PCA_analysis

Kevin Healy

19 June 2017

First we load the required packages including the mulTree package which will allow us to read back in the relavent information from the MCMCglmm models we ran previously. We will also require the `paran` package for the PCA test, `caper`, `phytools` and `MCMCglmm` to handle the phylogeny objects and related functions and `SIBER` and `ggplot2` to creat the ellipses and plot the overlaps. As in previous scripts we will also use some costume functions.

```
library(caper)
```

```
## Loading required package: ape
```

```
## Loading required package: MASS
```

```
## Loading required package: mvtnorm
```

```
library(phytools)
```

```
## Loading required package: maps
```

```
## Loading required package: rgl
```

```
library(MCMCglmm)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:phytools':
##
##     expm
```

```
## Loading required package: coda
```

```
library(mulTree)
```

```
## Loading required package: hdrcde
```

```
## This is hdrcde 3.3
```

```
## Loading required package: snow
```

```
library(paran)
library(SIBER)
library(ggplot2)

source("Demography_functions.R")
```

```
## Welcome to popdemo! This is version 1.3-0
## Use ?popdemo for an intro, or browseVignettes('popdemo') for vignettes
## Citation for popdemo is here: doi.org/10.1111/j.2041-210X.2012.00222.x
## Development and legacy versions are here: github.com/iainmstott/popdemo
```

Now we upload the previous data including both the life history metrics we calulated and also the body size, IUCN and other data.

```
pop_data <- read.csv("axis_analysis_data.csv",
                          sep = ",", header = T)
```

Next we upload the matching distribution of phylogenies calculated from the previous Phylogeny_construction and Pop_metric_calulation scripts.

```
axis_trees <- read.tree("axis_analysis_phylo.tre")
```

Log10 the non index based metrics

```
log_list <- c("life_time_La",
              "mean_repo_rate_stable_state",
              "mean_repo_rate",
              "gen_time",
              "M_rep_lif_exp",
              "gini",
              "surv_sd",
              "mass_g",
              "mxlxsd",
              "matrix_size")

pop_data_log <- pop_data

pop_data_log[,log_list] <- sapply(pop_data[,log_list], function(x) log10(x))
```

And mean center the data

```
mean_c_list  <- c( "life_time_La",
                   "mean_repo_rate",
                   "gen_time",
                   "M_rep_lif_exp",
                   "matrix_size",
                   "gini",
                   "mean_repo_rate_stable_state",
                   "mxlxsd",
                   "surv_sd",
                   "mass_g")

pop_data_log_mc <- pop_data_log
pop_data_log_mc[,mean_c_list] <- sapply(pop_data_log[,mean_c_list], function(x) mean_center(x))
```

We also need to make a multree object that holds both the data and the multiphylo object and from which we will calulate the residuals from each of the modeel

```
pop_multree <- as.mulTree(data = pop_data_log_mc, tree = axis_trees, taxa = "animal", rand.terms = ~ani
```

Read back in the the MCMCglmm model outputs

For each of the life history metrics we read back in the information we need from the 100 MCMCglmm models we ran. We will then calculate the residuals form these which will be then used for the PCA analysis. We will also look at the random terms of phylogentic effect, population level variance (species) and the residual term.

Age at first reproduction

```
La_models <- read.mulTree("la_run")
summary(La_models)
```

```
##                       Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                      0.19739324   -1.50849226  -0.36691793
## mass_g                           0.59886093    0.38364793   0.52569761
## matrix_size                      0.06146116   -0.01051713   0.03715002
## phylogenetic.variance            2.03170953    0.92566768   1.57139219
## residual.variance                0.29098872    0.17437855   0.24779242
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)             0.76238945      1.9105211
## mass_g                  0.67352270      0.8166247
## matrix_size             0.08760917      0.1367495
## phylogenetic.variance   2.54300412      3.8375234
## residual.variance       0.34484697      0.4648104
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
la_var <- read.mulTree("la_run", extract = "VCV")

#phylogenetic signal
la_phlyo <- list()
#population level variation
la_spec <- list()
#residual
la_unit <- list()

#extrace the random terms from across the models
for(i in 1:length(names(la_var))){

  la_phlyo[[i]] <-  la_var[[1]][,1]
  la_spec[[i]] <-  la_var[[1]][,2]
  la_unit[[i]] <-  la_var[[1]][,3]
  }

la_phlyo <- unlist(la_phlyo)
la_spec <- unlist(la_spec)
la_unit <- unlist(la_unit)

la_prop_phlyo <- la_phlyo/(la_phlyo + la_spec + la_unit)
la_prop_spec  <- la_spec/(la_phlyo + la_spec + la_unit)
la_prop_residuals  <- la_unit/(la_phlyo + la_spec + la_unit)

hdr(la_prop_phlyo)$mode
```

```
## [1] 0.8937611
```

```
hdr(la_prop_spec)$mode
```

```
## [1] 0.09525258
```

```
hdr(la_prop_residuals)$mode
```

```
## [1] 0.01032276
```

Next calculate the residuals from the allometric model for Age at first reproduction.

```
La_resids <- mul_resids(mul_output = La_models,
                        mul_data = pop_multree,
                        Y_data_col = c("life_time_La")
 )
```

Mean Reporductive Rate

```
repo_models <- read.mulTree("mean_repo_rate_run")
summary(repo_models)
```

```
##                         Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                      -0.3822022    -1.9569009   -0.8931370
## mass_g                           -0.3269168    -0.5526092   -0.4028669
## matrix_size                      -0.2602243    -0.3864657   -0.3039552
## phylogenetic.variance             1.3402815     0.4631826    0.9726178
## residual.variance                 0.3854063     0.2280886    0.3277340
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)                0.0856983     1.02958316
## mass_g                    -0.2466121    -0.09573137
## matrix_size               -0.2177105    -0.13615055
## phylogenetic.variance      1.8029621     2.99722645
## residual.variance          0.4586991     0.61797900
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
repo_var <- read.mulTree("mean_repo_rate_run", extract = "VCV")

repo_phlyo <- list()
repo_spec <- list()
repo_unit <- list()

for(i in 1:length(names(repo_var))){

  repo_phlyo[[i]] <-  repo_var[[1]][,1]
  repo_spec[[i]] <-  repo_var[[1]][,2]
  repo_unit[[i]] <-  repo_var[[1]][,3]
  }

repo_phlyo <- unlist(repo_phlyo)
repo_spec <- unlist(repo_spec)
repo_unit <- unlist(repo_unit)

repo_prop_phlyo <- repo_phlyo/(repo_phlyo + repo_spec + repo_unit)
repo_prop_spec  <- repo_spec/(repo_phlyo + repo_spec + repo_unit)
repo_prop_residuals  <- repo_unit/(repo_phlyo + repo_spec + repo_unit)
```

```
hdr(repo_prop_phlyo)$mode
```

```
## [1] 0.7496981
```

```
hdr(repo_prop_spec)$mode
```

```
## [1] 0.1708962
```

```
hdr(repo_prop_residuals)$mode
```

```
## [1] 0.06139931
```

Next calculate the residuals from the allometric model for mean reporductive rate.

```
repo_resids <- mul_resids(mul_output = repo_models,
                          mul_data = pop_multree,
                          Y_data_col = c("mean_repo_rate_stable_state")
 )
```

Mean Reporductive Rate not at the stabel state distribution

```
repo_nst_models <- read.mulTree("mean_repo_rate_nst_run")
summary(repo_nst_models)
```

```
##                         Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                      0.20589691    -1.9870390  -0.50271732
## mass_g                          -0.01382265    -0.2666632  -0.09757447
## matrix_size                     -0.18379386    -0.2820750  -0.21757480
## phylogenetic.variance            3.02524117     1.3125030   2.25240081
## residual.variance                0.35199822     0.1497043   0.27276714
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)              0.90937531     2.33620025
## mass_g                   0.07769653     0.24786217
## matrix_size             -0.15030139    -0.08606111
## phylogenetic.variance    4.00091615     6.52083699
## residual.variance        0.42884271     0.59876957
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
repo_nst_var <- read.mulTree("mean_repo_rate_nst_run", extract = "VCV")

repo_nst_phlyo <- list()
repo_nst_spec <- list()
repo_nst_unit <- list()

for(i in 1:length(names(repo_nst_var))){

  repo_nst_phlyo[[i]] <-  repo_nst_var[[1]][,1]
  repo_nst_spec[[i]] <-  repo_nst_var[[1]][,2]
  repo_nst_unit[[i]] <-  repo_nst_var[[1]][,3]
  }

repo_nst_phlyo <- unlist(repo_nst_phlyo)
```

```
repo_nst_spec <- unlist(repo_nst_spec)
repo_nst_unit <- unlist(repo_nst_unit)

repo_nst_prop_phlyo <- repo_nst_phlyo/(repo_nst_phlyo + repo_nst_spec + repo_nst_unit)
repo_nst_prop_spec  <- repo_nst_spec/(repo_nst_phlyo + repo_nst_spec + repo_nst_unit)
repo_nst_prop_residuals  <- repo_nst_unit/(repo_nst_phlyo + repo_nst_spec + repo_nst_unit)

hdr(repo_nst_prop_phlyo)$mode
```

```
## [1] 0.9288788
```

```
hdr(repo_nst_prop_spec)$mode
```

```
## [1] 0.08166688
```

```
hdr(repo_nst_prop_residuals)$mode
```

```
## [1] 0.009818453
```

Next calculate the residuals from the allometric model for mean reporductive rate not at the stable state.

```
repo_nst_resids <- mul_resids(mul_output = repo_nst_models,
                        mul_data = pop_multree,
                        Y_data_col = c("mean_repo_rate")
 )
```

Standard deviation of mxlx

```
mxlxsd_models <- read.mulTree("mxlxsd_logged_10_run")
summary(mxlxsd_models)
```

```
##                         Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                      -0.1638412   -1.65326399  -0.63005810
## mass_g                           -0.1924517   -0.43282306  -0.27444388
## matrix_size                       0.1277919   -0.03112925   0.07252077
## phylogenetic.variance             1.1090746    0.30291582   0.75205057
## residual.variance                 0.3781816    0.16402737   0.29921999
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)                0.2940226     1.21802454
## mass_g                    -0.1085893     0.05379125
## matrix_size                0.1801869     0.28327719
## phylogenetic.variance      1.5470674     2.85798056
## residual.variance          0.4665230     0.66387157
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
mxlxsd_var <- read.mulTree("mxlxsd_logged_10_run", extract = "VCV")

mxlxsd_phlyo <- list()
mxlxsd_spec <- list()
mxlxsd_unit <- list()

for(i in 1:length(names(mxlxsd_var))){
```

```
  mxlxsd_phlyo[[i]] <-  mxlxsd_var[[1]][,1]
  mxlxsd_spec[[i]] <-  mxlxsd_var[[1]][,2]
  mxlxsd_unit[[i]] <-  mxlxsd_var[[1]][,3]
  }

mxlxsd_phlyo <- unlist(mxlxsd_phlyo)
mxlxsd_spec <- unlist(mxlxsd_spec)
mxlxsd_unit <- unlist(mxlxsd_unit)

mxlxsd_prop_phlyo <- mxlxsd_phlyo/(mxlxsd_phlyo + mxlxsd_spec + mxlxsd_unit)
mxlxsd_prop_spec  <- mxlxsd_spec/(mxlxsd_phlyo + mxlxsd_spec + mxlxsd_unit)
mxlxsd_prop_residual  <- mxlxsd_unit/(mxlxsd_phlyo + mxlxsd_spec + mxlxsd_unit)

hdr(mxlxsd_prop_phlyo)$mode
```

```
## [1] 0.6703992
```

```
hdr(mxlxsd_prop_spec)$mode
```

```
## [1] 0.169454
```

```
hdr(mxlxsd_prop_residual)$mode
```

```
## [1] 0.1641695
```

Next calculate the residuals from the allometric model for mxlxsd

```
mxlxsd_resids <- mul_resids(mul_output = mxlxsd_models,
                        mul_data = pop_multree,
                        Y_data_col = c("mxlxsd")
 )
```

Generation Time

```
gen_time_models <- read.mulTree("gen_time_run")
summary(gen_time_models)
```

```
##                       Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                    0.60222796  -1.421124979  -0.07963985
## mass_g                         0.58944661   0.383174918   0.51837006
## matrix_size                   -0.03399182  -0.121444880  -0.06351619
## phylogenetic.variance          3.02574222   1.685129644   2.50955898
## residual.variance              0.07259825  -0.001586581   0.04181414
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)           1.268524941     2.61020858
## mass_g                0.658241758     0.79282154
## matrix_size          -0.002309643     0.05672409
## phylogenetic.variance 3.695115427     5.18097751
## residual.variance     0.105108941     0.17293323
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
gen_time_var <- read.mulTree("gen_time_run", extract = "VCV")

gen_time_phlyo <- list()
gen_time_spec <- list()
gen_time_unit <- list()

for(i in 1:length(names(gen_time_var))){

  gen_time_phlyo[[i]] <-  gen_time_var[[1]][,1]
  gen_time_spec[[i]] <-  gen_time_var[[1]][,2]
  gen_time_unit[[i]] <-  gen_time_var[[1]][,3]
  }

gen_time_phlyo <- unlist(gen_time_phlyo)
gen_time_spec <- unlist(gen_time_spec)
gen_time_unit <- unlist(gen_time_unit)

gen_time_prop_phlyo <- gen_time_phlyo/(gen_time_phlyo + gen_time_spec + gen_time_unit)
gen_time_prop_spec  <- gen_time_spec/(gen_time_phlyo + gen_time_spec + gen_time_unit)
gen_time_prop_residual  <- gen_time_unit/(gen_time_phlyo + gen_time_spec + gen_time_unit)

hdr(gen_time_prop_phlyo)$mode
```

```
## [1] 0.9692614
```

```
hdr(gen_time_prop_spec)$mode
```

```
## [1] 0.01628064
```

```
hdr(gen_time_prop_residual)$mode
```

```
## [1] 0.02179253
```

Next calculate the residuals from the allometric model for generation time

```
gen_time_resids <- mul_resids(mul_output = gen_time_models,
                        mul_data = pop_multree,
                        Y_data_col = c("gen_time")
 )
```

Life expectancy conditional on reaching sexual maturity

```
M_rep_lif_exp_models <- read.mulTree("M_rep_lif_exp_run")
summary(M_rep_lif_exp_models)
```

```
##                       Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                     0.31566955  -1.439073324  -0.26029085
## mass_g                          0.58795467   0.386374002   0.51712022
## matrix_size                     0.05466295  -0.049173230   0.02086479
## phylogenetic.variance           2.25003343   1.100327992   1.78279941
## residual.variance               0.07460030  -0.005803252   0.03870317
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)             0.90913758      2.0895158
## mass_g                  0.65293431      0.7840524
```

```
## matrix_size             0.09396444      0.1641853
## phylogenetic.variance   2.78689181      4.0508399
## residual.variance       0.11422750      0.1933473
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
M_rep_lif_exp_var <- read.mulTree("M_rep_lif_exp_run", extract = "VCV")

M_rep_lif_exp_phlyo <- list()
M_rep_lif_exp_spec <- list()
M_rep_lif_exp_unit <- list()

for(i in 1:length(names(M_rep_lif_exp_var))){

  M_rep_lif_exp_phlyo[[i]] <-  M_rep_lif_exp_var[[1]][,1]
  M_rep_lif_exp_spec[[i]] <-  M_rep_lif_exp_var[[1]][,2]
  M_rep_lif_exp_unit[[i]] <-  M_rep_lif_exp_var[[1]][,3]
  }

M_rep_lif_exp_phlyo <- unlist(M_rep_lif_exp_phlyo)
M_rep_lif_exp_spec <- unlist(M_rep_lif_exp_spec)
M_rep_lif_exp_unit <- unlist(M_rep_lif_exp_unit)

M_rep_lif_exp_prop_phlyo <- M_rep_lif_exp_phlyo/(M_rep_lif_exp_phlyo + M_rep_lif_exp_spec + M_rep_lif_ex
M_rep_lif_exp_prop_spec  <- M_rep_lif_exp_spec/(M_rep_lif_exp_phlyo + M_rep_lif_exp_spec + M_rep_lif_exp
M_rep_lif_exp_prop_residuals  <- M_rep_lif_exp_unit/(M_rep_lif_exp_phlyo + M_rep_lif_exp_spec + M_rep_li

hdr(M_rep_lif_exp_prop_phlyo)$mode
```

```
## [1] 0.9304621
```

```
hdr(M_rep_lif_exp_prop_spec)$mode
```

```
## [1] 0.01841453
```

```
hdr(M_rep_lif_exp_prop_residuals)$mode
```

```
## [1] 0.05572958
```

Next calculate the residuals from the allometric model for life expectancy conditional on reaching sexual maturity

```
M_rep_lif_exp_resids <- mul_resids(mul_output = M_rep_lif_exp_models,
                        mul_data = pop_multree,
                        Y_data_col = c("M_rep_lif_exp")
 )
```

Gini index

```
gini_models <- read.mulTree("gini_logged_run")

summary(gini_models)
```

```
##                       Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
```

```
## (Intercept)                   -0.42786957    -2.0464308   -0.9689482
## mass_g                        -0.19598964    -0.4244743   -0.2745812
## matrix_size                   -0.09997753    -0.2324764   -0.1452912
## phylogenetic.variance          1.79685251     0.9171813    1.4300579
## residual.variance              0.25795196     0.1276147    0.2086544
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)              0.11276001     1.19790360
## mass_g                  -0.11831778     0.03280006
## matrix_size             -0.05474636     0.03184340
## phylogenetic.variance    2.19374036     3.26168320
## residual.variance        0.31350172     0.44078630
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
gini_var <- read.mulTree("gini_logged_run", extract = "VCV")

gini_phlyo <- list()
gini_spec <- list()
gini_unit <- list()

for(i in 1:length(names(gini_var))){

  gini_phlyo[[i]] <-  gini_var[[1]][,1]
  gini_spec[[i]] <-  gini_var[[1]][,2]
  gini_unit[[i]] <-  gini_var[[1]][,3]
  }

gini_phlyo <- unlist(gini_phlyo)
gini_spec <- unlist(gini_spec)
gini_unit <- unlist(gini_unit)

gini_prop_phlyo <- gini_phlyo/(gini_phlyo + gini_spec + gini_unit)
gini_prop_spec  <- gini_spec/(gini_phlyo + gini_spec + gini_unit)
gini_prop_residuals  <- gini_unit/(gini_phlyo + gini_spec + gini_unit)

hdr(gini_prop_phlyo)$mode
```

```
## [1] 0.7922699
```

```
hdr(gini_prop_spec)$mode
```

```
## [1] 0.1048258
```

```
hdr(gini_prop_residuals)$mode
```

```
## [1] 0.09191186
```

Next calculate the residuals from the allometric model for the gini index

```
gini_resids <- mul_resids(mul_output = gini_models,
                          mul_data = pop_multree,
                          Y_data_col = c("gini")
 )
```

Standard deviation of mortality rates

```
surv_sd_models <- read.mulTree("surv_sd_logged_run")

summary(surv_sd_models)
```

```
##                       Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                     0.2918959   -1.17657840   -0.1883131
## mass_g                          0.2633114    0.03469115    0.1864596
## matrix_size                    -0.3503738   -0.50536667   -0.4040554
## phylogenetic.variance           1.1715798    0.38880294    0.8386642
## residual.variance               0.2432233    0.09920176    0.1882841
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)              0.7439676      1.7092964
## mass_g                   0.3445205      0.5007707
## matrix_size             -0.2995981     -0.1995475
## phylogenetic.variance    1.6285540      2.8397630
## residual.variance        0.3083792      0.4565600
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Now we calculate the proportion of variance between phylogenetic, population and residual variance.

```
surv_sd_var <- read.mulTree("surv_sd_logged_run", extract = "VCV")

surv_sd_phlyo <- list()
surv_sd_spec <- list()
surv_sd_unit <- list()

for(i in 1:length(names(surv_sd_var))){

  surv_sd_phlyo[[i]] <-  surv_sd_var[[1]][,1]
  surv_sd_spec[[i]] <-  surv_sd_var[[1]][,2]
  surv_sd_unit[[i]] <-  surv_sd_var[[1]][,3]
  }

surv_sd_phlyo <- unlist(surv_sd_phlyo)
surv_sd_spec <- unlist(surv_sd_spec)
surv_sd_unit <- unlist(surv_sd_unit)

surv_sd_prop_phlyo <- surv_sd_phlyo/(surv_sd_phlyo + surv_sd_spec + surv_sd_unit)
surv_sd_prop_spec  <- surv_sd_spec/(surv_sd_phlyo + surv_sd_spec + surv_sd_unit)
surv_sd_prop_residuals  <- surv_sd_unit/(surv_sd_phlyo + surv_sd_spec + surv_sd_unit)

hdr(surv_sd_prop_phlyo)$mode
```

```
## [1] 0.7197843
```

```
hdr(surv_sd_prop_spec)$mode
```

```
## [1] 0.1072864
```

```
hdr(surv_sd_prop_residuals)$mode
```

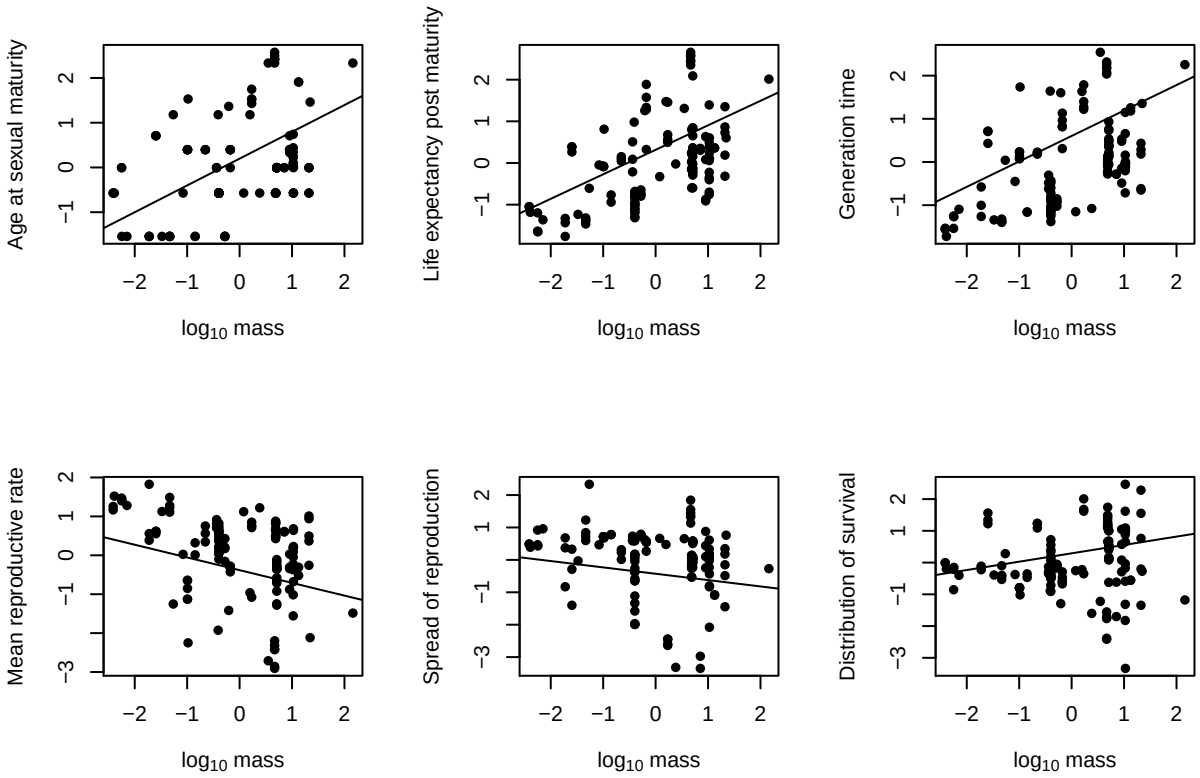
```
## [1] 0.2246864
```

Next calculate the residuals from the allometric model for the standard deviation

```
surv_sd_resids <- mul_resids(mul_output = surv_sd_models,
                         mul_data = pop_multree,
                         Y_data_col = c("surv_sd")
 )
```

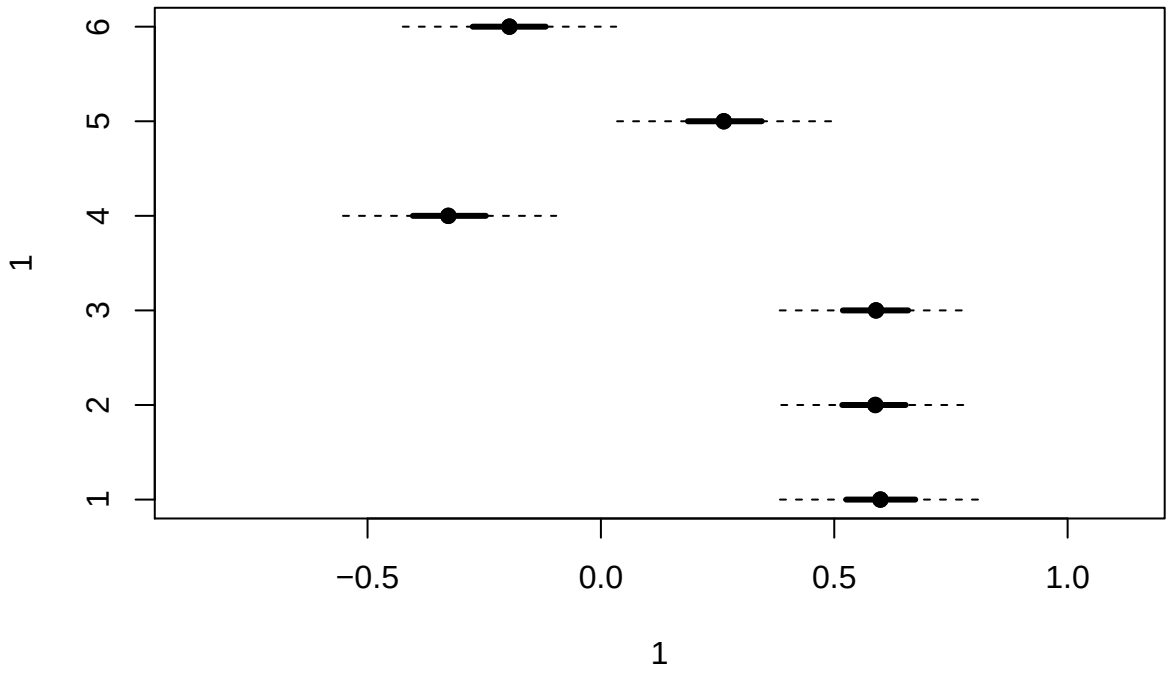
Using the outputs for the life history metrics lets plot them out.

```
par(mfrow=c(2,3))

##la
plot(pop_multree$data$life_time_La ~ pop_multree$data$mass_g, pch = 16,
     xlab = expression('log'[10]*" mass"),
     ylab = "Age at sexual maturity")
abline(summary(La_models)[1],summary(La_models)[2])

#M_rep_lif_exp
plot(pop_multree$data$M_rep_lif_exp ~ pop_multree$data$mass_g, pch = 16,
          xlab = expression('log'[10]*" mass"),
          ylab = "Life expectancy post maturity")
abline(summary(M_rep_lif_exp_models)[1],summary(M_rep_lif_exp_models)[2])

#generation time
plot(pop_multree$data$gen_time ~ pop_multree$data$mass_g, pch = 16,
          xlab = expression('log'[10]*" mass"),
          ylab = "Generation time")
abline(summary(gen_time_models)[1],summary(gen_time_models)[2])


##mean repo rate
plot(pop_multree$data$mean_repo_rate_stable_state ~ pop_multree$data$mass_g, pch = 16,
          xlab = expression('log'[10]*" mass"),
          ylab = "Mean reproductive rate")
abline(summary(repo_models)[1],summary(repo_models)[2])



#Gini
plot(pop_multree$data$gini ~ pop_multree$data$mass_g, pch = 16,
          xlab = expression('log'[10]*" mass"),
          ylab = "Spread of reproduction")
abline(summary(gini_models)[1],summary(gini_models)[2])

#SD of survival
plot(pop_multree$data$surv_sd ~ pop_multree$data$mass_g, pch = 16,
     xlab = expression('log'[10]*" mass"),
     ylab = "Distribution of survival")
abline(summary(surv_sd_models)[1],summary(surv_sd_models)[2])
```



Lets also plot out the model slope cooficents into a table.

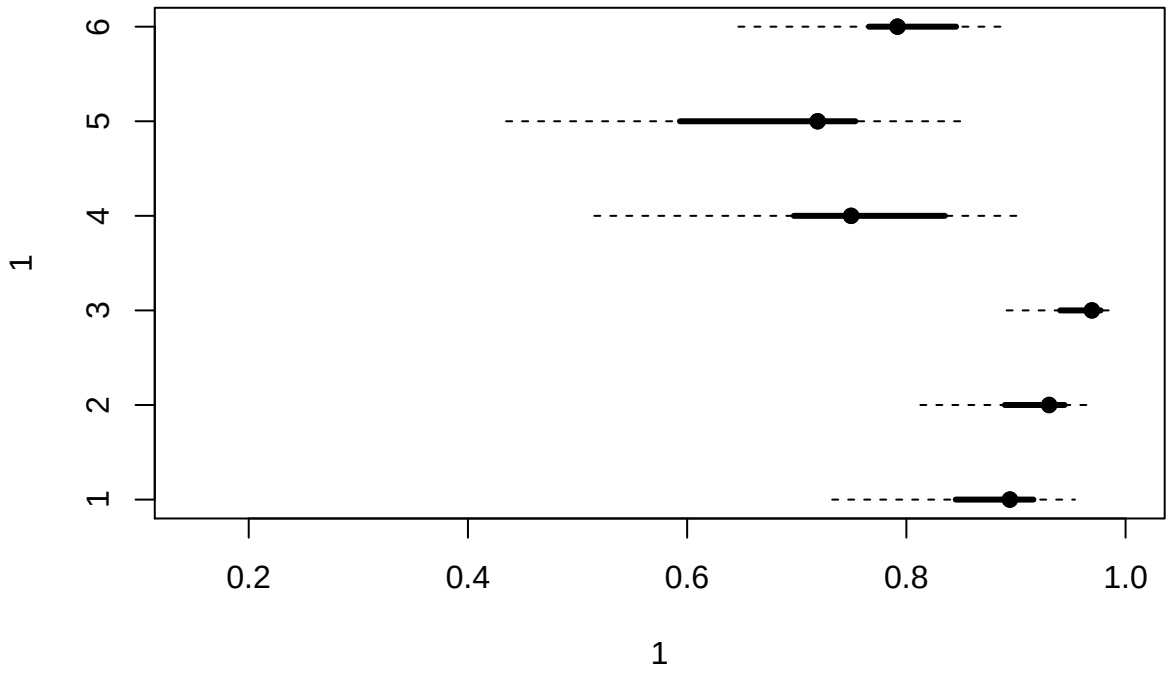
```
##Allometric scaling
#pdf("scaling_bar_plots.pdf")
scaling_list <- list( La_B = La_models$mass_g,
                      Sur_B = M_rep_lif_exp_models$mass_g,
                      T_B = gen_time_models$mass_g,
                      Repo_B = repo_models$mass_g,
                      life_shape_B = surv_sd_models$mass_g,
                      gini_B = gini_models$mass_g
                      )
MultiDisPlot(scaling_list)
```




```
#dev.off()
```

Lets also plot out the phylogentic signals into a table.

```
#pdf("phy_var_plots.pdf")
phy_var_list <- list( La_B = la_prop_phlyo,
                      Sur_B = M_rep_lif_exp_prop_phlyo,
                      T_B = gen_time_prop_phlyo,
                      Repo_B = repo_prop_phlyo,
                      life_shape_B = surv_sd_prop_phlyo,
                      gini_B = gini_prop_phlyo
                      )
MultiDisPlot(phy_var_list)
```

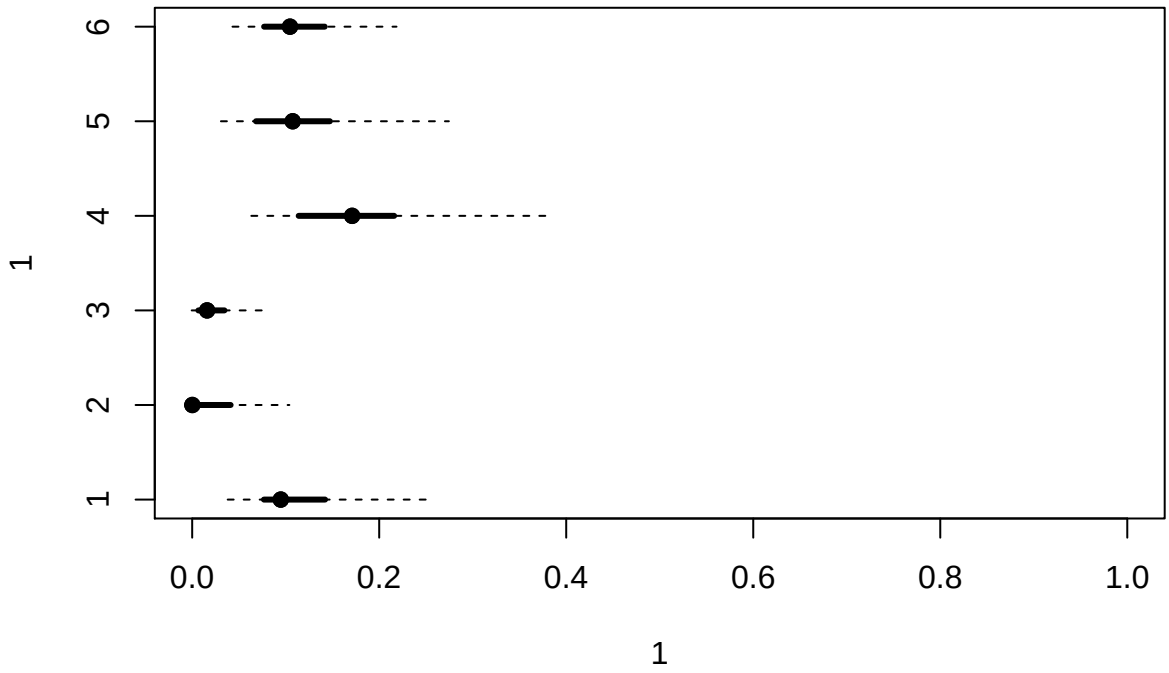



```
#dev.off()
```

Lets also plot out the population level variances into a table.

```
#pdf("species_var_plots.pdf")
species_var_list <- list(
                      La_B = la_prop_spec,
                      Sur_B = M_rep_lif_exp_prop_spec,
                      T_B = gen_time_prop_spec,
                      Repo_B = repo_prop_spec,
                      life_shape_B = surv_sd_prop_spec,
                      gini_B = gini_prop_spec
                      )
MultiDisPlot(species_var_list, xlim = c(0,1))
```

```
## Warning in if (xlim == "auto") {: the condition has length > 1 and only the
## first element will be used
```




```
#dev.off()
```

Now lets create a dataset of these residuals for each PCA analysis. This includes, the main analysis, the analysis using mean reporductive rate with the population not at its stable state distribution, the analysis using the standard deviation of mxlx curve as a measure of the Gini index and the analysis with generation time removed.

```
#Main PCA dataset
predicted_data <- data.frame(
                        SD_mort = surv_sd_resids,
                        La_r = La_resids,
                        gen_r = gen_time_resids,
                        M_repo = repo_resids,
                        M_suv = M_rep_lif_exp_resids,
                        gini_r = gini_resids
                        )

#PCA dataset using mean reporductive rate with the population not at its stable state distribution.
predicted_data_M_repo_nst <- data.frame(
                         SD_mort = surv_sd_resids,
                        La_r = La_resids,
                        gen_r = gen_time_resids,
                        M_repo_nst = repo_nst_resids,
                        M_suv = M_rep_lif_exp_resids,
                        gini_r = gini_resids
                        )

#PCA dataset using the standard deviation of the mxlx curve
predicted_data_mxlxsd <- data.frame(
                        SD_mort = surv_sd_resids,
                        La_r = La_resids,
                        gen_r = gen_time_resids,
                        M_repo = repo_resids,
```

```
                        M_suv = M_rep_lif_exp_resids,
                        mxlxsd = mxlxsd_resids
                        )
#PCA dataset without generation time.
predicted_data_noT <- data.frame(
                         SD_mort = surv_sd_resids,
                        La_r = La_resids,
                        M_repo = repo_resids,
                        M_suv = M_rep_lif_exp_resids,
                        gini = gini_resids
                        )
```

PCA

Now we run a PCA for each of the datasets of residuals from the life history allometric models.

```
pca_res <- prcomp(predicted_data)

pca_nst <- prcomp(predicted_data_M_repo_nst)

pca_mxlxsd <- prcomp(predicted_data_mxlxsd)

pca_noT <- prcomp(predicted_data_noT)
```

Next we use Horn's Parallel Analysis of Principal Components to test for the number of axis to retain.

```
horn_res <- paran(predicted_data)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 180 iterations, using the mean estimate
##
## ----------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## ----------------------------------------------------
## 1           2.904159    3.195047      0.290887
## 2           1.235338    1.382979      0.147641
## ----------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

```
horn_nst <- paran(predicted_data_M_repo_nst)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
```

```
## Results of Horn's Parallel Analysis for component retention
## 180 iterations, using the mean estimate
##
## ---------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## ---------------------------------------------------
## 1           2.599748    2.890173      0.290424
## 2           1.380244    1.530890      0.150646
## ---------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

```
horn_mxlx <- paran(predicted_data_mxlxsd)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 180 iterations, using the mean estimate
##
## ---------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## ---------------------------------------------------
## 1           2.765163    3.063570      0.298407
## 2           1.153609    1.303107      0.149498
## 3           1.175911    1.215409      0.039498
## ---------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (3 components retained)
```

```
horn_noT <- paran(predicted_data_noT)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 150 iterations, using the mean estimate
##
## ---------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## ---------------------------------------------------
## 1           2.197126    2.448004      0.250878
## 2           1.107872    1.204538      0.096666
## ---------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
```

```
## (2 components retained)
```