,  
 count=3)

## Warning: executing %dopar% sequentially: no parallel backend registered

## Did fixN converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$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 732.75 1.208e+02 2.205e+00 7.100e+00  
## width 30.02 3.273e+01 5.975e-01 3.491e+00  
## varH 382287.54 5.398e+05 9.855e+03 1.647e+05  
## model.LL -121.19 1.505e+00 2.747e-02 8.466e-02  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 561.2910 637.191 688.71 852.98 935.4  
## width 0.4821 7.735 23.18 40.87 163.9  
## varH 7.3163 35432.503 214300.41 387598.46 2038696.6  
## model.LL -124.6800 -121.370 -121.02 -120.30 -118.7

## Did optN converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$runs$chains[16:18],  
 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 612.2516 6.866e+01 1.254e+00 2.069e+00  
## width 24.4962 1.907e+01 3.482e-01 5.644e-01  
## muL 0.4415 2.566e-01 4.684e-03 7.124e-03  
## muR -1.5097 2.927e-01 5.343e-03 7.704e-03  
## varL 1.9594 5.831e-01 1.065e-02 1.796e-02  
## varR 2.9618 8.371e-01 1.528e-02 2.422e-02  
## varH 45605.1023 3.996e+04 7.295e+02 8.374e+03  
## model.LL -117.1525 1.663e+00 3.036e-02 5.571e-02  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 473.51372 565.5198 615.0415 660.4336 7.334e+02  
## width 0.78196 9.5319 20.7844 36.3869 6.127e+01  
## muL -0.06221 0.2748 0.4346 0.6115 9.472e-01  
## muR -2.10994 -1.7028 -1.5099 -1.3100 -9.464e-01  
## varL 1.13795 1.5503 1.8449 2.2704 3.398e+00  
## varR 1.63987 2.3855 2.8415 3.3917 4.963e+00  
## varH 749.40330 13891.9361 33869.2569 67765.2130 1.400e+05  
## model.LL -121.15888 -118.0047 -116.8282 -115.9889 -1.147e+02

## Did optB converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$runs$chains[28:30],  
 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 749.0297 139.0712 2.539081 5.741958  
## width 481.5082 345.8357 6.314067 17.786309  
## muL 0.5992 0.4142 0.007562 0.015901  
## muR -2.2184 0.8528 0.015570 0.043189  
## varL 1.8667 0.7198 0.013141 0.024132  
## varR 1.9564 1.1885 0.021699 0.056883  
## varH 7.6185 13.3541 0.243812 2.585674  
## deltaL 581.0122 310.1301 5.662175 9.513216  
## deltaR 511.4358 268.7864 4.907347 42.872045  
## tauL 0.5127 0.2871 0.005242 0.008713  
## tauR 0.5082 0.2882 0.005261 0.018504  
## model.LL -116.8789 1.8521 0.033814 0.073849  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 484.30682 662.5628 738.3885 832.9105 1051.5261  
## width 17.31942 164.8134 412.8781 804.4751 1078.4715  
## muL -0.03732 0.3258 0.5332 0.8092 1.6826  
## muR -4.41458 -2.5816 -1.9893 -1.6263 -1.1450  
## varL 0.53667 1.4159 1.7845 2.2526 3.5270  
## varR 0.12198 1.1011 1.8455 2.6712 4.4433  
## varH 0.14237 1.2619 2.5120 6.4491 58.7059  
## deltaL 48.20718 312.3861 584.0990 854.9171 1076.6671  
## deltaR 59.19079 303.9959 477.8662 708.0626 1037.8729  
## tauL 0.01845 0.2620 0.5222 0.7630 0.9750  
## tauR 0.02605 0.2650 0.5101 0.7640 0.9754  
## model.LL -121.36638 -117.8358 -116.5879 -115.6449 -114.0196

## Start aggregation of data for analysis  
  
## Clear out a spot to collect the data for analysis (note that  
## there is currenComp.1y no "null model" to compare against).  
mkn$Comp.1$analysis$initDGs <- list(  
)  
  
## Create a model data group (hzar.dataGroup object) for each  
## model from the initial runs.  
mkn$Comp.1$analysis$initDGs <-  
 c( mkn$Comp.1$analysis$initDGs,  
 sapply(mkn$Comp.1$runs$init,  
 hzar.dataGroup.add,  
 simplify=FALSE))  
  
## Create a hzar.obsDataGroup object from the four hzar.dataGroup  
## just created, copying the naming scheme (modelI, modelII,  
## modelIII).  
mkn$Comp.1$analysis$oDG <-  
 hzar.make.obsDataGroup(mkn$Comp.1$analysis$initDGs)

## [1] 10

mkn$Comp.1$analysis$oDG <-  
 hzar.copyModelLabels(mkn$Comp.1$analysis$initDGs,  
 mkn$Comp.1$analysis$oDG)  
  
## Convert all 90 runs to hzar.dataGroup objects, adding them to  
## the hzar.obsDataGroup object.  
mkn$Comp.1$analysis$oDG <-  
 hzar.make.obsDataGroup(lapply(mkn$Comp.1$runs$chains,  
 hzar.dataGroup.add),  
 mkn$Comp.1$analysis$oDG);  
  
## Check to make sure that there are only ten hzar.dataGroup  
## objects in the hzar.obsDataGroup object.  
print(summary(mkn$Comp.1$analysis$oDG$data.groups))

## Length Class Mode  
## fixN 6 hzar.dataGroup list  
## fixL 6 hzar.dataGroup list  
## fixR 6 hzar.dataGroup list  
## fixM 6 hzar.dataGroup list  
## fixB 6 hzar.dataGroup list  
## optN 6 hzar.dataGroup list  
## opComp.1 6 hzar.dataGroup list  
## optR 6 hzar.dataGroup list  
## optM 6 hzar.dataGroup list  
## optB 6 hzar.dataGroup list

## Look at the variation in parameters for cline models  
oDGkey <- which(!(names(mkn$Comp.1$analysis$oDG$data.groups) %in%  
 "nullModel"));  
print(hzar.getLLCutParam(mkn$Comp.1$analysis$oDG$data.groups[oDGkey ],  
 c("center","width")));

## center2LLLow center2LLHigh width2LLLow width2LLHigh  
## fixN 565.031824 887.1473 1.051574e+02 1096.40126  
## fixL 578.039540 914.9755 1.032566e+02 1109.12965  
## fixR 584.464948 888.5589 1.830397e+01 483.86665  
## fixM 595.635423 894.7678 1.238061e+02 819.84176  
## fixB 582.126667 903.7078 6.890622e+01 1093.99499  
## optN 445.469477 943.0301 3.161315e-01 981.94299  
## opComp.1 469.856500 823.3663 5.089741e+00 293.72153  
## optR -9.992227 282.5837 3.222879e-04 47.19304  
## optM 601.654143 926.2580 5.223992e+00 1093.65295  
## optB 504.351322 890.4339 2.553682e+01 1068.83467

## Compare the 3 cline models to the null model graphically  
##hzar.plot.cline(mkn$Comp.1$analysis$oDG);  
  
## Do model selection based on the AICc scores  
print(mkn$Comp.1$analysis$AICcTable <-  
 hzar.AICc.hzar.obsDataGroup(mkn$Comp.1$analysis$oDG));

## AICc  
## fixN 238.8490  
## fixL 243.4483  
## fixR 243.5759  
## fixM 243.5158  
## fixB 248.4193  
## optN 242.8341  
## opComp.1 247.6560  
## optR 274.3533  
## optM 247.1894  
## optB 252.8746

## Print out the model with the minimum AICc score  
print(mkn$Comp.1$analysis$model.name <-  
 rownames(mkn$Comp.1$analysis$AICcTable  
 )[[ which.min(mkn$Comp.1$analysis$AICcTable$AICc )]])

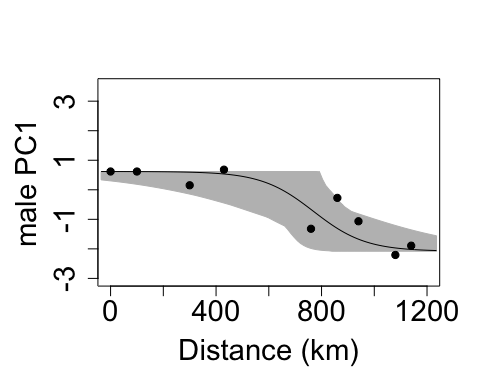
## [1] "fixN"

## Extract the hzar.dataGroup object for the selected model  
mkn$Comp.1$analysis$model.selected <-  
 mkn$Comp.1$analysis$oDG$data.groups[[mkn$Comp.1$analysis$model.name]]  
  
## Plot the maximum likelihood cline for the selected model  
##hzar.plot.cline(mkn$Comp.1$analysis$model.selected);  
  
print(hzar.get.ML.cline(mkn$Comp.1$analysis$model.selected))

## $param.free  
## center width varH  
## 675 769.6547 395.286 0.9632401  
##   
## $param.all  
## $param.all$center  
## [1] 769.6547  
##   
## $param.all$width  
## [1] 395.286  
##   
## $param.all$varH  
## [1] 0.9632401  
##   
## $param.all$muL  
## [1] 0.6188497  
##   
## $param.all$muR  
## [1] -2.080915  
##   
## $param.all$varL  
## [1] 3.381186  
##   
## $param.all$varR  
## [1] 1.605924  
##   
##   
## $clineFunc  
## function (x)   
## 0.618849669166667 + (-2.0809150238 - 0.618849669166667) \* (1/(1 +   
## exp(-4 \* (x - 769.654715525418)/395.286012357152)))  
##   
## $logLike  
## [1] -116.2245  
##   
## $isValid  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "hzar.cline"

check out your plot

## Plot the 95% credible cline region for the selected model  
##hzar.plot.fzCline(mkn$Comp.1$analysis$model.selected);  
  
par(mar=c(5,5,4,2)+.1)  
par(ps=22, cex=1, cex.main=1)  
male.pc1<-hzar.plot.fzCline(mkn$Comp.1$analysis$model.selected,  
 pch=19,  
 xlab="Distance (km)",  
 ylab="male PC1",  
 ylim=c(-3,3.5),xlim=c(0,1200))



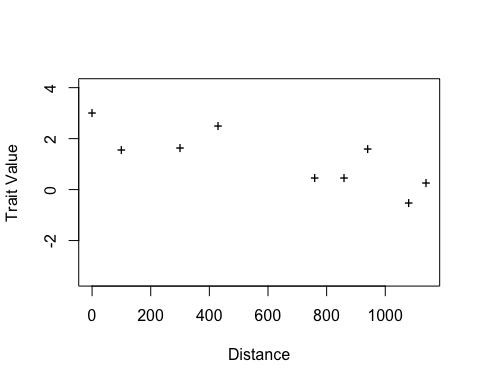
#export as 8"x8" PDF  
#dev.off()  
## End Comp.1

Open lists

## Blank out space in memory to hold morphological analysis  
if(length(apropos("^mkn$",ignore.case=FALSE)) == 0 ||  
 !is.list(mkn) ) mkn <- list()  
## We are doing just the one quantitative trait, but it is  
## good to stay organized.  
mkn$Comp.1 <- list();  
## Space to hold the observed data  
mkn$Comp.1$obs <- list();  
## Space to hold the models to fit  
mkn$Comp.1$models <- list();  
## Space to hold the compiled fit requests  
mkn$Comp.1$fitRs <- list();  
## Space to hold the output data chains  
mkn$Comp.1$runs <- list();  
## Space to hold the analysed data  
mkn$Comp.1$analysis <- list()

# set female body size, character 2

## Beard Length Trait from Brumfield et al 2001  
mkn$Comp.1$obs <-  
 hzar.doNormalData1DRaw(hzar.mapSiteDist(SiteLoc$Site,  
 SiteLoc$Distance),  
 femalemorph$Site,  
 femalemorph$Comp.1)  
  
## Look at a graph of the observed data  
hzar.plot.obsData(mkn$Comp.1$obs)



## Make a helper function  
mkn.loadComp.1model <- function(scaling,tails,  
 id=paste(scaling,tails,sep=".")){  
 mkn$Comp.1$models[[id]] <<-  
 hzar.makeCline1DNormal(mkn$Comp.1$obs, tails)  
 ## As there is no quick option for "fixed" scaling, and the  
 ## combined sites "A", "B" and "J","K" have a fair number of samples (> 20),  
 ## fix the mean and variance of the left and right sides of  
 ## the cline to the values observed by the combined sites.  
 if (all(regexpr("fixed",scaling,ignore.case=TRUE) == 1 )){  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$muL <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$muR <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$varL <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$varR <<- TRUE  
 }  
 ## Helper function to work around low sample size  
 getCombo <- function(id1,id2,colName,frame=mkn$Comp.1$obs$frame)  
 (frame[id1,colName]\*frame[id1,"nEff"]+  
 frame[id2,colName]\*frame[id2,"nEff"])/  
 (frame[id1,"nEff"]+frame[id2,"nEff"])  
   
 ## Site A, B is the "left" side of the cline, so pull the  
 ## fixed values from there.  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$muL <<-  
 getCombo("A","B","mu")  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varL <<-  
 getCombo("A","B","var")  
 ## Site J, K is the "right" side of the cline, so pull the  
 ## fixed values from there.  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$muR <<-  
 getCombo("I","J","mu")  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varR <<-  
 getCombo("I","J","var")  
 ## Make a better estimate of varH using site D, E  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varH <<-  
 getCombo("E","F","var")-  
 (getCombo("I","J","var")+getCombo("A","B","var"))/2  
   
}  
## mkn.loadComp.1model("fixed","none","modelI");  
## mkn.loadComp.1model("free" ,"none","modelII");  
## mkn.loadComp.1model("free" ,"both","modelIII");  
mkn.loadComp.1model("fixed","none" ,"fixN");  
mkn.loadComp.1model("fixed","left" ,"fixL");  
mkn.loadComp.1model("fixed","right" ,"fixR");  
mkn.loadComp.1model("fixed","mirror","fixM");  
mkn.loadComp.1model("fixed","both" ,"fixB");  
mkn.loadComp.1model("free" ,"none" ,"optN");  
mkn.loadComp.1model("free" ,"left" ,"opComp.1");  
mkn.loadComp.1model("free" ,"right" ,"optR");  
mkn.loadComp.1model("free" ,"mirror","optM");  
mkn.loadComp.1model("free" ,"both" ,"optB");  
  
  
## Check the default settings  
##print(mkn$Comp.1$models)  
  
## Modify all models to focus on the region where the observed  
## data were collected.  
## Observations were between -10 and 130000 km.  
mkn$Comp.1$models <- sapply(mkn$Comp.1$models,  
 hzar.model.addBoxReq,  
 -10 , 1100,  
 simplify=FALSE)  
  
## Due to the large number of free variables, it is prudent to  
## reduce the tune setting of optB from 1.5 to 1.1  
hzar:::meta.tune(mkn$Comp.1$models$optB)<-1.1  
  
  
## Check the updated settings  
##print(mkn$Comp.1$models)  
  
## Compile each of the models to prepare for fitting  
## Note that we are using hzar.first.fitRequest.gC for fitting  
## guassian (aka "normal") clines.  
mkn$Comp.1$fitRs$init <- sapply(mkn$Comp.1$models,  
 hzar.first.fitRequest.gC,  
 obsData=mkn$Comp.1$obs,  
 verbose=TRUE,  
 simplify=FALSE)

## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauR  
## [1] 0.5  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 247.5  
##   
## $tauM  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 247.5  
##   
## $tauM  
## [1] 0.5  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauR  
## [1] 0.5  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 247.5  
##   
## $tauM  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaM  
## [1] 247.5  
##   
## $tauM  
## [1] 0.5  
##   
##   
## a$center  
## [1] 1026  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5  
##   
## Ab$center  
## [1] 1080  
##   
## $width  
## [1] 330  
##   
## $muL  
## [1] 2.282268  
##   
## $muR  
## [1] -0.3916515  
##   
## $varL  
## [1] 0.515512  
##   
## $varR  
## [1] 1.900138  
##   
## $varH  
## [1] NA  
##   
## $deltaL  
## [1] 247.5  
##   
## $deltaR  
## [1] 247.5  
##   
## $tauL  
## [1] 0.5  
##   
## $tauR  
## [1] 0.5

## Update the settings for the fitter if desired.  
mkn$Comp.1$fitRs$init <- sapply(mkn$Comp.1$fitRs$init,  
 function(mdl) {  
 mdl$mcmcParam$chainLength <-  
 chainLength; #1e5  
 mdl$mcmcParam$burnin <-  
 chainLength %/% 10; #1e4  
 mdl },  
 simplify=FALSE)  
  
mkn$Comp.1$fitRs$init <-  
 rotateModelSeeds(mkn$Comp.1$fitRs$init)  
  
  
## Check fit request settings  
##print(mkn$Comp.1$fitRs$init)  
  
## Do just one run of the models for an initial chain  
mkn$Comp.1$runs$init <-  
 hzar.doFit.multi(mkn$Comp.1$fitRs$init)

## MCMCmetrop1R iteration 1 of 110000   
## function value = -83.92694  
## theta =   
## -9.99993  
## 852.93868  
## 11.44809  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.02945  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -71.53670  
## theta =   
## 1100.00000  
## 859.82272  
## 4.46944  
## 825.14003  
## 0.05039  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.04486  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -84.76982  
## theta =   
## -10.00000  
## 814.77986  
## 14.31376  
## 933.71165  
## 0.94225  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00248  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -66.24693  
## theta =   
## 584.69993  
## 124.05846  
## 15.39251  
## 191.67960  
## 0.00000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.29199  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -89.23192  
## theta =   
## 195.59025  
## 65.33330  
## 107.56207  
## 62.27664  
## 433.86889  
## 0.19103  
## 0.24741  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.05071  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -167.36965  
## theta =   
## 1100.00000  
## 539.57465  
## -0.98180  
## 2.09214  
## 0.00161  
## 7003.03309  
## 0.22485  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00020  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -174.57507  
## theta =   
## 737.43680  
## 307.39470  
## -0.98632  
## 3.44243  
## 0.00081  
## 2832.29363  
## 0.05707  
## 214.79204  
## 0.66633  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00020  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -90.67659  
## theta =   
## 89.78146  
## 333.56342  
## -1.21821  
## -0.29645  
## 3.85925  
## 5.37147  
## 0.24238  
## 1011.64665  
## 0.00113  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00015  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -180.07928  
## theta =   
## 546.07324  
## 283.28914  
## -0.28581  
## -2.70285  
## 0.00076  
## 2122.71648  
## 1.61671  
## 426.37857  
## 0.93520  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00040  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = -154.87738  
## theta =   
## 1100.00000  
## 501.11537  
## -3.42872  
## 3.92421  
## 0.00411  
## 2263.45922  
## 0.03513  
## 524.15300  
## 0.00000  
## 0.18367  
## 0.00000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.00020  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

names(mkn$Comp.1$runs$init) <- names(mkn$Comp.1$fitRs$init)  
  
  
  
## Compile a new set of fit requests using the initial chains   
mkn$Comp.1$fitRs$chains <-  
 lapply(mkn$Comp.1$runs$init,  
 hzar.next.fitRequest)

## 0ac 2.9e+04dfg0ac 6.1e+04dfg0ac 1.2e+04dfg0ac 1.6e+07dfg0ac 1.4e+11dfg0ac 0.0e+00dfg0ac 3.1e-04dfg0ac 7.1e+03dfg0ac 9.3e+03dfg0ac 7.0e-01dfg

## Replicate each fit request 3 times, keeping the original  
## seeds while switching to a new seed channel.  
mkn$Comp.1$fitRs$chains <-  
 hzar.multiFitRequest(mkn$Comp.1$fitRs$chains,  
 each=3,  
 baseSeed=NULL)  
  
## Just to be thorough, randomize the initial value for each fit  
  
  
## Go ahead and run a chain of 3 runs for every fit request  
mkn$Comp.1$runs$chains <- hzar.doChain.multi(mkn$Comp.1$fitRs$chains,  
 doPar=TRUE,  
 inOrder=FALSE,  
 count=3)

## Did fixN converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$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 5.963e+02 6.734e+01 1.229e+00 1.230e+00  
## width 1.248e+01 8.942e+00 1.633e-01 1.720e-01  
## varH 2.421e+07 1.780e+07 3.250e+05 6.691e+06  
## model.LL -6.640e+01 2.683e-01 4.898e-03 5.756e-03  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 4.694e+02 5.473e+02 5.968e+02 6.437e+02 7.251e+02  
## width 5.906e-01 5.068e+00 1.073e+01 1.859e+01 3.192e+01  
## varH 4.000e+05 7.109e+06 2.493e+07 3.351e+07 6.540e+07  
## model.LL -6.704e+01 -6.635e+01 -6.635e+01 -6.635e+01 -6.634e+01

## Did optN converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$runs$chains[16:18],  
 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 866.6661 195.0237 3.560629 13.04683  
## width 619.9348 321.0909 5.862291 15.07420  
## muL 2.1628 0.3853 0.007035 0.02420  
## muR -1.2574 1.2245 0.022355 0.05108  
## varL 0.9206 1.2253 0.022371 0.13284  
## varR 2.3683 1.5482 0.028266 0.06863  
## varH 2.1843 3.4568 0.063112 0.23895  
## model.LL -66.0329 2.3860 0.043562 0.16874  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 496.17387 736.8297 919.4716 1027.9758 1093.184  
## width 16.08820 393.3804 670.0300 870.3822 1091.936  
## muL 1.57861 1.9494 2.1172 2.3331 2.987  
## muR -3.91551 -2.0961 -1.1442 -0.2349 0.601  
## varL 0.24640 0.5352 0.7288 0.9896 2.142  
## varR 0.19654 1.4011 2.0687 2.9656 6.224  
## varH 0.02949 0.4006 1.0019 1.9862 12.924  
## model.LL -72.02126 -66.9097 -65.5354 -64.4750 -63.314

## Did optB converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$runs$chains[28:30],  
 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 1078.9620 17.9617 0.327934 3.576595  
## width 796.5550 204.9339 3.741563 5.272435  
## muL 2.2184 0.4568 0.008341 0.043571  
## muR -2.9811 1.1328 0.020683 0.073012  
## varL 0.6765 0.3849 0.007027 0.011582  
## varR 3.0426 2.3801 0.043455 0.063121  
## varH 1.1812 1.0124 0.018484 0.033387  
## deltaL 578.3159 313.0084 5.714725 8.729609  
## deltaR 550.6770 323.5462 5.907118 8.030626  
## tauL 0.5018 0.2804 0.005120 0.007384  
## tauR 0.5153 0.2891 0.005278 0.007739  
## model.LL -65.8682 2.0876 0.038114 0.061757  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 1036.63349 1066.5094 1084.9542 1093.6403 1099.3353  
## width 349.99071 662.5651 818.3168 963.6014 1095.9787  
## muL 1.54114 1.9591 2.1579 2.3941 3.6813  
## muR -5.67405 -3.6380 -2.9327 -2.2543 -0.8263  
## varL 0.07985 0.4149 0.6165 0.8764 1.6334  
## varR 0.13276 1.3114 2.6402 4.2440 8.5976  
## varH 0.05102 0.4319 0.9709 1.6345 3.7313  
## deltaL 36.26436 308.6770 607.8520 832.2222 1079.6530  
## deltaR 30.70837 260.6294 551.6318 832.4530 1084.8413  
## tauL 0.03331 0.2675 0.4918 0.7392 0.9754  
## tauR 0.02932 0.2638 0.5200 0.7664 0.9762  
## model.LL -70.87846 -66.9970 -65.4744 -64.3195 -63.0890

## Start aggregation of data for analysis  
  
## Clear out a spot to collect the data for analysis (note that  
## there is currenComp.1y no "null model" to compare against).  
mkn$Comp.1$analysis$initDGs <- list(  
)  
  
## Create a model data group (hzar.dataGroup object) for each  
## model from the initial runs.  
mkn$Comp.1$analysis$initDGs <-  
 c( mkn$Comp.1$analysis$initDGs,  
 sapply(mkn$Comp.1$runs$init,  
 hzar.dataGroup.add,  
 simplify=FALSE))  
  
## Create a hzar.obsDataGroup object from the four hzar.dataGroup  
## just created, copying the naming scheme (modelI, modelII,  
## modelIII).  
mkn$Comp.1$analysis$oDG <-  
 hzar.make.obsDataGroup(mkn$Comp.1$analysis$initDGs)

## [1] 10

mkn$Comp.1$analysis$oDG <-  
 hzar.copyModelLabels(mkn$Comp.1$analysis$initDGs,  
 mkn$Comp.1$analysis$oDG)  
  
## Convert all 90 runs to hzar.dataGroup objects, adding them to  
## the hzar.obsDataGroup object.  
mkn$Comp.1$analysis$oDG <-  
 hzar.make.obsDataGroup(lapply(mkn$Comp.1$runs$chains,  
 hzar.dataGroup.add),  
 mkn$Comp.1$analysis$oDG);  
  
## Check to make sure that there are only ten hzar.dataGroup  
## objects in the hzar.obsDataGroup object.  
print(summary(mkn$Comp.1$analysis$oDG$data.groups))

## Length Class Mode  
## fixN 6 hzar.dataGroup list  
## fixL 6 hzar.dataGroup list  
## fixR 6 hzar.dataGroup list  
## fixM 6 hzar.dataGroup list  
## fixB 6 hzar.dataGroup list  
## optN 6 hzar.dataGroup list  
## opComp.1 6 hzar.dataGroup list  
## optR 6 hzar.dataGroup list  
## optM 6 hzar.dataGroup list  
## optB 6 hzar.dataGroup list

## Look at the variation in parameters for cline models  
oDGkey <- which(!(names(mkn$Comp.1$analysis$oDG$data.groups) %in%  
 "nullModel"));  
print(hzar.getLLCutParam(mkn$Comp.1$analysis$oDG$data.groups[oDGkey ],  
 c("center","width")));

## center2LLLow center2LLHigh width2LLLow width2LLHigh  
## fixN 632.7804 888.8020 1.944070e+02 1075.10884  
## fixL 584.9400 890.8806 2.031896e+02 1105.36053  
## fixR 596.4628 886.0415 8.695846e+01 1093.68117  
## fixM 431.7655 758.4416 5.309517e-03 74.59295  
## fixB 434.1827 758.5236 8.175199e-04 69.39300  
## optN 465.0657 1099.9541 9.178632e-01 1109.53868  
## opComp.1 469.2681 1099.6282 2.480258e+00 1109.37484  
## optR 434.8182 1098.4947 5.418195e-02 1109.78642  
## optM 433.5872 753.2317 3.050117e-03 86.77621  
## optB 1007.2615 1099.9999 3.084437e+02 1109.97243

## Compare the 3 cline models to the null model graphically  
##hzar.plot.cline(mkn$Comp.1$analysis$oDG);  
  
## Do model selection based on the AICc scores  
print(mkn$Comp.1$analysis$AICcTable <-  
 hzar.AICc.hzar.obsDataGroup(mkn$Comp.1$analysis$oDG));

## AICc  
## fixN 132.2324  
## fixL 136.0709  
## fixR 136.9674  
## fixM 143.6045  
## fixB 149.5118  
## optN 142.4566  
## opComp.1 148.1728  
## optR 148.9269  
## optM 150.2358  
## optB 155.8364

## Print out the model with the minimum AICc score  
print(mkn$Comp.1$analysis$model.name <-  
 rownames(mkn$Comp.1$analysis$AICcTable  
 )[[ which.min(mkn$Comp.1$analysis$AICcTable$AICc )]])

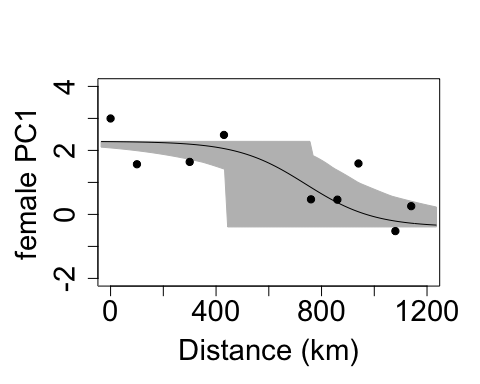
## [1] "fixN"

## Extract the hzar.dataGroup object for the selected model  
mkn$Comp.1$analysis$model.selected <-  
 mkn$Comp.1$analysis$oDG$data.groups[[mkn$Comp.1$analysis$model.name]]  
  
## Plot the maximum likelihood cline for the selected model  
##hzar.plot.cline(mkn$Comp.1$analysis$model.selected);  
  
print(hzar.get.ML.cline(mkn$Comp.1$analysis$model.selected))

## $param.free  
## center width varH  
## 846 740.1891 519.7283 0.07829332  
##   
## $param.all  
## $param.all$center  
## [1] 740.1891  
##   
## $param.all$width  
## [1] 519.7283  
##   
## $param.all$varH  
## [1] 0.07829332  
##   
## $param.all$muL  
## [1] 2.282268  
##   
## $param.all$muR  
## [1] -0.3916515  
##   
## $param.all$varL  
## [1] 0.515512  
##   
## $param.all$varR  
## [1] 1.900138  
##   
##   
## $clineFunc  
## function (x)   
## 2.2822678625 + (-0.391651469083333 - 2.2822678625) \* (1/(1 +   
## exp(-4 \* (x - 740.189090363207)/519.728334624764)))  
##   
## $logLike  
## [1] -62.8004  
##   
## $isValid  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "hzar.cline"

check out your plot

## Plot the 95% credible cline region for the selected model  
##hzar.plot.fzCline(mkn$Comp.1$analysis$model.selected);  
  
par(mar=c(5,5,4,2)+.1)  
par(ps=22, cex=1, cex.main=1)  
male.pc1<-hzar.plot.fzCline(mkn$Comp.1$analysis$model.selected,  
 pch=19,  
 xlab="Distance (km)",  
 ylab="female PC1",  
 ylim=c(-2,4),xlim=c(0,1200))



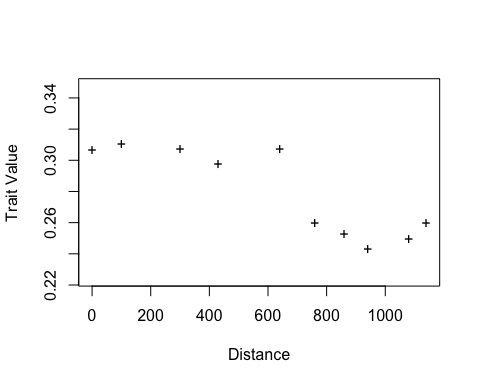
#export as 8"x8" PDF  
#dev.off()  
## End Comp.1

Open lists

## Blank out space in memory to hold morphological analysis  
if(length(apropos("^mkn$",ignore.case=FALSE)) == 0 ||  
 !is.list(mkn) ) mkn <- list()  
## We are doing just the one quantitative trait, but it is  
## good to stay organized.  
mkn$Comp.1 <- list();  
## Space to hold the observed data  
mkn$Comp.1$obs <- list();  
## Space to hold the models to fit  
mkn$Comp.1$models <- list();  
## Space to hold the compiled fit requests  
mkn$Comp.1$fitRs <- list();  
## Space to hold the output data chains  
mkn$Comp.1$runs <- list();  
## Space to hold the analysed data  
mkn$Comp.1$analysis <- list()

# Set s1blue, character 3

## Beard Length Trait from Brumfield et al 2001  
mkn$Comp.1$obs <-  
 hzar.doNormalData1DRaw(hzar.mapSiteDist(SiteLoc$Site,  
 SiteLoc$Distance),  
 S1blue$Site,  
 S1blue$S1.blue)  
  
## Look at a graph of the observed data  
hzar.plot.obsData(mkn$Comp.1$obs)



## Make a helper function  
mkn.loadComp.1model <- function(scaling,tails,  
 id=paste(scaling,tails,sep=".")){  
 mkn$Comp.1$models[[id]] <<-  
 hzar.makeCline1DNormal(mkn$Comp.1$obs, tails)  
 ## As there is no quick option for "fixed" scaling, and the  
 ## combined sites "A", "B" and "J","K" have a fair number of samples (> 20),  
 ## fix the mean and variance of the left and right sides of  
 ## the cline to the values observed by the combined sites.  
 if (all(regexpr("fixed",scaling,ignore.case=TRUE) == 1 )){  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$muL <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$muR <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$varL <<- TRUE  
 hzar:::meta.fix(mkn$Comp.1$models[[id]])$varR <<- TRUE  
 }  
 ## Helper function to work around low sample size  
 getCombo <- function(id1,id2,colName,frame=mkn$Comp.1$obs$frame)  
 (frame[id1,colName]\*frame[id1,"nEff"]+  
 frame[id2,colName]\*frame[id2,"nEff"])/  
 (frame[id1,"nEff"]+frame[id2,"nEff"])  
   
 ## Site A, B is the "left" side of the cline, so pull the  
 ## fixed values from there.  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$muL <<-  
 getCombo("A","B","mu")  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varL <<-  
 getCombo("A","B","var")  
 ## Site J, K is the "right" side of the cline, so pull the  
 ## fixed values from there.  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$muR <<-  
 getCombo("I","J","mu")  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varR <<-  
 getCombo("I","J","var")  
 ## Make a better estimate of varH using site D, E  
 hzar:::meta.init(mkn$Comp.1$models[[id]])$varH <<-  
 getCombo("E","F","var")-  
 (getCombo("I","J","var")+getCombo("A","B","var"))/2  
   
}  
## mkn.loadComp.1model("fixed","none","modelI");  
## mkn.loadComp.1model("free" ,"none","modelII");  
## mkn.loadComp.1model("free" ,"both","modelIII");  
mkn.loadComp.1model("fixed","none" ,"fixN");  
mkn.loadComp.1model("fixed","left" ,"fixL");  
mkn.loadComp.1model("fixed","right" ,"fixR");  
mkn.loadComp.1model("fixed","mirror","fixM");  
mkn.loadComp.1model("fixed","both" ,"fixB");  
mkn.loadComp.1model("free" ,"none" ,"optN");  
mkn.loadComp.1model("free" ,"left" ,"opComp.1");  
mkn.loadComp.1model("free" ,"right" ,"optR");  
mkn.loadComp.1model("free" ,"mirror","optM");  
mkn.loadComp.1model("free" ,"both" ,"optB");  
  
  
## Check the default settings  
##print(mkn$Comp.1$models)  
  
## Modify all models to focus on the region where the observed  
## data were collected.  
## Observations were between -10 and 130000 km.  
mkn$Comp.1$models <- sapply(mkn$Comp.1$models,  
 hzar.model.addBoxReq,  
 -10 , 1100,  
 simplify=FALSE)  
  
## Due to the large number of free variables, it is prudent to  
## reduce the tune setting of optB from 1.5 to 1.1  
hzar:::meta.tune(mkn$Comp.1$models$optB)<-1.1  
  
  
## Check the updated settings  
##print(mkn$Comp.1$models)  
  
## Compile each of the models to prepare for fitting  
## Note that we are using hzar.first.fitRequest.gC for fitting  
## guassian (aka "normal") clines.  
mkn$Comp.1$fitRs$init <- sapply(mkn$Comp.1$models,  
 hzar.first.fitRequest.gC,  
 obsData=mkn$Comp.1$obs,  
 verbose=TRUE,  
 simplify=FALSE)

## a  
## ab  
## a  
## a  
## ab  
## a  
## ab  
## a  
## a  
## ab

## Update the settings for the fitter if desired.  
mkn$Comp.1$fitRs$init <- sapply(mkn$Comp.1$fitRs$init,  
 function(mdl) {  
 mdl$mcmcParam$chainLength <-  
 chainLength; #1e5  
 mdl$mcmcParam$burnin <-  
 chainLength %/% 10; #1e4  
 mdl },  
 simplify=FALSE)  
  
mkn$Comp.1$fitRs$init <-  
 rotateModelSeeds(mkn$Comp.1$fitRs$init)  
  
  
## Check fit request settings  
##print(mkn$Comp.1$fitRs$init)  
  
## Do just one run of the models for an initial chain  
mkn$Comp.1$runs$init <-  
 hzar.doFit.multi(mkn$Comp.1$fitRs$init)

## MCMCmetrop1R iteration 1 of 110000   
## function value = 287.16608  
## theta =   
## 757.23661  
## 252.72654  
## 0.00038  
## Metropolis acceptance rate = 1.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.47674  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 286.84935  
## theta =   
## 760.00000  
## 120.00000  
## 0.00018  
## 72.00000  
## 0.07500  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.33501  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 285.76987  
## theta =   
## 760.00000  
## 120.00000  
## 0.00018  
## 90.00000  
## 0.50000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.19086  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 285.78336  
## theta =   
## 760.00000  
## 120.00000  
## 0.00018  
## 90.00000  
## 0.50000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.13858  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 286.72165  
## theta =   
## 760.00000  
## 120.00000  
## 0.00018  
## 72.00000  
## 90.00000  
## 0.07500  
## 0.50000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.12671  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 285.89757  
## theta =   
## 760.00000  
## 120.00000  
## 0.30869  
## 0.25269  
## 0.00036  
## 0.00010  
## 0.00018  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.16401  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 286.84935  
## theta =   
## 760.00000  
## 120.00000  
## 0.30869  
## 0.25269  
## 0.00036  
## 0.00010  
## 0.00018  
## 72.00000  
## 0.07500  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.07089  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 285.76987  
## theta =   
## 760.00000  
## 120.00000  
## 0.30869  
## 0.25269  
## 0.00036  
## 0.00010  
## 0.00018  
## 90.00000  
## 0.50000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.06147  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 285.78336  
## theta =   
## 760.00000  
## 120.00000  
## 0.30869  
## 0.25269  
## 0.00036  
## 0.00010  
## 0.00018  
## 90.00000  
## 0.50000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.04587  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## MCMCmetrop1R iteration 1 of 110000   
## function value = 286.72165  
## theta =   
## 760.00000  
## 120.00000  
## 0.30869  
## 0.25269  
## 0.00036  
## 0.00010  
## 0.00018  
## 72.00000  
## 90.00000  
## 0.07500  
## 0.50000  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.09531  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

names(mkn$Comp.1$runs$init) <- names(mkn$Comp.1$fitRs$init)  
  
  
  
## Compile a new set of fit requests using the initial chains   
mkn$Comp.1$fitRs$chains <-  
 lapply(mkn$Comp.1$runs$init,  
 hzar.next.fitRequest)

## 0ac 3.0e+01dfg0ac 3.4e+01dfg0ac 3.1e+04dfg0ac 2.3e-13dfg0ac 4.0e-10dfg0ac 1.5e-10dfg0ac 1.0e-10dfg0ac 1.2e-06dfg

## Replicate each fit request 3 times, keeping the original  
## seeds while switching to a new seed channel.  
mkn$Comp.1$fitRs$chains <-  
 hzar.multiFitRequest(mkn$Comp.1$fitRs$chains,  
 each=3,  
 baseSeed=NULL)  
  
## Just to be thorough, randomize the initial value for each fit  
  
  
## Go ahead and run a chain of 3 runs for every fit request  
mkn$Comp.1$runs$chains <- hzar.doChain.multi(mkn$Comp.1$fitRs$chains,  
 doPar=TRUE,  
 inOrder=FALSE,  
 count=3)

## Did fixN converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$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 686.1232 43.1558 0.78791 4.94760  
## width 132.9994 117.3898 2.14324 16.20156  
## varH 0.1507 0.2525 0.00461 0.09525  
## model.LL 291.2692 1.3987 0.02554 0.06288  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 5.964e+02 6.542e+02 6.987e+02 719.3787 748.3993  
## width 6.506e+00 2.390e+01 1.160e+02 224.1925 373.6463  
## varH 3.046e-04 7.915e-04 2.715e-03 0.2487 0.9422  
## model.LL 2.879e+02 2.906e+02 2.913e+02 292.3351 293.3162

## Did optN converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$runs$chains[16:18],  
 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 7.133e+02 2.596e+01 4.739e-01 1.292e+00  
## width 2.441e+01 3.031e+01 5.535e-01 2.499e+00  
## muL 3.042e-01 2.674e-03 4.882e-05 1.309e-04  
## muR 2.517e-01 1.968e-03 3.593e-05 1.057e-04  
## varL 3.310e-04 7.011e-05 1.280e-06 3.446e-06  
## varR 1.893e-04 5.480e-05 1.000e-06 2.963e-06  
## varH 8.708e-01 8.079e-01 1.475e-02 1.411e-01  
## model.LL 2.924e+02 2.444e+00 4.462e-02 1.684e-01  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 6.489e+02 7.032e+02 7.152e+02 7.284e+02 7.521e+02  
## width 1.629e+00 1.153e+01 1.851e+01 2.549e+01 1.353e+02  
## muL 2.990e-01 3.026e-01 3.042e-01 3.059e-01 3.095e-01  
## muR 2.479e-01 2.503e-01 2.516e-01 2.531e-01 2.554e-01  
## varL 2.176e-04 2.806e-04 3.214e-04 3.703e-04 4.926e-04  
## varR 1.176e-04 1.507e-04 1.793e-04 2.131e-04 3.254e-04  
## varH 1.109e-03 2.457e-01 6.711e-01 1.206e+00 2.912e+00  
## model.LL 2.864e+02 2.910e+02 2.930e+02 2.941e+02 2.954e+02

## Did optB converge?  
summary(do.call(mcmc.list,  
 lapply(mkn$Comp.1$runs$chains[28:30],  
 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 7.032e+02 3.375e+01 6.163e-01 1.626e+00  
## width 8.653e+01 8.605e+01 1.571e+00 1.028e+01  
## muL 3.044e-01 3.298e-03 6.022e-05 2.467e-04  
## muR 2.511e-01 2.081e-03 3.799e-05 1.127e-04  
## varL 3.299e-04 8.310e-05 1.517e-06 5.458e-06  
## varR 1.751e-04 5.196e-05 9.486e-07 4.083e-06  
## varH 5.208e-02 6.847e-02 1.250e-03 2.288e-02  
## deltaL 5.771e+02 3.254e+02 5.940e+00 1.880e+01  
## deltaR 5.209e+02 3.237e+02 5.910e+00 2.068e+01  
## tauL 4.978e-01 3.023e-01 5.520e-03 1.718e-02  
## tauR 5.286e-01 2.911e-01 5.314e-03 1.614e-02  
## model.LL 2.926e+02 2.154e+00 3.933e-02 1.489e-01  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## center 6.157e+02 6.904e+02 7.104e+02 7.266e+02 7.491e+02  
## width 5.282e+00 2.411e+01 4.392e+01 1.358e+02 3.043e+02  
## muL 2.986e-01 3.022e-01 3.043e-01 3.062e-01 3.109e-01  
## muR 2.469e-01 2.497e-01 2.512e-01 2.526e-01 2.551e-01  
## varL 1.939e-04 2.742e-04 3.261e-04 3.796e-04 5.020e-04  
## varR 9.559e-05 1.403e-04 1.673e-04 1.996e-04 3.130e-04  
## varH 1.928e-04 8.960e-04 6.809e-03 9.236e-02 2.044e-01  
## deltaL 5.052e+01 2.770e+02 5.994e+02 8.644e+02 1.082e+03  
## deltaR 2.817e+01 2.445e+02 4.907e+02 7.958e+02 1.082e+03  
## tauL 1.190e-02 2.265e-01 5.174e-01 7.742e-01 9.720e-01  
## tauR 3.220e-02 2.763e-01 5.269e-01 7.804e-01 9.819e-01  
## model.LL 2.877e+02 2.914e+02 2.931e+02 2.941e+02 2.958e+02

## Start aggregation of data for analysis  
  
## Clear out a spot to collect the data for analysis (note that  
## there is currenComp.1y no "null model" to compare against).  
mkn$Comp.1$analysis$initDGs <- list(  
)  
  
## Create a model data group (hzar.dataGroup object) for each  
## model from the initial runs.  
mkn$Comp.1$analysis$initDGs <-  
 c( mkn$Comp.1$analysis$initDGs,  
 sapply(mkn$Comp.1$runs$init,  
 hzar.dataGroup.add,  
 simplify=FALSE))  
  
## Create a hzar.obsDataGroup object from the four hzar.dataGroup  
## just created, copying the naming scheme (modelI, modelII,  
## modelIII).  
mkn$Comp.1$analysis$oDG <-  
 hzar.make.obsDataGroup(mkn$Comp.1$analysis$initDGs)

## [1] 10

mkn$Comp.1$analysis$oDG <-  
 hzar.copyModelLabels(mkn$Comp.1$analysis$initDGs,  
 mkn$Comp.1$analysis$oDG)  
  
## Convert all 90 runs to hzar.dataGroup objects, adding them to  
## the hzar.obsDataGroup object.  
mkn$Comp.1$analysis$oDG <-  
 hzar.make.obsDataGroup(lapply(mkn$Comp.1$runs$chains,  
 hzar.dataGroup.add),  
 mkn$Comp.1$analysis$oDG);  
  
## Check to make sure that there are only ten hzar.dataGroup  
## objects in the hzar.obsDataGroup object.  
print(summary(mkn$Comp.1$analysis$oDG$data.groups))

## Length Class Mode  
## fixN 6 hzar.dataGroup list  
## fixL 6 hzar.dataGroup list  
## fixR 6 hzar.dataGroup list  
## fixM 6 hzar.dataGroup list  
## fixB 6 hzar.dataGroup list  
## optN 6 hzar.dataGroup list  
## opComp.1 6 hzar.dataGroup list  
## optR 6 hzar.dataGroup list  
## optM 6 hzar.dataGroup list  
## optB 6 hzar.dataGroup list

## Look at the variation in parameters for cline models  
oDGkey <- which(!(names(mkn$Comp.1$analysis$oDG$data.groups) %in%  
 "nullModel"));  
print(hzar.getLLCutParam(mkn$Comp.1$analysis$oDG$data.groups[oDGkey ],  
 c("center","width")));

## center2LLLow center2LLHigh width2LLLow width2LLHigh  
## fixN 578.4043 758.0223 1.8036559 445.6374  
## fixL 580.9194 756.9412 6.4940211 426.1471  
## fixR 599.8896 744.0916 18.5841118 387.7142  
## fixM 595.3337 743.7049 6.6255177 404.4922  
## fixB 601.4668 741.3346 17.0857666 378.5956  
## optN 616.8794 759.6671 0.9619093 315.9263  
## opComp.1 623.1333 758.0597 3.5737139 337.1956  
## optR 637.5813 756.6937 3.8762821 340.7050  
## optM 621.3724 758.1869 2.2940114 365.4111  
## optB 619.9877 759.4977 1.4708540 352.2028

## Compare the 3 cline models to the null model graphically  
##hzar.plot.cline(mkn$Comp.1$analysis$oDG);  
  
## Do model selection based on the AICc scores  
print(mkn$Comp.1$analysis$AICcTable <-  
 hzar.AICc.hzar.obsDataGroup(mkn$Comp.1$analysis$oDG));

## AICc  
## fixN -580.7619  
## fixL -576.8585  
## fixR -578.4070  
## fixM -578.1884  
## fixB -574.1163  
## optN -578.3998  
## opComp.1 -573.8543  
## optR -574.0469  
## optM -573.8801  
## optB -569.2569

## Print out the model with the minimum AICc score  
print(mkn$Comp.1$analysis$model.name <-  
 rownames(mkn$Comp.1$analysis$AICcTable  
 )[[ which.min(mkn$Comp.1$analysis$AICcTable$AICc )]])

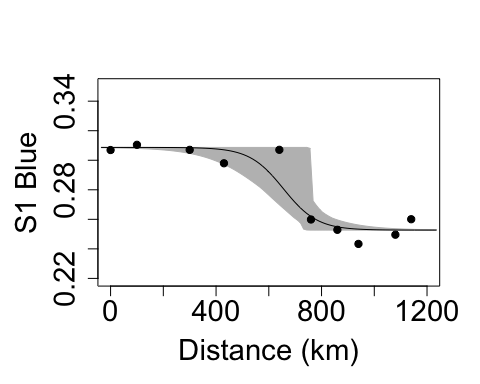
## [1] "fixN"

## Extract the hzar.dataGroup object for the selected model  
mkn$Comp.1$analysis$model.selected <-  
 mkn$Comp.1$analysis$oDG$data.groups[[mkn$Comp.1$analysis$model.name]]  
  
## Plot the maximum likelihood cline for the selected model  
##hzar.plot.cline(mkn$Comp.1$analysis$model.selected);  
  
print(hzar.get.ML.cline(mkn$Comp.1$analysis$model.selected))

## $param.free  
## center width varH  
## 960 655.4965 257.2473 0.0005967894  
##   
## $param.all  
## $param.all$center  
## [1] 655.4965  
##   
## $param.all$width  
## [1] 257.2473  
##   
## $param.all$varH  
## [1] 0.0005967894  
##   
## $param.all$muL  
## [1] 0.3086877  
##   
## $param.all$muR  
## [1] 0.2526924  
##   
## $param.all$varL  
## [1] 0.0003566026  
##   
## $param.all$varR  
## [1] 0.0001029839  
##   
##   
## $clineFunc  
## function (x)   
## 0.3086876914 + (0.252692425333333 - 0.3086876914) \* (1/(1 + exp(-4 \*   
## (x - 655.496475157108)/257.247342689255)))  
##   
## $logLike  
## [1] 293.4963  
##   
## $isValid  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "hzar.cline"

check out your plot

## Plot the 95% credible cline region for the selected model  
##hzar.plot.fzCline(mkn$Comp.1$analysis$model.selected);  
  
par(mar=c(5,5,4,2)+.1)  
par(ps=22, cex=1, cex.main=1)  
male.pc1<-hzar.plot.fzCline(mkn$Comp.1$analysis$model.selected,  
 pch=19,  
 xlab="Distance (km)",  
 ylab="S1 Blue",  
 ylim=c(.22,.35),xlim=c(0,1200))



#export as 8"x8" PDF  
#dev.off()  
## End Comp.1