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 = ~animal + species)
```

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)                     0.19747134   -1.50849226  -0.36692148
## mass_g                          0.59885721    0.38364820   0.52569656
## matrix_size                     0.06146852   -0.01051303   0.03715236
## phylogenetic.variance           2.03307751    0.92585738   1.57129554
## residual.variance               0.29103448    0.17437952   0.24779145
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)             0.76238085      1.9104997
## mass_g                  0.67352076      0.8166299
## matrix_size             0.08761325      0.1367536
## phylogenetic.variance   2.54291734      3.8375481
## residual.variance       0.34484428      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)

#Phylogenetic signal
hdr(la_prop_phlyo)$mode
```

```
## [1] 0.8936751
```

```
#Population level variance
hdr(la_prop_spec)$mode
```

```
## [1] 0.09491442
```

```
#Residual term
hdr(la_prop_residuals)$mode
```

```
## [1] 0.01031528
```

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)                      -0.3821796    -1.9569241   -0.8931401
## mass_g                           -0.3269163    -0.5526083   -0.4028667
## matrix_size                      -0.2602267    -0.3864657   -0.3039558
## phylogenetic.variance             1.3403244     0.4631857    0.9726163
## residual.variance                 0.3852503     0.2280839    0.3277378
##                        upper.CI(75) upper.CI(97.5)
## (Intercept)               0.0856870     1.02958316
## mass_g                   -0.2466111    -0.09573132
## matrix_size              -0.2177105    -0.13615120
## phylogenetic.variance     1.8029584     2.99722645
## residual.variance         0.4587044     0.61796005
## 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)

#Phylogenetic signal
hdr(repo_prop_phlyo)$mode
```

```
## [1] 0.7498984
```

```
#Population level variance
hdr(repo_prop_spec)$mode
```

```
## [1] 0.171035
```

```
#Residual term
hdr(repo_prop_residuals)$mode
```

```
## [1] 0.06143344
```

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)                    0.20587434    -1.9870391  -0.50269567
## mass_g                        -0.01381948    -0.2666529  -0.09755688
## matrix_size                   -0.18379335    -0.2820732  -0.21757472
## phylogenetic.variance          3.02741680     1.3126973   2.25227288
## residual.variance              0.35185781     0.1496883   0.27277345
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)             0.90939664      2.3361914
## mass_g                  0.07770733      0.2478651
## matrix_size            -0.15030098     -0.0860611
## phylogenetic.variance   4.00080256      6.5209695
## residual.variance       0.42884686      0.5987686
## 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)


#Phylogenetic signal
hdr(repo_nst_prop_phlyo)$mode
```

```
## [1] 0.9289587
```

```
#Population level variance
hdr(repo_nst_prop_spec)$mode
```

```
## [1] 0.08177648
```

```
#Residual term
hdr(repo_nst_prop_residuals)$mode
```

```
## [1] 0.009848131
```

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)                     -0.1635107   -1.65326374  -0.63001884
## mass_g                          -0.1924576   -0.43282306  -0.27444772
## matrix_size                      0.1277956   -0.03112926   0.07252093
## phylogenetic.variance            1.1087434    0.30288752   0.75206045
## residual.variance                0.3783354    0.16403320   0.29920713
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)              0.2940519     1.21802905
## mass_g                  -0.1085906     0.05379147
## matrix_size              0.1801876     0.28327683
## phylogenetic.variance    1.5470773     2.85798056
## residual.variance        0.4665128     0.66387366
## 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)


#Phylogenetic signal
hdr(mxlxsd_prop_phlyo)$mode
```

```
## [1] 0.6702649
```

```
#Population level variance
hdr(mxlxsd_prop_spec)$mode
```

```
## [1] 0.1697088
```

```
#Residual term
hdr(mxlxsd_prop_residual)$mode
```

```
## [1] 0.1641848
```

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)                    0.60237913  -1.421124979  -0.07969169
## mass_g                         0.58946067   0.383169700   0.51837197
## matrix_size                   -0.03398869  -0.121445918  -0.06351646
```

```
## phylogenetic.variance          3.02776553   1.685256044   2.50942926
## residual.variance              0.07255637  -0.001590761   0.04181470
##                        upper.CI(75) upper.CI(97.5)
## (Intercept)             1.268474269     2.61018832
## mass_g                  0.658243780     0.79282154
## matrix_size            -0.002310125     0.05672399
## phylogenetic.variance   3.694971664     5.18105881
## residual.variance       0.105109278     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)


#Phylogenetic signal
hdr(gen_time_prop_phlyo)$mode
```

```
## [1] 0.9690505
```

```
#Population level variance
hdr(gen_time_prop_spec)$mode
```

```
## [1] 0.01624795
```

```
#Residual term
hdr(gen_time_prop_residual)$mode
```

```
## [1] 0.02174675
```

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)                    0.31580239  -1.439073835  -0.26028814
## mass_g                         0.58795350   0.386374260   0.51712046
## matrix_size                    0.05466408  -0.049172640   0.02086333
## phylogenetic.variance          2.24980397   1.100305459   1.78280214
## residual.variance              0.07502929  -0.005531941   0.03867796
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)              0.9091395      2.0895027
## mass_g                   0.6529346      0.7840525
## matrix_size              0.0939637      0.1641865
## phylogenetic.variance    2.7868984      4.0508399
## residual.variance        0.1141998      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_e
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


#Phylogenetic signal
hdr(M_rep_lif_exp_prop_phlyo)$mode
```

```
## [1] 0.9303974
```

```
#Population level variance
hdr(M_rep_lif_exp_prop_spec)$mode
```

```
## [1] 0.01849731
```

```
#Residual term
hdr(M_rep_lif_exp_prop_residuals)$mode
```

```
## [1] 0.05573421
```

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)                    -0.42794181    -2.0464228   -0.9689459
## mass_g                         -0.19599691    -0.4244735   -0.2745817
## matrix_size                    -0.09997641    -0.2324744   -0.1452908
## phylogenetic.variance           1.79736608     0.9172384    1.4300211
## residual.variance               0.25793064     0.1276131    0.2086544
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)             0.11276323     1.19790403
## mass_g                 -0.11831885     0.03280006
## matrix_size            -0.05474589     0.03184347
## phylogenetic.variance   2.19371282     3.26168320
## residual.variance       0.31350150     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)
```

```
#Phylogenetic signal
hdr(gini_prop_phlyo)$mode
```

```
## [1] 0.7922173
```

```
#Population level variance
hdr(gini_prop_spec)$mode
```

```
## [1] 0.1047816
```

```
#Residual term
hdr(gini_prop_residuals)$mode
```

```
## [1] 0.09193673
```

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)                       0.2921592   -1.17657819   -0.1883418
## mass_g                            0.2633087    0.03469292    0.1864593
## matrix_size                      -0.3503796   -0.50536667   -0.4040569
## phylogenetic.variance             1.1640853    0.38884786    0.8386304
## residual.variance                 0.2433491    0.09921485    0.1882762
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)                0.7439494      1.7093005
## mass_g                     0.3445201      0.5007707
## matrix_size               -0.2995992     -0.1995487
## phylogenetic.variance      1.6285460      2.8398280
## residual.variance          0.3083719      0.4565608
## 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)


#Phylogenetic signal
hdr(surv_sd_prop_phlyo)$mode
```

```
## [1] 0.7196334
```

```
#Population level variance
hdr(surv_sd_prop_spec)$mode
```

```
## [1] 0.107074
```

```
#Residual term
hdr(surv_sd_prop_residuals)$mode
```

```
## [1] 0.2247204
```

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

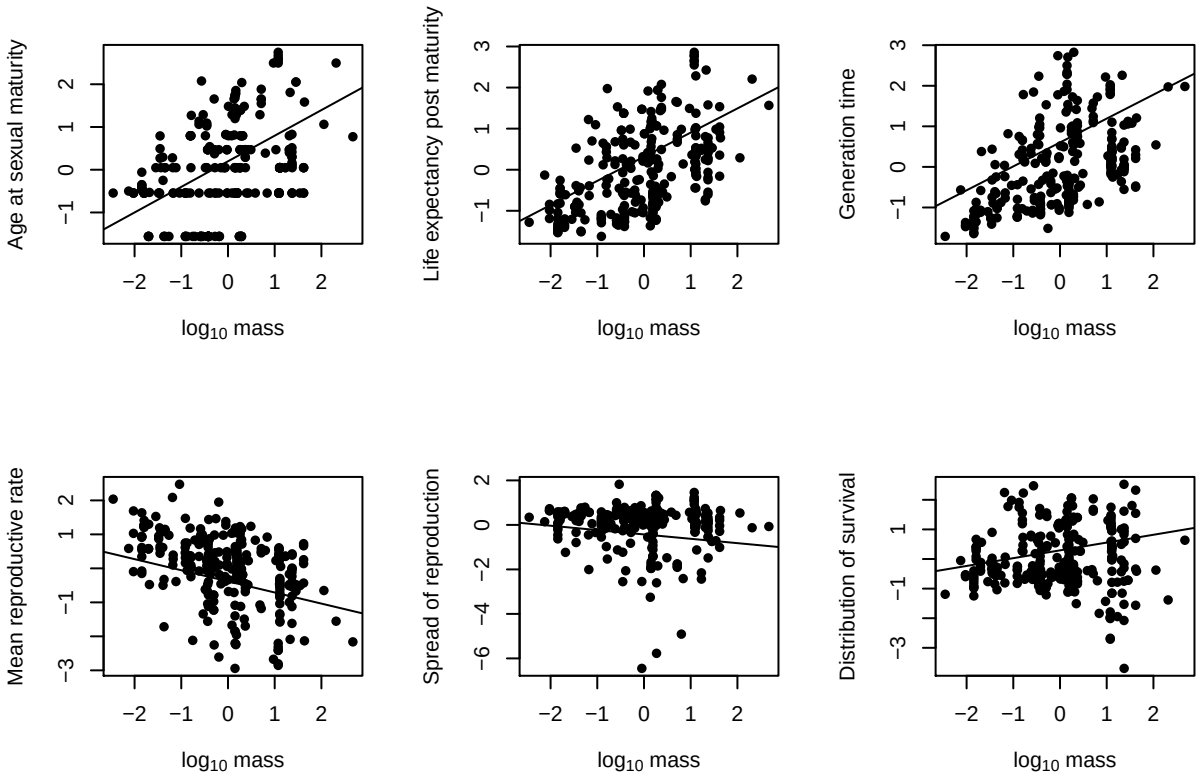
```
surv_sd_resids <- mul_resids(mul_output = surv_sd_models,
                         mul_data = pop_multree,
                         Y_data_col = c("surv_sd")
 )
```

Plotting allometry model output

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])
```

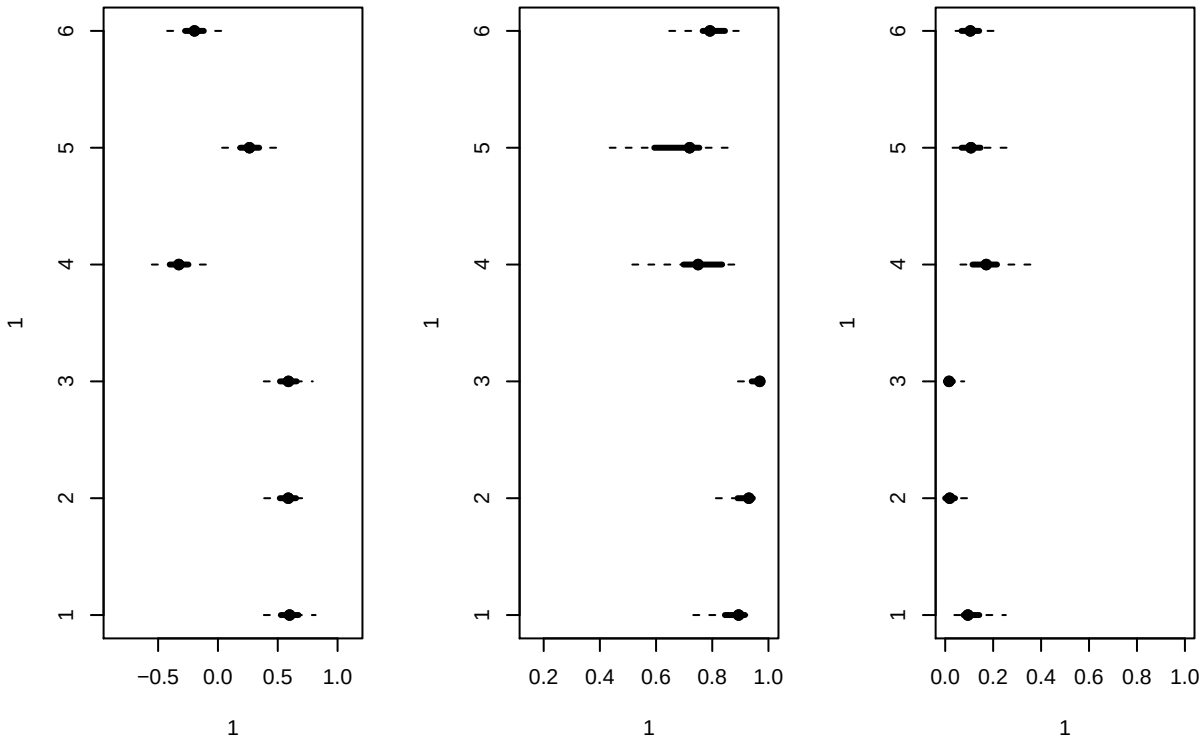


Lets also plot out the model slope coefficients into a table.

```
##Allometric scaling

par(mfrow=c(1,3))

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)


##phylogentic signals
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)


##population level variance
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
```



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. We use Horn's Parallel Analysis of Principal Components to test for the number of axis to retain. Note that for ease of interpretation the sign of some axis was reversed in the corresponding supplementary table 3 so that the fast slow axis alway went from fast on the left to slow on the right.

First the main analysis

```
pca_res <- prcomp(predicted_data)
pca_res
```

```
## Standard deviations (1, .., p=6):
## [1] 1.5214900 1.1418273 0.8361469 0.6983264 0.4966644 0.2823536
##
## Rotation (n x k) = (6 x 6):
##                PC1        PC2         PC3         PC4         PC5
## SD_mort -0.3373610 -0.5056419 -0.64858897 -0.30533698  0.30602653
## La_r    -0.5012398  0.2074782  0.12294518  0.59463208  0.54472281
## gen_r   -0.5557434  0.1251703  0.01911192 -0.14338153 -0.11159244
## M_repo   0.1458494 -0.5755156 -0.14330980  0.69111082 -0.26505288
## M_suv   -0.4812860  0.1440881 -0.14512090  0.09118089 -0.72576868
## gini_r   0.2705314  0.5776088 -0.72267129  0.21604545  0.01336949
##                PC6
## SD_mort -0.1515365
## La_r    -0.2007131
## gen_r    0.8013147
## M_repo   0.2812202
## M_suv   -0.4375936
## gini_r   0.1551539
```

```
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           2.601002    2.804730      0.203727
## 2           1.380085    1.481356      0.101271
## ----------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

Next the analysis with reporductive rate with the population not at the stable state distribution,

```
pca_nst <- prcomp(predicted_data_M_repo_nst)

pca_nst
```

```
## Standard deviations (1, .., p=6):
## [1] 1.5247503 1.1510163 0.8861361 0.7921989 0.4446327 0.3047726
##
## Rotation (n x k) = (6 x 6):
##                   PC1        PC2          PC3        PC4         PC5
## SD_mort    -0.3812794 -0.2234415  0.713723340 -0.4832885  0.21567840
## La_r       -0.4898175  0.1879261 -0.515947355 -0.1201502  0.56296273
## gen_r      -0.5181094  0.2802035  0.035606085  0.1384867  0.02155379
## M_repo_nst -0.1788388 -0.6792153 -0.455790782 -0.4051659 -0.29258542
## M_suv      -0.4480816  0.3229370  0.004226023 -0.0225824 -0.74060487
## gini_r      0.3369081  0.5202133 -0.123961409 -0.7537561 -0.04460302
##                   PC6
## SD_mort    -0.1233449
## La_r       -0.3566481
## gen_r       0.7950709
## M_repo_nst  0.2217486
## M_suv      -0.3819831
## gini_r      0.1742611
```

```
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           2.560805    2.759187      0.198382
## 2           1.305770    1.410377      0.104607
## ---------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

Next the analysis with generation time not included.

```
pca_mxlxsd <- prcomp(predicted_data_mxlxsd)

pca_mxlxsd
```

```
## Standard deviations (1, .., p=6):
## [1] 1.5510226 1.1188432 0.9079943 0.6290138 0.4439734 0.2063080
##
## Rotation (n x k) = (6 x 6):
##                PC1         PC2         PC3         PC4        PC5
## SD_mort -0.2541265 -0.58578261 -0.63423830 -0.19751156  0.3849226
## La_r    -0.4827626 -0.11207474  0.40710084  0.59465655  0.4703647
## gen_r   -0.5579208 -0.04688062  0.01038699 -0.04873374 -0.3433282
## M_repo   0.2690637 -0.55268899 -0.09269274  0.58351718 -0.5214535
## M_suv   -0.4406493 -0.24037911  0.30866469 -0.38899450 -0.4261037
## mxlxsd   0.3528700 -0.52806530  0.57274510 -0.33643537  0.2433912
##                 PC6
## SD_mort  0.05332925
## La_r     0.11742655
## gen_r   -0.75244233
## M_repo   0.03378873
## M_suv    0.56558915
## mxlxsd  -0.31010446
```

```
horn_mxlxsd <- 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           2.711875    2.911102      0.199226
## 2           1.333164    1.437719      0.104555
## ---------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

Next the analysis with generation time not included.

```
pca_noT <- prcomp(predicted_data_noT)

pca_noT
```

```
## Standard deviations (1, .., p=5):
## [1] 1.2902907 1.1185509 0.8359158 0.6884132 0.4932431
##
## Rotation (n x k) = (5 x 5):
##                  PC1        PC2        PC3         PC4         PC5
## SD_mort -0.508465012 -0.3638112  0.6467465 -0.33951272 -0.27487234
## La_r    -0.537540340  0.4176595 -0.1433677  0.50111012 -0.51472833
## M_repo   0.001115311 -0.6034195  0.1458530  0.75663818  0.20520538
## M_suv   -0.508048272  0.3288245  0.1320249  0.02472698  0.78467791
## gini     0.440907500  0.4700652  0.7228146  0.24598171 -0.04088118
```

```
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           1.757631    1.934123      0.176491
## 2           1.394326    1.466940      0.072614
## ----------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

PCA plots

Note that for ease of interpretation the sign of some axis was reversed so that the fast slow axis goes from fast on the left to slow on the right.

```
results <- pca_res
results$rotation[,"PC1"] <- -results$rotation[,"PC1"]
results$x[,"PC1"] <- -results$x[,"PC1"]
results$rotation[,"PC2"] <- results$rotation[,"PC2"]
results$x[,"PC2"] <- results$x[,"PC2"]
```

We then set up the colors we will give to each of the taxinommic groups, the color of the PC arrows and the life history symbols

```
result <- results

loadings <- as.data.frame(result$rotation)
```

```
loadings[,"col"]=c("gray50",
                   "gray50",
                   "gray50",
                   "gray50",
                   "gray50",
                   "gray50"
                   )

loadings$LHT=rownames(loadings)


loadings$LHT=c("surv_sd",
               "La",
               "gen_time",
               "mean_repo_rate",
               "M_suv"
               ,"gini_r"
               )

loadings$LHTexpr <-  list(
                         expression(sigma),
                         expression("L"[alpha]),
                         expression("T"),
                         expression(phi),
                         expression(Rep["e"]),
                         expression("G")
                       )
arrowThickness=2.9
sizeArrowLetters=1
scalingArrows=2.5
scalingLetters=2.9

class_match <- vector()
species_match <- vector()
loads_taxa <- data.frame(results$x)


for(i in 1:length(loads_taxa[,1])){

   class_match[i] <- as.vector(pop_multree$data[i,"taxa_name"])
   species_match[i] <- as.vector(pop_multree$data[i,"species"])

   }


#combine the pca results data with the external data
pca_data <- data.frame(loads_taxa,
                       class_match = as.vector(pop_multree$data[,"taxa_name"]),
                       species_match = as.vector(pop_multree$data[,"species"]),
                       PCA_moblist = as.vector(pop_multree$data[,"mode_of_life"]),
                       PCA_met = as.vector(pop_multree$data[,"met_rate"]),
                       PCA_repo = as.vector(pop_multree$data[,"repo_output"]),
                       PCA_iucn = as.vector(pop_multree$data[,"IUCN"]),
```

```
                      PCA_met_type = as.vector(pop_multree$data[,"met_type"]),
                      animal = as.vector(pop_multree$data[,"species"])
                      )



mam_col <- rgb(0,136,170,
               max= 255)

bird_col <- rgb(255,153,85,
                max= 255)

rep_col <- rgb(147,172,147,
               max= 255)

fish_col <- rgb(135,205,222,
                max= 255)

sponge_col <- rgb(211,95,141,
                  max= 255)

coral_col <- rgb(153,85,255,
                 max= 255)

gast_col <- rgb(255,170,238,
                max= 255)

biv_col <- rgb(205,135,222,
               max= 255)

shark_col <- rgb(85,0,212,
                 max= 255)
```

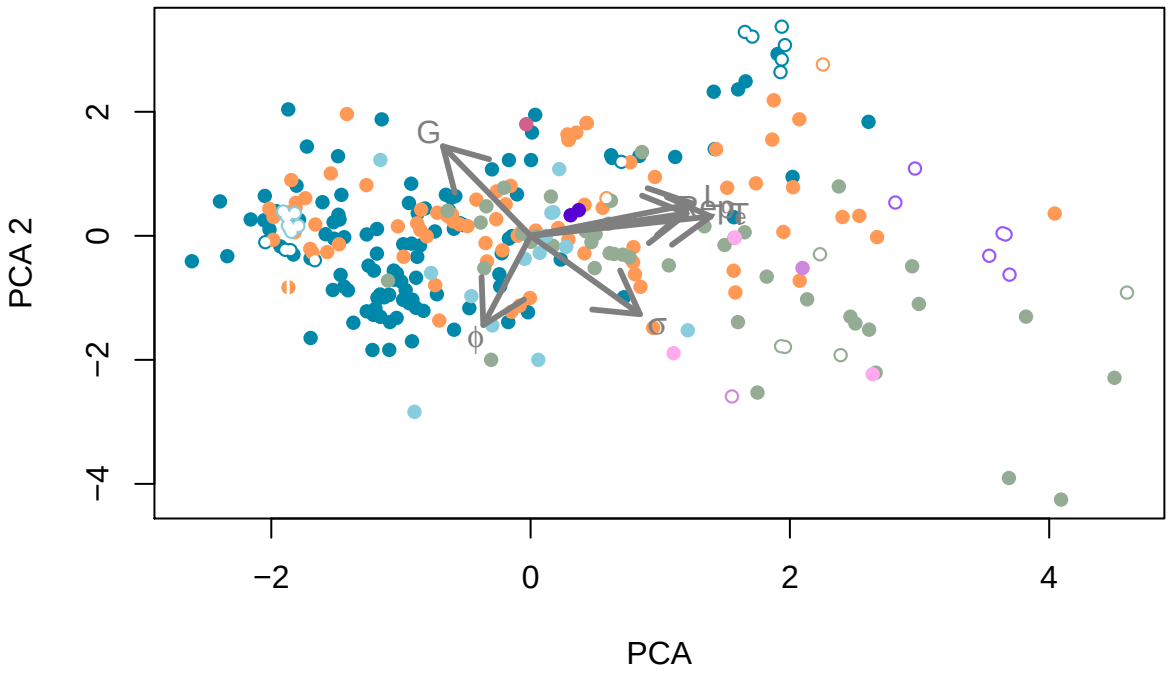
Now lets plot out the two retained axis from our PCA.

```
plot(pca_data[,1],
     pca_data[,2],
     pch=16,
     cex = 0.1,
     col = "white",
     xlab= "PCA",
     ylab= "PCA 2")

points(pca_data[pca_data$class_match == "Mammalia",1],
       pca_data[pca_data$class_match == "Mammalia",2],
       pch=16,
       col = mam_col)

points(pca_data[pca_data$class_match == "Aves",1],
       pca_data[pca_data$class_match == "Aves",2],
       pch=16,
       col = bird_col)

points(pca_data[pca_data$class_match == "Reptilia",1],
```

```
       pca_data[pca_data$class_match == "Reptilia",2],
       pch=16,
       col = rep_col)

points(pca_data[pca_data$class_match == "Actinopterygii",1],
       pca_data[pca_data$class_match == "Actinopterygii",2],
       pch=16,
       col = fish_col)

points(pca_data[pca_data$class_match == "Gastropoda",1],
       pca_data[pca_data$class_match == "Gastropoda",2],
       pch=16,
       col = gast_col)

points(pca_data[pca_data$class_match == "Demospongiae",1],
       pca_data[pca_data$class_match == "Demospongiae",2],
       pch=16,
       col = sponge_col)

points(pca_data[pca_data$class_match == "Anthozoa",1],
       pca_data[pca_data$class_match == "Anthozoa",2],
       pch=16,
       col = coral_col)

points(pca_data[pca_data$class_match == "Bivalvia",1],
       pca_data[pca_data$class_match == "Bivalvia",2],
       pch=16,
       col = biv_col)

points(pca_data[pca_data$class_match == "Elasmobranchii",1],
       pca_data[pca_data$class_match == "Elasmobranchii",2],
       pch=16,
       col = shark_col)

###And lets add Humans
points(pca_data[pca_data$species_match == "Homo_sapiens",1],
       pca_data[pca_data$species_match == "Homo_sapiens",2],
       pch= 16,
       col = "white",
       cex = 0.7)

##and other points
points(pca_data[pca_data$species_match == "Elephas_maximus",1],
       pca_data[pca_data$species_match == "Elephas_maximus",2],
       pch= 16,
       col = "white",
       cex = 0.7)

points(pca_data[pca_data$species_match == "Fulmarus_glacialis",1],
       pca_data[pca_data$species_match == "Fulmarus_glacialis",2],
       pch= 16,
       col = "white",
       cex = 0.7)
```

```
points(pca_data[pca_data$species_match == "Tympanuchus_cupido",1],
       pca_data[pca_data$species_match == "Tympanuchus_cupido",2],
       pch= "T",
       col = "white")

points(pca_data[pca_data$species_match == "Gyps_coprotheres",1],
       pca_data[pca_data$species_match == "Gyps_coprotheres",2],
       pch= 16,
       col = "white",
       cex = 0.7)

points(pca_data[pca_data$species_match == "Crocodylus_johnsoni",1],
       pca_data[pca_data$species_match == "Crocodylus_johnsoni",2],
       pch= 16,
       col = "white",
       cex = 0.7)

points(pca_data[pca_data$species_match == "Urocitellus_armatus",1],
       pca_data[pca_data$species_match == "Urocitellus_armatus",2],
       pch= 16,
       col = "white",
       cex = 0.7)

points(pca_data[pca_data$species_match == "Paramuricea_clavata",1],
       pca_data[pca_data$species_match == "Paramuricea_clavata",2],
       pch= 16,
       col = "white",
       cex = 0.7)

points(pca_data[pca_data$species_match == "Oncorhynchus_tshawytscha",1],
       pca_data[pca_data$species_match == "Oncorhynchus_tshawytscha",2],
       pch= 16,
       col = "white",
       cex = 0.7)

points(pca_data[pca_data$species_match == "Mya_arenaria",1],
       pca_data[pca_data$species_match == "Mya_arenaria",2],
       pch= 16,
       col = "white",
       cex = 0.7)

points(pca_data[pca_data$species_match == "Clemmys_guttata",1],
       pca_data[pca_data$species_match == "Clemmys_guttata",2],
       pch= 16,
       col = "white",
       cex = 0.7)


arrows(x0=0,
       y0=0,
       x1=loadings[,1]*scalingArrows,
       y1=loadings[,2]*scalingArrows,
       col="black",
```

```
       lwd=2)

arrows(x0=0,
       y0=0,
       x1=loadings[,1]*scalingArrows,
       y1=loadings[,2]*scalingArrows,
       col=as.character(loadings$col),
       lwd=arrowThickness)


text(loadings[1,"PC1"]*scalingLetters-.0,
     loadings[1,"PC2"]*scalingLetters,
     loadings$LHTexpr[[1]],
     col = loadings$col[1],
     cex=sizeArrowLetters)

text(loadings[2,"PC1"]*scalingLetters-.0,
     loadings[2,"PC2"]*scalingLetters,
     loadings$LHTexpr[[2]],
     col = loadings$col[2],
     cex=sizeArrowLetters)

text(loadings[3,"PC1"]*scalingLetters+.0,
     loadings[3,"PC2"]*scalingLetters,
     loadings$LHTexpr[[3]],
     col = loadings$col[3],
     cex=sizeArrowLetters)

text(loadings[4,"PC1"]*scalingLetters-.0,
     loadings[4,"PC2"]*scalingLetters,
     loadings$LHTexpr[[4]],
     col = loadings$col[4],
     cex=sizeArrowLetters)

text(loadings[5,"PC1"]*scalingLetters-.0,
     loadings[5,"PC2"]*scalingLetters,
     loadings$LHTexpr[[5]],
     col = loadings$col[5],
     cex=sizeArrowLetters)

text(loadings[6,"PC1"]*scalingLetters+.0,
     loadings[6,"PC2"]*scalingLetters,
     loadings$LHTexpr[[6]],
     col = loadings$col[6],
     cex=sizeArrowLetters)
```



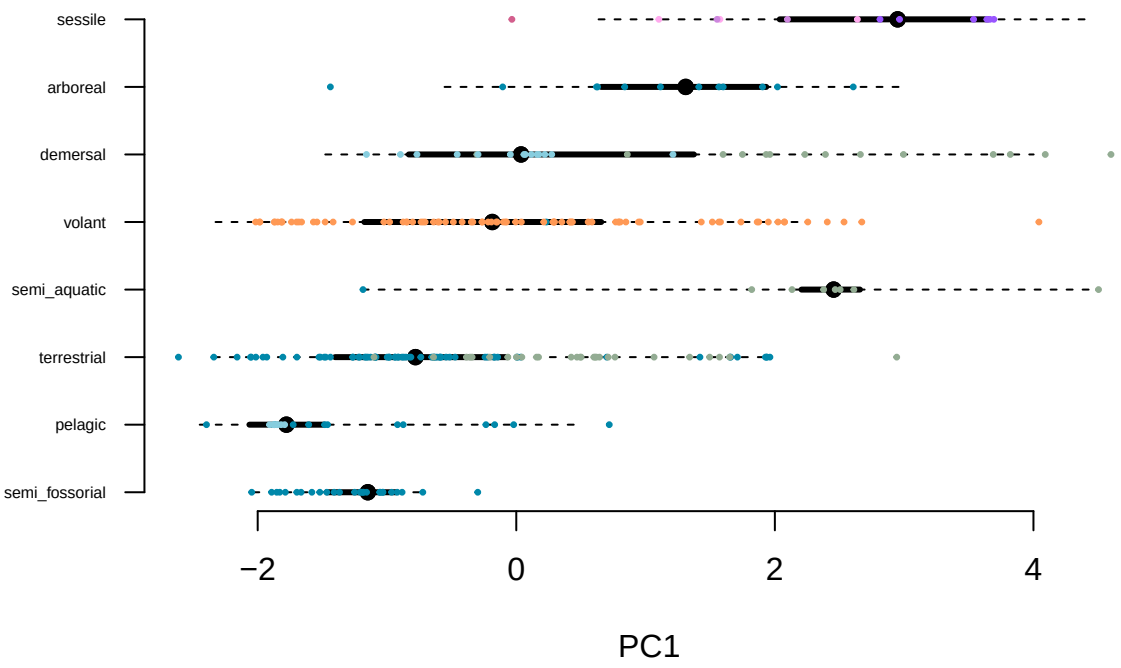
Mode of life analysis analysis

Using PC1 we test how different modes of life are distributed across the portion of life histroy space associated with the fast-slow continuum.

First lets plot out how differnt modes of life are distributed across PC1.

```
#put the mode-of-life data into a formate that can be plotted with the
#MultiDisPlot function

PCA_moblist <- list(semi_fossorial = pca_data[pca_data$PCA_moblist == "semi_fossorial",1],
                    pelagic = pca_data[pca_data$PCA_moblist == "pelagic",1],
                    terrestrial = pca_data[pca_data$PCA_moblist == "terrestrial",1],
                    semi_aquatic = pca_data[pca_data$PCA_moblist == "semi_aquatic",1],
                    volant = pca_data[pca_data$PCA_moblist == "volant",1],
                    demersal = pca_data[pca_data$PCA_moblist == "demersal",1],
                    arboreal = pca_data[pca_data$PCA_moblist == "arboreal",1],
                    sessile = pca_data[pca_data$PCA_moblist == "sessile",1]
                    )


#use the MultiDisPlot function to plot out the groups
#with credibility intervals
MultiDisPlot(PCA_moblist,
             yaxt = "n",
             xlab = "PC1",
             ylab = "",
             bty = "n")

#Plot the mode of life groups on y axis
tick_lables <- names(PCA_moblist)
```

```r
axis(2,
     1:length(tick_lables),
     labels = tick_lables,
     las=1,
     cex.axis = 0.5)

#To plot the species on we need to convert the names into ther associated tick
#mark.
mob_match <- as.vector(pca_data$PCA_moblist)

for(i in 1:(length(tick_lables))){
mob_match[mob_match == tick_lables[i]] <- i
}

mob_match <- as.numeric(mob_match)
pca_data$mob_match <- mob_match


#Plot the points onto the figure.
points(pca_data[pca_data$class_match == "Mammalia","mob_match"] ~ pca_data[pca_data$class_match == "Mamm
       col = mam_col, pch = 20, cex = 0.5)

points(pca_data[pca_data$class_match == "Aves","mob_match"] ~ pca_data[pca_data$class_match == "Aves",1]
       col = bird_col, pch = 20, cex = 0.5)

points(pca_data[pca_data$class_match == "Reptilia","mob_match"] ~ pca_data[pca_data$class_match == "Rep
       col = rep_col, pch = 20, cex = 0.5)

points(pca_data[pca_data$class_match == "Actinopterygii","mob_match"] ~ pca_data[pca_data$class_match =

points(pca_data[pca_data$class_match == "Gastropoda","mob_match"] ~ pca_data[pca_data$class_match == "Ga

points(pca_data[pca_data$class_match == "Demospongiae","mob_match"] ~ pca_data[pca_data$class_match ==

points(pca_data[pca_data$class_match == "Anthozoa","mob_match"] ~ pca_data[pca_data$class_match == "Anth

points(pca_data[pca_data$class_match == "Bivalvia","mob_match"] ~ pca_data[pca_data$class_match == "Biva
```



Now lets test whether these differences of mode of life on the fast slow axis are differnt from each other whilst accounting for phylogentic relationship and population level variation. We do this using a MCMCglmm model both seperately for terrestrial groups and aquatic groups.

First we run the aquatic groups. To avoid long runs for this script we run it for one tree over two chains.

```
#subset to aquatic species
aquatic_res <- pca_data[pca_data$PCA_moblist == "sessile"
                        | pca_data$PCA_moblist == "demersal"
                        | pca_data$PCA_moblist == "pelagic",]


#We set pelagic species as our base level.
aquatic_res$PCA_moblist <- factor(aquatic_res$PCA_moblist,
                                  levels = c("pelagic",
                                             "sessile",
                                             "demersal"))


#set a prior
prior<-list(R = list(V = 1/2, nu=0.002),
            G = list(G1=list(V = 1/2,
                             n = 1,
                             alpha.mu=rep(0,1),
                             alpha.V= diag(1)*10^3),
                     G1=list(V = 1/2,
                             n = 1,
                             alpha.mu=rep(0,1),
                             alpha.V= diag(1)*10^3)))

#run the analysis
#Chain 1
aquatic_pc1_c1 <- MCMCglmm(PC1 ~ PCA_moblist,
                     data = aquatic_res,
```

```
                    random=~animal + species_match,
                    pedigree = axis_trees[[1]],
                    prior = prior,
                    nitt = c(1100000),
                    burnin = 100000,
                    thin = 500,
                    verbose = F)
#Chain 2
aquatic_pc1_c2 <- MCMCglmm(PC1 ~ PCA_moblist,
                    data = aquatic_res,
                    random=~animal + species_match,
                    pedigree = axis_trees[[1]],
                    prior = prior,
                    nitt = c(1100000),
                    burnin = 100000,
                    thin = 500,
                    verbose = F)


#Check the chains converge
gelman.diag(mcmc.list(aquatic_pc1_c1$Sol,
                      aquatic_pc1_c2$Sol))
```

```
## Potential scale reduction factors:
##
##                      Point est. Upper C.I.
## (Intercept)                   1          1
## PCA_moblistsessile            1          1
## PCA_moblistdemersal           1          1
##
## Multivariate psrf
##
## 1
```

```
gelman.diag(mcmc.list(aquatic_pc1_c1$VCV,
                      aquatic_pc1_c2$VCV))
```

```
## Potential scale reduction factors:
##
##                Point est. Upper C.I.
## animal               1.01       1.02
## species_match        1.00       1.01
## units                1.00       1.02
##
## Multivariate psrf
##
## 1
```

```
#summary
summary(aquatic_pc1_c1)
```

```
##
##  Iterations = 100001:1099501
##  Thinning interval  = 500
##  Sample size  = 2000
```

```
##
##  DIC: 142.0327
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal     2.652 7.35e-06    6.242     2000
##
##                ~species_match
##
##               post.mean  l-95% CI u-95% CI eff.samp
## species_match    0.7488 3.795e-06    1.613     2000
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.4106   0.2177   0.6687     1872
##
##  Location effects: PC1 ~ PCA_moblist
##
##                     post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)           -1.0520  -3.7804   1.4420     2027 0.399
## PCA_moblistsessile     2.6764   0.1755   5.4938     2079 0.049 *
## PCA_moblistdemersal    1.6700   0.1668   3.2321     2000 0.036 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Calculate phyogentic signal, population level variation and residual variation.
aquatic_pc1_vcv_phylo <- aquatic_pc1_c1$VCV[,1]
aquatic_pc1_vcv_spec <- aquatic_pc1_c1$VCV[,2]
aquatic_pc1_vcv_units <- aquatic_pc1_c1$VCV[,3]

#Phyogentic signal
aquatic_pc1_phylo <- aquatic_pc1_vcv_phylo/
                          c(aquatic_pc1_vcv_phylo
                            + aquatic_pc1_vcv_spec
                            + aquatic_pc1_vcv_units)
#Population level variation
aquatic_pc1_spec <- aquatic_pc1_vcv_spec/
                          c(aquatic_pc1_vcv_phylo
                            + aquatic_pc1_vcv_spec
                            + aquatic_pc1_vcv_units)
#Residual
aquatic_pc1_units <- aquatic_pc1_vcv_units/
                          c(aquatic_pc1_vcv_phylo
                            + aquatic_pc1_vcv_spec
                            + aquatic_pc1_vcv_units)


hdr(aquatic_pc1_phylo)$mode
```

```
## [1] 0.7285542
```

```
hdr(aquatic_pc1_spec)$mode
```

```
## [1] 0.1366372
```

```
hdr(aquatic_pc1_units)$mode
```

```
## [1] 0.09454943
```

Now lets do the same for terrestrial species.

```
##terrestiral species

ter_res <- pca_data[pca_data$PCA_moblist == "terrestrial"
                    | pca_data$PCA_moblist == "arboreal"
                    | pca_data$PCA_moblist == "volant"
                    | pca_data$PCA_moblist == "semi_aquatic"
                    | pca_data$PCA_moblist == "semi_fossorial",]

ter_res$PCA_moblist <- factor(ter_res$PCA_moblist,
                              levels = c("terrestrial",
                                         "arboreal",
                                         "volant",
                                         "semi_aquatic",
                                         "semi_fossorial"))


#set a prior
prior<-list(R = list(V = 1/2, nu=0.002),
            G = list(G1=list(V = 1/2,
                             n = 1,
                             alpha.mu=rep(0,1),
                             alpha.V= diag(1)*10^3),
                     G1=list(V = 1/2,
                             n = 1,
                             alpha.mu=rep(0,1),
                             alpha.V= diag(1)*10^3)))


ter_pc1_c1 <- MCMCglmm(PC1 ~ PCA_moblist,
                    data = ter_res,
                    random=~animal + species_match,
                    pedigree = axis_trees[[1]],
                    prior = prior,
                    nitt = c(1100000),
                    burnin = 100000,
                    thin = 500,
                    verbose = F)


ter_pc1_c2 <- MCMCglmm(PC1 ~ PCA_moblist,
                    data = ter_res,
                    random=~animal + species_match,
                    pedigree = axis_trees[[1]],
                    prior = prior,
                    nitt = c(1100000),
```

```
                    burnin = 100000,
                    thin = 500,
                    verbose = F)


#Check the chains converge
gelman.diag(mcmc.list(ter_pc1_c1$Sol,
                      ter_pc1_c2$Sol))
```

```
## Potential scale reduction factors:
##
##                            Point est. Upper C.I.
## (Intercept)                         1       1.02
## PCA_moblistarboreal                 1       1.00
## PCA_moblistvolant                   1       1.00
## PCA_moblistsemi_aquatic             1       1.00
## PCA_moblistsemi_fossorial           1       1.01
##
## Multivariate psrf
##
## 1
```

```
gelman.diag(mcmc.list(ter_pc1_c1$VCV,
                      ter_pc1_c2$VCV))
```

```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## animal                 1       1.01
## species_match          1       1.00
## units                  1       1.00
##
## Multivariate psrf
##
## 1
```

```
summary(ter_pc1_c1)
```

```
##
##  Iterations = 100001:1099501
##  Thinning interval  = 500
##  Sample size  = 2000
##
##  DIC: 319.4351
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal     10.62    6.018    16.23     2000
##
##                ~species_match
##
##               post.mean  l-95% CI u-95% CI eff.samp
## species_match    0.1346 6.456e-07   0.3288     1696
##
```

```
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.1762   0.1314   0.2165     2000
##
##  Location effects: PC1 ~ PCA_moblist
##
##                            post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)                  0.57368 -4.64811  6.41147     2000 0.844
## PCA_moblistarboreal          0.23139 -0.63376  1.15915     1913 0.616
## PCA_moblistvolant            0.05750 -2.06626  2.14367     2000 0.950
## PCA_moblistsemi_aquatic      0.03251 -1.76657  1.72798     2000 0.966
## PCA_moblistsemi_fossorial   -0.40138 -1.66672  0.96700     2000 0.555
```

```
ter_pc1_vcv_phylo <- ter_pc1_c1$VCV[,1]
ter_pc1_vcv_spec <- ter_pc1_c1$VCV[,2]
ter_pc1_vcv_units <- ter_pc1_c1$VCV[,3]

ter_pc1_phylo <- ter_pc1_vcv_phylo/
  c(ter_pc1_vcv_phylo + ter_pc1_vcv_spec + ter_pc1_vcv_units)

ter_pc1_spec <- ter_pc1_vcv_spec/
  c(ter_pc1_vcv_phylo + ter_pc1_vcv_spec + ter_pc1_vcv_units)

ter_pc1_units <- ter_pc1_vcv_units/
  c(ter_pc1_vcv_phylo + ter_pc1_vcv_spec + ter_pc1_vcv_units)

hdr(ter_pc1_phylo)$mode
```

```
## [1] 0.9760544
```

```
hdr(ter_pc1_spec)$mode
```

```
## [1] 0.0006235144
```

```
hdr(ter_pc1_units)$mode
```

```
## [1] 0.01518916
```

Metabolic rate analysis

Next we analysis how metabolic rate is associated with the fast slow continuum by regressing mass soecific metabolic rate against PC1.

```
#subset to species with metabolic rate data
pca_data_met <- na.omit(pca_data[,c("PC1",
                                    "species_match",
                                    "class_match",
                                    "animal",
                                    "PCA_met")])


#set a prior
prior_met <-list(R = list(V = 1/2, nu=0.002),
            G = list(G1=list(V = 1/2,
```

```
                            n = 1,
                            alpha.mu=rep(0,1),
                            alpha.V= diag(1)*10^3),
                        G1=list(V = 1/2,
                            n = 1,
                            alpha.mu=rep(0,1),
                            alpha.V= diag(1)*10^3)))



met_mod_ch1 <- MCMCglmm(PC1 ~ log10(PCA_met) ,
                        data = pca_data_met,
                        random=~animal + species_match,
                        pedigree = axis_trees[[1]],
                        prior = prior_met,
                        nitt = c(1100000),
                        burnin = 100000,
                        thin = 500,
                        verbose = F)


met_mod_ch2 <- MCMCglmm(PC1 ~ log10(PCA_met) ,
                        data = pca_data_met,
                        random=~animal + species_match,
                        pedigree = axis_trees[[1]],
                        prior = prior_met,
                        nitt = c(1100000),
                        burnin = 100000,
                        thin = 500,
                        verbose = F)


#Check the chains converge
gelman.diag(mcmc.list(met_mod_ch1$Sol,
                      met_mod_ch2$Sol))
```

```
## Potential scale reduction factors:
##
##                 Point est. Upper C.I.
## (Intercept)              1       1.00
## log10(PCA_met)           1       1.01
##
## Multivariate psrf
##
## 1
```

```
gelman.diag(mcmc.list(met_mod_ch1$VCV,
                      met_mod_ch2$VCV))
```

```
## Potential scale reduction factors:
##
##                Point est. Upper C.I.
## animal                 1          1
## species_match          1          1
## units                  1          1
```

```
##
## Multivariate psrf
##
## 1
```

```
summary(met_mod_ch1)
```

```
##
##  Iterations = 100001:1099501
##  Thinning interval  = 500
##  Sample size  = 2000
##
##  DIC: 221.3889
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal      7.18    3.191    11.45     2000
##
##                ~species_match
##
##               post.mean  l-95% CI u-95% CI eff.samp
## species_match   0.06473 5.938e-09   0.2389     1628
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.2072   0.1556   0.2645     2000
##
##  Location effects: PC1 ~ log10(PCA_met)
##
##                post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      0.59897 -3.43038  4.11072     2000 0.753
## log10(PCA_met)  -1.02322 -1.88377 -0.09075     2000 0.022 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
met_mod_phylo <- met_mod_ch1$VCV[,1]
met_mod_spec <- met_mod_ch1$VCV[,2]
met_mod_units <- met_mod_ch1$VCV[,3]

met_mod_phylo_H <- met_mod_phylo/c(met_mod_phylo + met_mod_spec + met_mod_units)
met_mod_spec_H <- met_mod_spec/c(met_mod_phylo + met_mod_spec + met_mod_units)
met_mod_units_H <- met_mod_units/c(met_mod_phylo + met_mod_spec + met_mod_units)

hdr(met_mod_phylo_H)$mode
```

```
## [1] 0.9711183
```

```
hdr(met_mod_spec_H)$mode
```

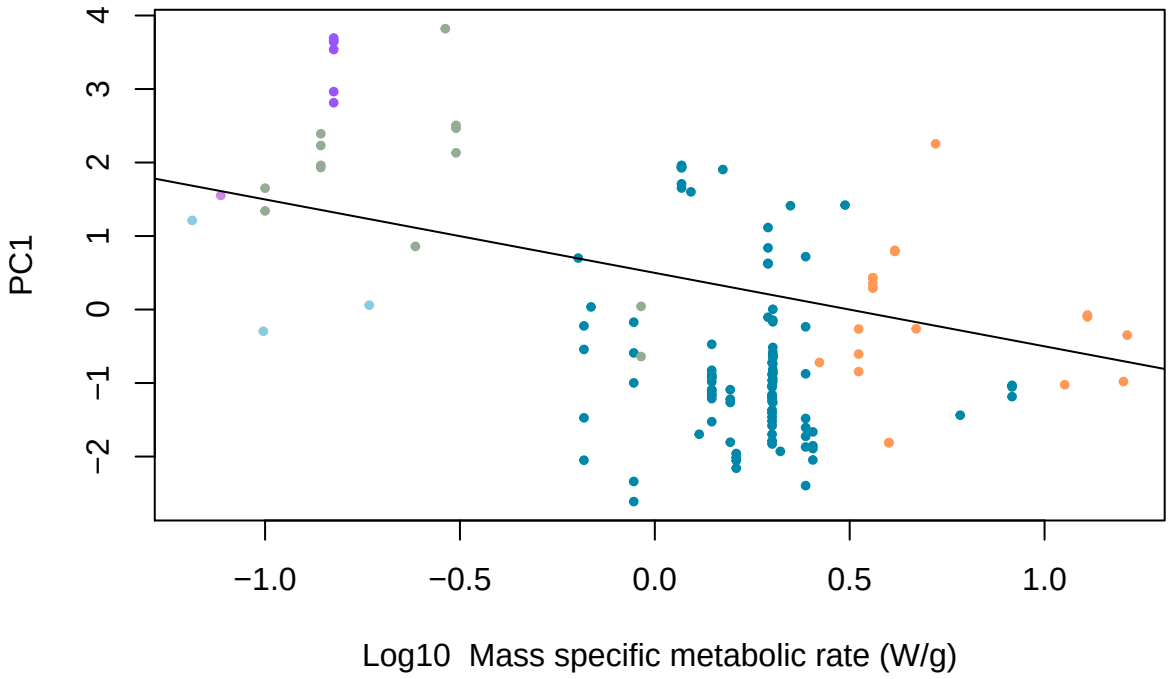
```
## [1] 0.0002180417
```

```
hdr(met_mod_units_H)$mode
```

```
## [1] 0.02678225
```

Lets also plot out how metabolic rate changes with a species position on the fast slow contiuum.

```
plot(log10(pca_data_met$PCA_met),
     pca_data_met$PC1,
     pch = 16,
     col = "white",
     cex = 0.1,
     xlab = "Log10  Mass specific metabolic rate (W/g)",
     ylab = "PC1")

points(log10(pca_data_met[pca_data_met$class_match == "Mammalia","PCA_met"]),
       pca_data_met[pca_data_met$class_match == "Mammalia","PC1"],
       col = mam_col, pch = 20, cex = 0.8)

points(log10(pca_data_met[pca_data_met$class_match == "Aves","PCA_met"]),
       pca_data_met[pca_data_met$class_match == "Aves","PC1"],
       col = bird_col,
       pch = 20,
       cex = 0.8)

points(log10(pca_data_met[pca_data_met$class_match == "Reptilia","PCA_met"]),
       pca_data_met[pca_data_met$class_match == "Reptilia","PC1"],
       col = rep_col,
       pch = 20,
       cex = 0.8)

points(log10(pca_data_met[pca_data_met$class_match == "Actinopterygii","PCA_met"]),
       pca_data_met[pca_data_met$class_match == "Actinopterygii","PC1"],
       col = fish_col,
       pch = 20,
       cex = 0.8)

points(log10(pca_data_met[pca_data_met$class_match == "Gastropoda","PCA_met"]),
       pca_data_met[pca_data_met$class_match == "Gastropoda","PC1"],
       col = gast_col,
       pch = 20,
       cex = 0.8)

points(log10(pca_data_met[pca_data_met$class_match == "Demospongiae","PCA_met"]),
       pca_data_met[pca_data_met$class_match == "Demospongiae","PC1"],
       col = sponge_col,
       pch = 20,
       cex = 0.8)

points(log10(pca_data_met[pca_data_met$class_match == "Anthozoa","PCA_met"]), pca_data_met[pca_data_met

points(log10(pca_data_met[pca_data_met$class_match == "Bivalvia","PCA_met"]), pca_data_met[pca_data_met

abline(hdr(met_mod_ch1$Sol[,1])$mode, hdr(met_mod_ch1$Sol[,2])$mode)
```



Reporductive productivity analysis

As reporductive output is only defined by numbers of offspring in matrix models we also tested if reporductive productivity, the amount of reporductive mass produced per year, would be more strongly associated with the fast-slow contiuum as represented by PC1, then we found in the main PCA.

```
pca_data_repo <- na.omit(pca_data[,c("PC1",
                                     "species_match",
                                     "class_match",
                                     "animal",
                                     "PCA_repo")])


#set a prior
prior_repo <-list(R = list(V = 1/2, nu=0.002),
            G = list(G1=list(V = 1/2,
                             n = 1,
                             alpha.mu=rep(0,1),
                             alpha.V= diag(1)*10^3),
                     G1=list(V = 1/2,
                             n = 1,
                             alpha.mu=rep(0,1),
                             alpha.V= diag(1)*10^3)))


egg_size_ch1 <- MCMCglmm(PC1 ~ PCA_repo,
                     data = pca_data_repo,
                     random=~animal + species_match,
                     pedigree = axis_trees[[1]],
                     prior = prior_repo,
                     nitt = c(11000000),
```

```
                     burnin = 1000000,
                     thin = 5000,
                     verbose = F)

egg_size_ch2 <- MCMCglmm(PC1 ~ PCA_repo,
                     data = pca_data_repo,
                     random=~animal + species_match,
                     pedigree = axis_trees[[1]],
                     prior = prior_repo,
                     nitt = c(11000000),
                     burnin = 1000000,
                     thin = 5000,
                     verbose = F)


gelman.diag(mcmc.list(egg_size_ch1$Sol,
                      egg_size_ch2$Sol))
```

```
## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## (Intercept)          1       1.00
## PCA_repo             1       1.01
##
## Multivariate psrf
##
## 1
```

```
gelman.diag(mcmc.list(egg_size_ch1$VCV,
                      egg_size_ch2$VCV))
```

```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## animal                1       1.00
## species_match         1       1.00
## units                 1       1.01
##
## Multivariate psrf
##
## 1
```

```
summary(egg_size_ch1)
```

```
##
##  Iterations = 1000001:10995001
##  Thinning interval  = 5000
##  Sample size  = 2000
##
##  DIC: 361.6741
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal     9.556    4.981    14.31     2000
```

```
##
##                ~species_match
##
##               post.mean  l-95% CI u-95% CI eff.samp
## species_match    0.1341 1.628e-10   0.3386     2000
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.1999   0.1537    0.245     1831
##
##  Location effects: PC1 ~ PCA_repo
##
##             post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)    0.9176  -3.1126   5.5581     2000 0.675
## PCA_repo      -0.7736  -1.4770  -0.0721     2000 0.027 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
egg_size_phylo <- egg_size_ch1$VCV[,1]/(egg_size_ch1$VCV[,1] +
                                        egg_size_ch1$VCV[,2] +
                                        egg_size_ch1$VCV[,3])

egg_size_species <- egg_size_ch1$VCV[,2]/(egg_size_ch1$VCV[,1] +
                                        egg_size_ch1$VCV[,2] +
                                        egg_size_ch1$VCV[,3])

egg_size_units <- egg_size_ch1$VCV[,3]/(egg_size_ch1$VCV[,1] +
                                        egg_size_ch1$VCV[,2] +
                                        egg_size_ch1$VCV[,3])

hdr(egg_size_phylo)$mode
```

```
## [1] 0.9725956
```

```
hdr(egg_size_species)$mode
```

```
## [1] 0.0007247752
```

```
hdr(egg_size_units)$mode
```

```
## [1] 0.01931435
```

Ellipses

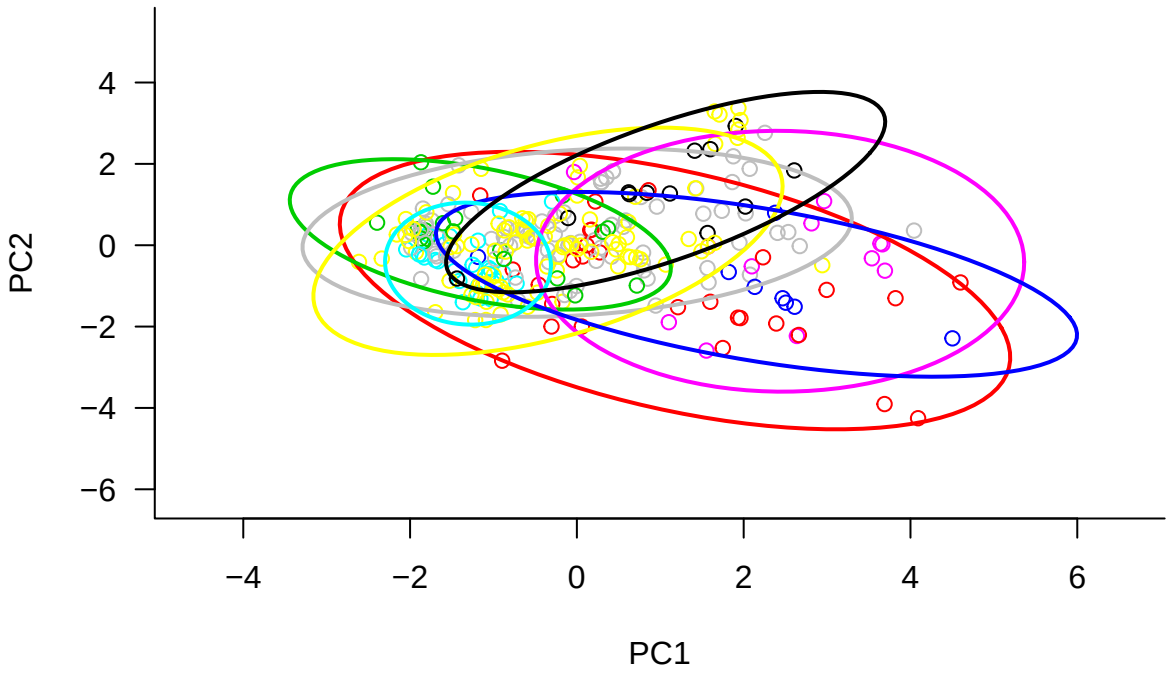
To test how different groupings are associated with PCA space we fit a series of ellipses using the SIBER package. Whilst SIBER is developed for fitting Baysian ellipses for stable isotope data it can be applied in the same way. TO get the figures to the standerd seen in the paper I used inkscape.

```
##Set up the data
siber_pca_data <- pca_data[pca_data$class_match =="Actinopterygii" |
                 pca_data$class_match == "Anthozoa"|
                 pca_data$class_match == "Aves"|
                 pca_data$class_match == "Gastropoda"|
                 pca_data$class_match == "Mammalia"|
```

```
                pca_data$class_match == "Reptilia",]

pca_data$PCA_iucn <- factor(pca_data$PCA_iucn, levels = c("NA", "CE", "E", "LC", "LR", "NT", "V"))
pca_data$PCA_iucn[is.na(pca_data$PCA_iucn)] <- "NA"


# Create lists of plotting arguments to be passed onwards to each
# of the three plotting functions.
community.hulls.args <- list(col = 1, lty = 1, lwd = 1)

group.ellipses.args  <- list(n = 100, p.interval = 0.95,
                             lty = 1, lwd = 2)

group.hull.args      <- list(lty = 2, col = "grey20")
```

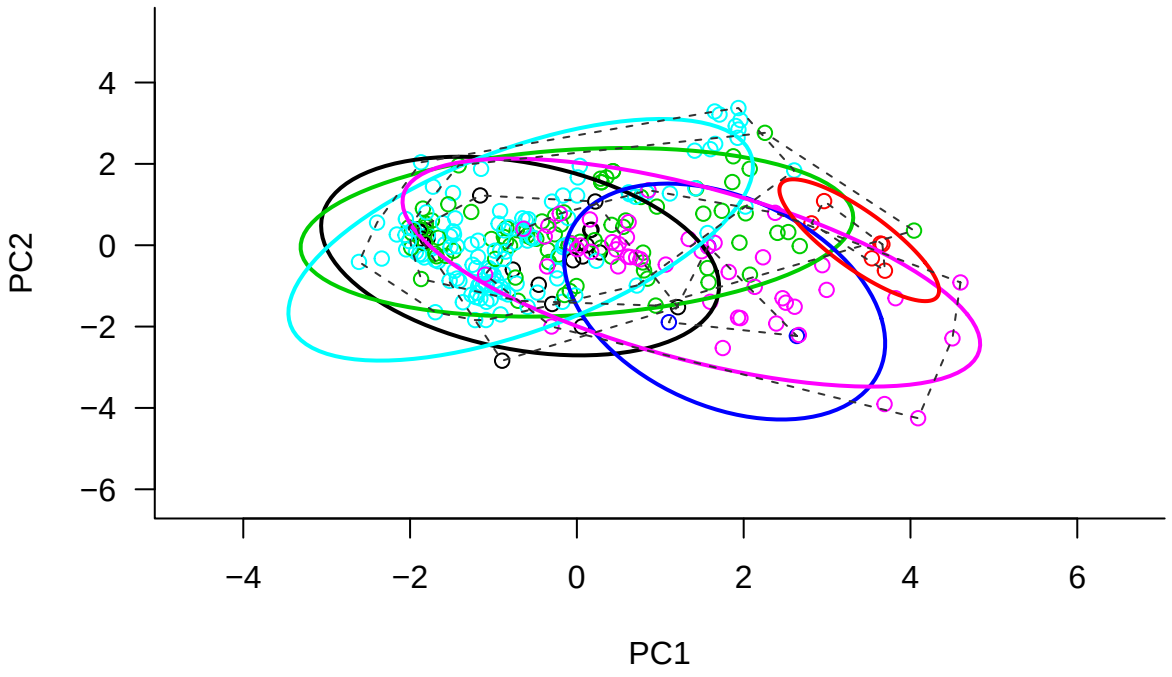
Mode of life Ellipse

```
sidmob <- data.frame(iso1 = pca_data$PC1,
                     iso2 = pca_data$PC2,
                     group = as.numeric(pca_data$PCA_moblist),
                     community = rep(1,length(pca_data$class_match)))

siber.mob <- createSiberObject(sidmob)


#plot for mode-of-life
plotSiberObject(siber.mob,
                  ax.pad = 2,
                  hulls = F, community.hulls.args,
                  ellipses = T, group.ellipses.args,
                  group.hulls = F, group.hull.args,
                  bty = "L",
                  iso.order = c(1,2),
                  xlab = "PC1",
                  ylab = "PC2"
                )
```



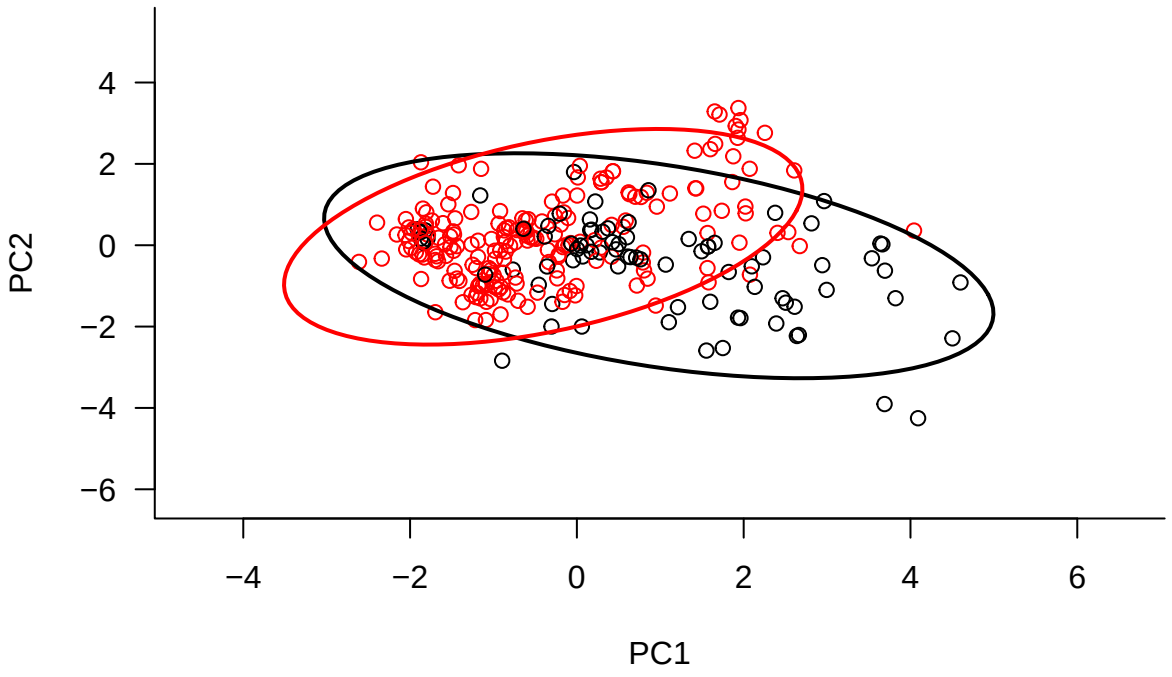
Taxinomic groupings Ellipse

```
sidpca <- data.frame(iso1 = siber_pca_data$PC1,
                     iso2 = siber_pca_data$PC2,
                     group = as.numeric(siber_pca_data$class_match),
                     community = rep(1,length(siber_pca_data$class_match)))


siber.plots <- createSiberObject(sidpca)

#plot for taxa
plotSiberObject(siber.plots,
                  ax.pad = 2,
                  hulls = F,
                  community.hulls.args,
                  ellipses = T,
                  group.ellipses.args,
                  group.hulls = T,
                  group.hull.args,
                  bty = "L",
                  iso.order = c(1,2),
                  xlab = "PC1",
                  ylab = "PC2"
                  )
```



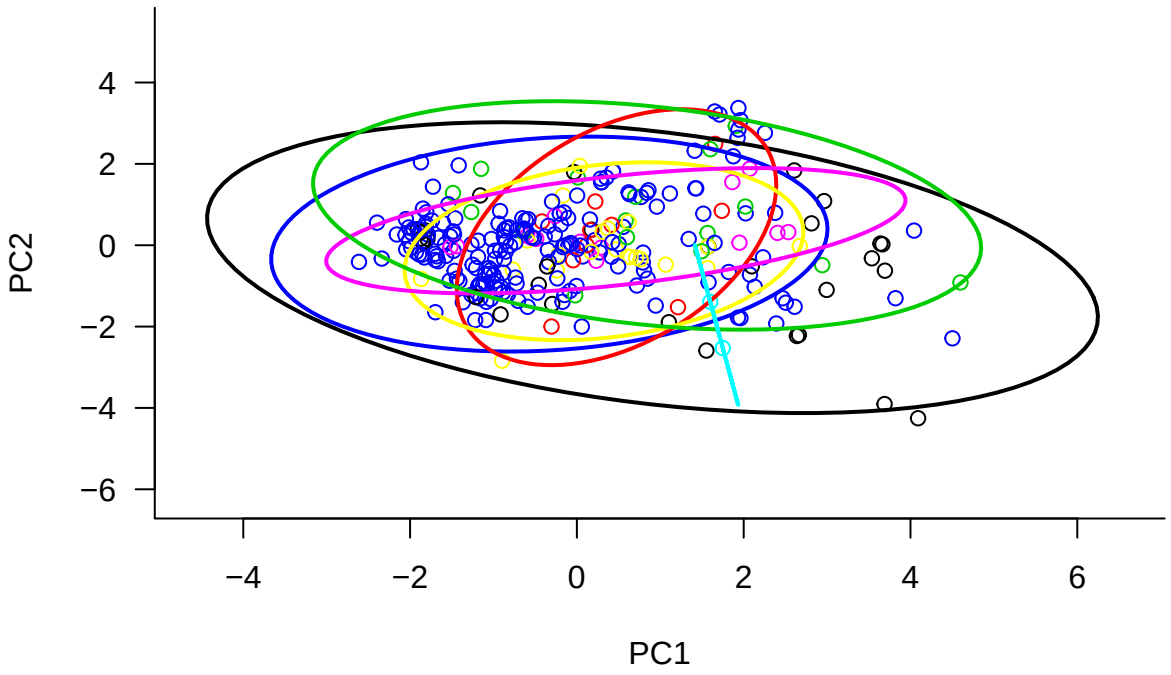
Thermal groupings Ellipse

```
sidtherm <- data.frame(iso1 = pca_data$PC1,
                       iso2 = pca_data$PC2,
                       group = as.numeric(pca_data$PCA_met_type),
                       community = rep(1,length(pca_data$class_match)))


siber.therm <- createSiberObject(sidtherm)

#plot for ecto endo
plotSiberObject(siber.therm,
                    ax.pad = 2,
                    hulls = F, community.hulls.args,
                    ellipses = T, group.ellipses.args,
                    group.hulls = F, group.hull.args,
                    bty = "L",
                    iso.order = c(1,2),
                    xlab = "PC1",
                    ylab = "PC2"
                    )
```



IUCN Ellipse

```
sired <- na.omit(data.frame(iso1 = pca_data$PC1,
                      iso2 = pca_data$PC2,
                      group = as.numeric(pca_data$PCA_iucn),
                      community = rep(1,length(pca_data$class_match))))


siber.iucn<- createSiberObject(sired)


#plot for iucn
plotSiberObject(siber.iucn,
                    ax.pad = 2,
                    hulls = F, community.hulls.args,
                    ellipses = T, group.ellipses.args,
                    group.hulls = F, group.hull.args,
                    bty = "L",
                    iso.order = c(1,2),
                    xlab = "PC1",
                    ylab = "PC2"
                    )
```



Ellipse overlap

Using a Baysian resampling approach we see how much each of the elliplses overlap with each other and plot these out. These calculations require `jags`. For more see the `SIBER` package

Mode of life ellipse overlap

```
group.ML <- groupMetricsML(siber.plots)
group.MLmob <- groupMetricsML(siber.mob)


# options for running jags
parms <- list()
parms$n.iter <- 2 * 10^4   # number of iterations to run the model for
parms$n.burnin <- 1 * 10^3 # discard the first set of values
parms$n.thin <- 10     # thin the posterior by this many
parms$n.chains <- 2        # run this many chains

# define the priors
priors <- list()
priors$R <- 1 * diag(2)
priors$k <- 2
priors$tau.mu <- 1.0E-3



ellipses.posterior_mob <- siberMVN(siber.mob, parms, priors)
```

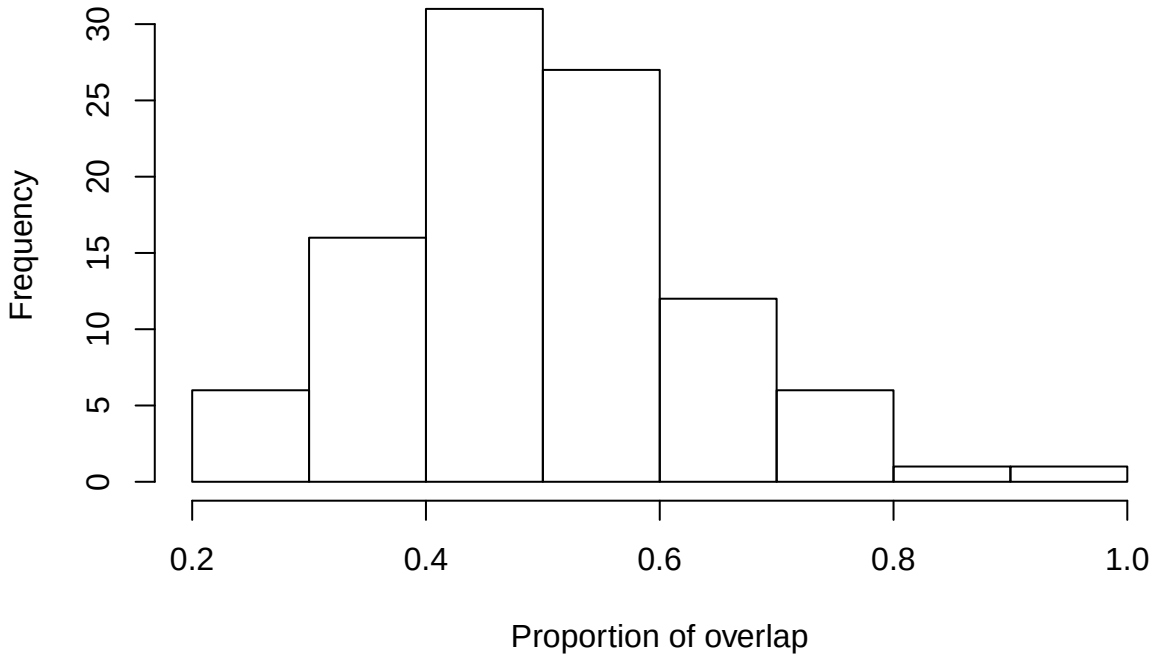
```
## Compiling model graph
```

```
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 28
##    Unobserved stochastic nodes: 3
##    Total graph size: 43
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 3
##    Total graph size: 36
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 82
##    Unobserved stochastic nodes: 3
##    Total graph size: 97
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 12
##    Unobserved stochastic nodes: 3
##    Total graph size: 27
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 95
##    Unobserved stochastic nodes: 3
##    Total graph size: 110
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
```

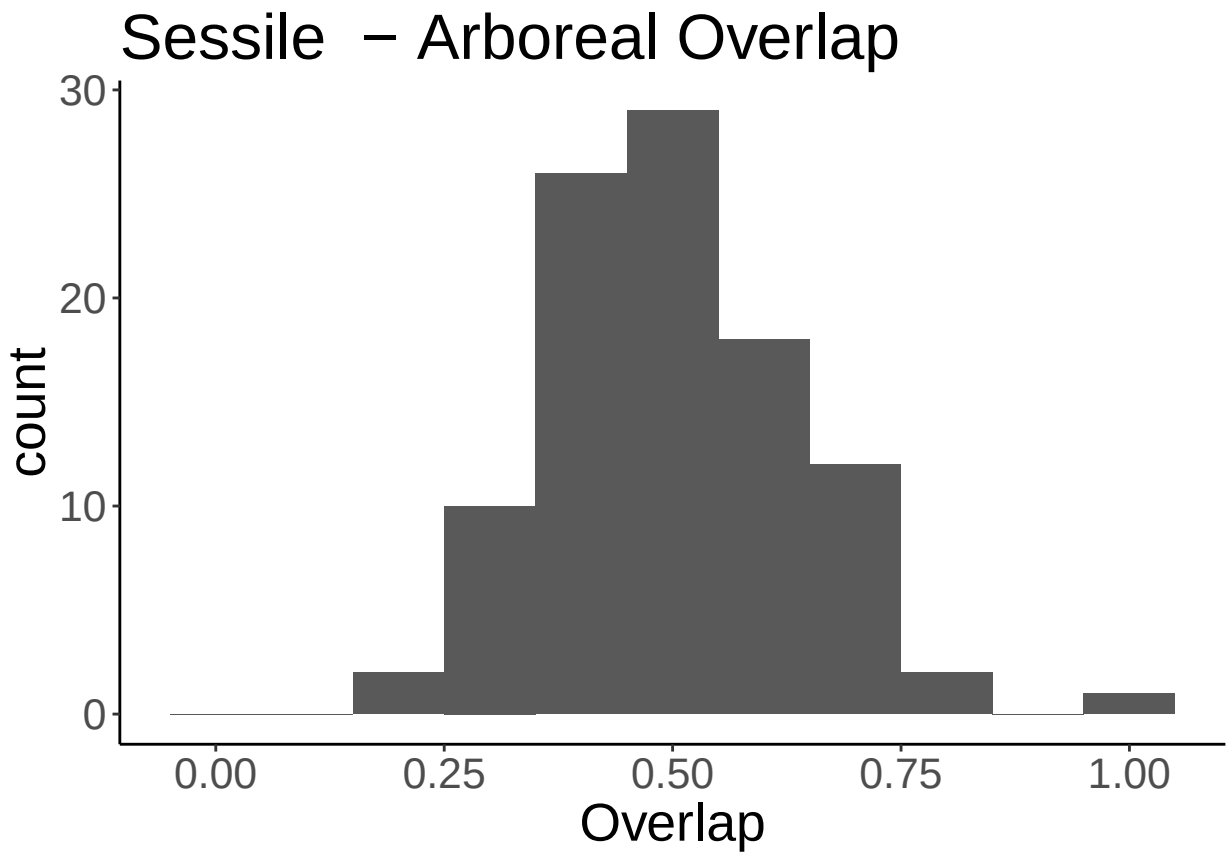
```
##    Unobserved stochastic nodes: 3
##    Total graph size: 23
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 3
##    Total graph size: 42
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 12
##    Unobserved stochastic nodes: 3
##    Total graph size: 27
##
## Initializing model
```

```r
# The first ellipse is referenced using a character string representation where
# in "x.y", "x" is the community, and "y" is the group within that community.
# So in this example: community 1, group 1

#ellipse group numbers
ellipse_sessile <- "1.1"
ellipse_arboreal <- "1.2"
ellipse_benthic <- "1.3"
ellipse_volant <- "1.4"
ellipse_semiaquatic <- "1.5"
ellipse_terrestrial <- "1.6"
ellipse_pelagic <- "1.7"
ellipse_semifossorial <- "1.8"


#####sessile  - arboreal
SA_95.overlap <- bayesianOverlap(ellipse_sessile,
                                 ellipse_arboreal,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
SA_95.overlap_prop <- vector()
for(i in 1:length(SA_95.overlap$overlap)){

SA_95.overlap_prop[i]  <- SA_95.overlap$overlap[i]/min(SA_95.overlap[i,1:2])

}
```

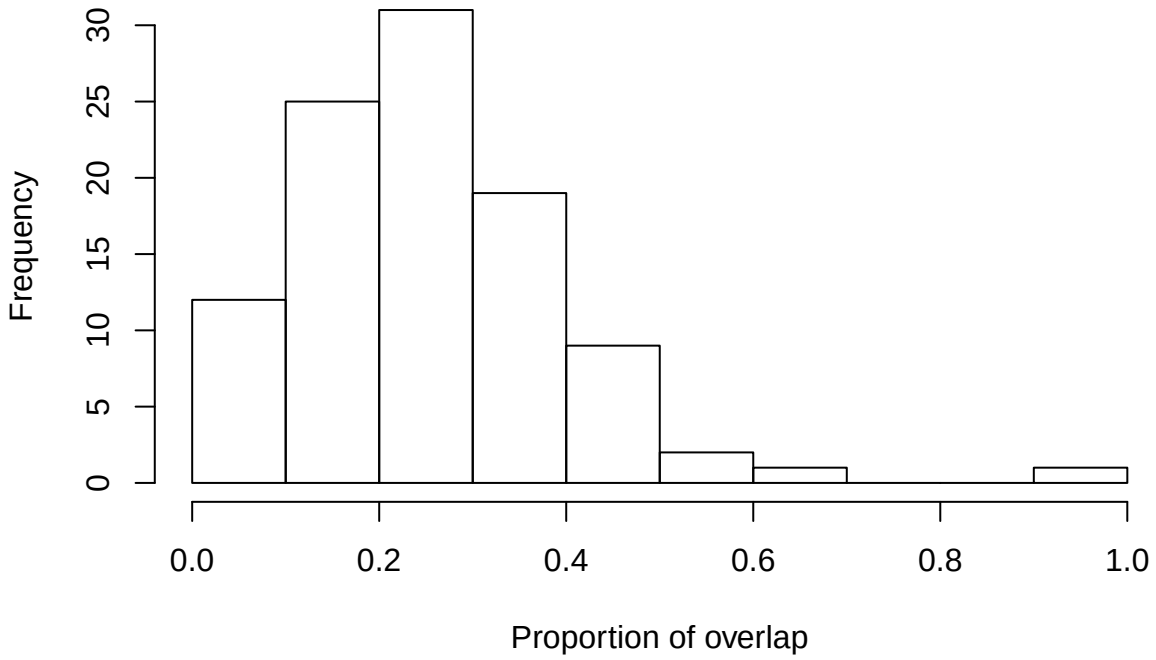
```r
#Plot
hist(SA_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



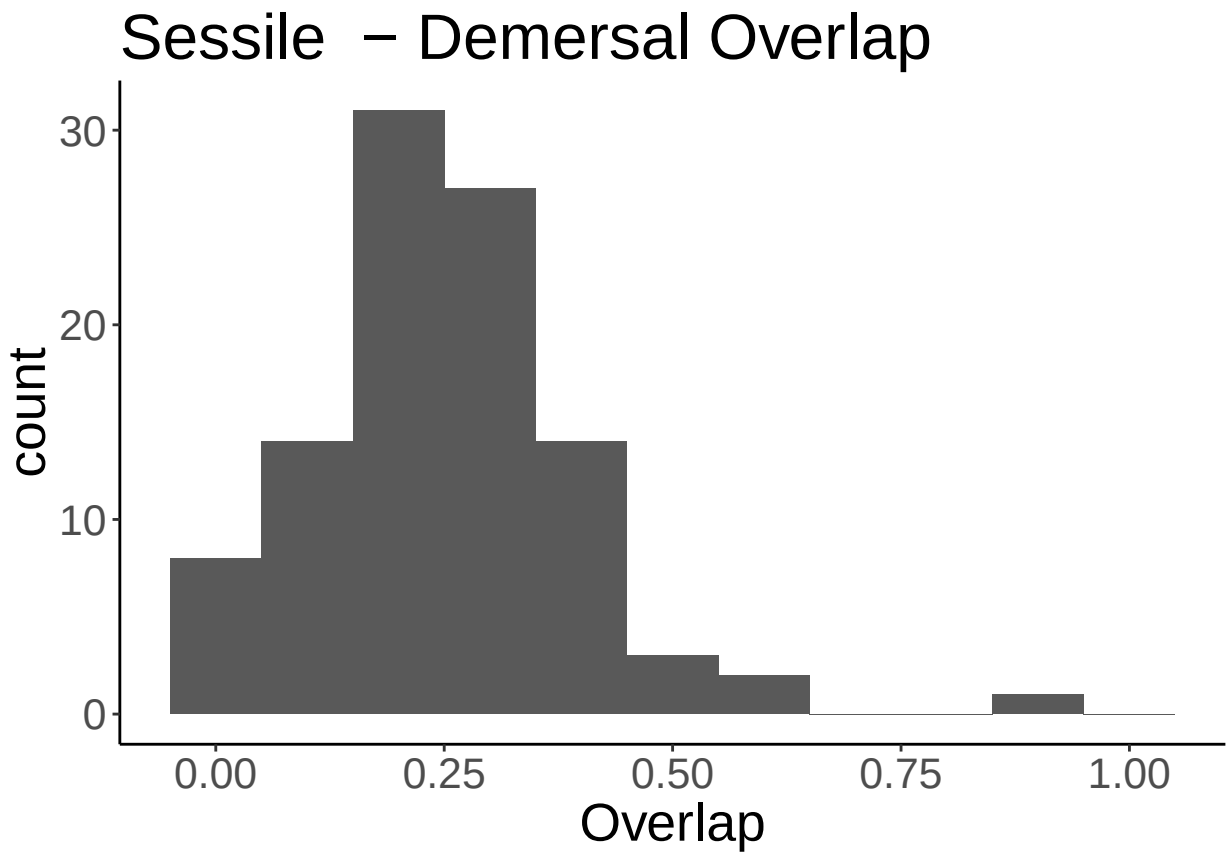

```r
myplot_SA = ggplot(data.frame(Overlap =  SA_95.overlap_prop),
                   aes(Overlap)) +
                   geom_histogram(binwidth=0.1) +
                   ggtitle("Sessile  - Arboreal Overlap") +
                   scale_x_continuous(limits = c(-0.05, 1.05))


myplot_SA + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



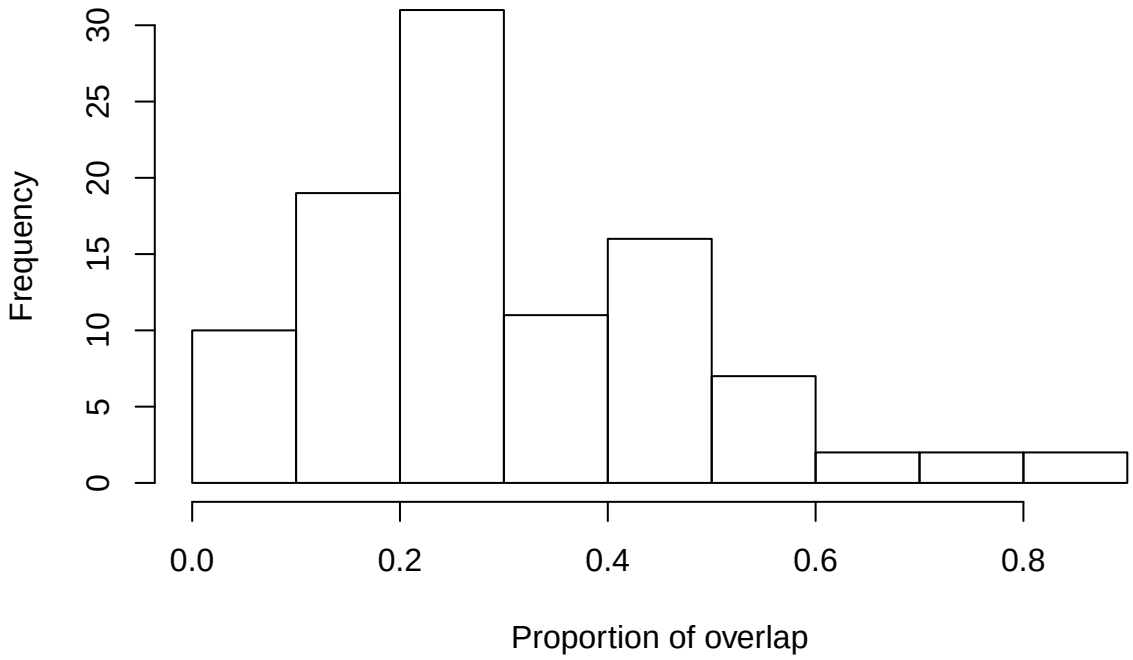

```
#####sessile  - benthic
SB_95.overlap <- bayesianOverlap(ellipse_sessile,
                                 ellipse_benthic,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)

SB_95.overlap_prop <- vector()
for(i in 1:length(SB_95.overlap$overlap)){

SB_95.overlap_prop[i]  <- SB_95.overlap$overlap[i]/min(SB_95.overlap[i,1:2])

}

hist(SB_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



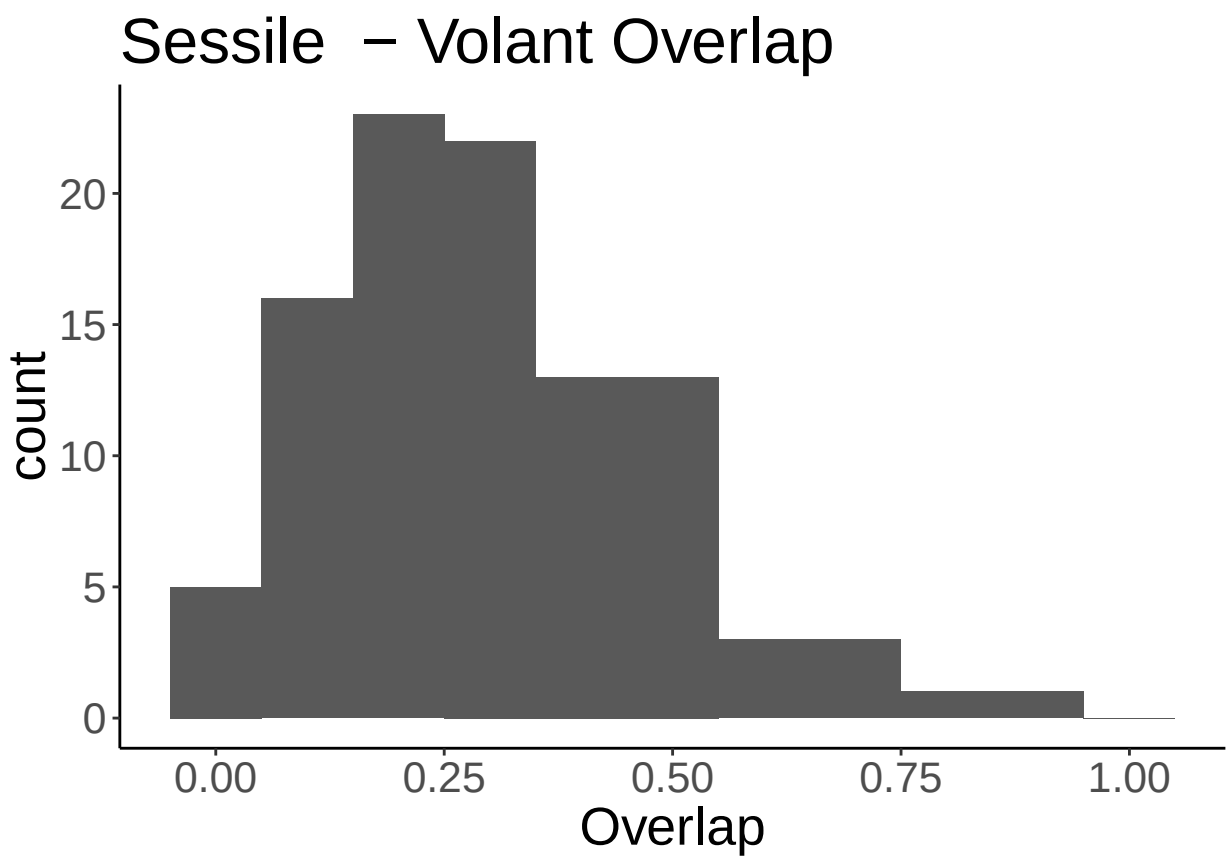

```
myplot_SB = ggplot(data.frame(Overlap =  SB_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Sessile  - Demersal Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_SB + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



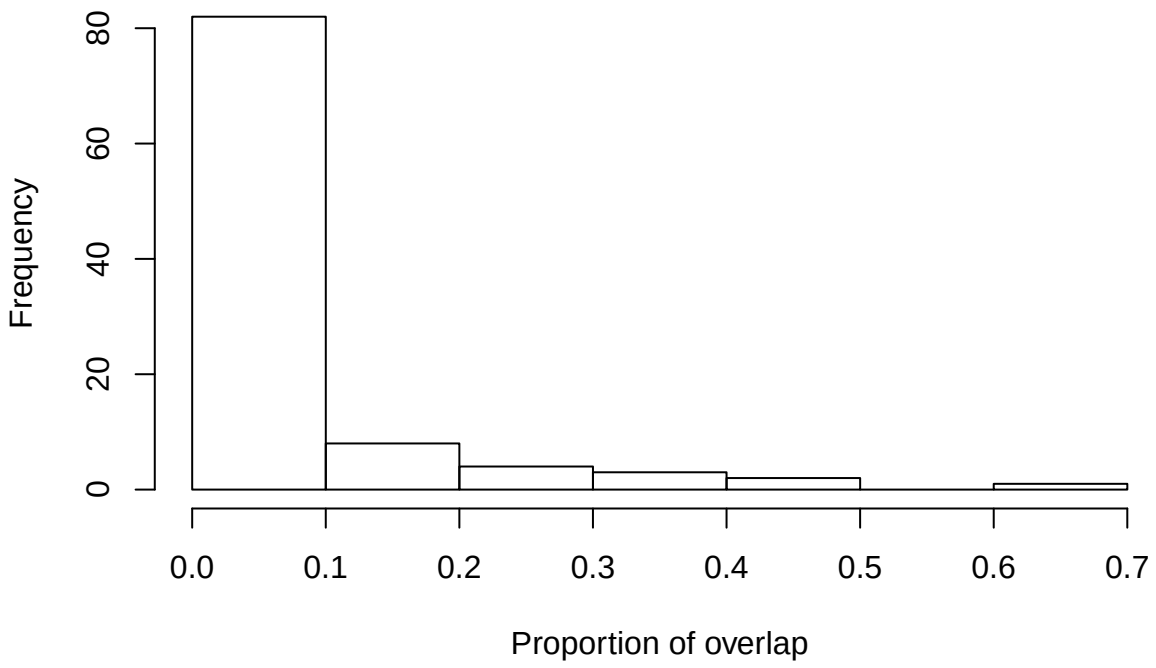

```
#####sessile  - volant
Sv_95.overlap <- bayesianOverlap(ellipse_sessile,
                                 ellipse_volant,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Sv_95.overlap_prop <- vector()
for(i in 1:length(Sv_95.overlap$overlap)){

Sv_95.overlap_prop[i]  <- Sv_95.overlap$overlap[i]/min(Sv_95.overlap[i,1:2])

}

hist(Sv_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



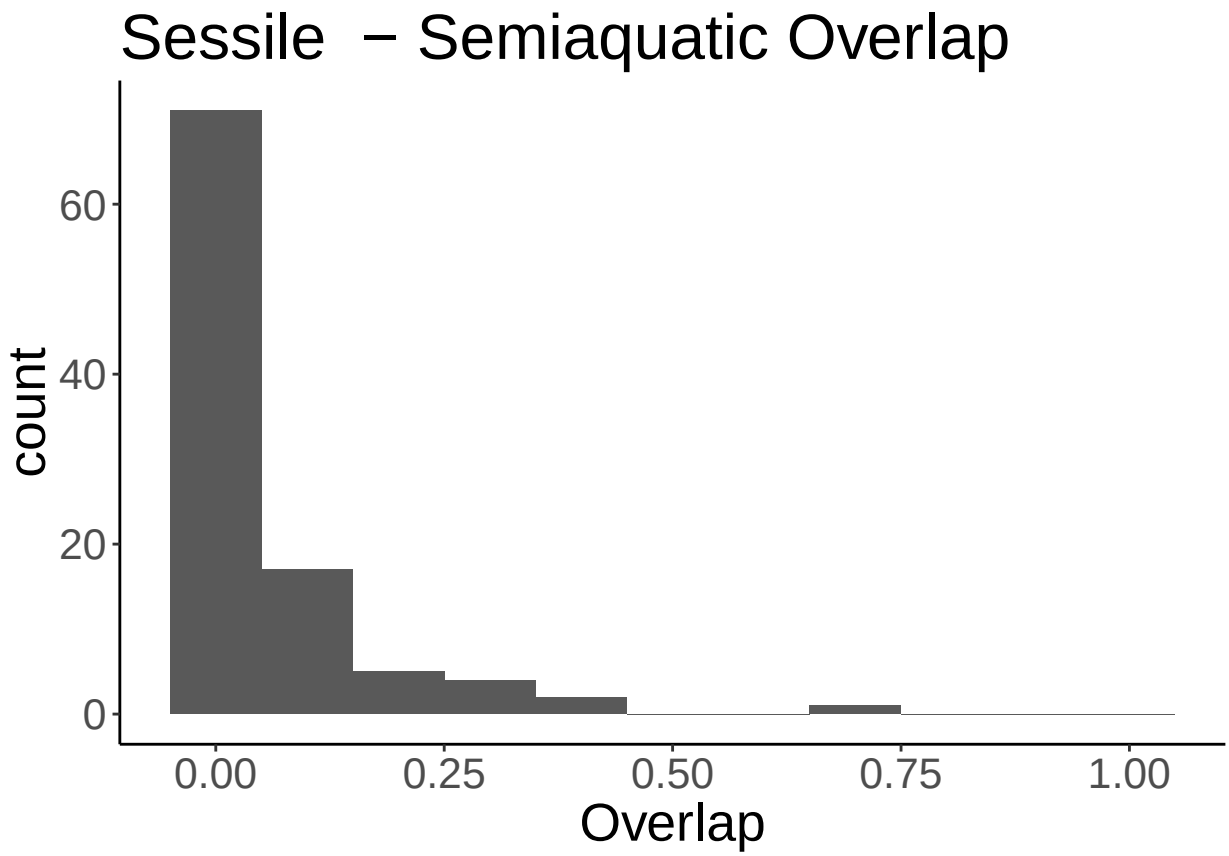

```
myplot_Sv = ggplot(data.frame(Overlap =  Sv_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Sessile  - Volant Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Sv + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



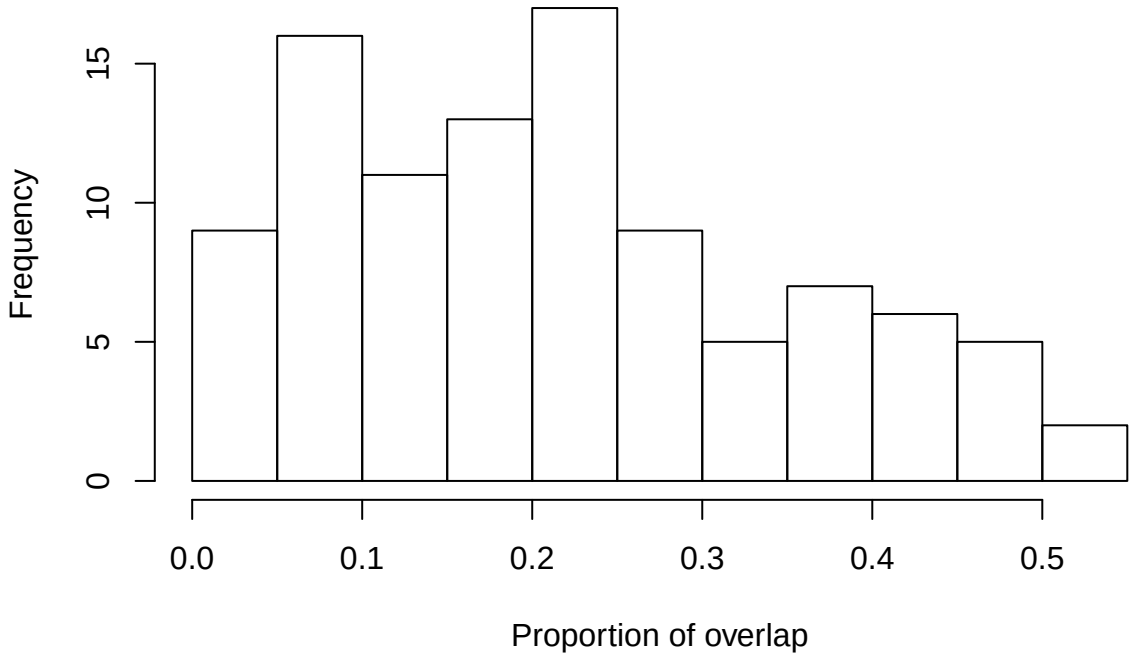

```
#####sessile  - semiaquatic
Ssem_95.overlap <- bayesianOverlap(ellipse_sessile,
                                   ellipse_semiaquatic,
                                   ellipses.posterior_mob,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
Ssem_95.overlap_prop <- vector()
for(i in 1:length(Ssem_95.overlap$overlap)){

Ssem_95.overlap_prop[i]  <- Ssem_95.overlap$overlap[i]/min(Ssem_95.overlap[i,1:2])

}

hist(Ssem_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



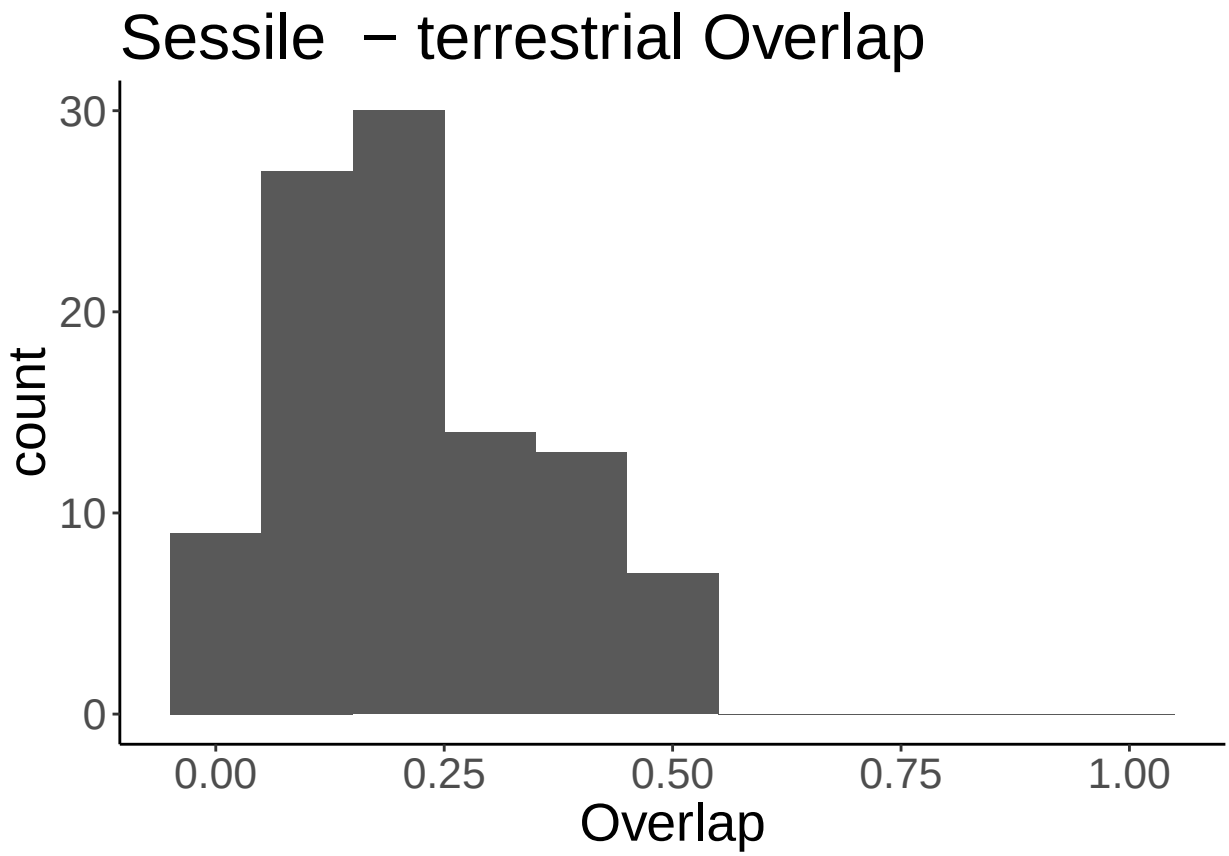

```
myplot_Ssem = ggplot(data.frame(Overlap =  Ssem_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Sessile  - Semiaquatic Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Ssem + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



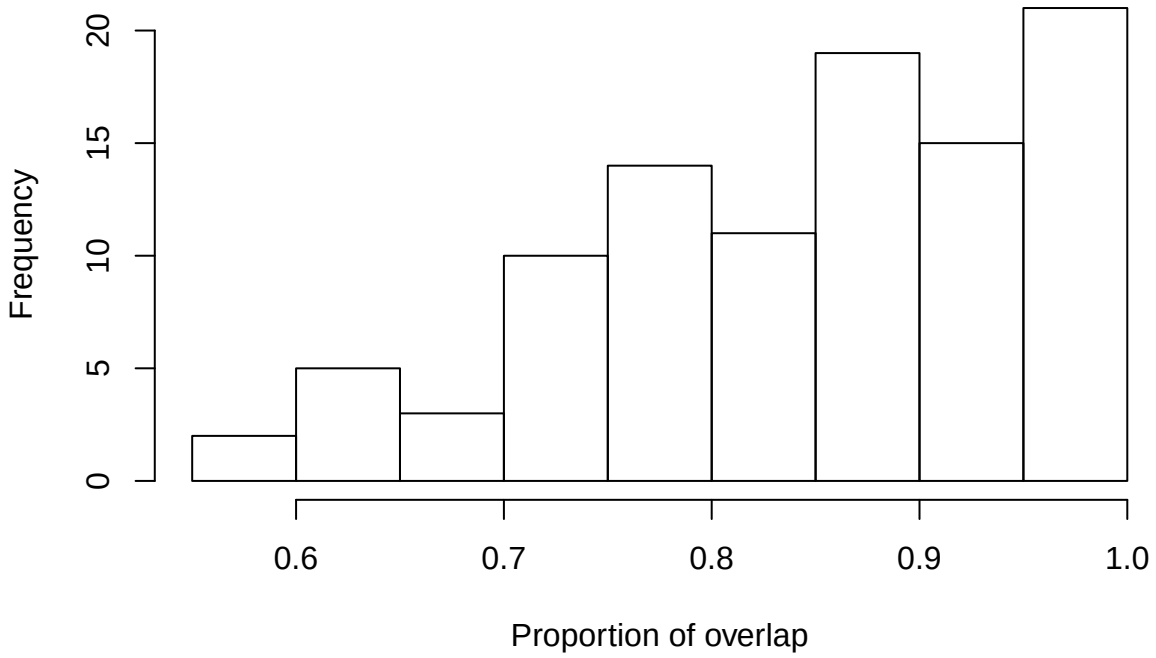

```
#####sessile  - terrestrial

St_95.overlap <- bayesianOverlap(ellipse_sessile,
                                 ellipse_terrestrial,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
St_95.overlap_prop <- vector()
for(i in 1:length(St_95.overlap$overlap)){

St_95.overlap_prop[i]  <- St_95.overlap$overlap[i]/min(St_95.overlap[i,1:2])

}

hist(St_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



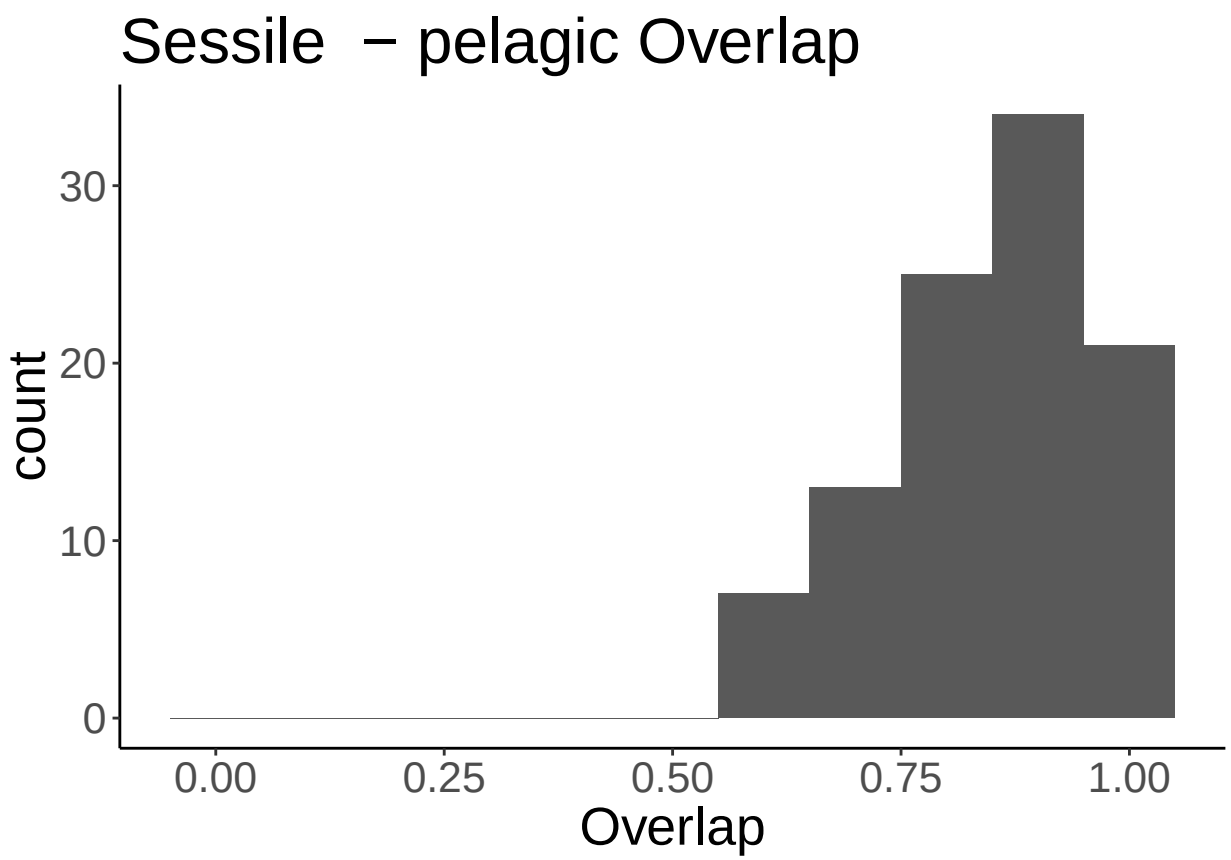

```
myplot_St = ggplot(data.frame(Overlap =  St_95.overlap_prop),
                   aes(Overlap)) +
                   geom_histogram(binwidth=0.1) +
                   ggtitle("Sessile  - terrestrial Overlap") +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_St + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



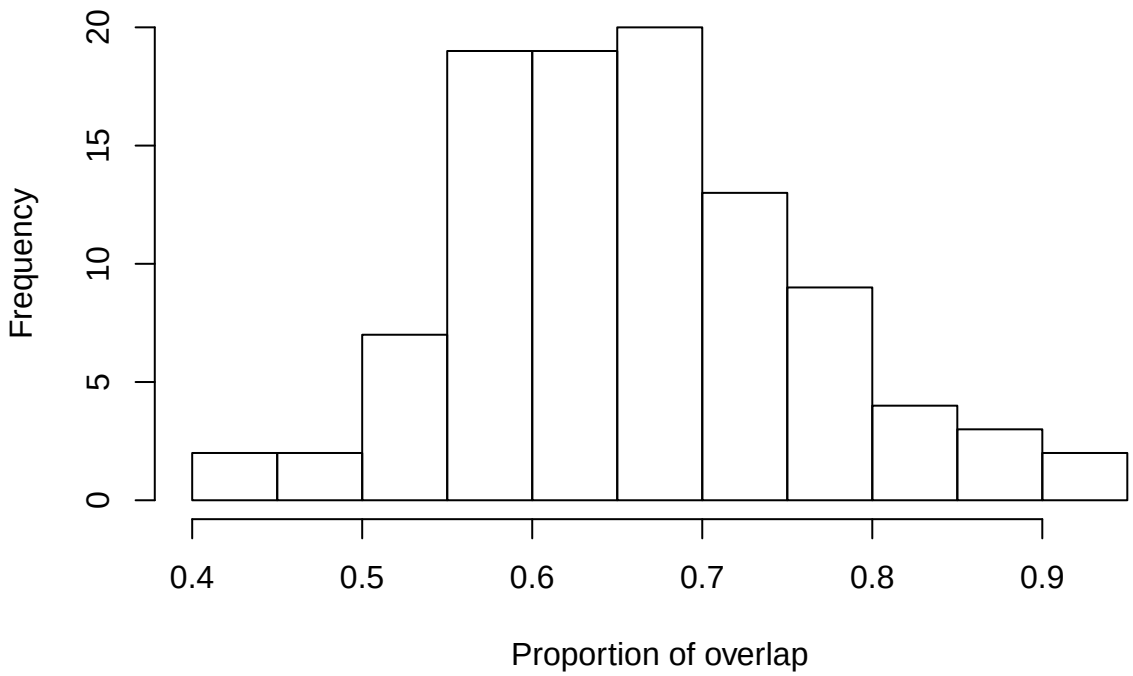

```
#####sessile  - pelagic
Sp_95.overlap <- bayesianOverlap(ellipse_sessile,
                                 ellipse_pelagic,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Sp_95.overlap_prop <- vector()
for(i in 1:length(Sp_95.overlap$overlap)){

Sp_95.overlap_prop[i]  <- Sp_95.overlap$overlap[i]/min(Sp_95.overlap[i,1:2])

}

hist(Sp_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



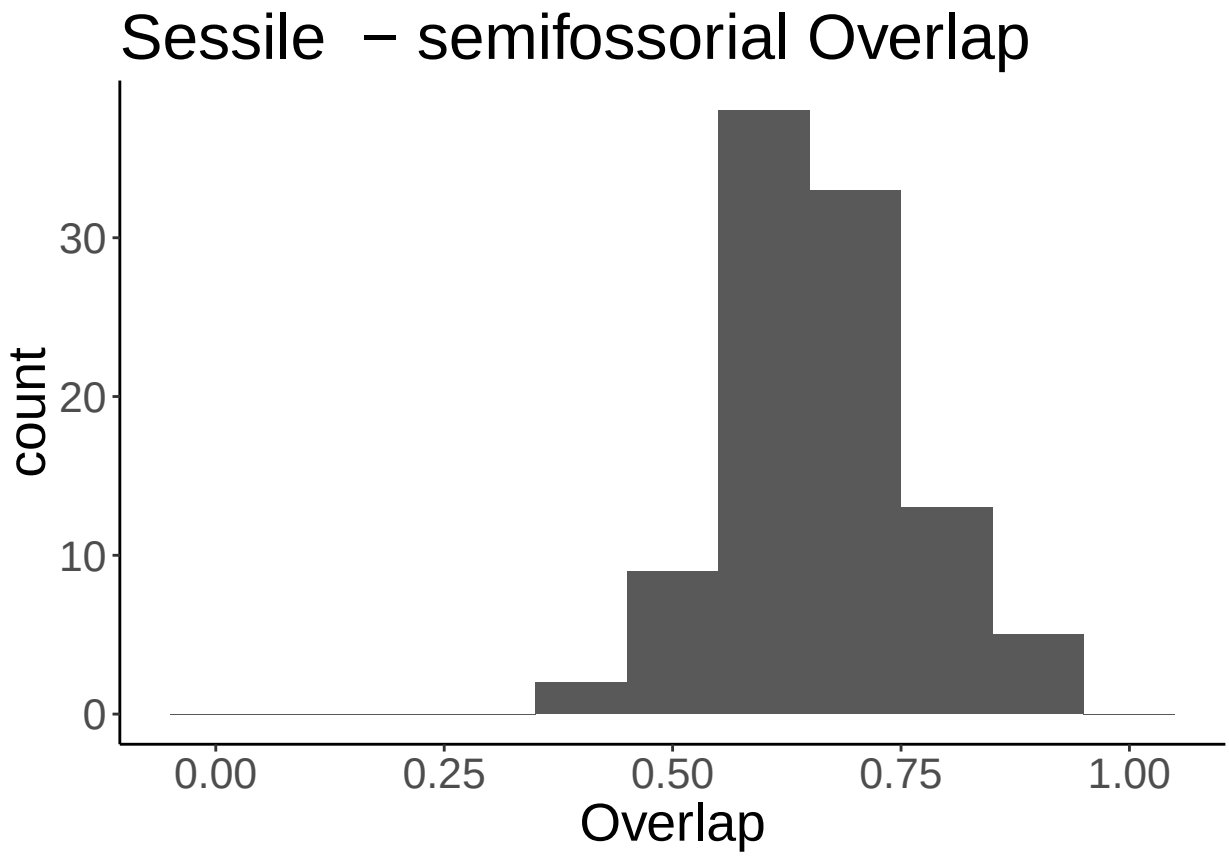

```
myplot_Sp = ggplot(data.frame(Overlap =  Sp_95.overlap_prop),
                   aes(Overlap)) +
                   geom_histogram(binwidth=0.1) +
                   ggtitle("Sessile  - pelagic Overlap") +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Sp + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



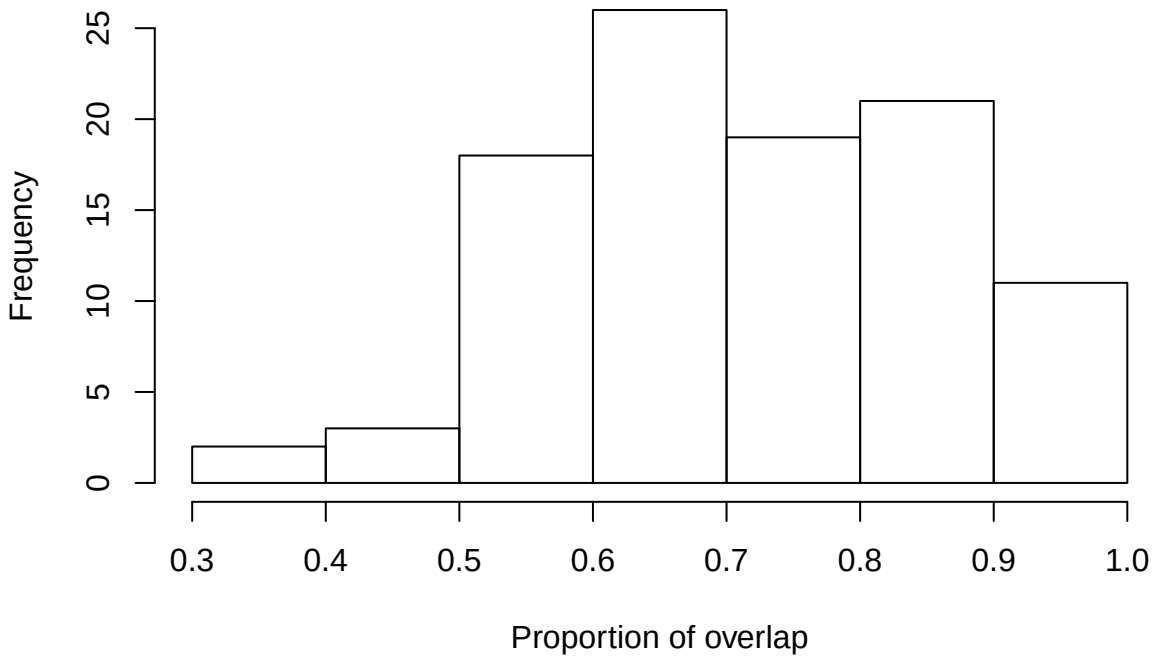

```
#####sessile  - semifossorial
Ssf_95.overlap <- bayesianOverlap(ellipse_sessile,
                                  ellipse_semifossorial,
                                  ellipses.posterior_mob,
                                  draws = 100,
                                  p.interval = 0.95,
                                  n = 100)
Ssf_95.overlap_prop <- vector()
for(i in 1:length(Ssf_95.overlap$overlap)){

Ssf_95.overlap_prop[i]  <- Ssf_95.overlap$overlap[i]/min(Ssf_95.overlap[i,1:2])

}

hist(Ssf_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



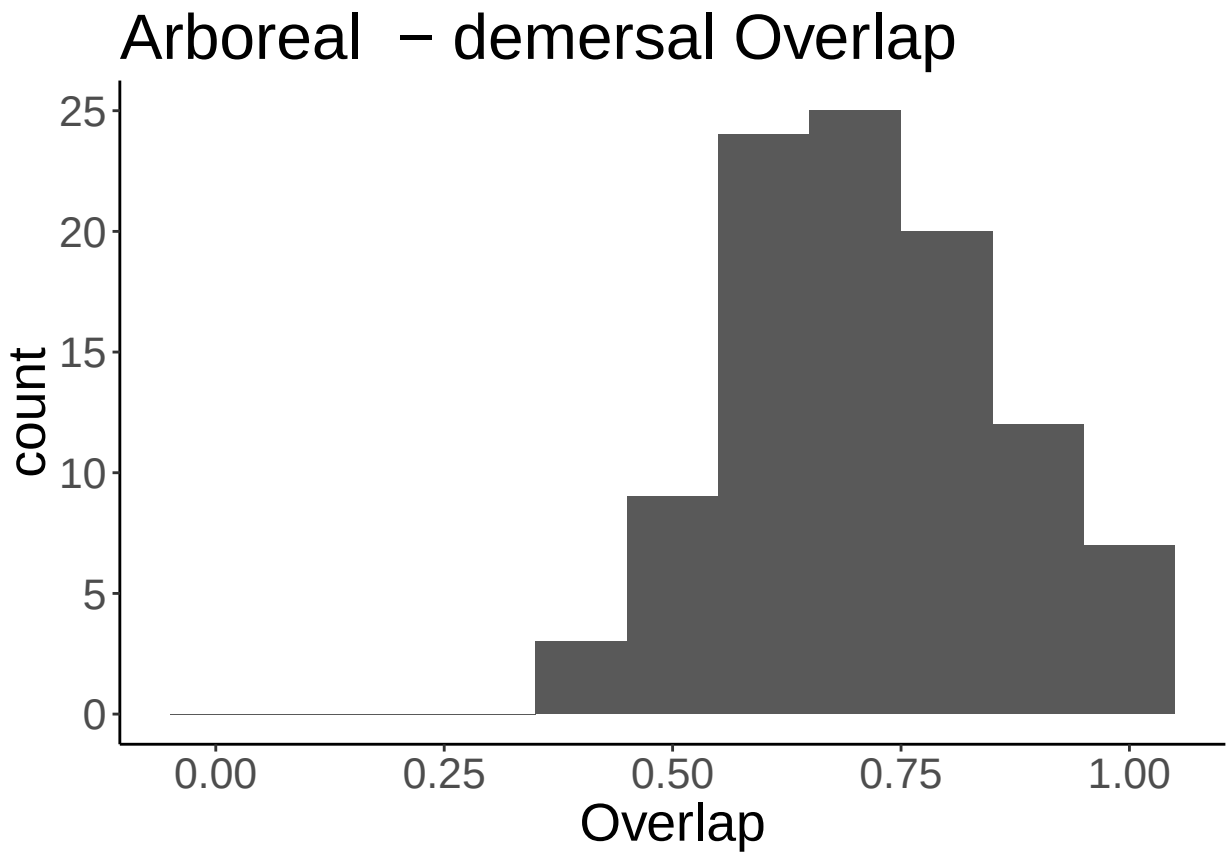

```
myplot_Ssf = ggplot(data.frame(Overlap =  Ssf_95.overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Sessile  - semifossorial Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Ssf + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



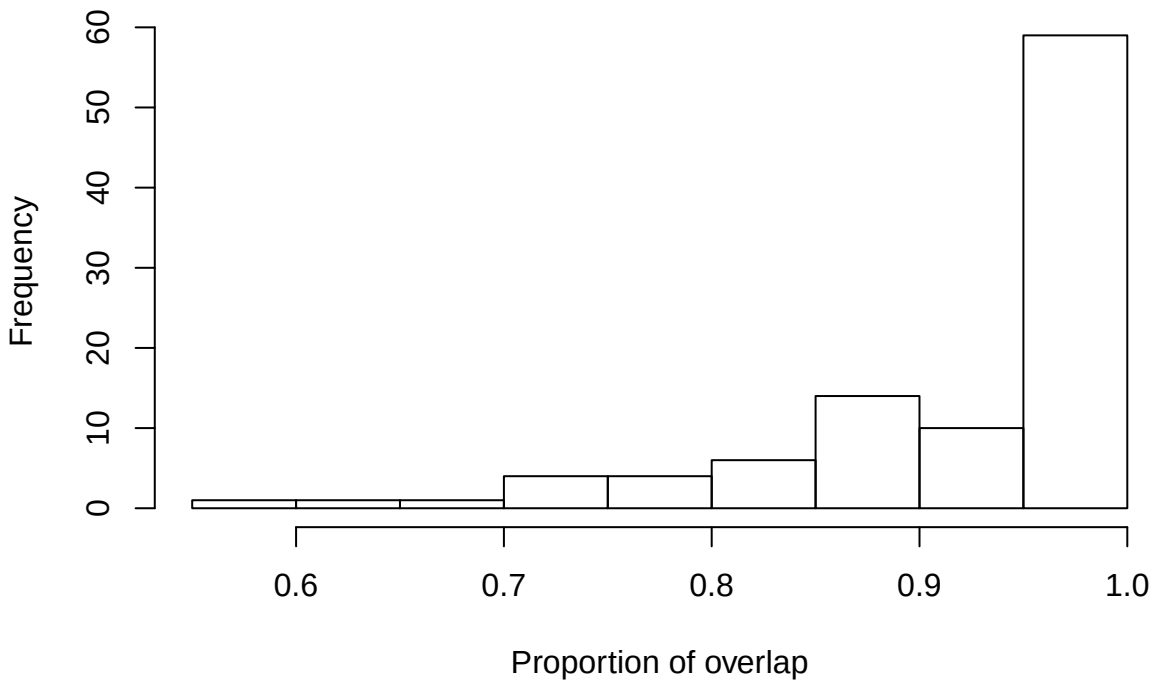

```
#####arboreal  - demersal
Ab_95.overlap <- bayesianOverlap(ellipse_arboreal,
                                 ellipse_benthic,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Ab_95.overlap_prop <- vector()
for(i in 1:length(Ab_95.overlap$overlap)){

Ab_95.overlap_prop[i]  <- Ab_95.overlap$overlap[i]/min(Ab_95.overlap[i,1:2])

}

hist(Ab_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



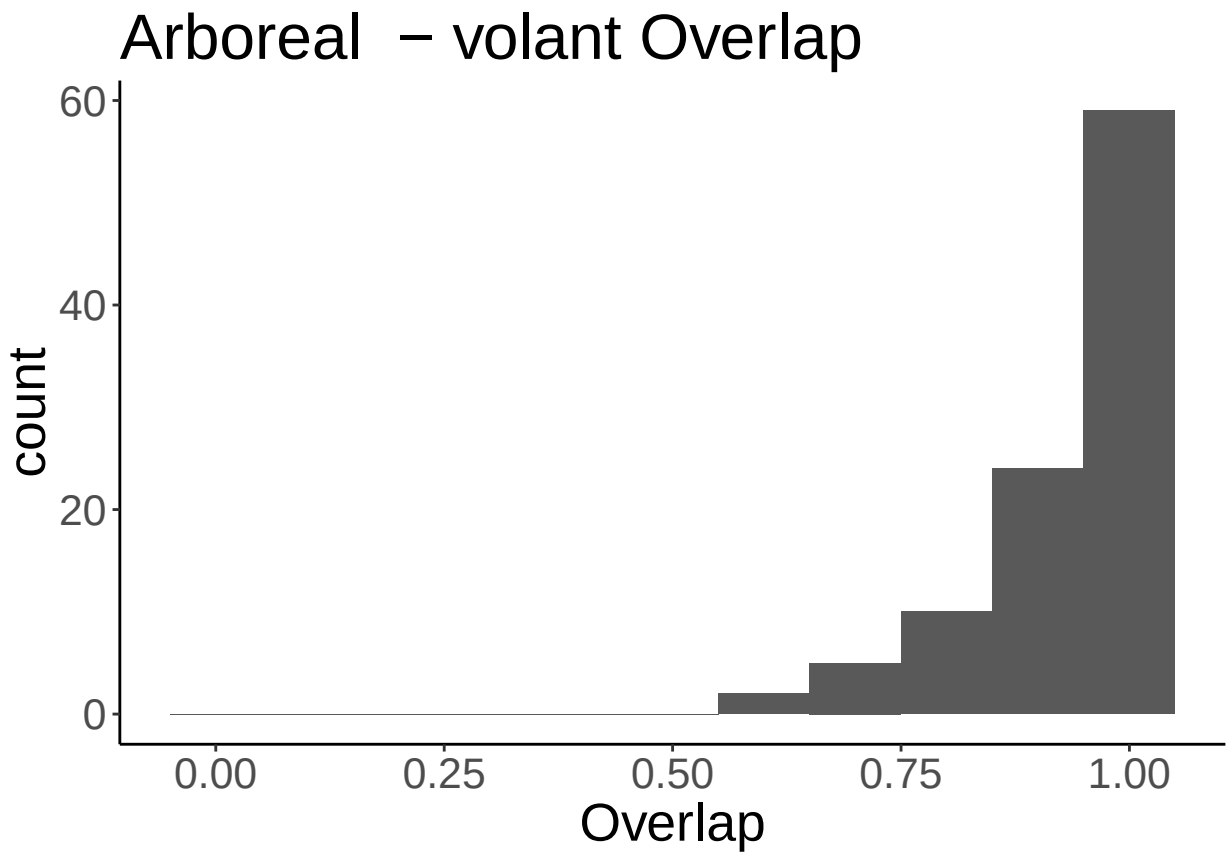

```
myplot_Ab = ggplot(data.frame(Overlap =  Ab_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Arboreal  - demersal Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Ab + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



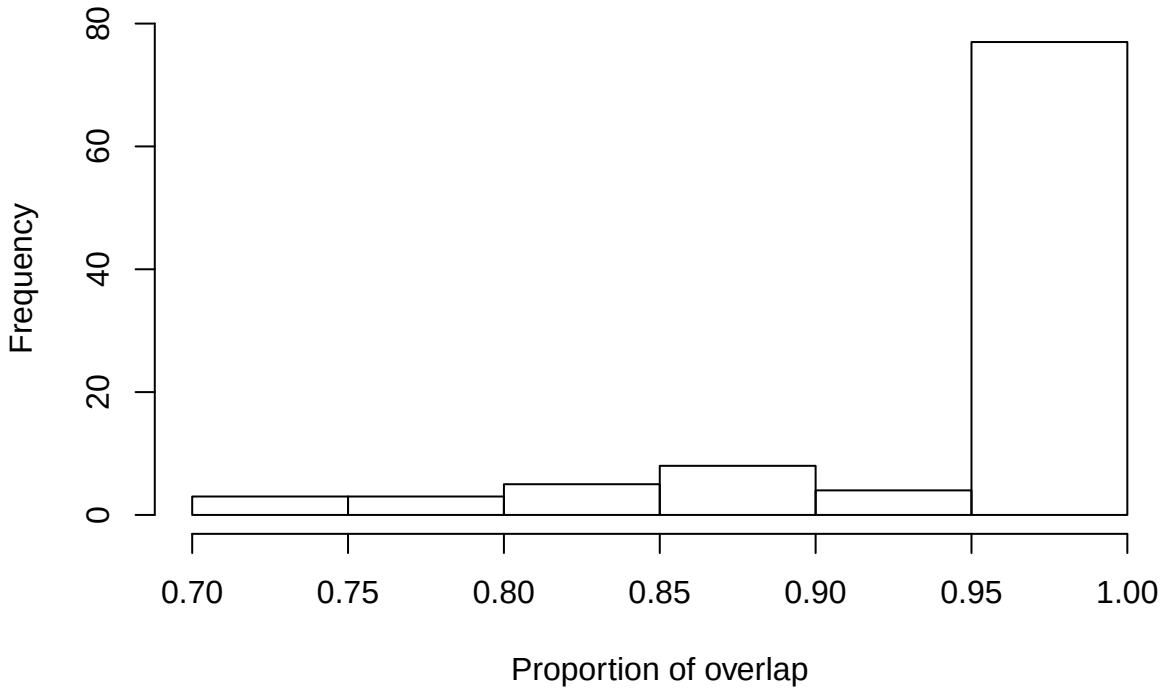

```
#####arboreal  - volant
Av_95.overlap <- bayesianOverlap(ellipse_arboreal,
                                 ellipse_volant,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Av_95.overlap_prop <- vector()
for(i in 1:length(Av_95.overlap$overlap)){

Av_95.overlap_prop[i]  <- Av_95.overlap$overlap[i]/min(Av_95.overlap[i,1:2])

}

hist(Av_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



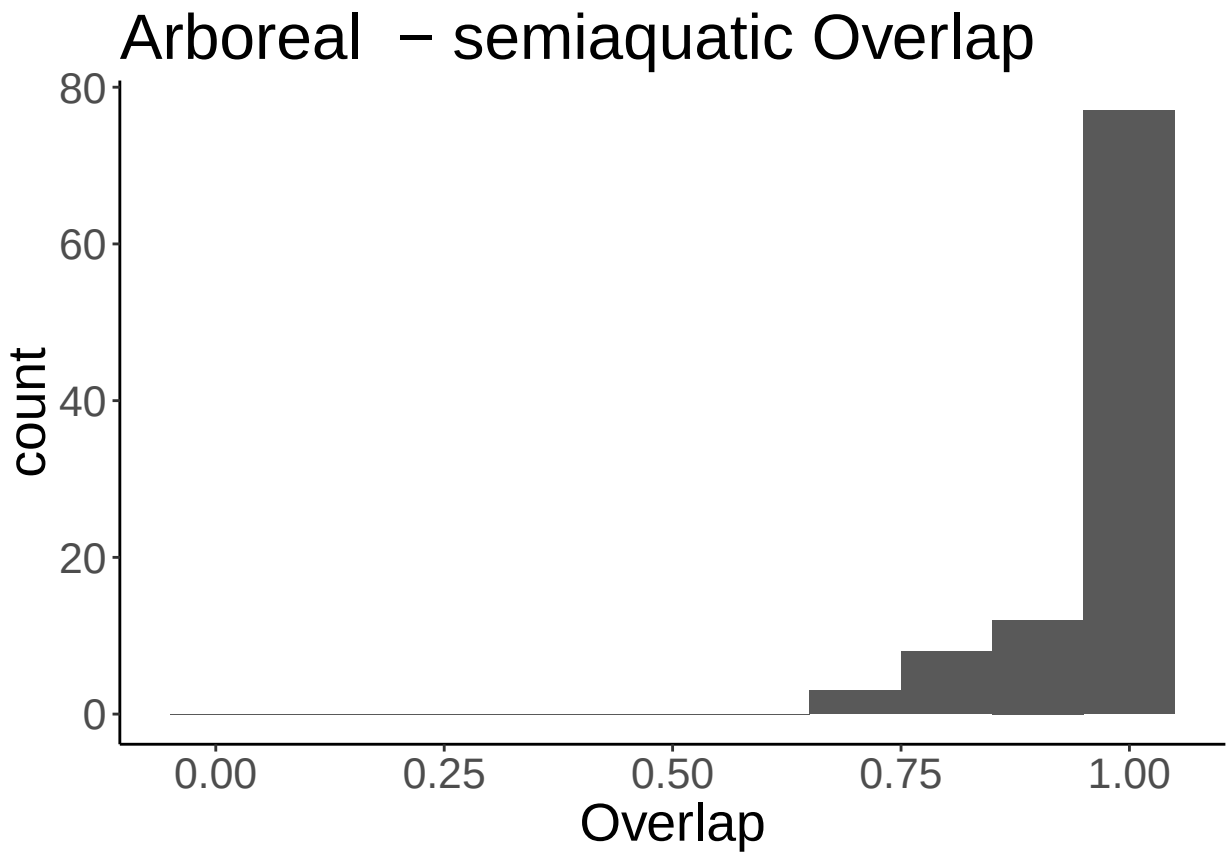

```
myplot_Av = ggplot(data.frame(Overlap =  Av_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Arboreal  - volant Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Av + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



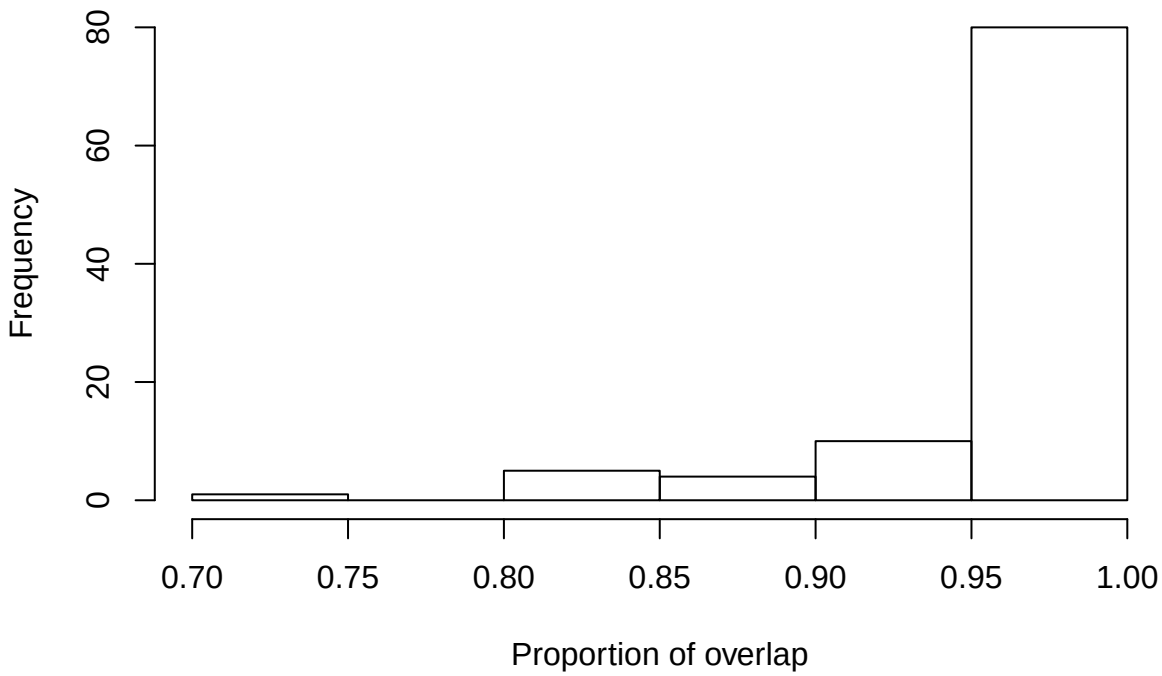

```
#####arboreal  - semiaquatic
Asemi_95.overlap <- bayesianOverlap(ellipse_arboreal,
                                    ellipse_semiaquatic,
                                    ellipses.posterior_mob,
                                    draws = 100,
                                    p.interval = 0.95,
                                    n = 100)
Asemi_95.overlap_prop <- vector()
for(i in 1:length(Av_95.overlap$overlap)){

Asemi_95.overlap_prop[i]  <- Asemi_95.overlap$overlap[i]/min(Asemi_95.overlap[i,1:2])

}

hist(Asemi_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



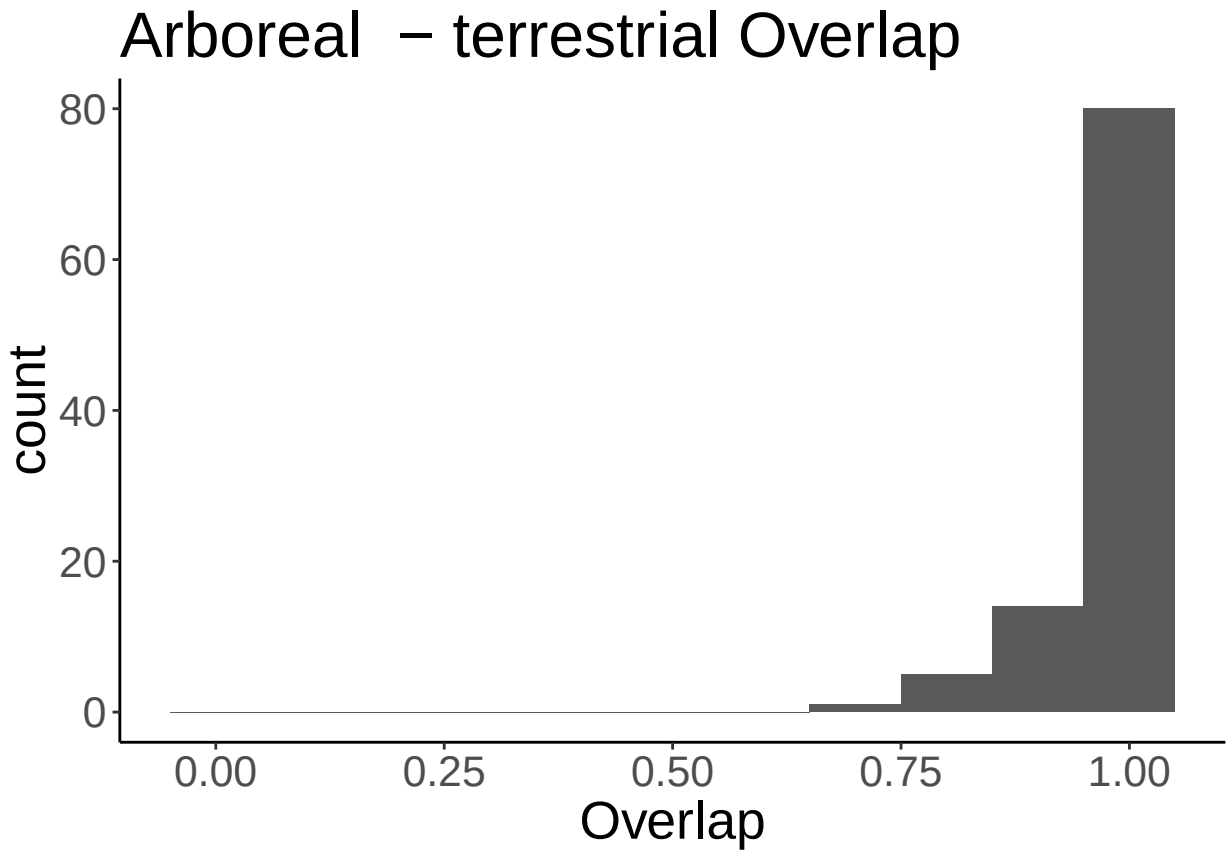

```
myplot_Asemi = ggplot(data.frame(Overlap =  Asemi_95.overlap_prop),
                      aes(Overlap)) +
                      ggtitle("Arboreal  - semiaquatic Overlap") +
                      geom_histogram(binwidth=0.1) +
                      scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Asemi + theme_bw() + theme(panel.border = element_blank(),
                                  panel.grid.major = element_blank(),
                                  panel.grid.minor = element_blank(),
                                  axis.line = element_line(colour = "black"),
                                  text = element_text(size=20))
```



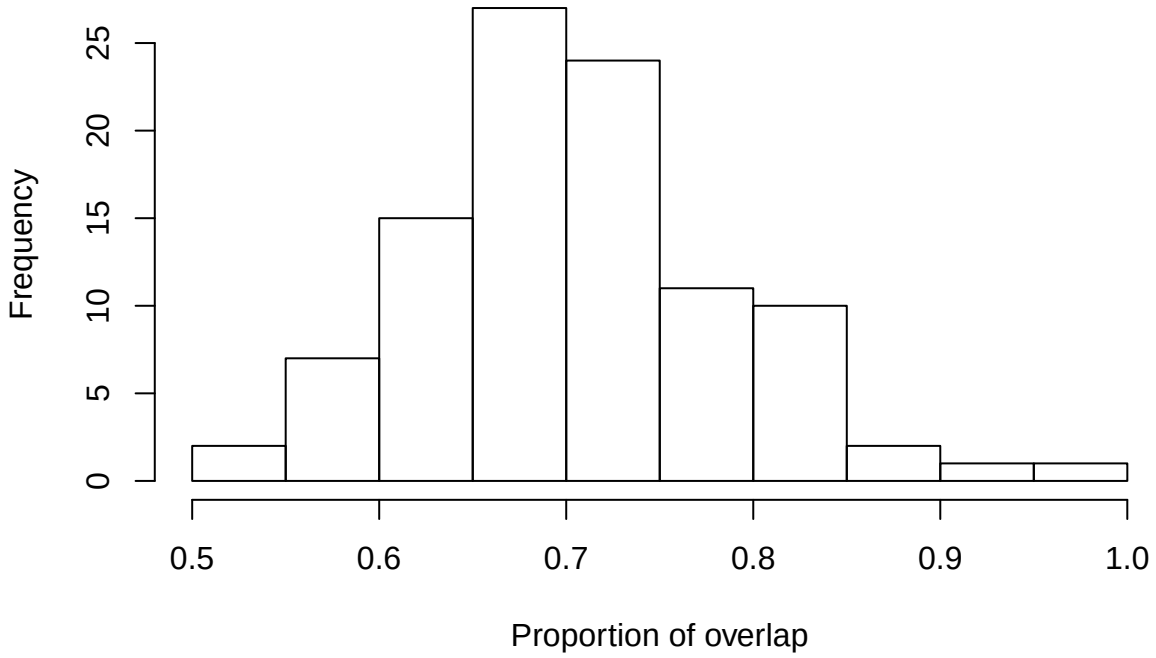

```
#####arboreal  - terrestrial
At_95.overlap <- bayesianOverlap(ellipse_arboreal,
                                 ellipse_terrestrial,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
At_95.overlap_prop <- vector()
for(i in 1:length(At_95.overlap$overlap)){

At_95.overlap_prop[i]  <- At_95.overlap$overlap[i]/min(At_95.overlap[i,1:2])

}

hist(At_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



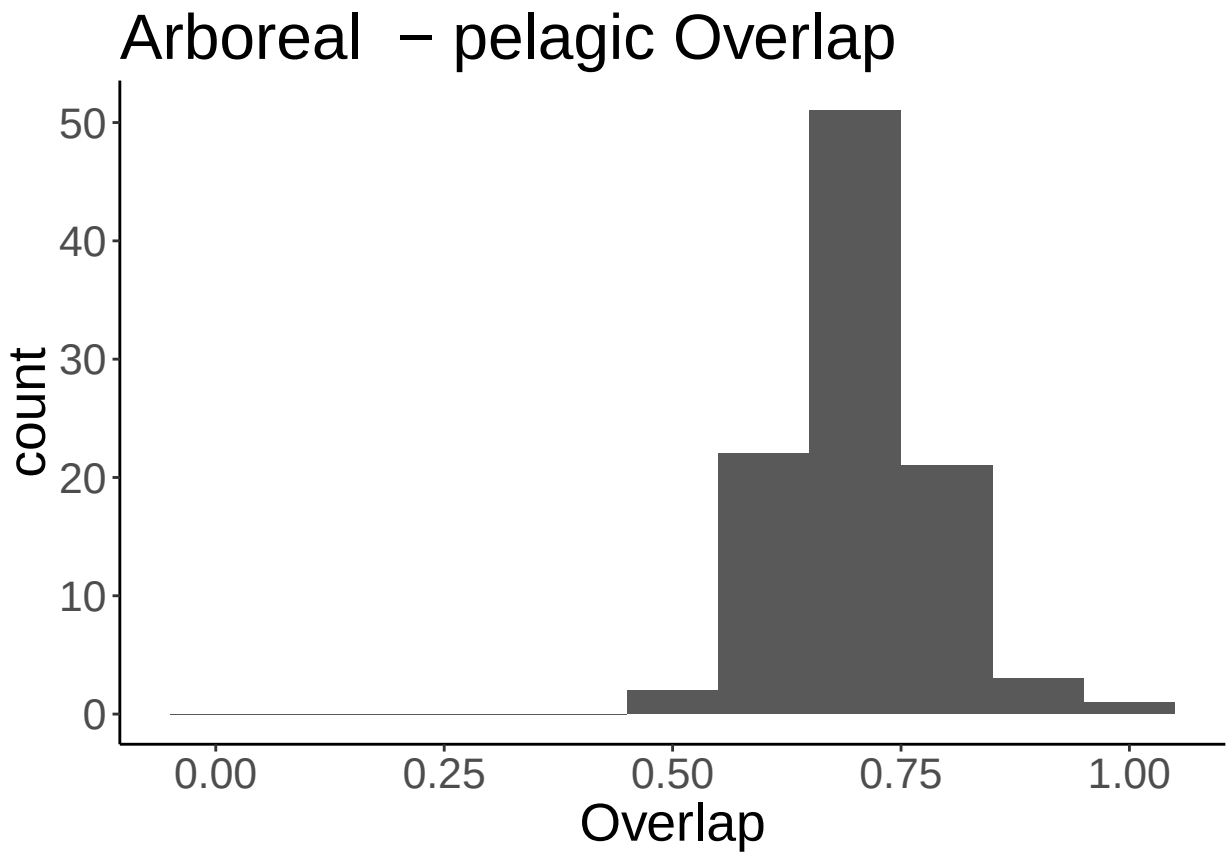

```
myplot_At = ggplot(data.frame(Overlap =  At_95.overlap_prop),
                   aes(Overlap)) +
                   geom_histogram(binwidth=0.1) +
                   ggtitle("Arboreal  - terrestrial Overlap") +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_At + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



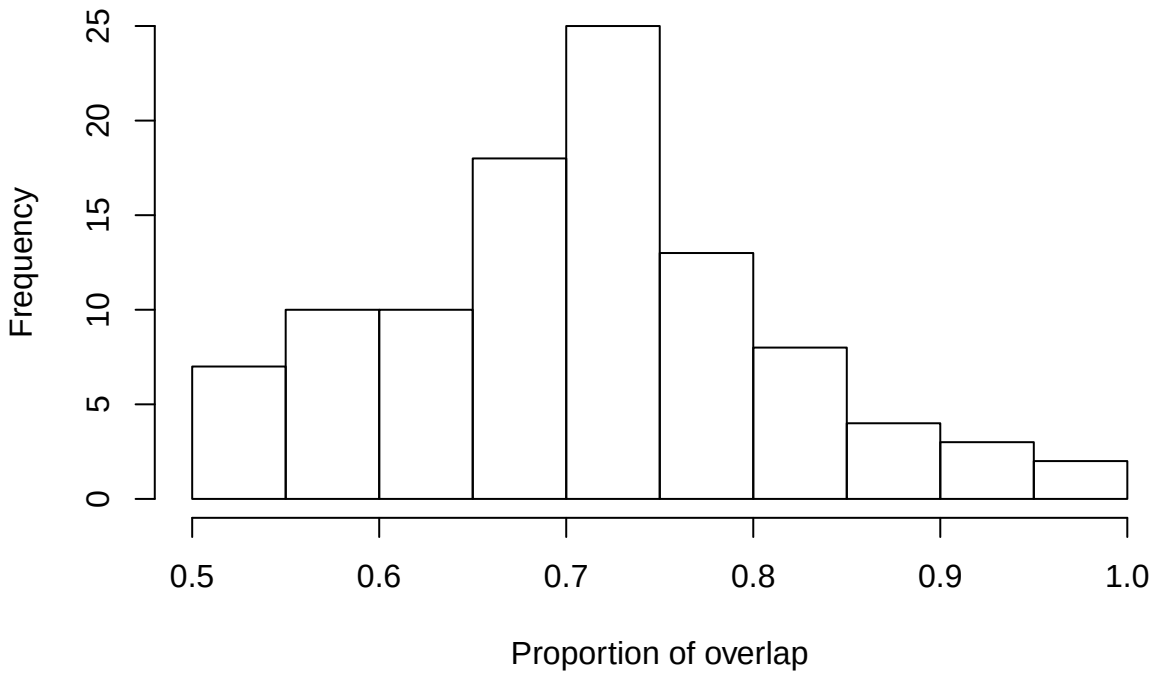

```
#####arboreal  - pelagic
Ap_95.overlap <- bayesianOverlap(ellipse_arboreal,
                                 ellipse_pelagic,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Ap_95.overlap_prop <- vector()
for(i in 1:length(Ap_95.overlap$overlap)){

Ap_95.overlap_prop[i]  <- Ap_95.overlap$overlap[i]/min(Ap_95.overlap[i,1:2])

}

hist(Ap_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



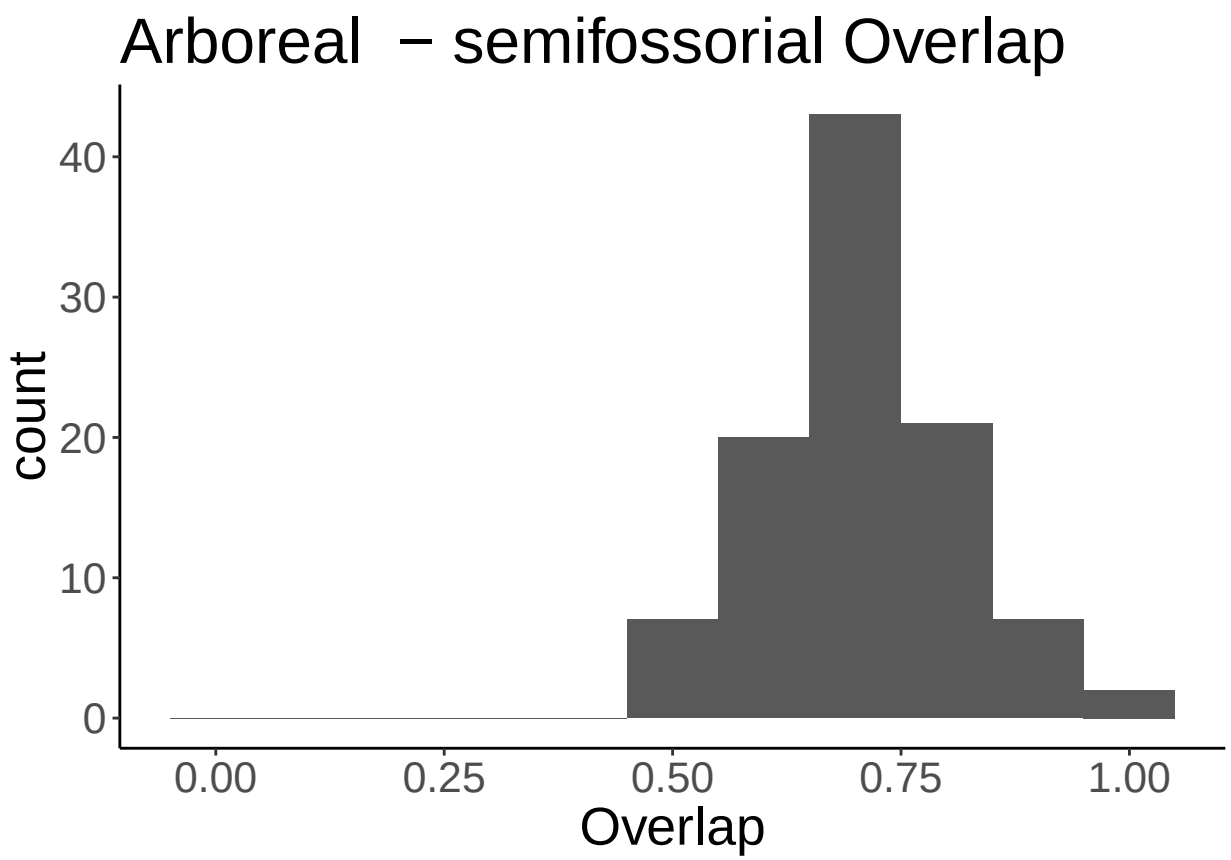

```
myplot_Ap = ggplot(data.frame(Overlap =  Ap_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Arboreal  - pelagic Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Ap + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



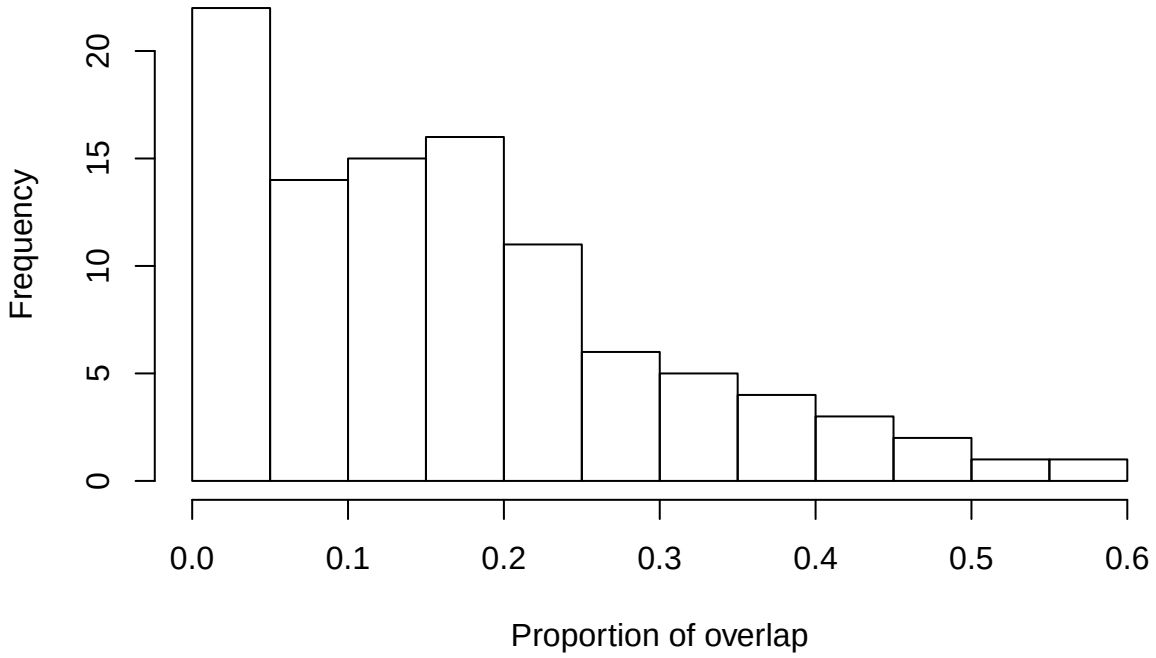

```
#####arboreal  - semifossorial
Afoss_95.overlap <- bayesianOverlap(ellipse_arboreal,
                                    ellipse_semifossorial,
                                    ellipses.posterior_mob,
                                    draws = 100,
                                    p.interval = 0.95,
                                    n = 100)
Afoss_95.overlap_prop <- vector()
for(i in 1:length(Afoss_95.overlap$overlap)){

Afoss_95.overlap_prop[i]  <- Afoss_95.overlap$overlap[i]/min(Afoss_95.overlap[i,1:2])

}

hist(Afoss_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



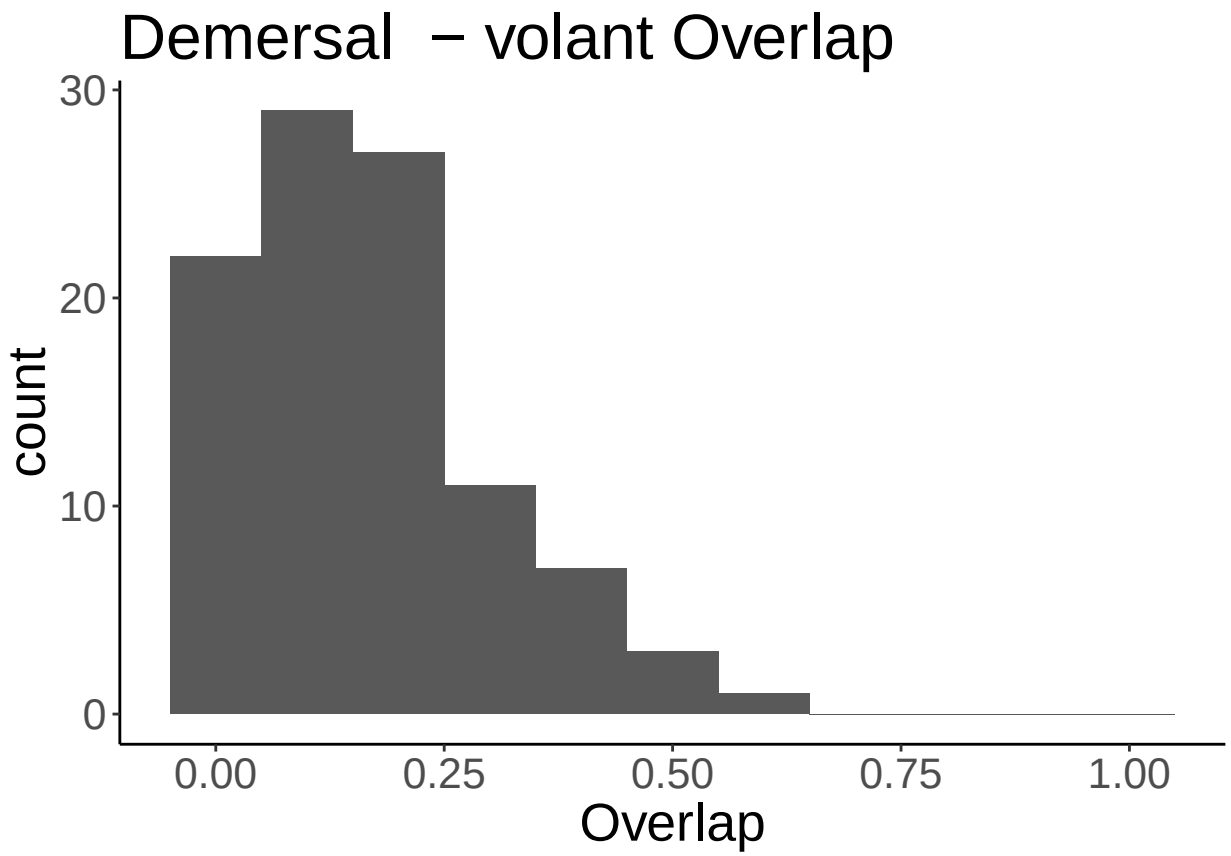

```
myplot_Afoss = ggplot(data.frame(Overlap =  Afoss_95.overlap_prop),
                      aes(Overlap)) +
                        ggtitle("Arboreal  - semifossorial Overlap") +

  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Afoss + theme_bw() + theme(panel.border = element_blank(),
                                  panel.grid.major = element_blank(),
                                  panel.grid.minor = element_blank(),
                                  axis.line = element_line(colour = "black"),
                                  text = element_text(size=20))
```



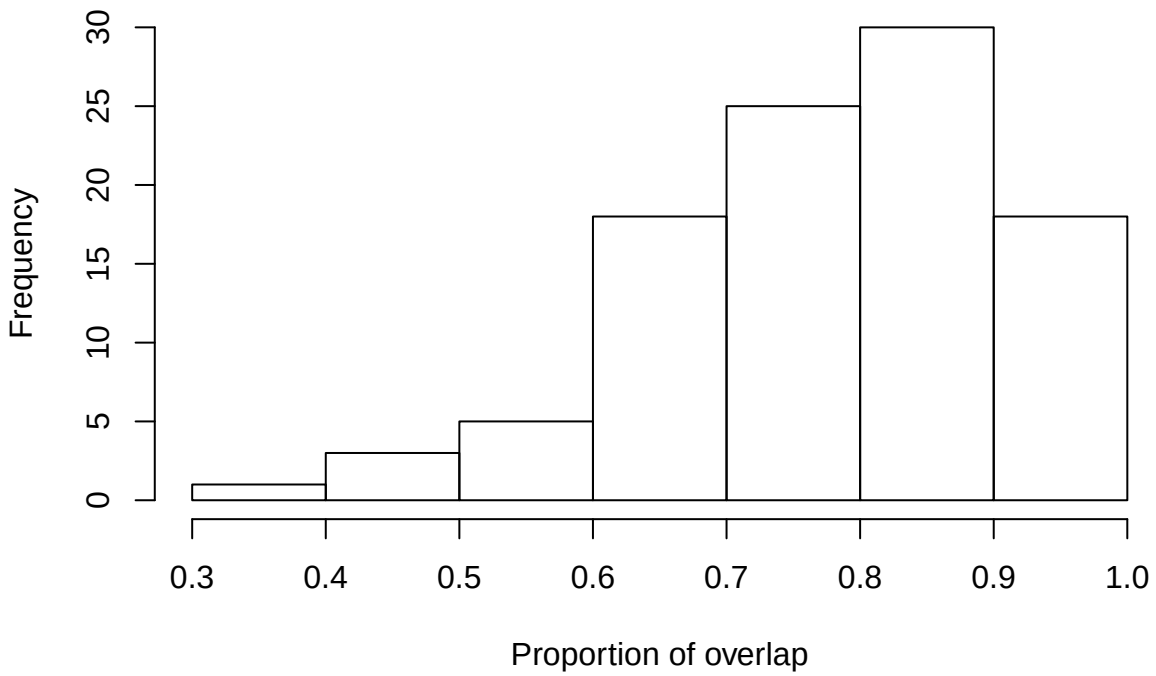

```
#####benthic  - volant
Bv_95.overlap <- bayesianOverlap(ellipse_benthic,
                                 ellipse_volant,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Bv_95.overlap_prop <- vector()
for(i in 1:length(Bv_95.overlap$overlap)){

Bv_95.overlap_prop[i]  <- Bv_95.overlap$overlap[i]/min(Bv_95.overlap[i,1:2])

}

hist(Bv_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



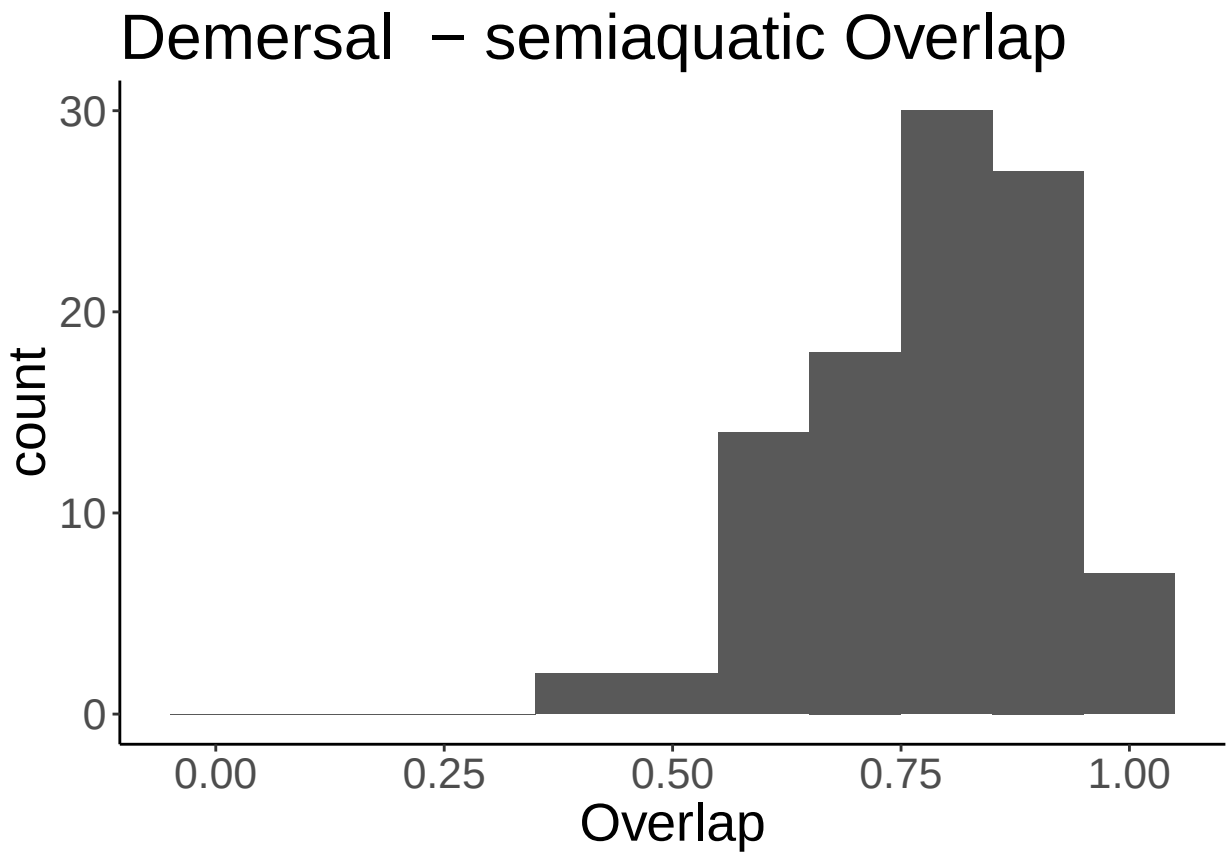

```
myplot_Bv = ggplot(data.frame(Overlap =  Bv_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Demersal  - volant Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Bv + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



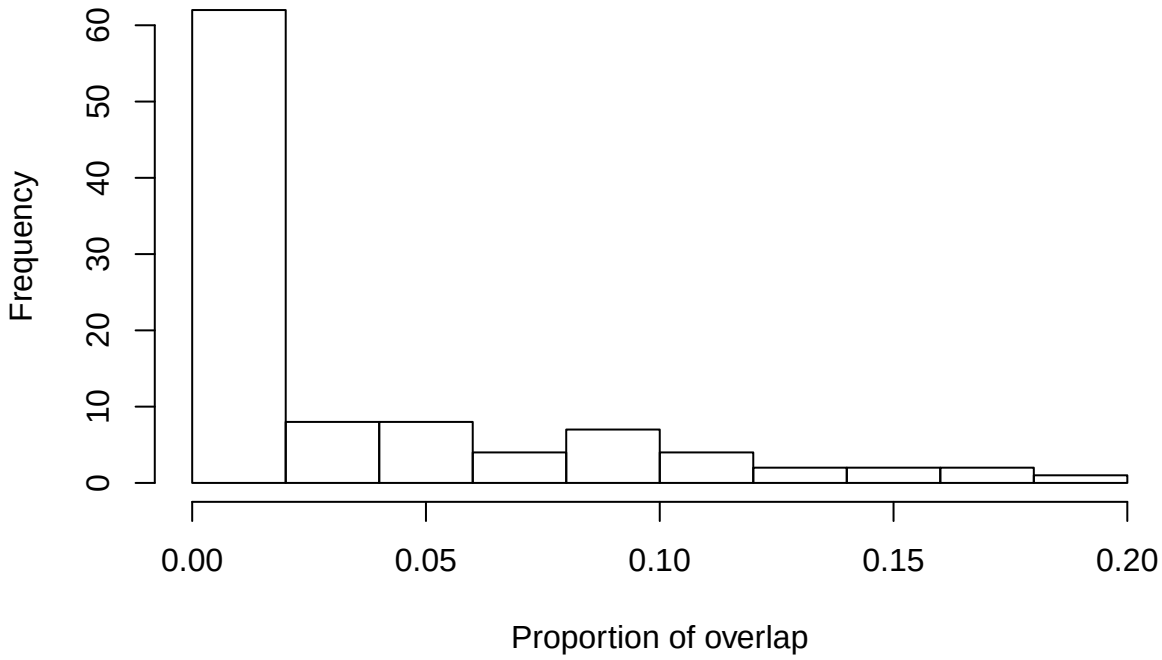

```
#####benthic  - semiaquatic
Bsemia_95.overlap <- bayesianOverlap(ellipse_benthic,
                                     ellipse_semiaquatic,
                                     ellipses.posterior_mob,
                                     draws = 100,
                                     p.interval = 0.95,
                                     n = 100)
Bsemia_95.overlap_prop <- vector()
for(i in 1:length(Bsemia_95.overlap$overlap)){

Bsemia_95.overlap_prop[i]  <- Bsemia_95.overlap$overlap[i]/min(Bsemia_95.overlap[i,1:2])

}

hist(Bsemia_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



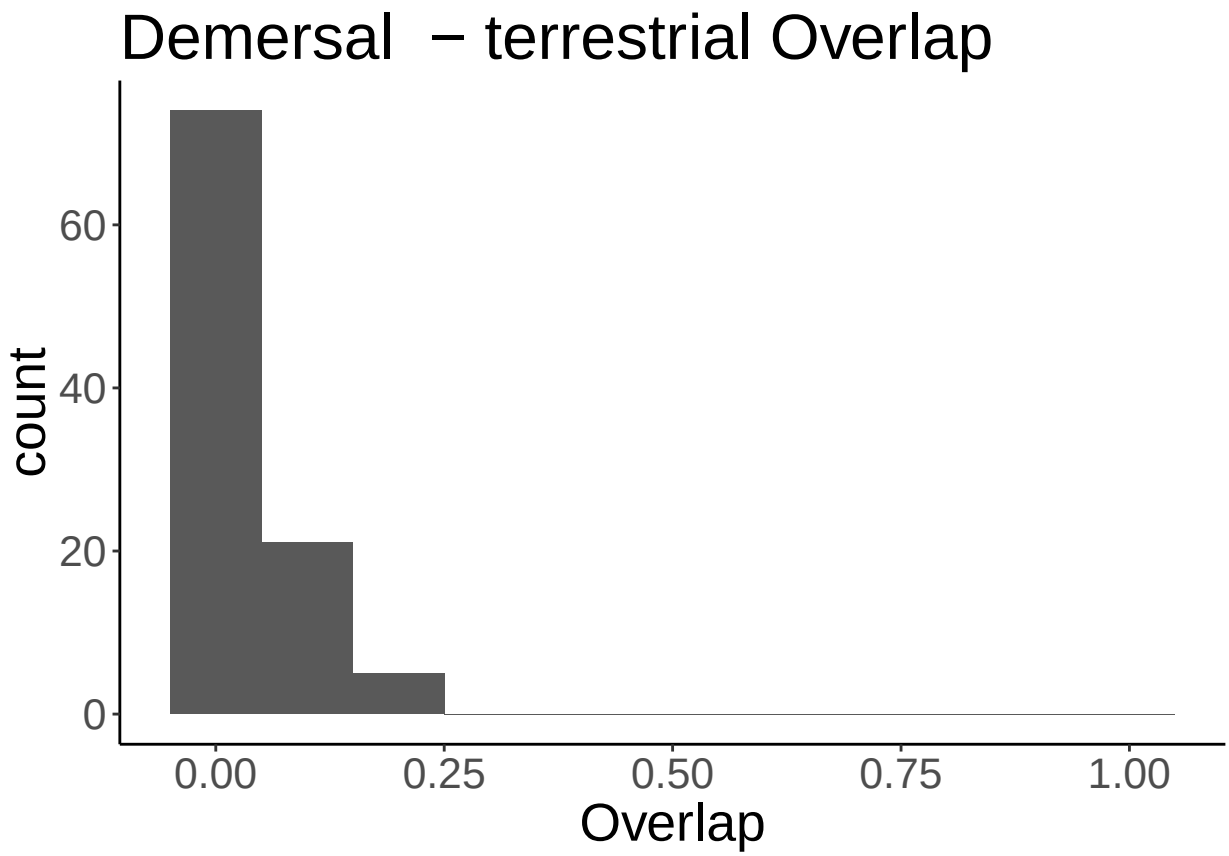

```
myplot_Bsemia = ggplot(data.frame(Overlap =  Bsemia_95.overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Demersal  - semiaquatic Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Bsemia + theme_bw() + theme(panel.border = element_blank(),
                                   panel.grid.major = element_blank(),
                                   panel.grid.minor = element_blank(),
                                   axis.line = element_line(colour = "black"),
                                   text = element_text(size=20))
```



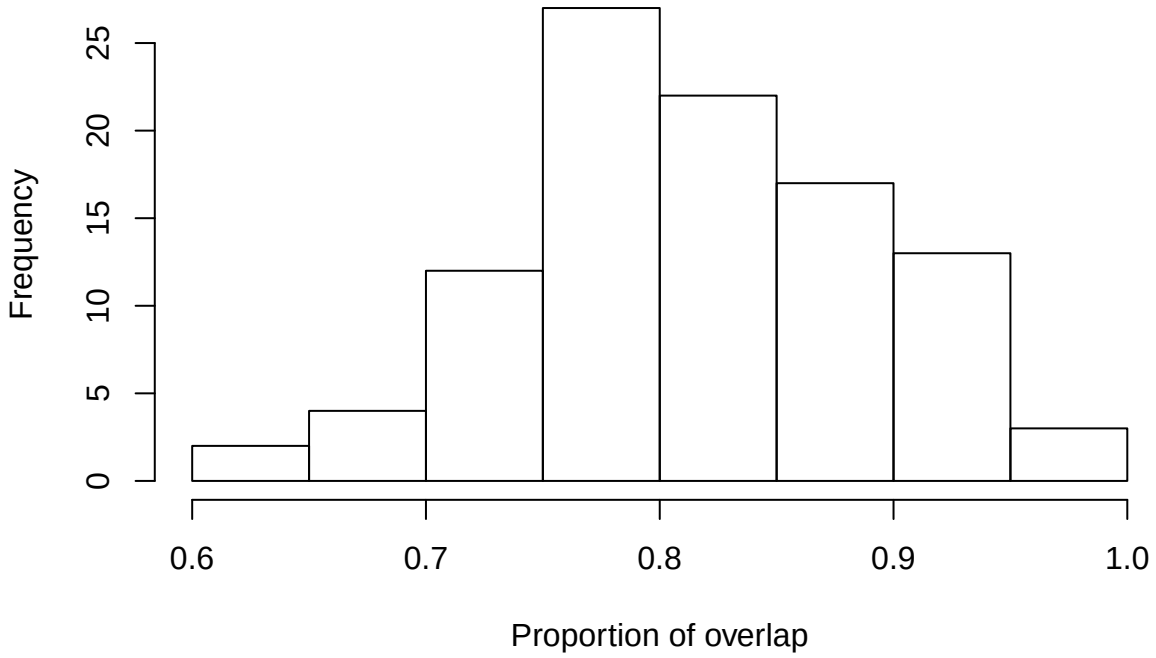

```
#####benthic  - terrestrial
Bt_95.overlap <- bayesianOverlap(ellipse_benthic,
                                 ellipse_terrestrial,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Bt_95.overlap_prop <- vector()
for(i in 1:length(Bt_95.overlap$overlap)){

Bt_95.overlap_prop[i]  <- Bt_95.overlap$overlap[i]/min(Bt_95.overlap[i,1:2])

}

hist(Bt_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



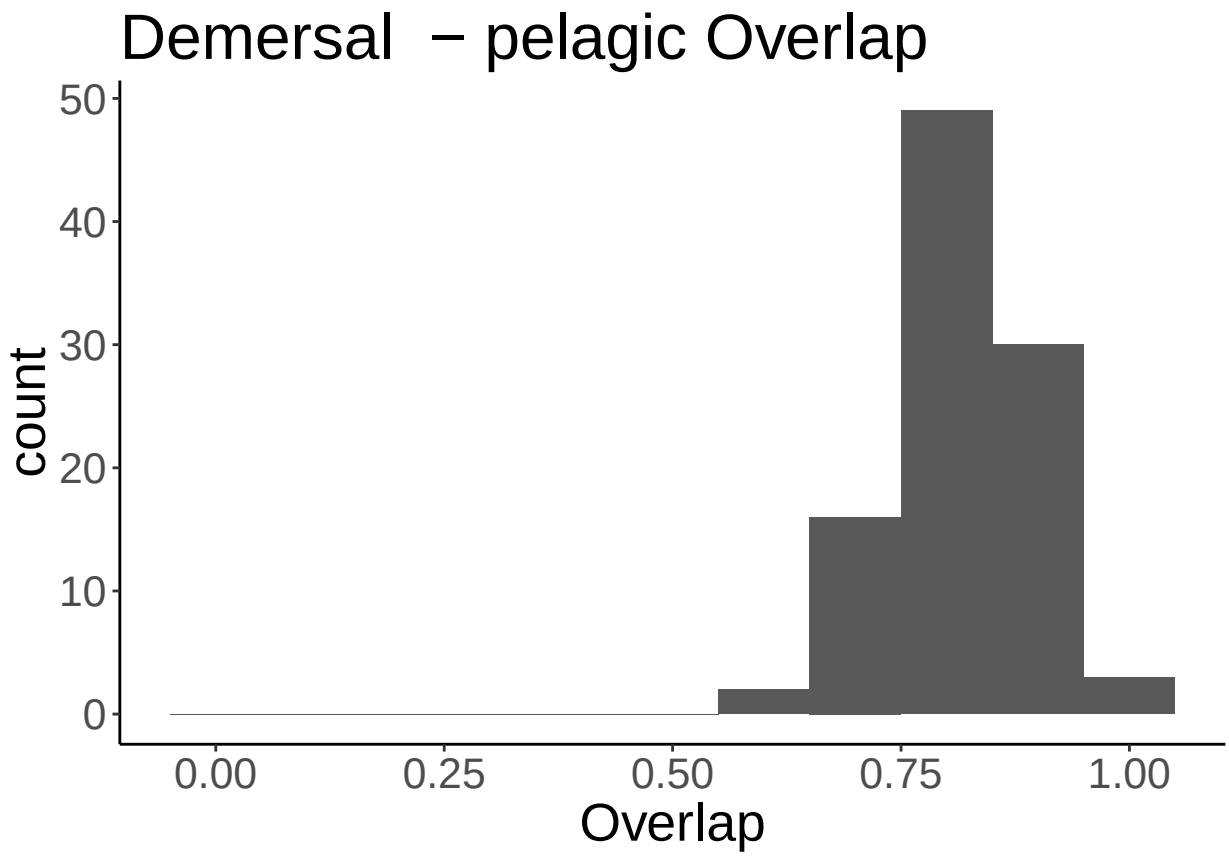

```
myplot_Bt = ggplot(data.frame(Overlap =  Bt_95.overlap_prop),
                   aes(Overlap)) +
                     ggtitle("Demersal  - terrestrial Overlap") +

  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Bt + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



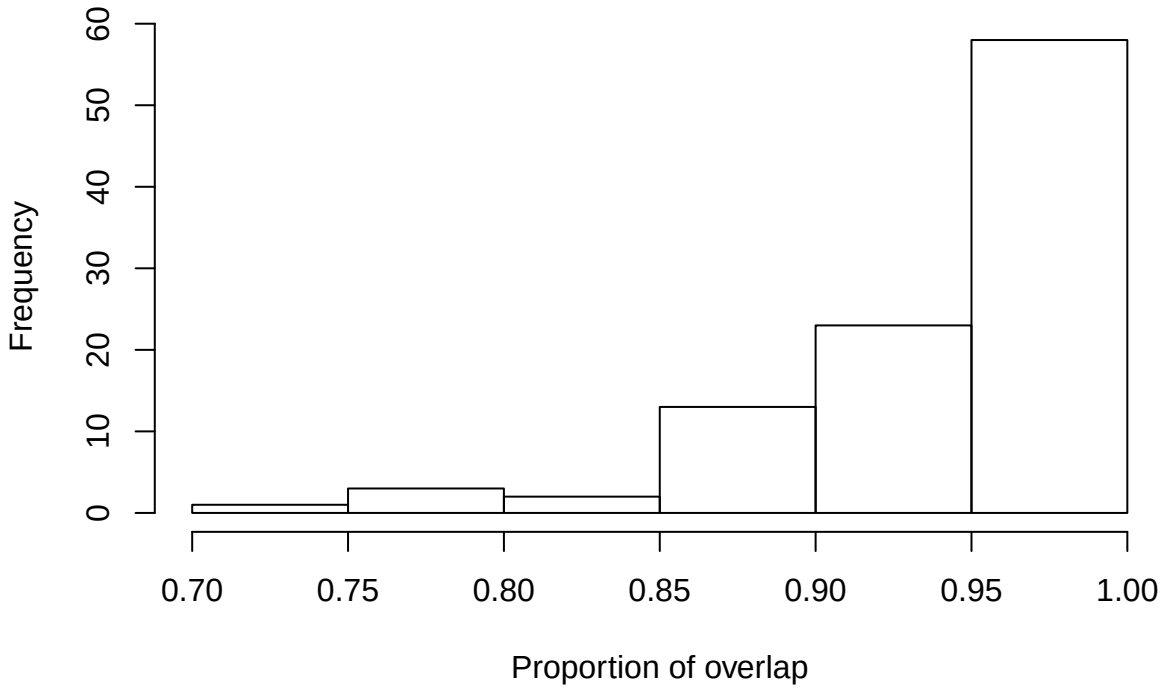

```
#####benthic  - pelagic
Bp_95.overlap <- bayesianOverlap(ellipse_benthic,
                                 ellipse_pelagic,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Bp_95.overlap_prop <- vector()
for(i in 1:length(Bp_95.overlap$overlap)){

Bp_95.overlap_prop[i]  <- Bp_95.overlap$overlap[i]/min(Bp_95.overlap[i,1:2])

}

hist(Bp_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



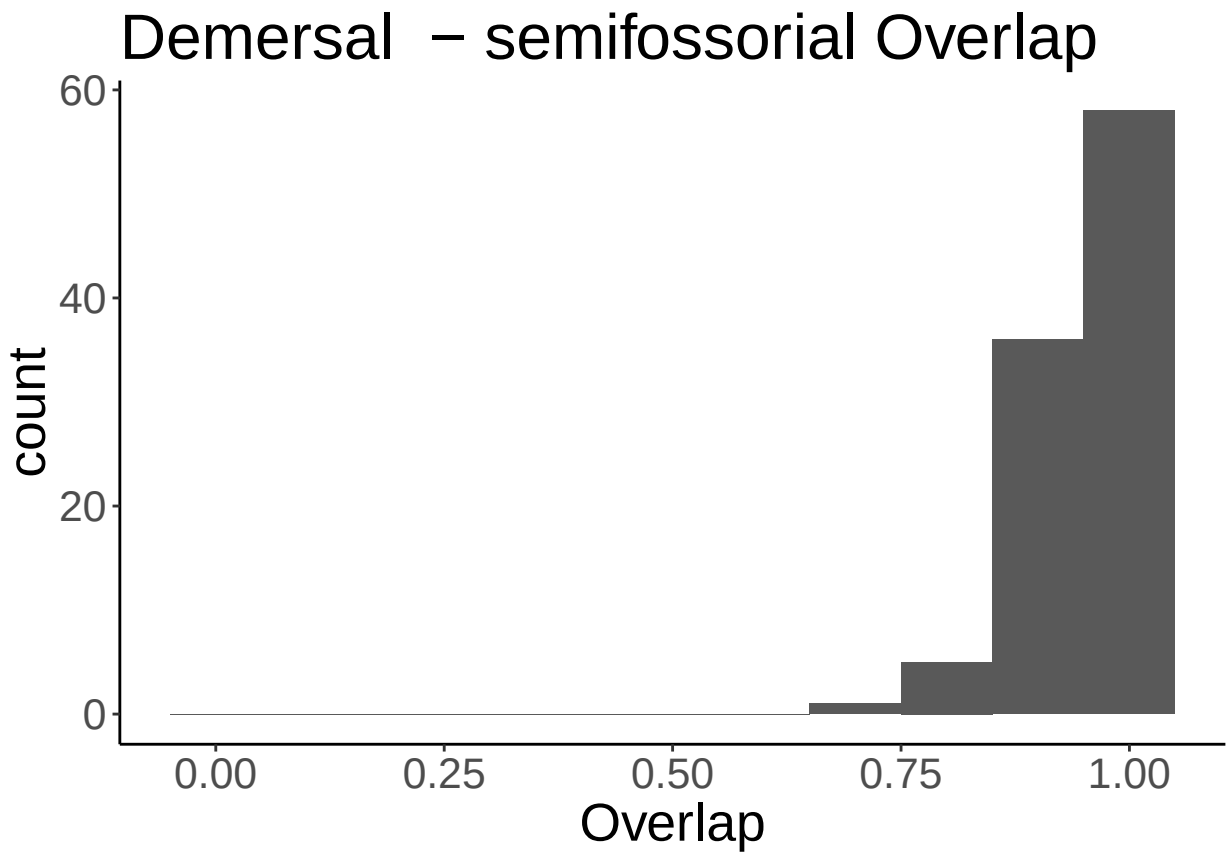

```
myplot_Bp = ggplot(data.frame(Overlap =  Bp_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Demersal  - pelagic Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))


myplot_Bp + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



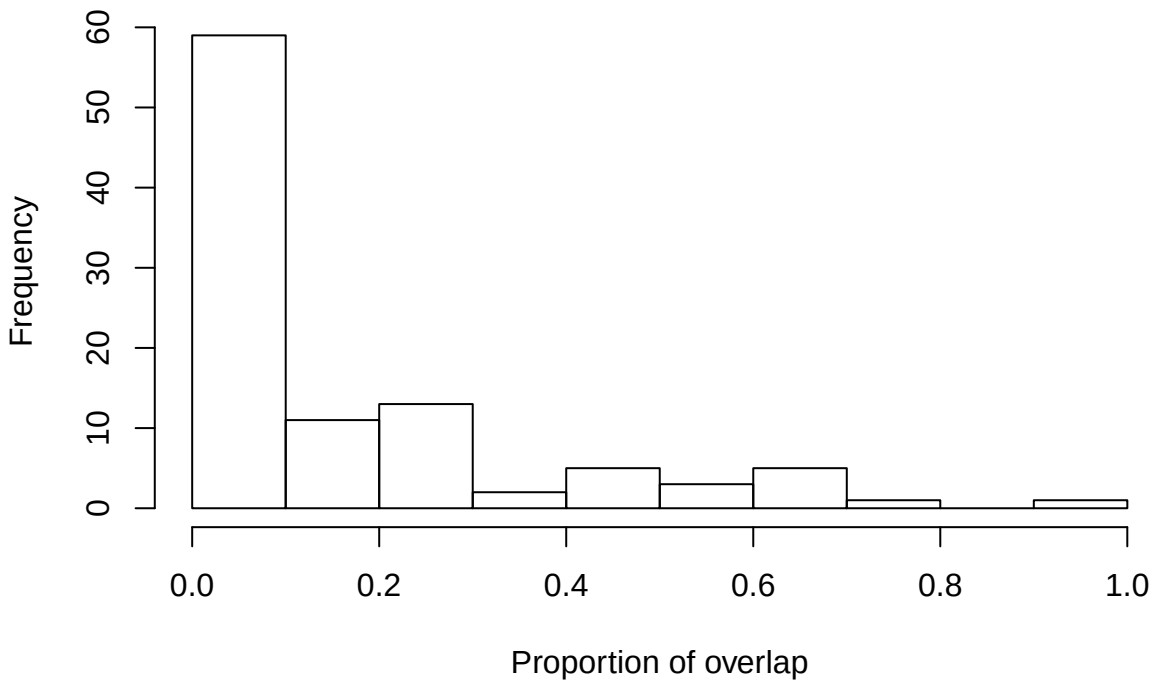

```
#####benthic  - semifossorial
Bfoss_95.overlap <- bayesianOverlap(ellipse_benthic,
                                    ellipse_semifossorial,
                                    ellipses.posterior_mob,
                                    draws = 100,
                                    p.interval = 0.95,
                                    n = 100)
Bfoss_95.overlap_prop <- vector()
for(i in 1:length(Bfoss_95.overlap$overlap)){

Bfoss_95.overlap_prop[i]  <- Bfoss_95.overlap$overlap[i]/min(Bfoss_95.overlap[i,1:2])

}

hist(Bfoss_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



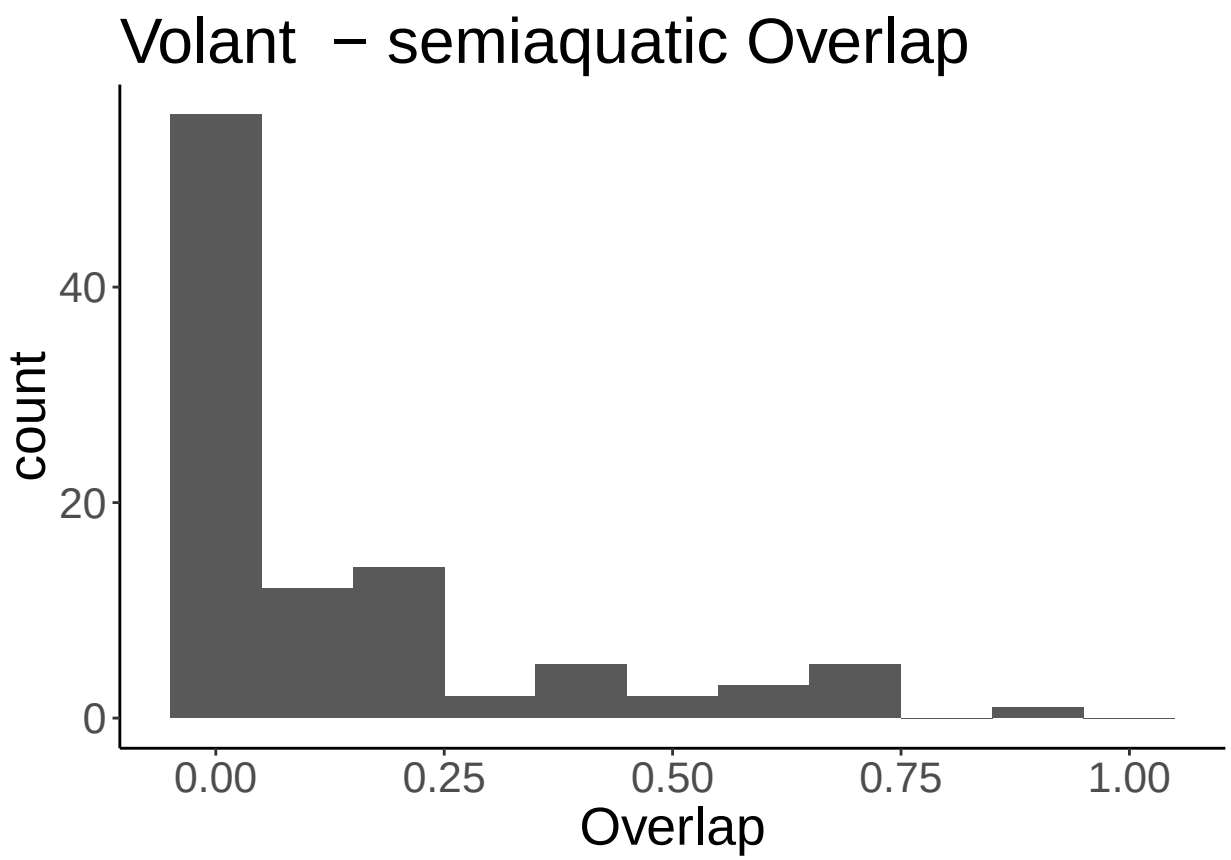

```
myplot_Bfoss = ggplot(data.frame(Overlap =  Bfoss_95.overlap_prop),
                      aes(Overlap)) +
                      ggtitle("Demersal  - semifossorial Overlap") +
                      geom_histogram(binwidth=0.1) +
                      scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Bfoss + theme_bw() + theme(panel.border = element_blank(),
                                  panel.grid.major = element_blank(),
                                  panel.grid.minor = element_blank(),
                                  axis.line = element_line(colour = "black"),
                                  text = element_text(size=20))
```



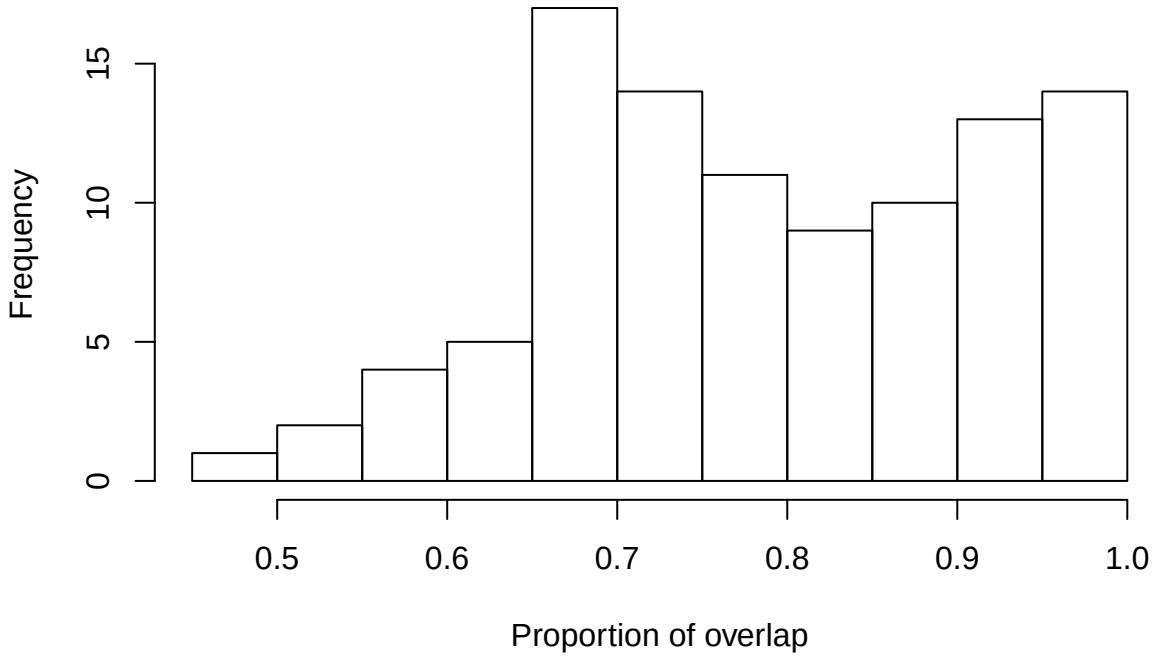

```
#####volant  - semiaquatic
Vsem_95.overlap <- bayesianOverlap(ellipse_volant,
                                   ellipse_semiaquatic,
                                   ellipses.posterior_mob,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
Vsem_95.overlap_prop <- vector()
for(i in 1:length(Vsem_95.overlap$overlap)){

Vsem_95.overlap_prop[i]  <- Vsem_95.overlap$overlap[i]/min(Vsem_95.overlap[i,1:2])

}

hist(Vsem_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



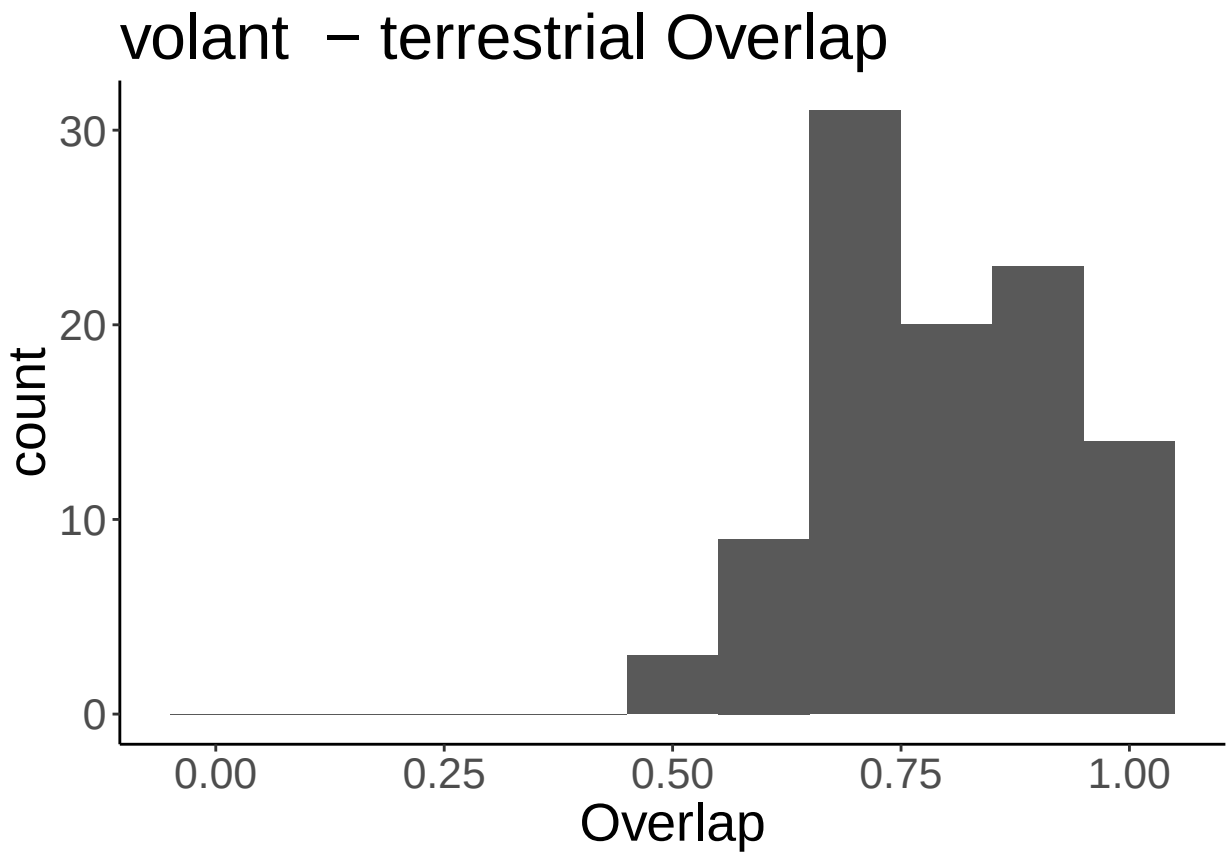

```
myplot_Vsem = ggplot(data.frame(Overlap =  Vsem_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Volant  - semiaquatic Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))


myplot_Vsem + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



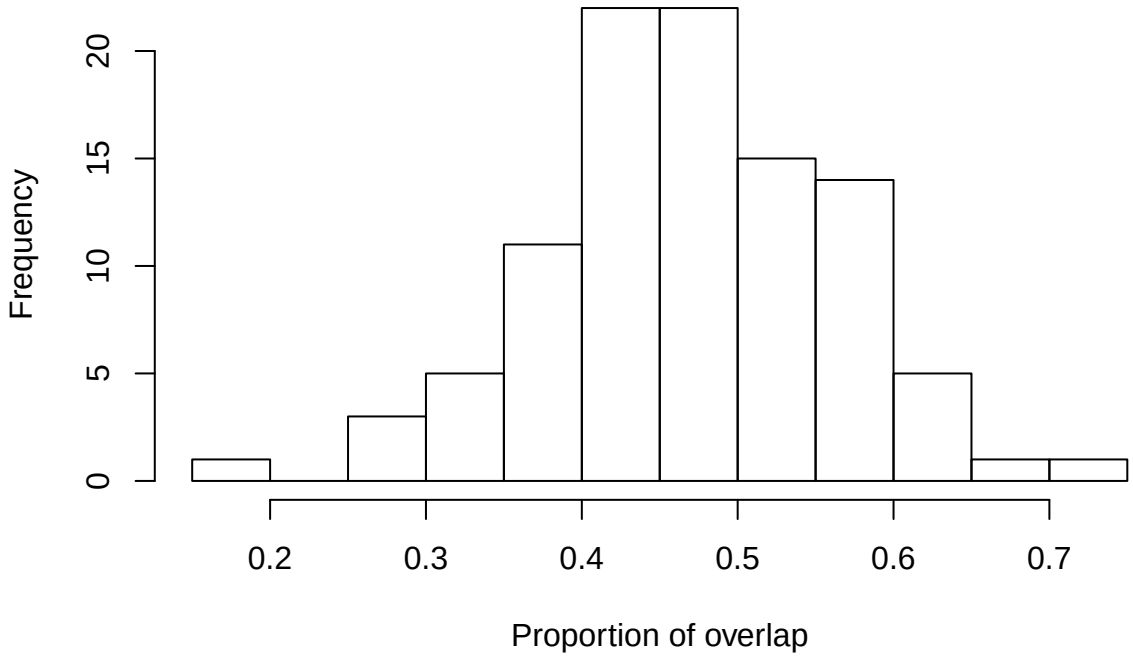

```
#####volant  - terrestrial
Vt_95.overlap <- bayesianOverlap(ellipse_volant,
                                 ellipse_terrestrial,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Vt_95.overlap_prop <- vector()
for(i in 1:length(Vt_95.overlap$overlap)){

Vt_95.overlap_prop[i]  <- Vt_95.overlap$overlap[i]/min(Vt_95.overlap[i,1:2])

}

hist(Vt_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



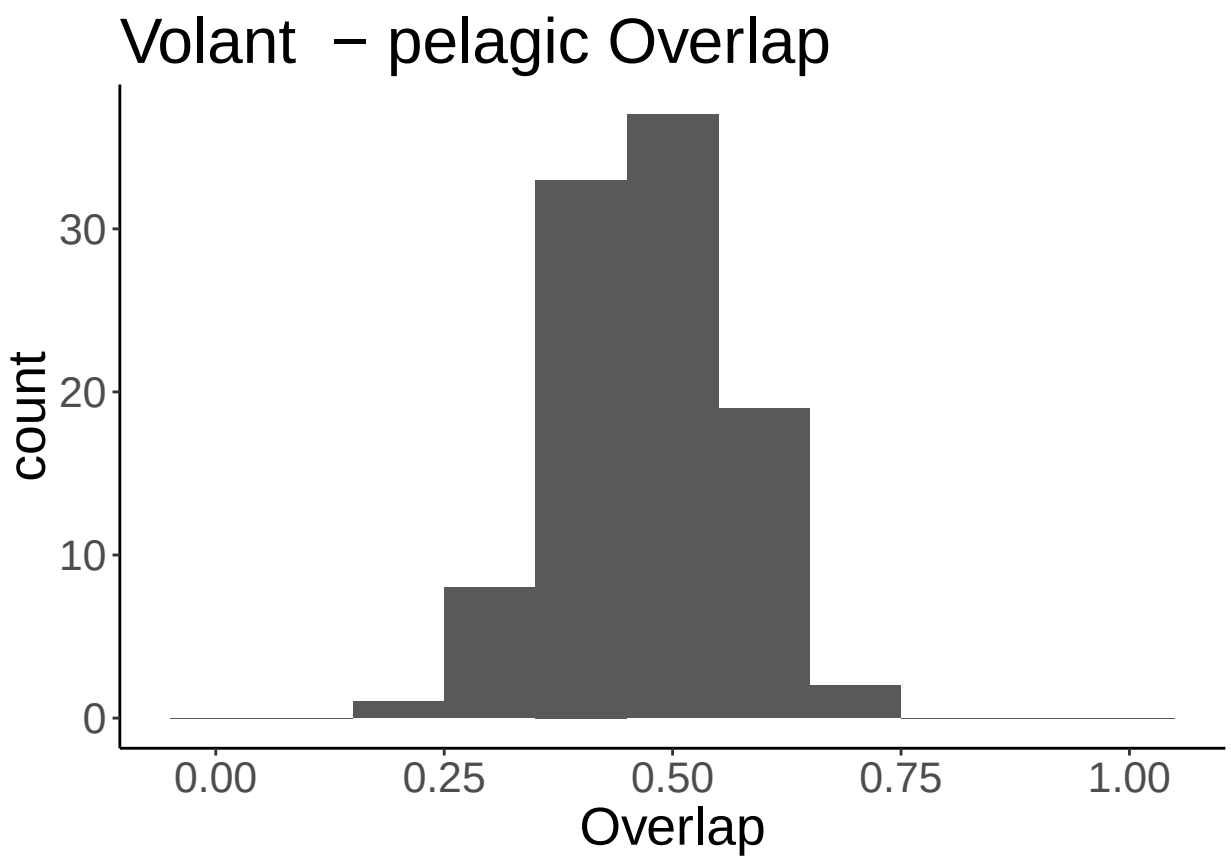

```
myplot_Vt = ggplot(data.frame(Overlap =  Vt_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("volant  - terrestrial Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Vt + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



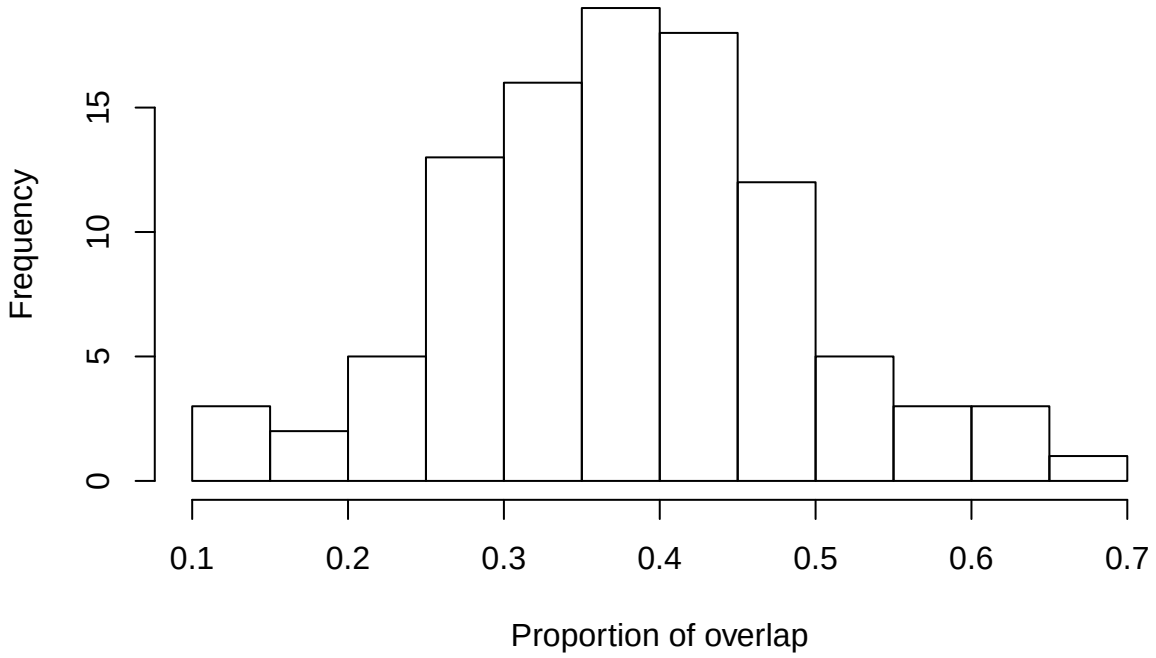

```
#####volant  - pelagic
Vp_95.overlap <- bayesianOverlap(ellipse_volant,
                                 ellipse_pelagic,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)

Vp_95.overlap_prop <- vector()
for(i in 1:length(Vp_95.overlap$overlap)){

Vp_95.overlap_prop[i]  <- Vp_95.overlap$overlap[i]/min(Vp_95.overlap[i,1:2])

}

hist(Vp_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



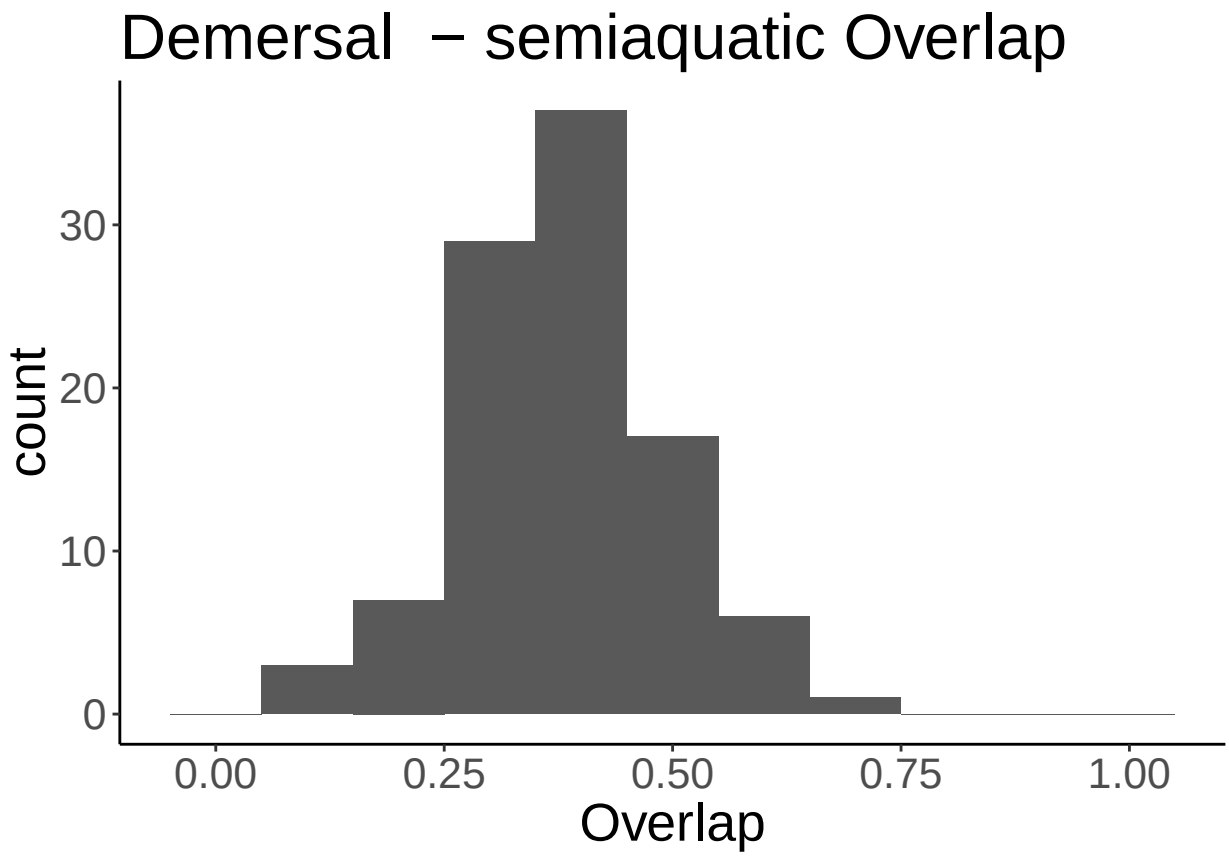

```
myplot_Vp = ggplot(data.frame(Overlap =  Vp_95.overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Volant  - pelagic Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Vp + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



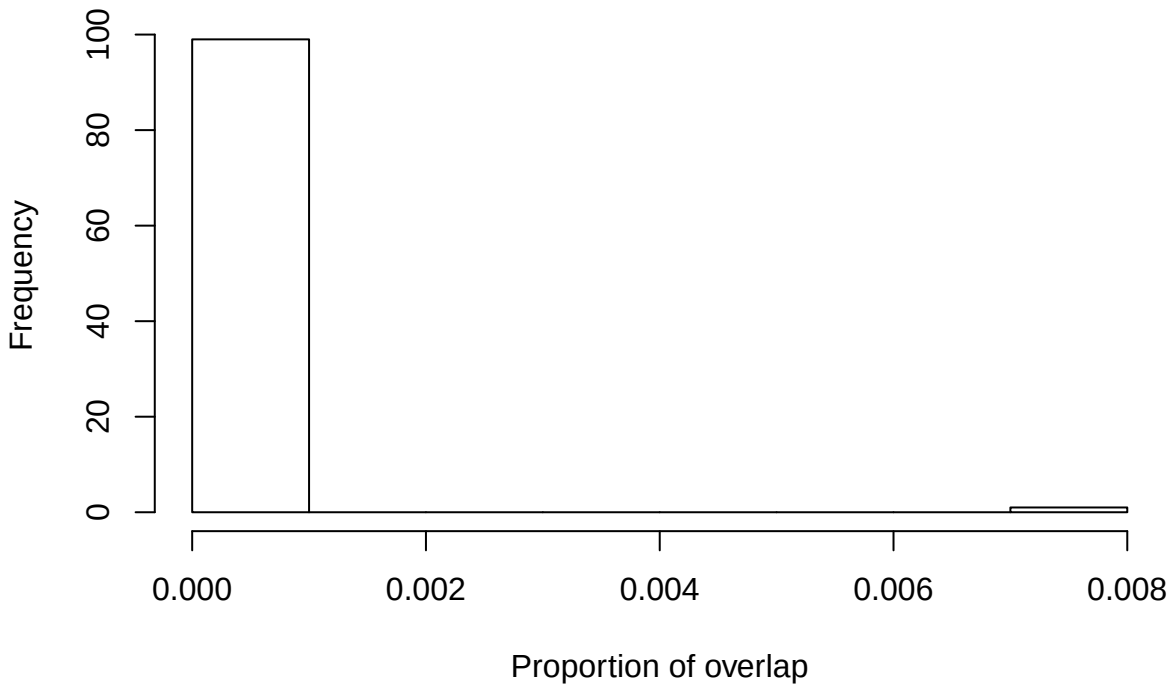

```
#####volant  - semifossorial
Vsfoss_95.overlap <- bayesianOverlap(ellipse_volant,
                                     ellipse_semifossorial,
                                     ellipses.posterior_mob,
                                     draws = 100,
                                     p.interval = 0.95,
                                     n = 100)
Vsfoss_95.overlap_prop <- vector()
for(i in 1:length(Vsfoss_95.overlap$overlap)){

Vsfoss_95.overlap_prop[i]  <- Vsfoss_95.overlap$overlap[i]/min(Vsfoss_95.overlap[i,1:2])

}

hist(Vsfoss_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



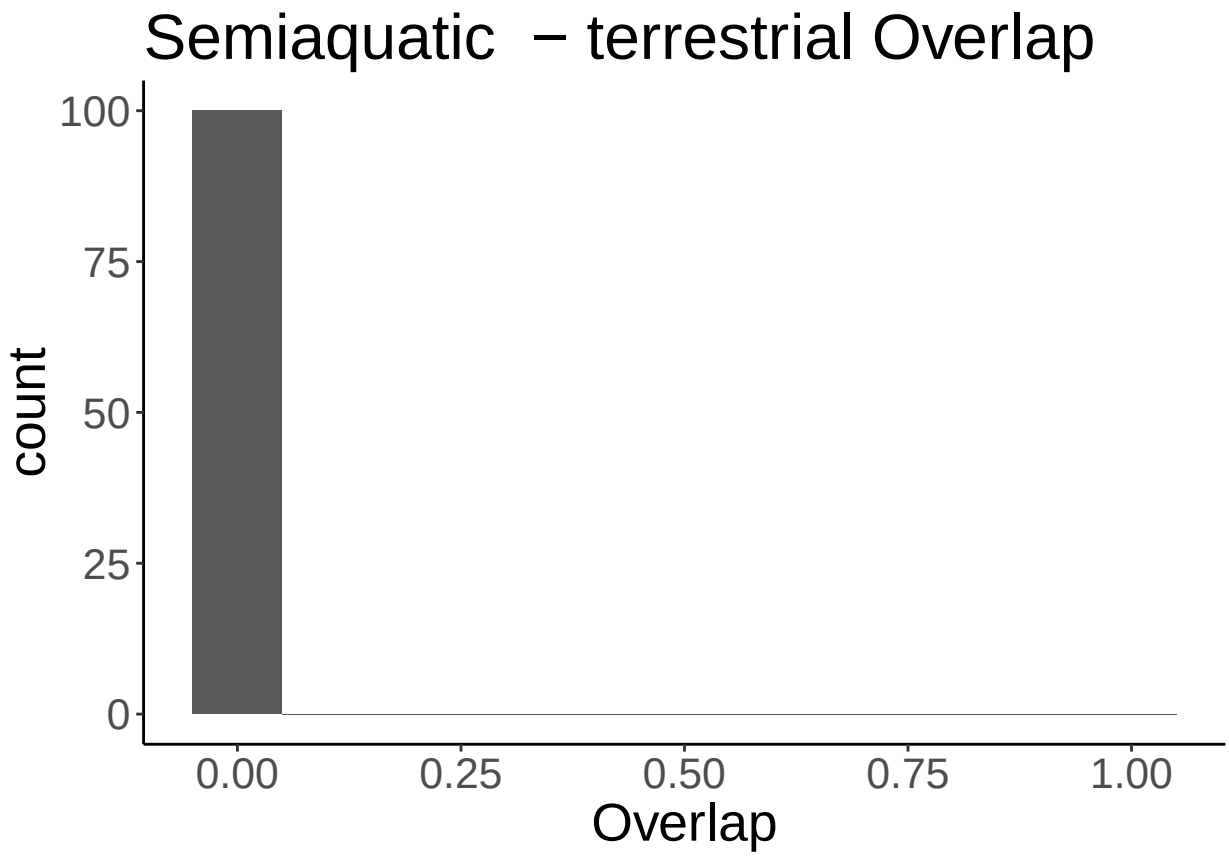

```
myplot_Vsfoss = ggplot(data.frame(Overlap = Vsfoss_95.overlap_prop),
                       aes(Overlap)) +
                       ggtitle("Demersal  - semiaquatic Overlap") +
                       geom_histogram(binwidth=0.1) +
                       scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Vsfoss + theme_bw() + theme(panel.border = element_blank(),
                                   panel.grid.major = element_blank(),
                                   panel.grid.minor = element_blank(),
                                   axis.line = element_line(colour = "black"),
                                   text = element_text(size=20))
```



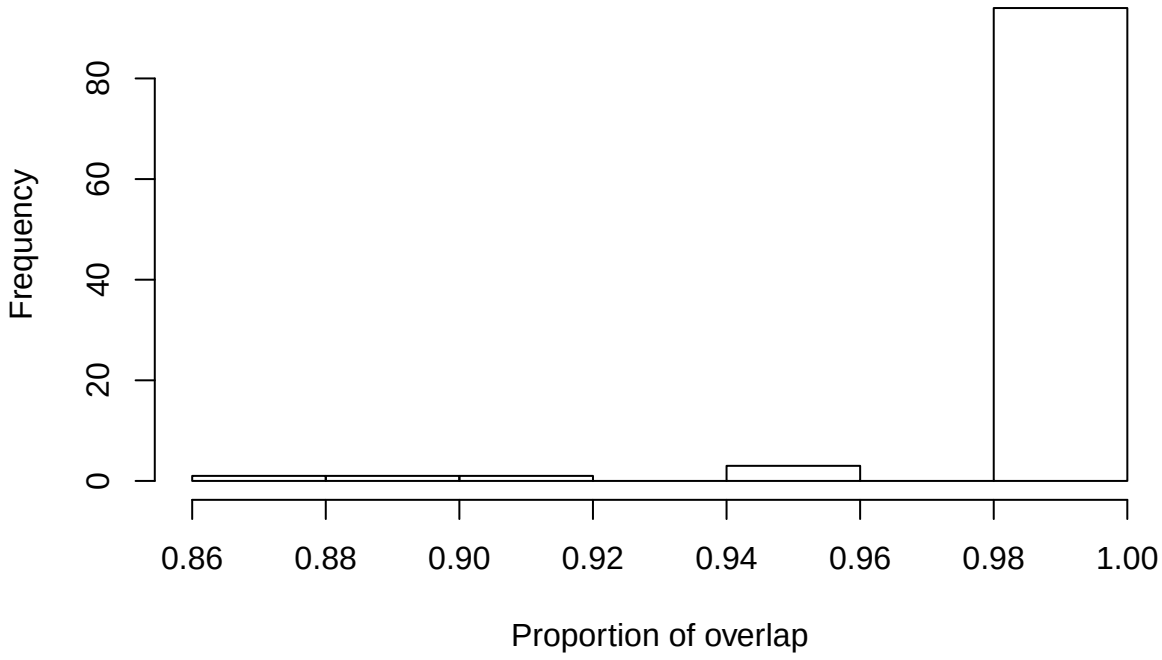

```
#####semiaquatic  - terrestrial
SAt_95.overlap <- bayesianOverlap(ellipse_semiaquatic,
                                  ellipse_terrestrial,
                                  ellipses.posterior_mob,
                                  draws = 100,
                                  p.interval = 0.95,
                                  n = 100)
SAt_95.overlap_prop <- vector()
for(i in 1:length(SAt_95.overlap$overlap)){

SAt_95.overlap_prop[i]  <- SAt_95.overlap$overlap[i]/min(SAt_95.overlap[i,1:2])

}

hist(SAt_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



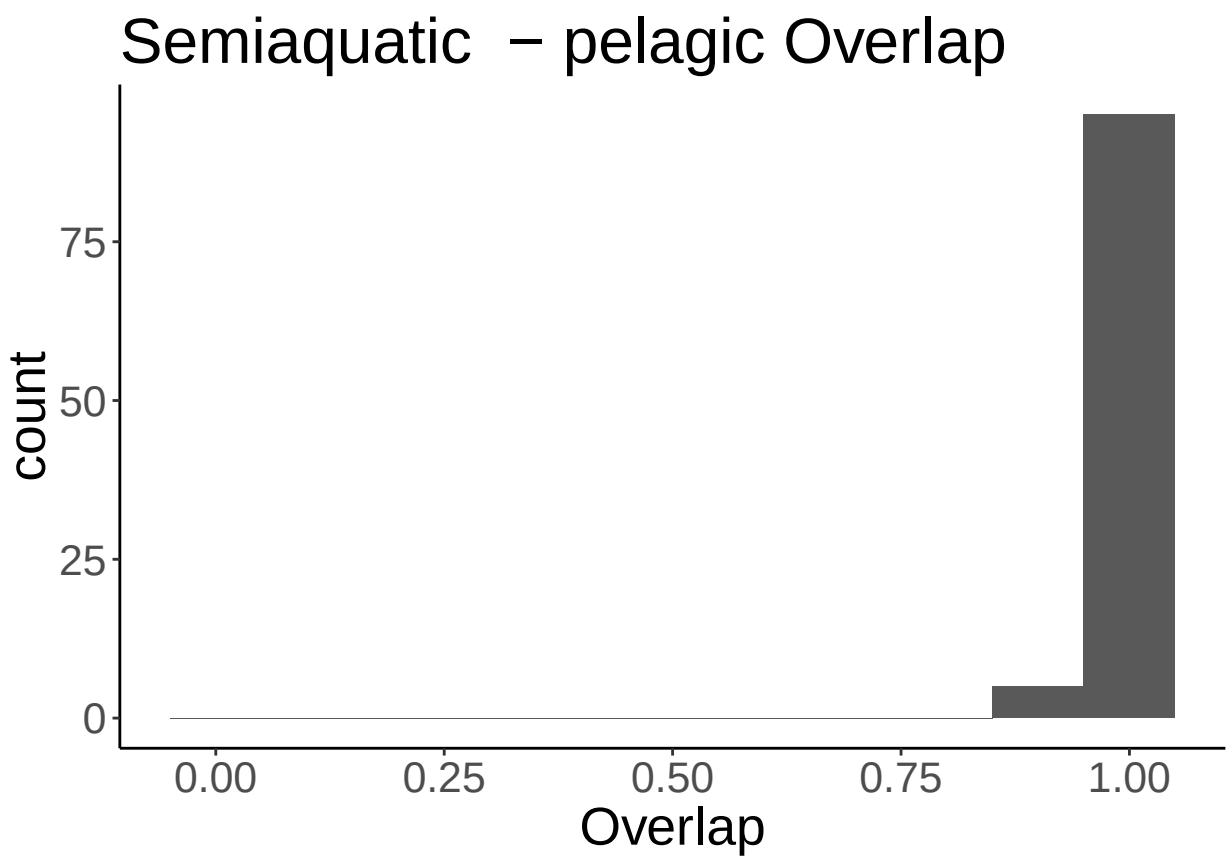

```
myplot_SAt = ggplot(data.frame(Overlap = SAt_95.overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Semiaquatic  - terrestrial Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_SAt + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



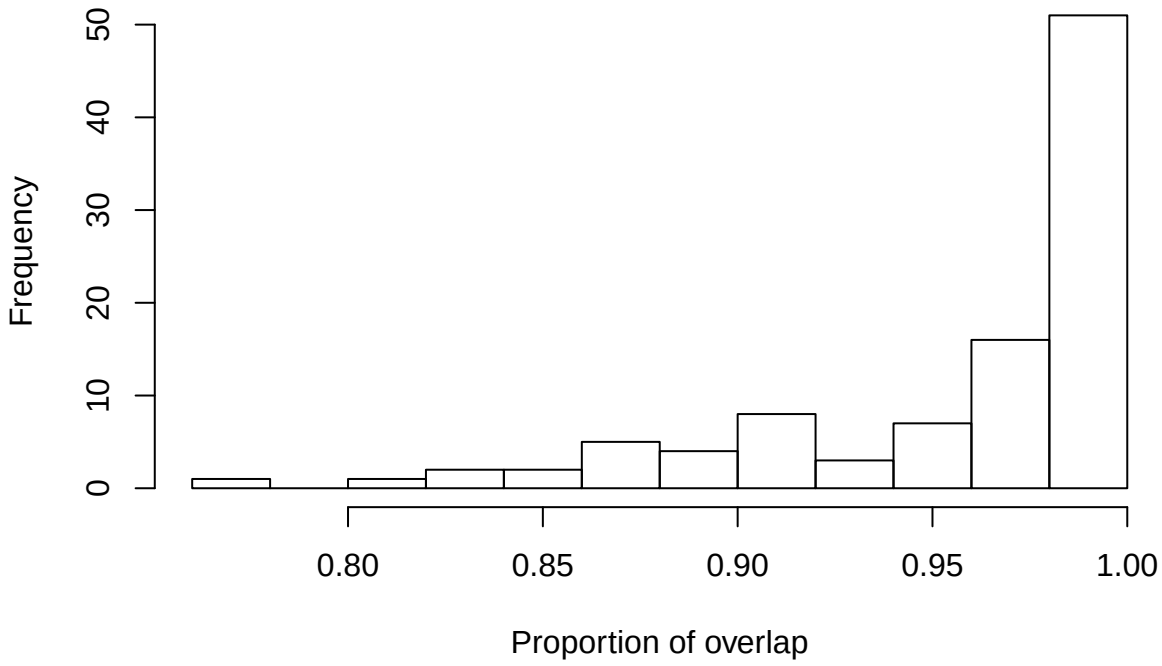

```
#####semiaquatic  - pelagic
SAp_95.overlap <- bayesianOverlap(ellipse_semiaquatic,
                                  ellipse_pelagic,
                                  ellipses.posterior_mob,
                                  draws = 100,
                                  p.interval = 0.95,
                                  n = 100)
SAp_95.overlap_prop <- vector()
for(i in 1:length(SAp_95.overlap$overlap)){

SAp_95.overlap_prop[i]  <- SAp_95.overlap$overlap[i]/min(SAp_95.overlap[i,1:2])

}

hist(SAp_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



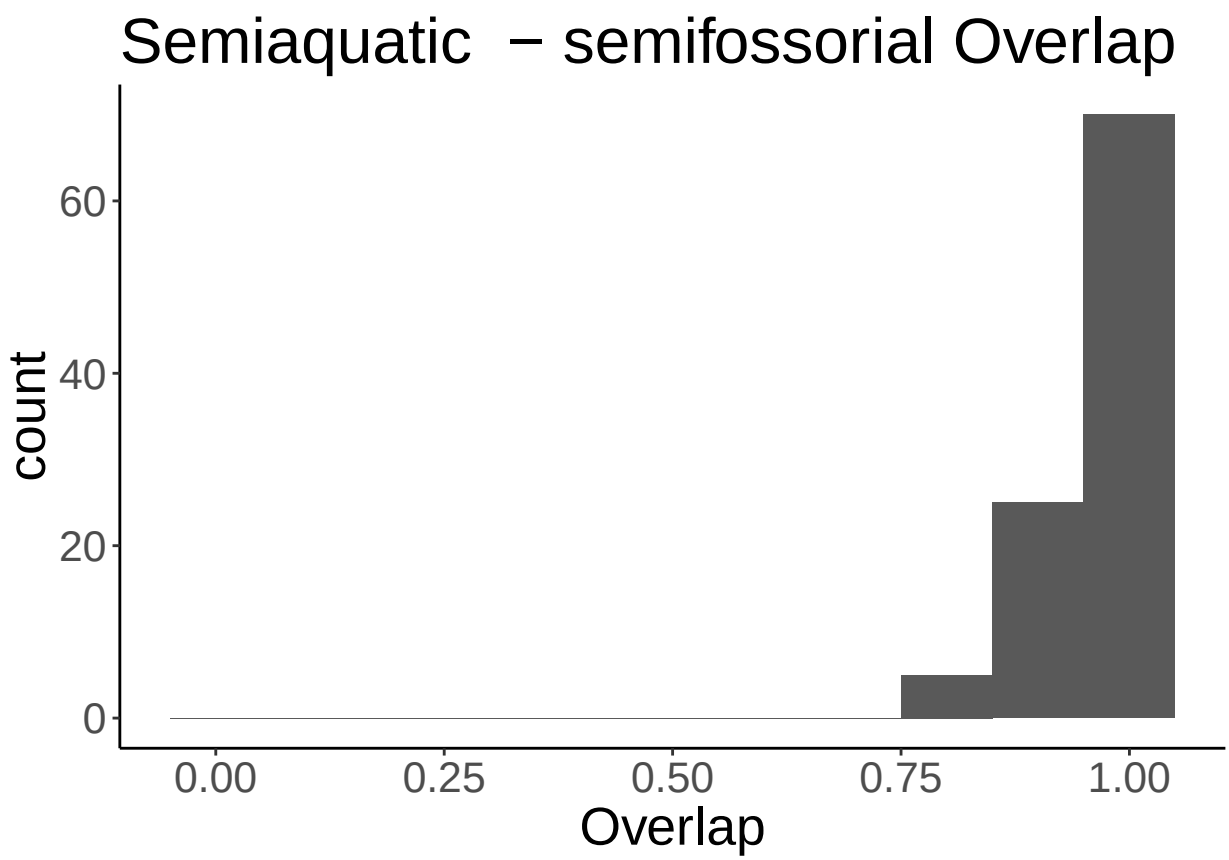

```
myplot_SAp = ggplot(data.frame(Overlap = SAp_95.overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Semiaquatic  - pelagic Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_SAp + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



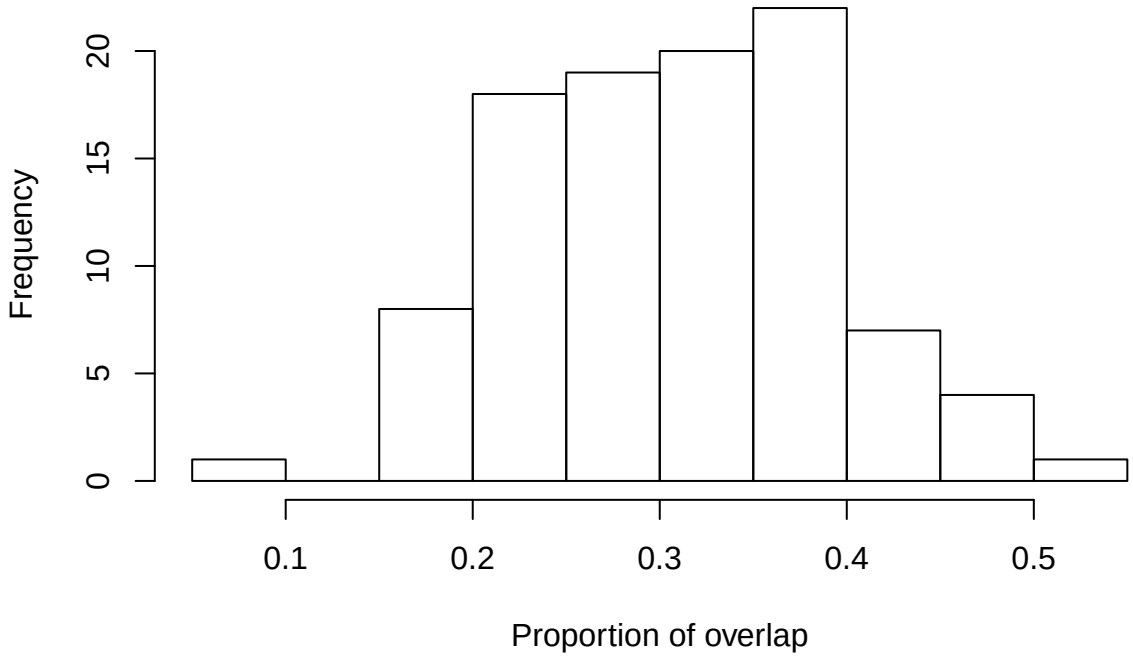

```
#####semiaquatic  - semifossorial
SAsfoss_95.overlap <- bayesianOverlap(ellipse_semiaquatic,
                                      ellipse_semifossorial,
                                      ellipses.posterior_mob,
                                      draws = 100,
                                      p.interval = 0.95,
                                      n = 100)
SAsfoss_95.overlap_prop <- vector()
for(i in 1:length(SAsfoss_95.overlap$overlap)){

SAsfoss_95.overlap_prop[i]  <- SAsfoss_95.overlap$overlap[i]/min(SAsfoss_95.overlap[i,1:2])

}

hist(SAsfoss_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



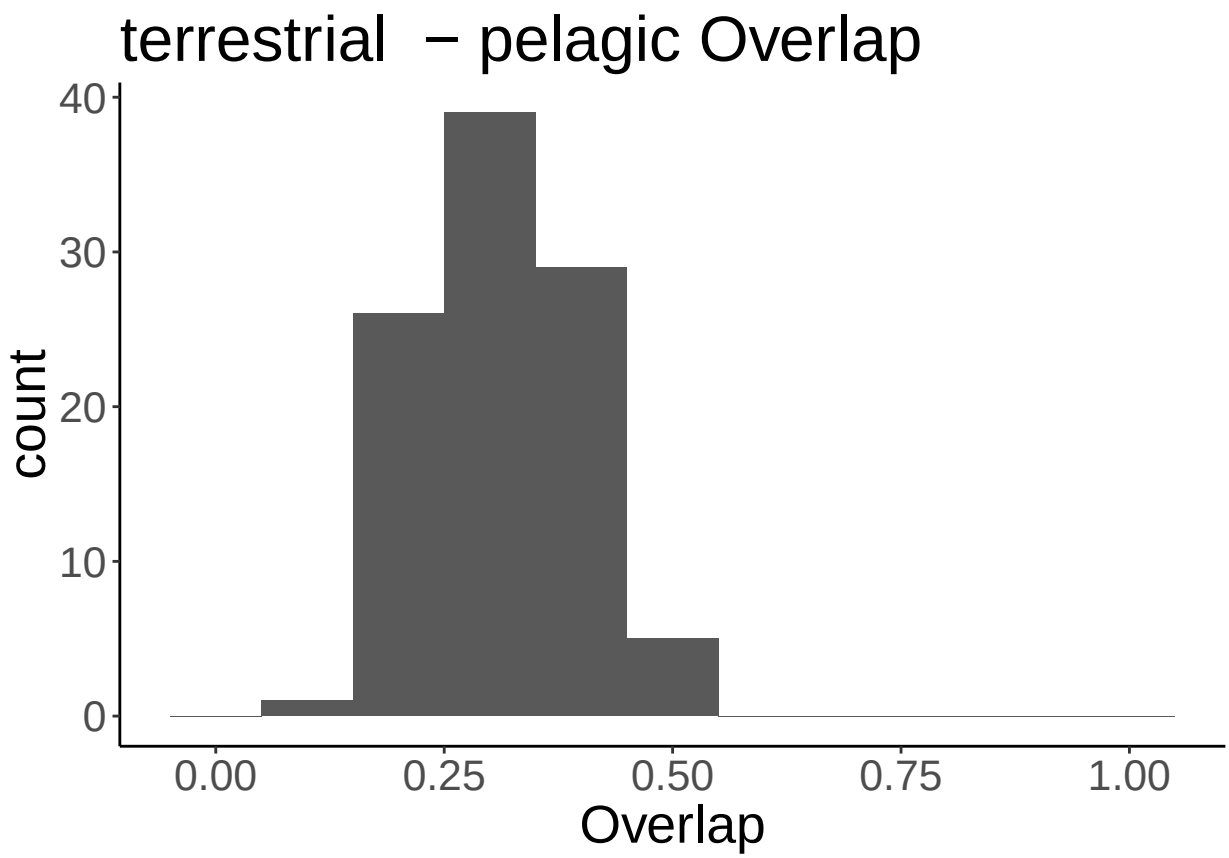

```
myplot_SAsfoss = ggplot(data.frame(Overlap = SAsfoss_95.overlap_prop),
                    aes(Overlap)) +
                        ggtitle("Semiaquatic  - semifossorial Overlap") +
                        geom_histogram(binwidth=0.1) +
                        scale_x_continuous(limits = c(-0.05, 1.05))

myplot_SAsfoss + theme_bw() + theme(panel.border = element_blank(),
                                    panel.grid.major = element_blank(),
                                    panel.grid.minor = element_blank(),
                                    axis.line = element_line(colour = "black"),
                                    text = element_text(size=20))
```



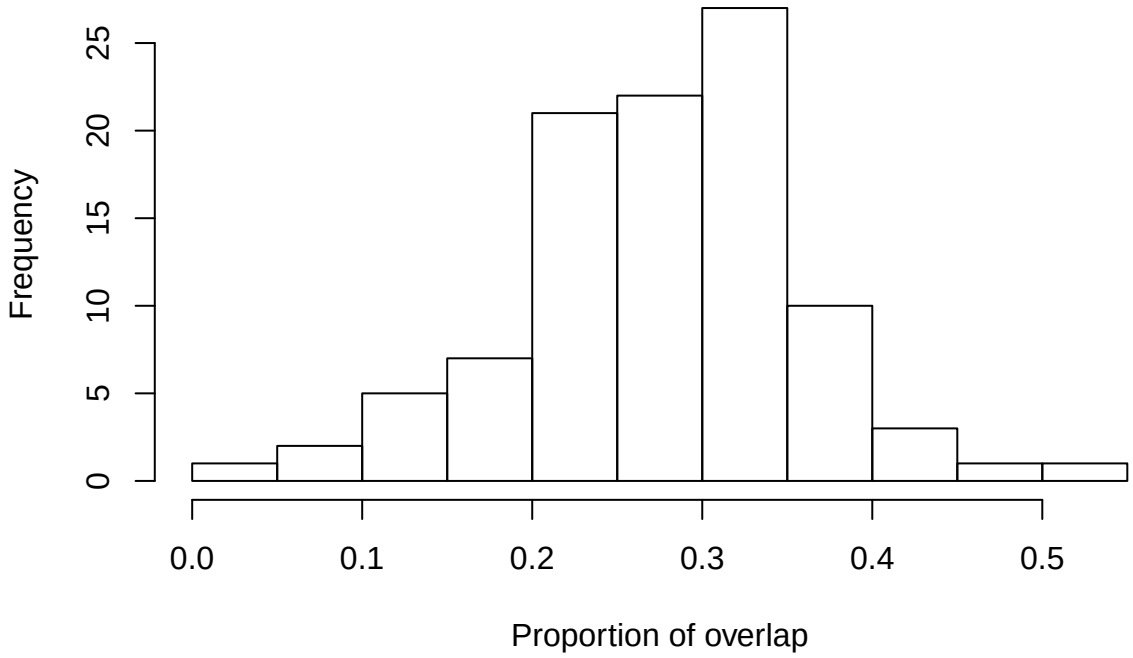

```
#####terrestrial  - pelagic
Tp_95.overlap <- bayesianOverlap(ellipse_terrestrial,
                                 ellipse_pelagic,
                                 ellipses.posterior_mob,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)
Tp_95.overlap_prop <- vector()
for(i in 1:length(Tp_95.overlap$overlap)){

Tp_95.overlap_prop[i]  <- Tp_95.overlap$overlap[i]/min(Tp_95.overlap[i,1:2])

}

hist(Tp_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



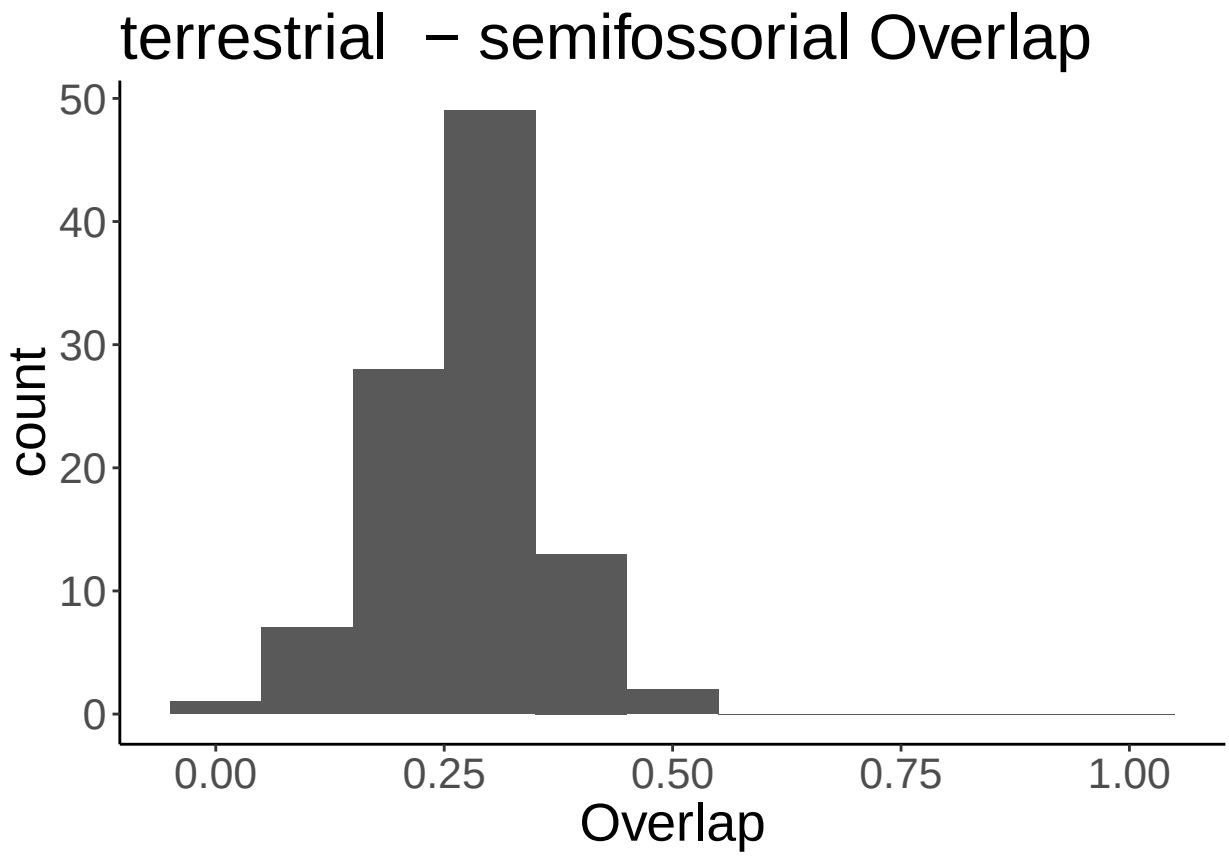

```
myplot_Tp = ggplot(data.frame(Overlap = Tp_95.overlap_prop),
                   aes(Overlap)) +
                   geom_histogram(binwidth=0.1) +
                   ggtitle("terrestrial  - pelagic Overlap") +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Tp + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



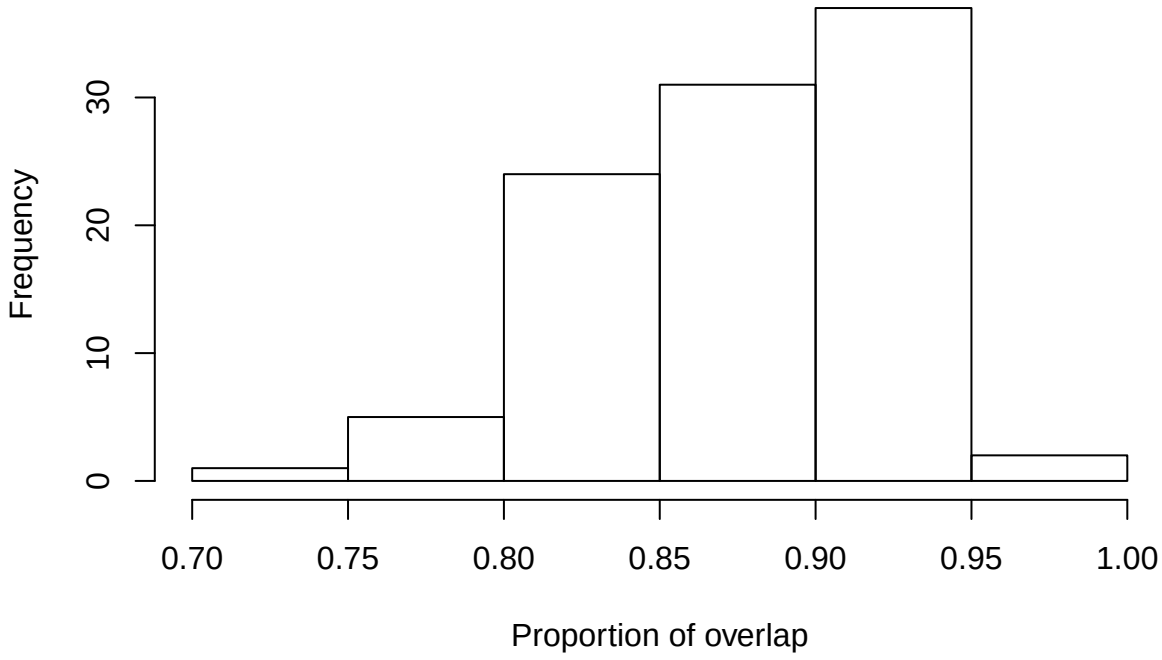

```
#####terrestrial  - semifossorial
Tfoss_95.overlap <- bayesianOverlap(ellipse_terrestrial,
                                    ellipse_semifossorial,
                                    ellipses.posterior_mob,
                                    draws = 100,
                                    p.interval = 0.95,
                                    n = 100)
Tfoss_95.overlap_prop <- vector()
for(i in 1:length(Tfoss_95.overlap$overlap)){

Tfoss_95.overlap_prop[i]  <- Tfoss_95.overlap$overlap[i]/min(Tfoss_95.overlap[i,1:2])

}

hist(Tfoss_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



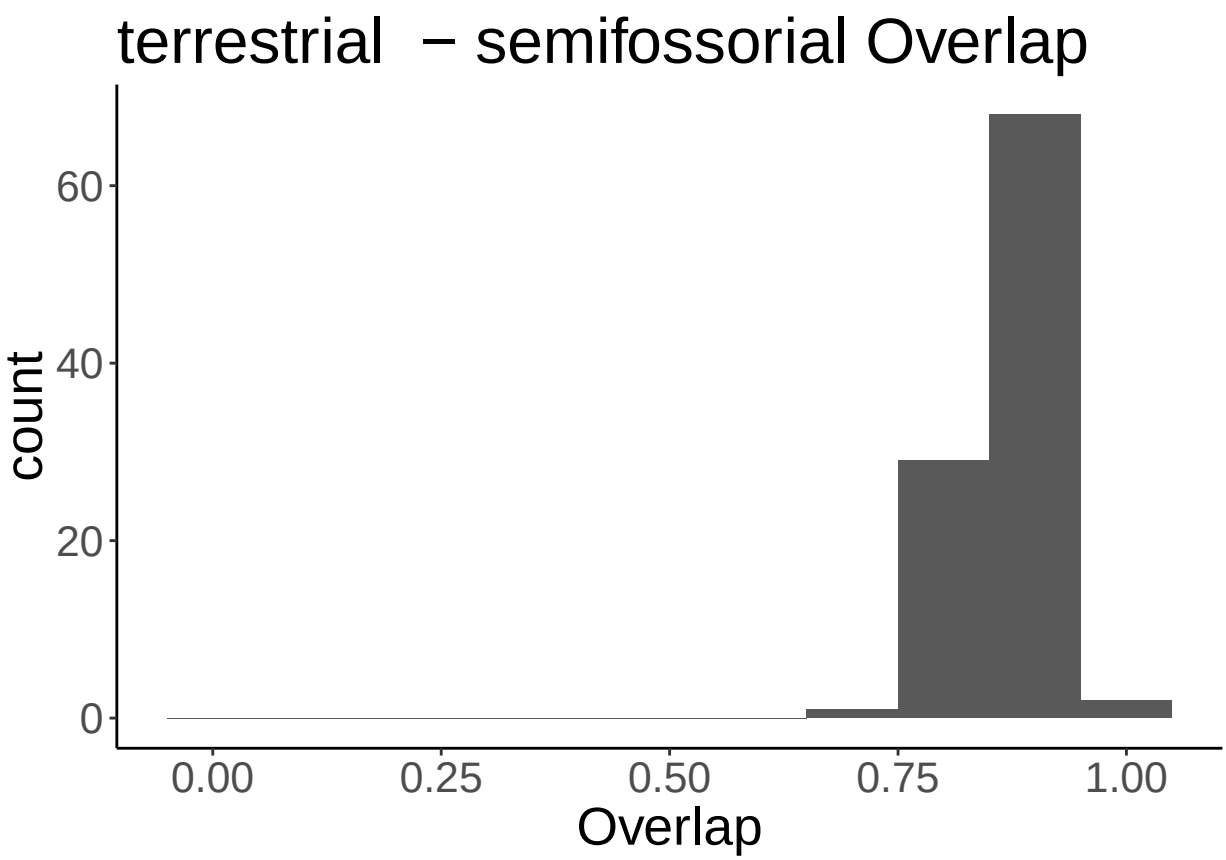

```
myplot_Tfoss = ggplot(data.frame(Overlap = Tfoss_95.overlap_prop),
                      aes(Overlap)) +
                      ggtitle("terrestrial  - semifossorial Overlap") +
                      geom_histogram(binwidth=0.1) +
                      scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Tfoss + theme_bw() + theme(panel.border = element_blank(),
                                  panel.grid.major = element_blank(),
                                  panel.grid.minor = element_blank(),
                                  axis.line = element_line(colour = "black"),
                                  text = element_text(size=20))
```




```
#####pelagic  - semifossorial
Psfoss_95.overlap <- bayesianOverlap(ellipse_pelagic,
                                     ellipse_semifossorial,
                                     ellipses.posterior_mob,
                                     draws = 100,
                                     p.interval = 0.95,
                                     n = 100)
Psfoss_95.overlap_prop <- vector()
for(i in 1:length(Psfoss_95.overlap$overlap)){

Psfoss_95.overlap_prop[i]  <- Psfoss_95.overlap$overlap[i]/min(Psfoss_95.overlap[i,1:2])

}

hist(Psfoss_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```




```
myplot_Psfoss = ggplot(data.frame(Overlap = Psfoss_95.overlap_prop),
                       aes(Overlap)) +
                       ggtitle("terrestrial  - semifossorial Overlap") +
                       geom_histogram(binwidth=0.1) +
                       scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Psfoss + theme_bw() + theme(panel.border = element_blank(),
                                   panel.grid.major = element_blank(),
                                   panel.grid.minor = element_blank(),
                                   axis.line = element_line(colour = "black"),
                                   text = element_text(size=20))
```



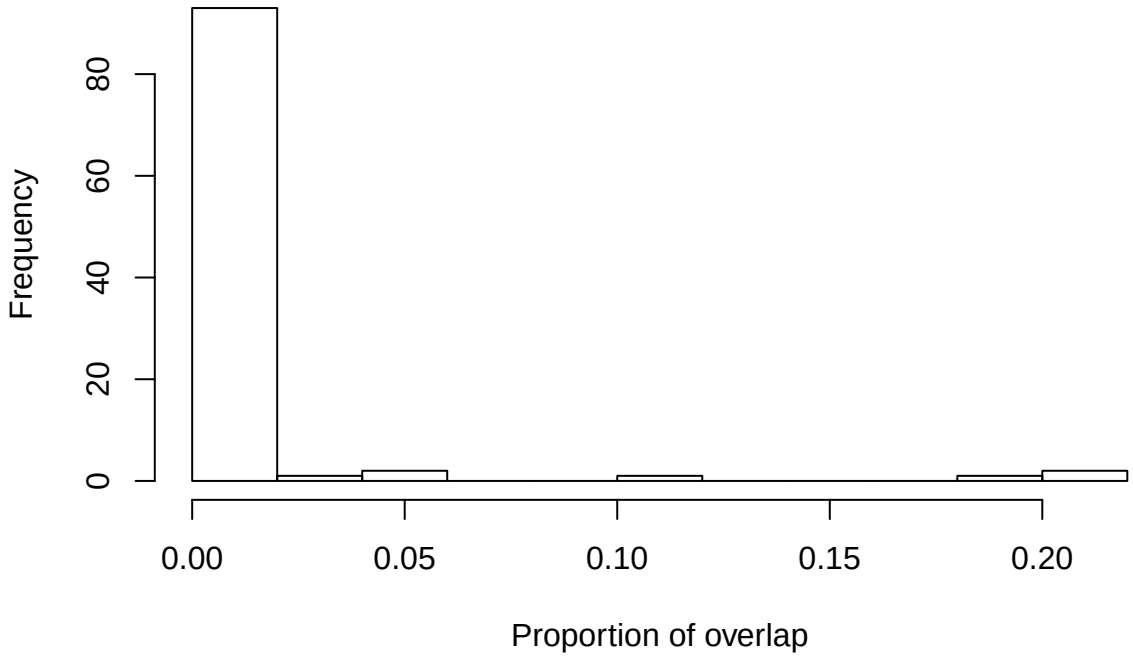
Ellipse overlap calculations for taxa

```
group.MLtaxa <- groupMetricsML(siber.plots)
group.MLmob <- groupMetricsML(siber.mob)


# options for running jags
parms <- list()
parms$n.iter <- 2 * 10^4   # number of iterations to run the model for
parms$n.burnin <- 1 * 10^3 # discard the first set of values
parms$n.thin <- 10     # thin the posterior by this many
parms$n.chains <- 2        # run this many chains

# define the priors
priors <- list()
priors$R <- 1 * diag(2)
priors$k <- 2
priors$tau.mu <- 1.0E-3



ellipses.posterior <- siberMVN(siber.plots,
                               parms,
                               priors)
```

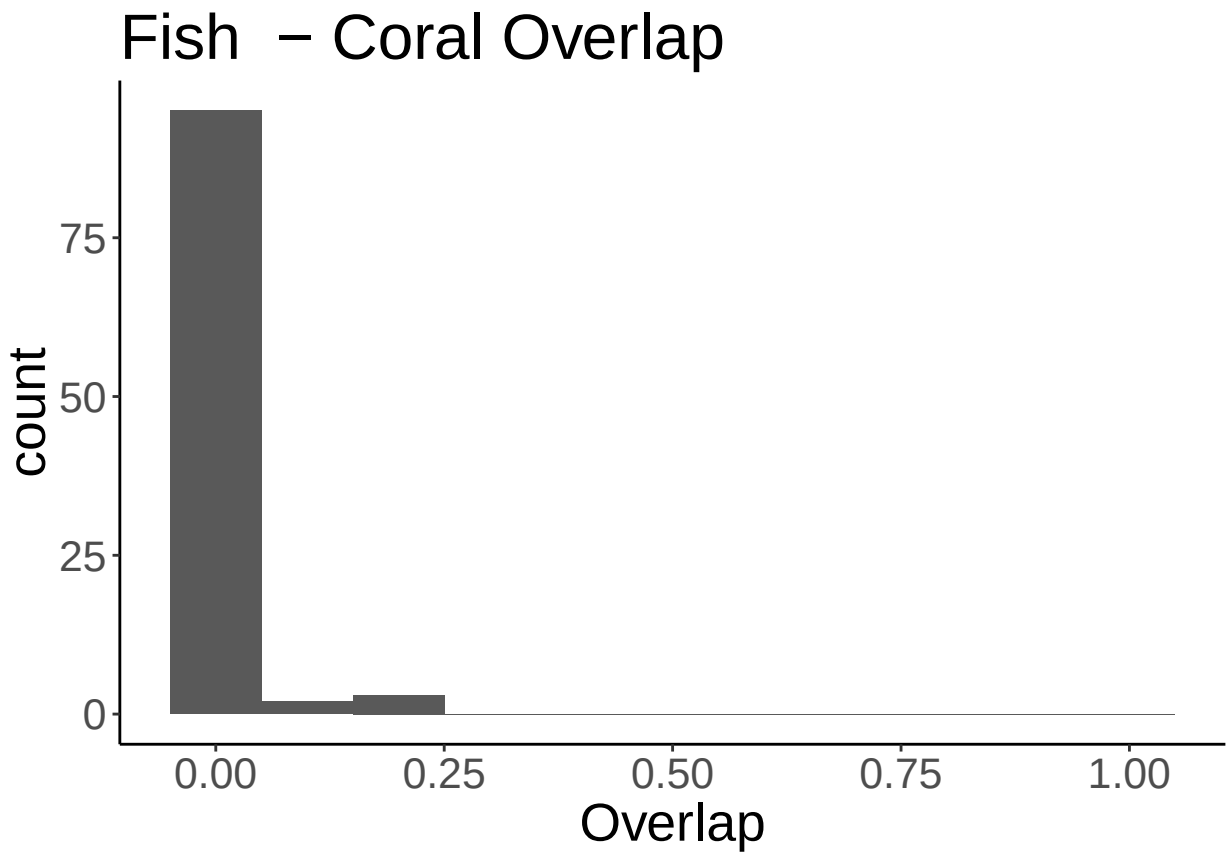
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 3
##    Total graph size: 36
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 81
##    Unobserved stochastic nodes: 3
##    Total graph size: 96
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3
##    Unobserved stochastic nodes: 3
##    Total graph size: 18
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 121
##    Unobserved stochastic nodes: 3
##    Total graph size: 136
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 48
##    Unobserved stochastic nodes: 3
##    Total graph size: 63
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

```
##    Observed stochastic nodes: 6
##    Unobserved stochastic nodes: 3
##    Total graph size: 21
##
## Initializing model
```

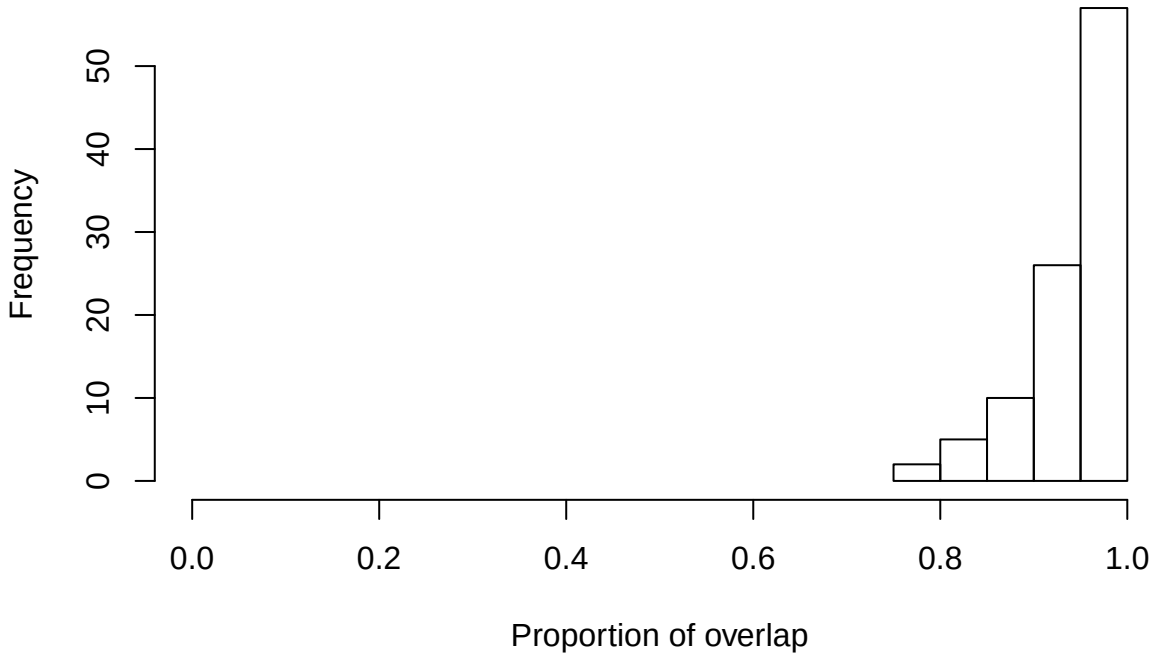
```
# The first ellipse is referenced using a character string representation where
# in "x.y", "x" is the community, and "y" is the group within that community.
# So in this example: community 1, group 1

#Actinopterygii
ellipse_Actinopterygii <- "1.1"

#Anthozoa
ellipse_Anthozoa <- "1.2"

#Aves
ellipse_Aves <- "1.3"

#Gastropoda
ellipse_Gastropoda <- "1.7"

#Mammalia
ellipse_Mammalia <- "1.8"

#Reptilia
ellipse_Reptilia <- "1.9"


#####fish - coral
AAn_95.overlap <- bayesianOverlap(ellipse_Actinopterygii,
                                  ellipse_Anthozoa,
                                  ellipses.posterior,
                                  draws = 100,
                                  p.interval = 0.95,
                                  n = 100)
AAn_95_overlap_prop <- vector()
for(i in 1:length(AAn_95.overlap$overlap)){

AAn_95_overlap_prop[i]  <- AAn_95.overlap$overlap[i]/min(AAn_95.overlap[i,1:2])

}

hist(AAn_95_overlap_prop, xlab = "Proportion of overlap", main = "")
```




```
myplot_AAn = ggplot(data.frame(Overlap =  AAn_95_overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Fish  - Coral Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AAn + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



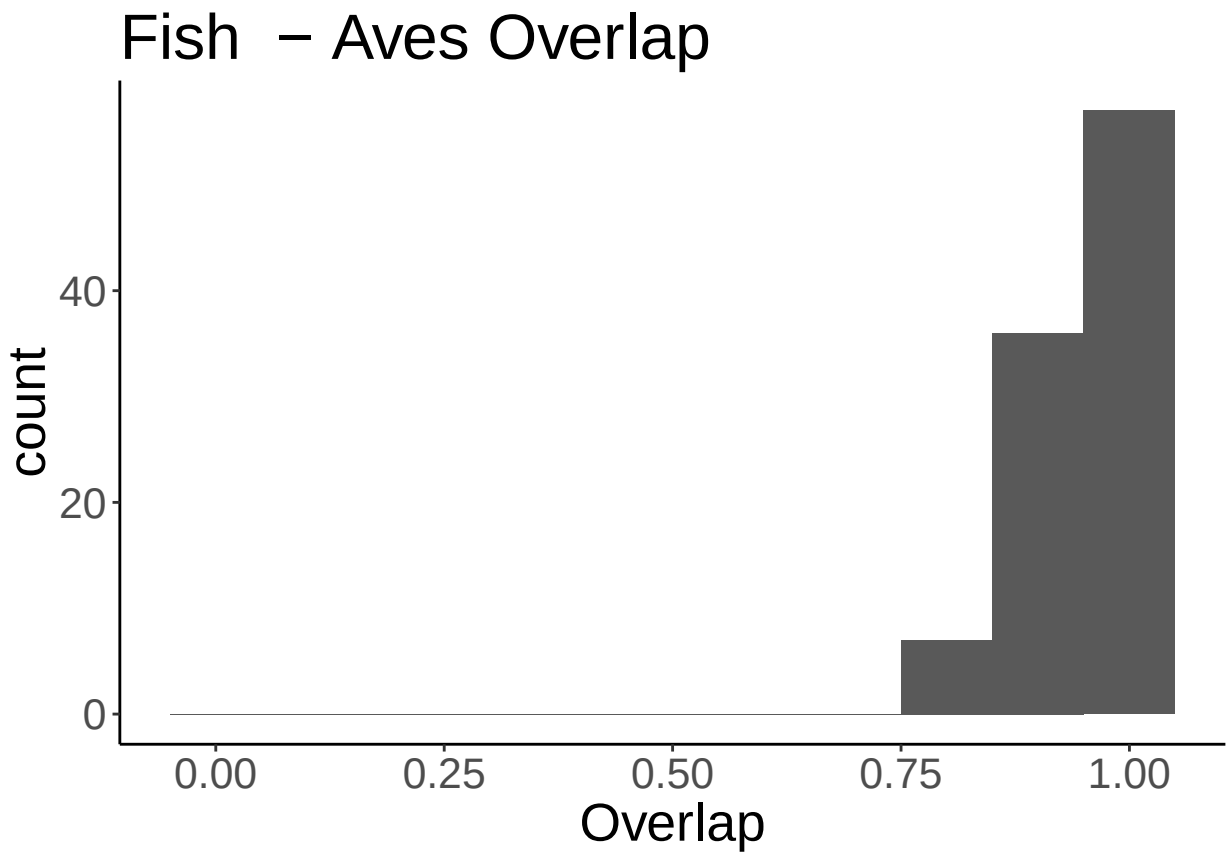

```
#####fish - aves
AAv95.overlap <- bayesianOverlap(ellipse_Actinopterygii,
                                 ellipse_Aves,
                                 ellipses.posterior,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)

AAv_95_overlap_prop <- vector()
for(i in 1:length(AAv95.overlap$overlap)){

AAv_95_overlap_prop[i]  <- AAv95.overlap$overlap[i]/min(AAv95.overlap[i,1:2])

}

hist(AAv_95_overlap_prop, xlab = "Proportion of overlap", main = "", xlim = c(0,1))
```



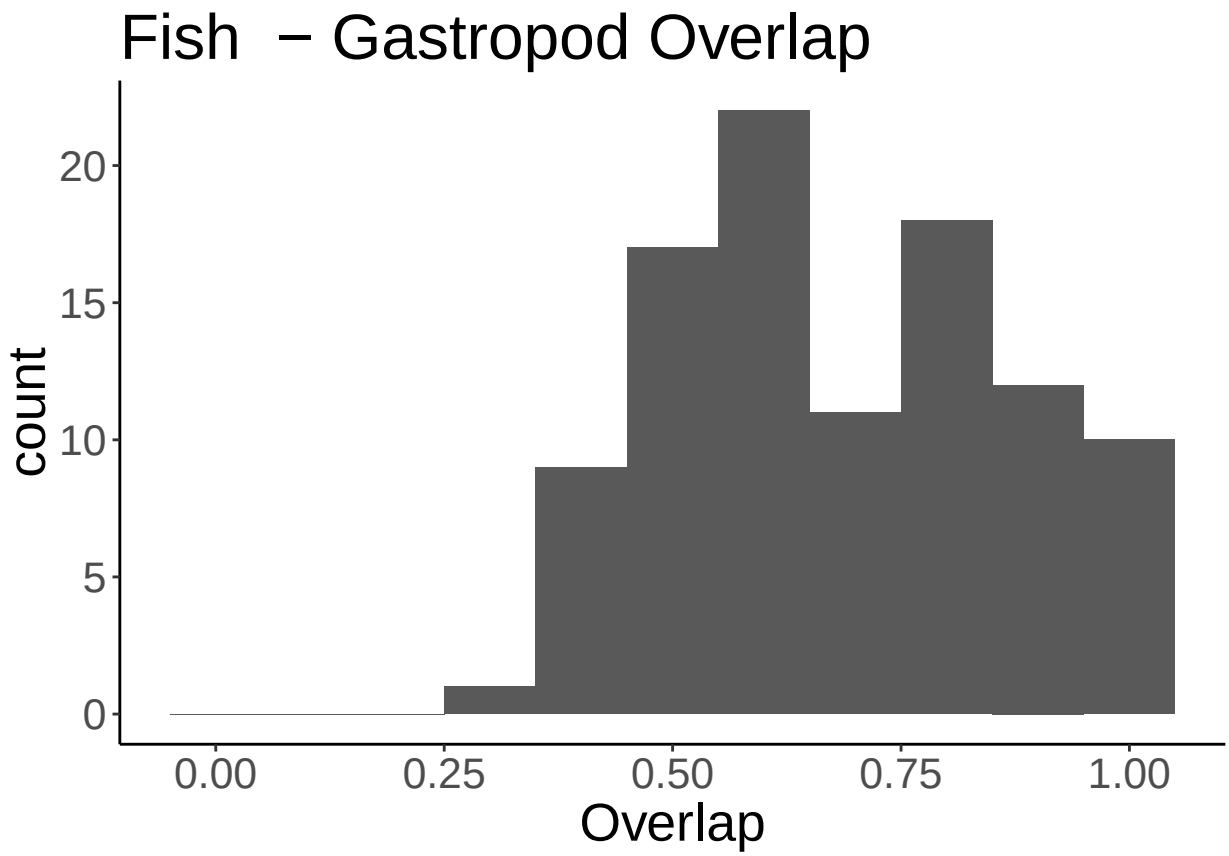

```
hdr(AAv_95_overlap_prop)
```

```
## $hdr
##          [,1]      [,2]      [,3]     [,4]
## 99% 0.7754661 1.0334730        NA       NA
## 95% 0.8120950 0.8156312 0.8401105 1.029851
## 50% 0.9656204 1.0142572        NA       NA
##
## $mode
## [1] 0.9962782
##
## $falpha
##        1%        5%       50%
## 0.6094313 1.0703473 5.9849876
```

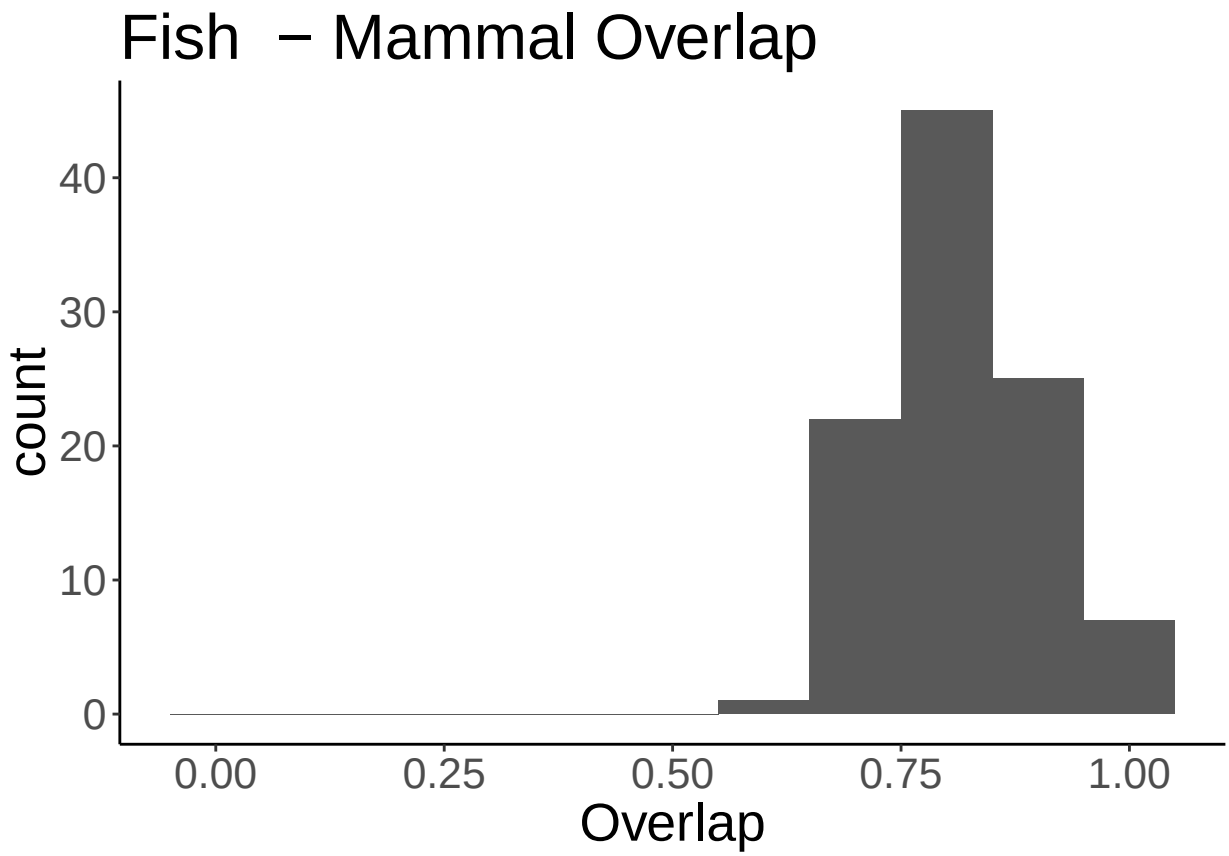
```
myplot_Aav = ggplot(data.frame(Overlap =  AAv_95_overlap_prop),
                    aes(Overlap)) +
                    geom_histogram(binwidth=0.1) +
                    ggtitle("Fish  - Aves Overlap") +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_Aav + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



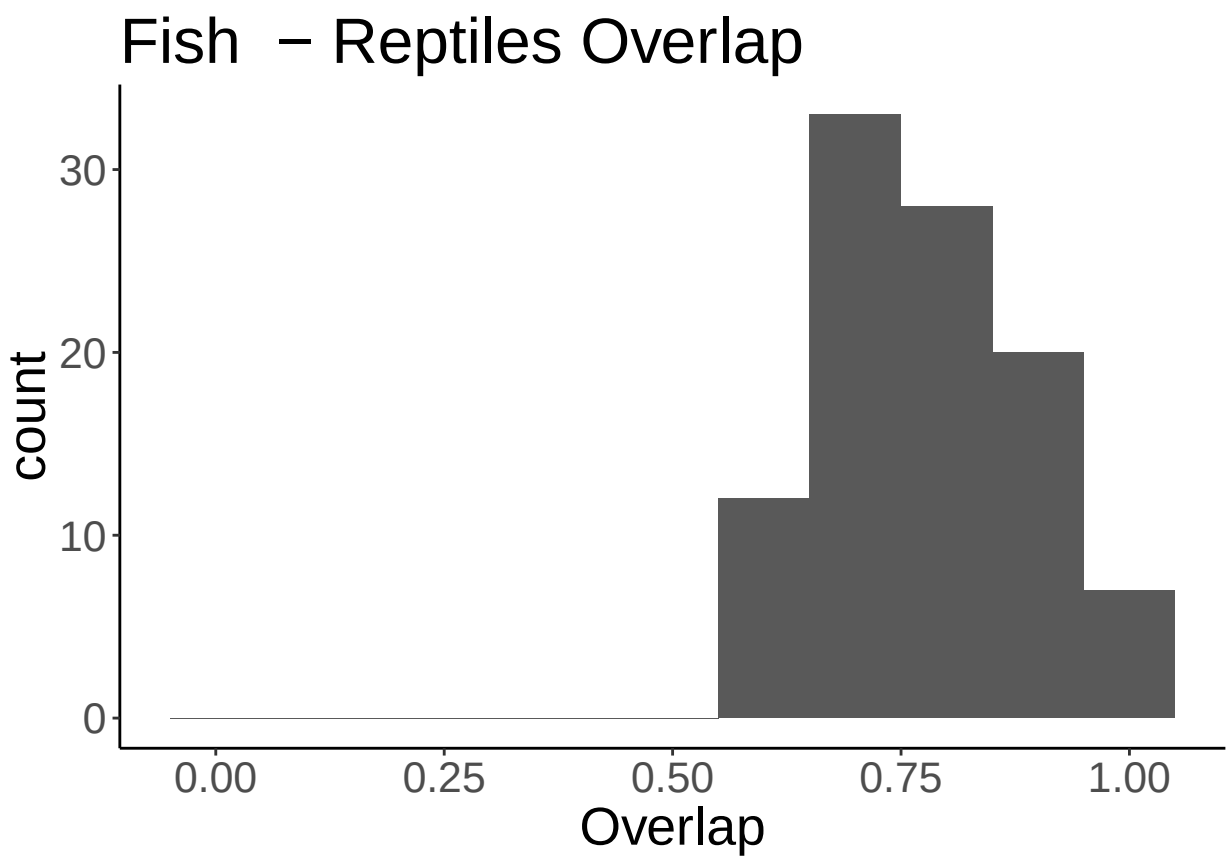

```
#####fish - gastropod
AG95.overlap <- bayesianOverlap(ellipse_Actinopterygii,
                                ellipse_Gastropoda,
                                ellipses.posterior,
                                draws = 100,
                                p.interval = 0.95,
                                n = 100)

AG_95_overlap_prop <- vector()
for(i in 1:length(AG95.overlap$overlap)){

AG_95_overlap_prop[i]  <- AG95.overlap$overlap[i]/min(AG95.overlap[i,1:2])

}


myplot_AG = ggplot(data.frame(Overlap =  AG_95_overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Fish  - Gastropod Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AG + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



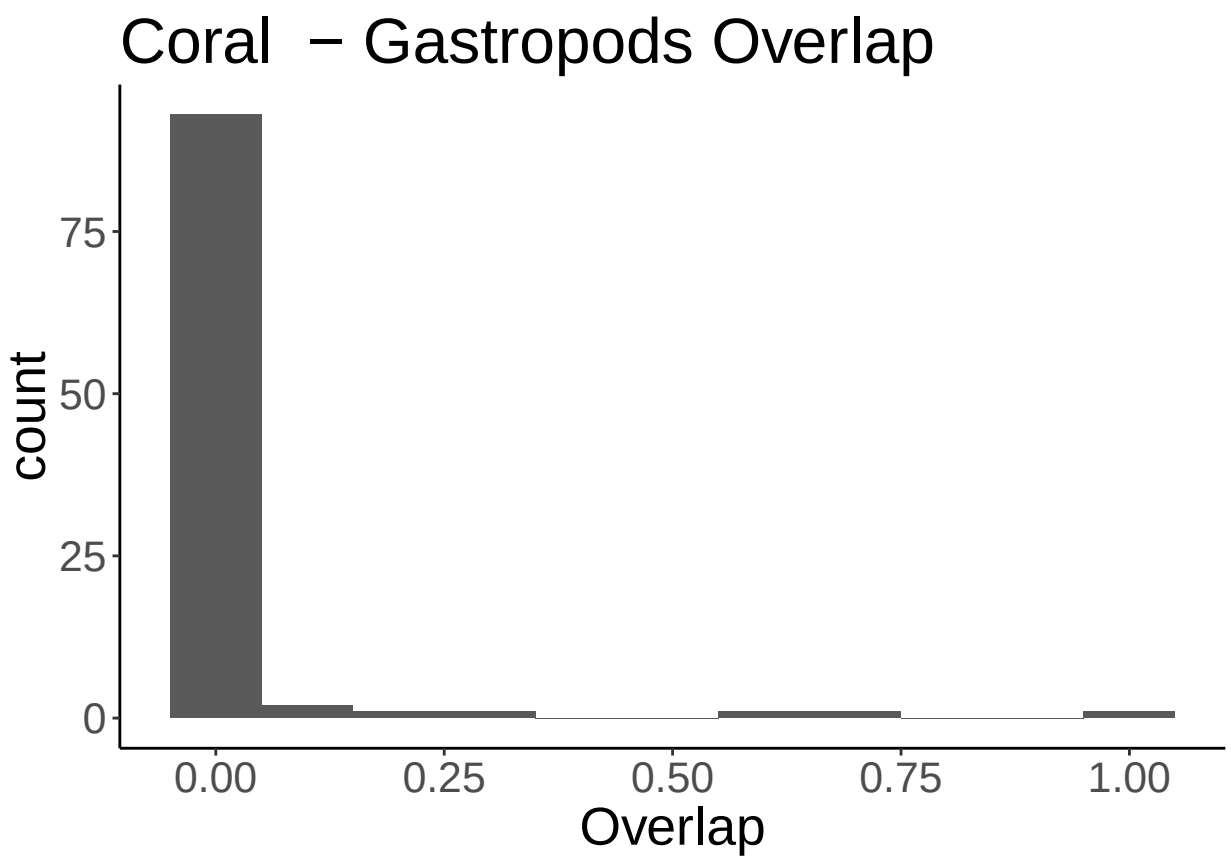

```
##fish - mammal
AM95.overlap <- bayesianOverlap(ellipse_Actinopterygii,
                                ellipse_Mammalia,
                                ellipses.posterior,
                                draws = 100,
                                p.interval = 0.95,
                                n = 100)

Am_95_overlap_prop <- vector()
for(i in 1:length(AM95.overlap$overlap)){

Am_95_overlap_prop[i]  <- AM95.overlap$overlap[i]/min(AM95.overlap[i,1:2])

}

myplot_AM = ggplot(data.frame(Overlap =  Am_95_overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Fish  - Mammal Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AM + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



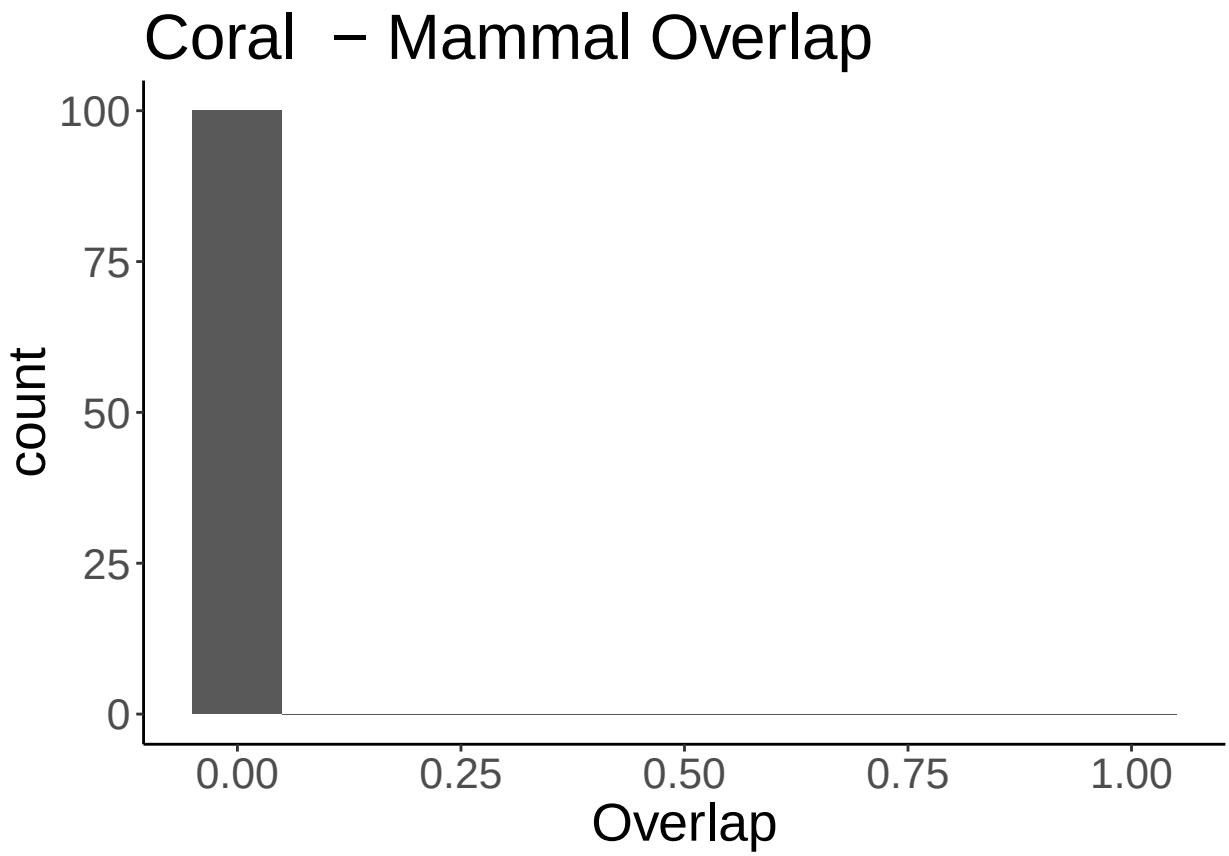

```
###fish and reptiles
AR95.overlap <- bayesianOverlap(ellipse_Actinopterygii,
                                ellipse_Reptilia,
                                ellipses.posterior,
                                draws = 100,
                                p.interval = 0.95,
                                n = 100)

AR_95_overlap_prop <- vector()
for(i in 1:length(AR95.overlap$overlap)){

AR_95_overlap_prop[i]  <- AR95.overlap$overlap[i]/min(AR95.overlap[i,1:2])

}

myplot_AR = ggplot(data.frame(Overlap =  AR_95_overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Fish  - Reptiles Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AR + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



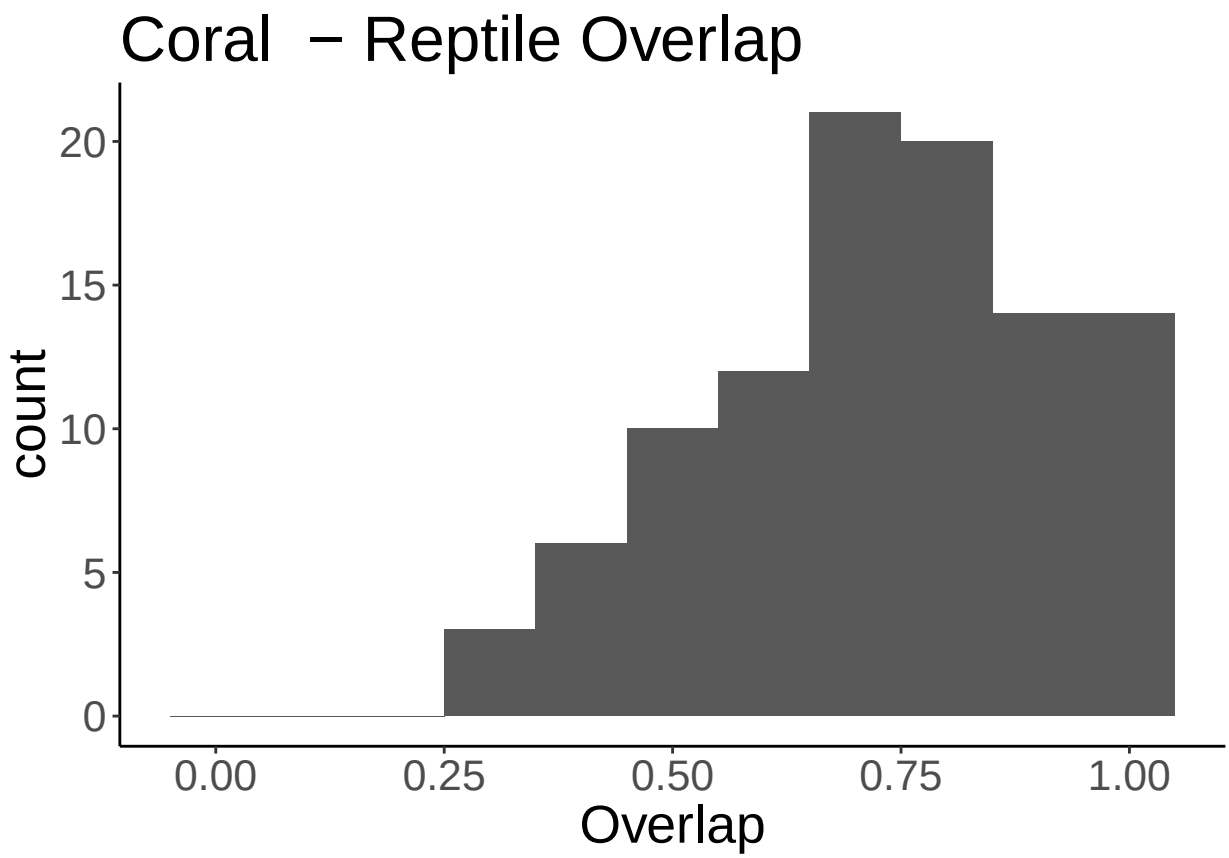

```
###coral-gastropod
AnG95.overlap <- bayesianOverlap(ellipse_Anthozoa,
                                 ellipse_Gastropoda,
                                 ellipses.posterior,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)

AnG95_overlap_prop <- vector()
for(i in 1:length(AnG95.overlap$overlap)){

AnG95_overlap_prop[i]  <- AnG95.overlap$overlap[i]/min(AnG95.overlap[i,1:2])

}

myplot_AnG = ggplot(data.frame(Overlap =  AnG95_overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Coral  - Gastropods Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AnG + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



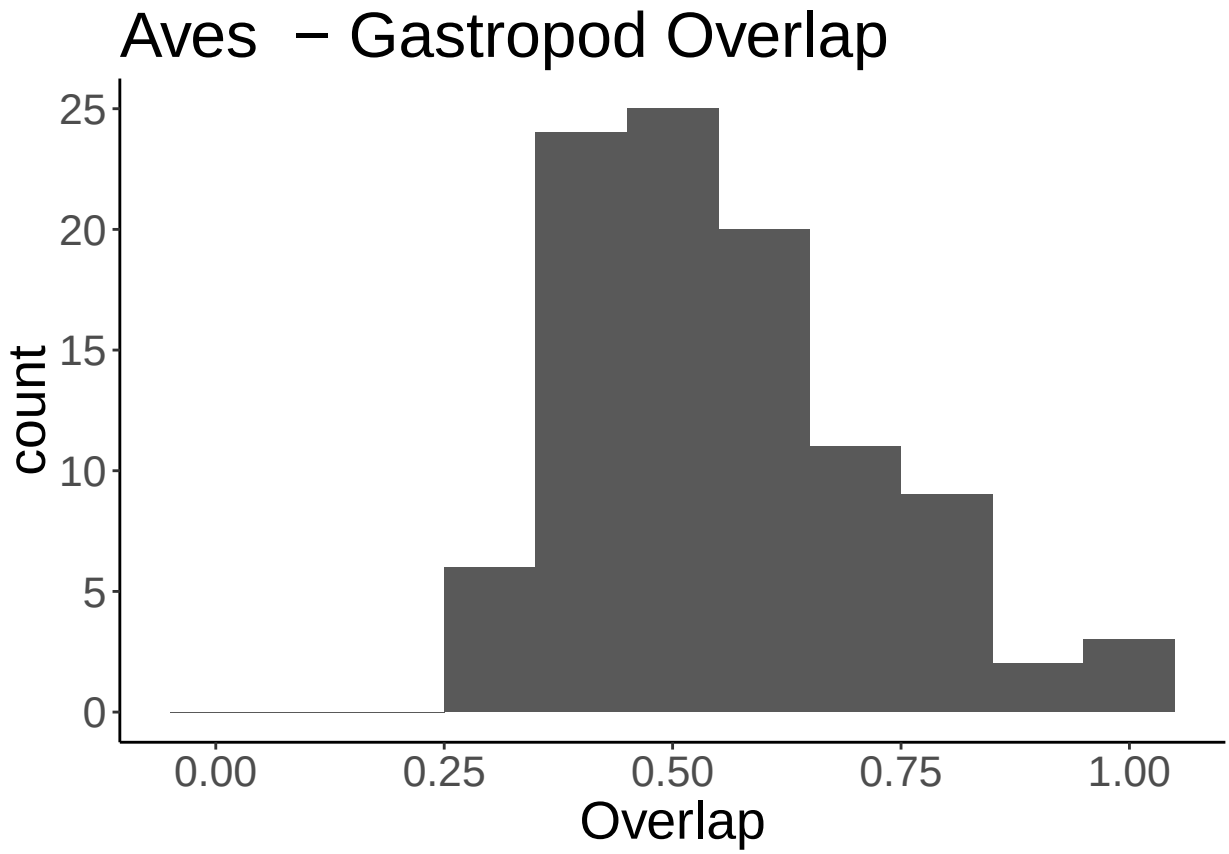

```
##coral mammal
AnM95.overlap <- bayesianOverlap(ellipse_Anthozoa,
                                 ellipse_Mammalia,
                                 ellipses.posterior,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)


AnM95_overlap_prop <- vector()
for(i in 1:length(AnG95.overlap$overlap)){

AnM95_overlap_prop[i]  <- AnM95.overlap$overlap[i]/min(AnM95.overlap[i,1:2])

}

myplot_AnM = ggplot(data.frame(Overlap =  AnM95_overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Coral  - Mammal Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AnM + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



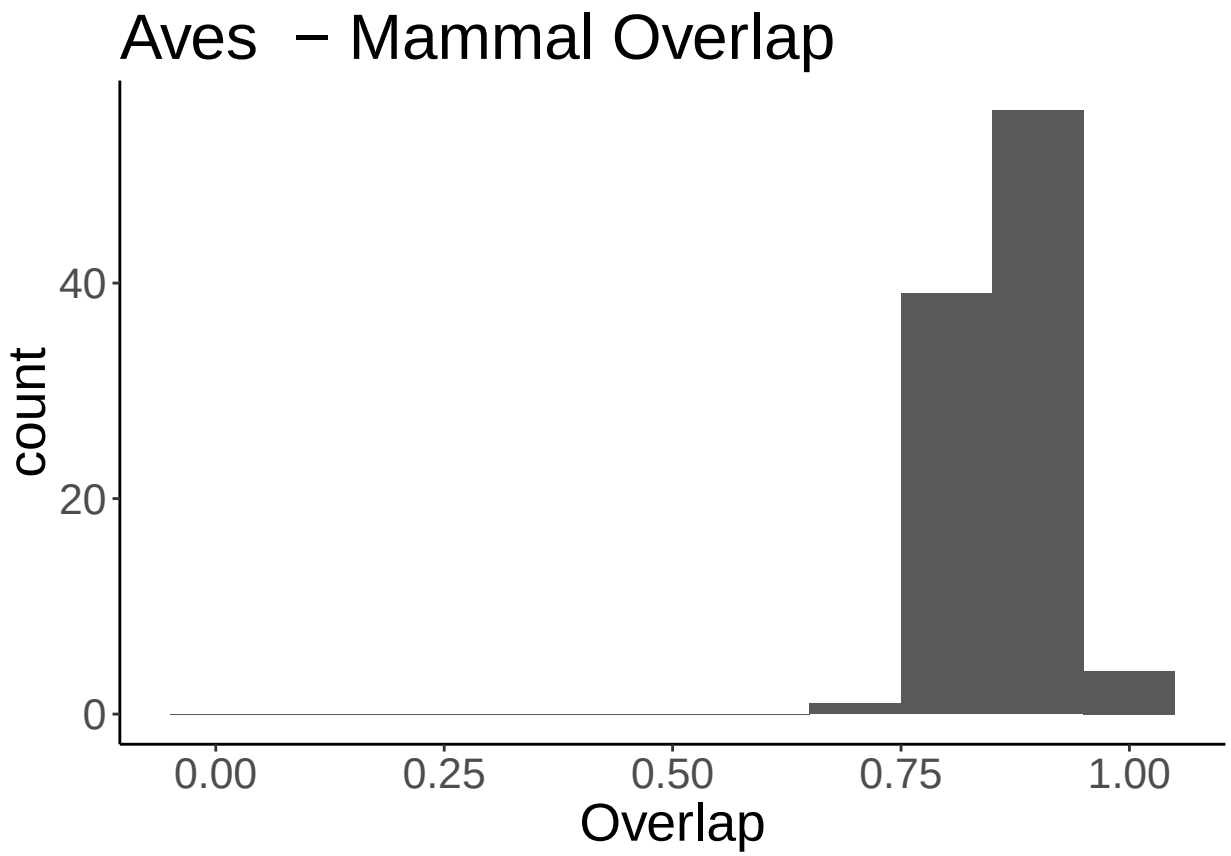

```
##coral reptile
AnR95.overlap <- bayesianOverlap(ellipse_Anthozoa,
                                 ellipse_Reptilia,
                                 ellipses.posterior,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)


AnR95_overlap_prop <- vector()
for(i in 1:length(AnR95.overlap$overlap)){

AnR95_overlap_prop[i]  <- AnR95.overlap$overlap[i]/min(AnR95.overlap[i,1:2])

}

myplot_AnR = ggplot(data.frame(Overlap =  AnR95_overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Coral  - Reptile Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AnR + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



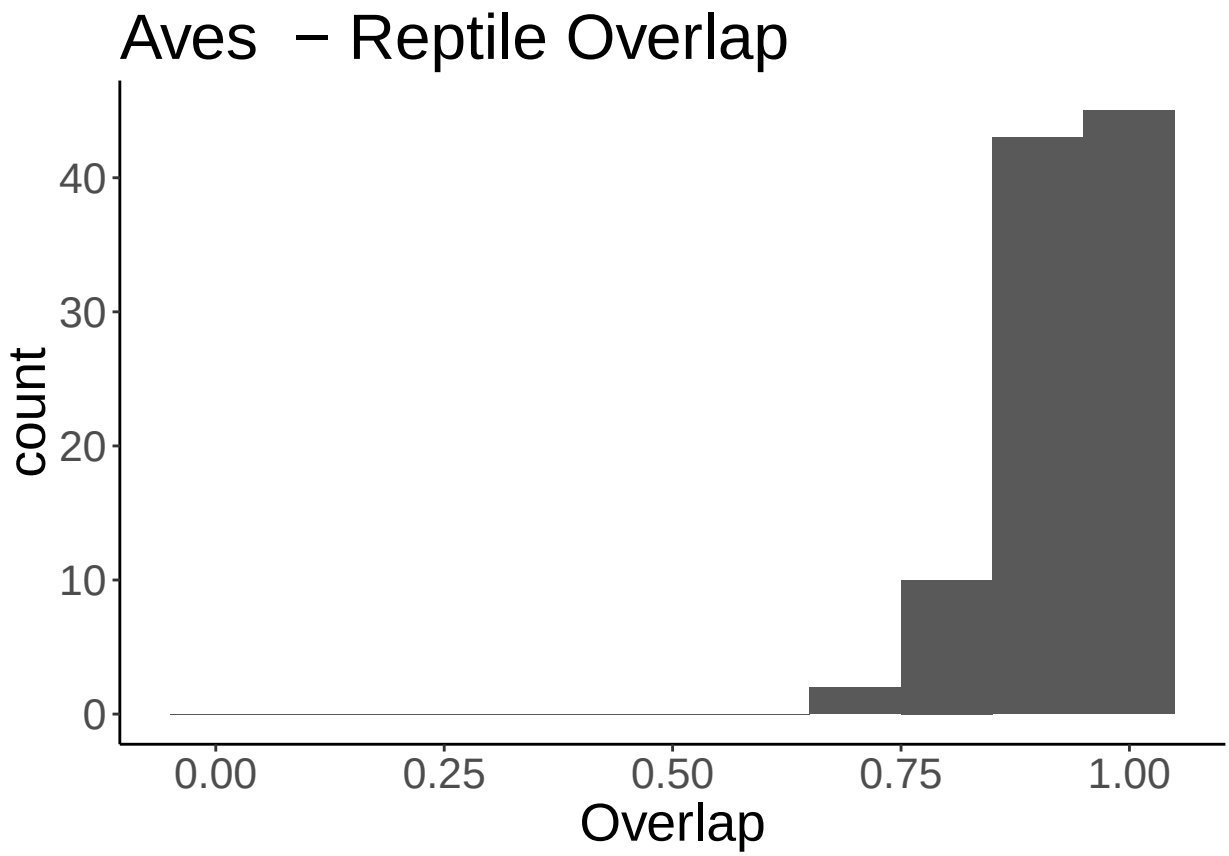

```
###bird - gastropd

AvG95.overlap <- bayesianOverlap(ellipse_Aves,
                                 ellipse_Gastropoda,
                                 ellipses.posterior,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)

AvG95_overlap_prop <- vector()
for(i in 1:length(AvG95.overlap$overlap)){

AvG95_overlap_prop[i]  <- AvG95.overlap$overlap[i]/min(AvG95.overlap[i,1:2])

}

myplot_AvG = ggplot(data.frame(Overlap =  AvG95_overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Aves  - Gastropod Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AvG + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



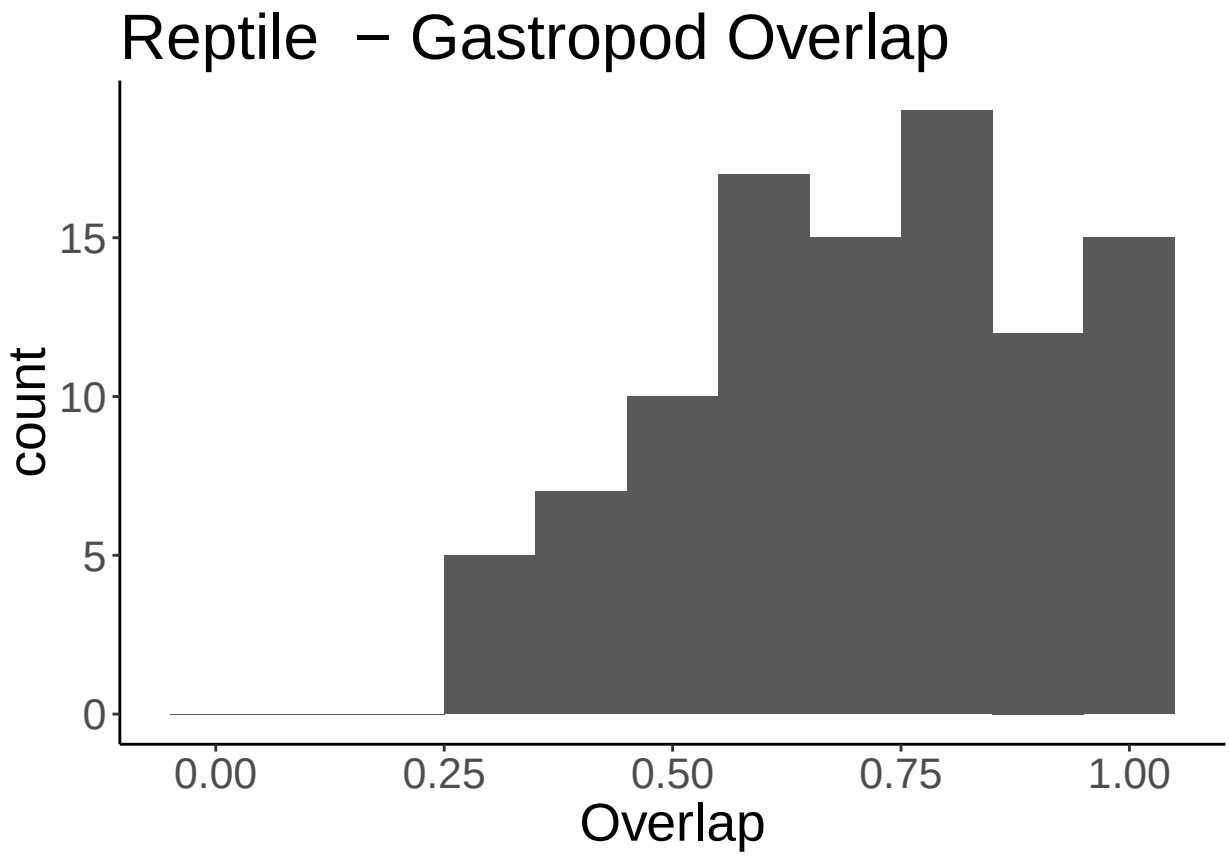

```
####aves mammal
AvM95.overlap <- bayesianOverlap(ellipse_Aves,
                                 ellipse_Mammalia,
                                 ellipses.posterior,
                                 draws = 100,
                                 p.interval = 0.95,
                                 n = 100)

AvM95_overlap_prop <- vector()
for(i in 1:length(AvM95.overlap$overlap)){

AvM95_overlap_prop[i]  <- AvM95.overlap$overlap[i]/min(AvM95.overlap[i,1:2])

}

myplot_AvM = ggplot(data.frame(Overlap =  AvM95_overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Aves  - Mammal Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AvM + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



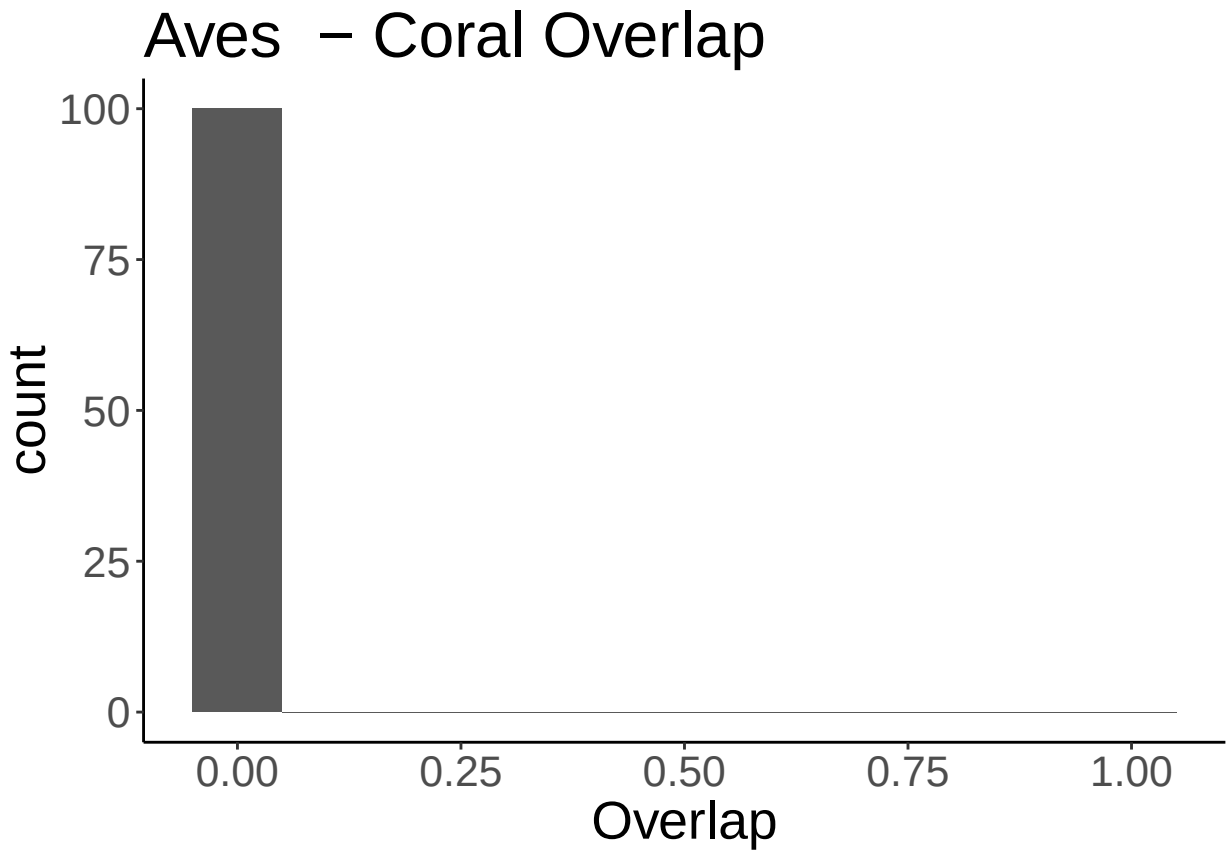

```
##### Aves reptile

AvRR95.overlap <- bayesianOverlap(ellipse_Aves,
                                  ellipse_Reptilia,
                                  ellipses.posterior,
                                  draws = 100,
                                  p.interval = 0.95,
                                  n = 100)

AvR95_overlap_prop <- vector()
for(i in 1:length(AvRR95.overlap$overlap)){

AvR95_overlap_prop[i]  <- AvRR95.overlap$overlap[i]/min(AvRR95.overlap[i,1:2])

}

myplot_AvR = ggplot(data.frame(Overlap =  AvR95_overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Aves  - Reptile Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AvR + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```



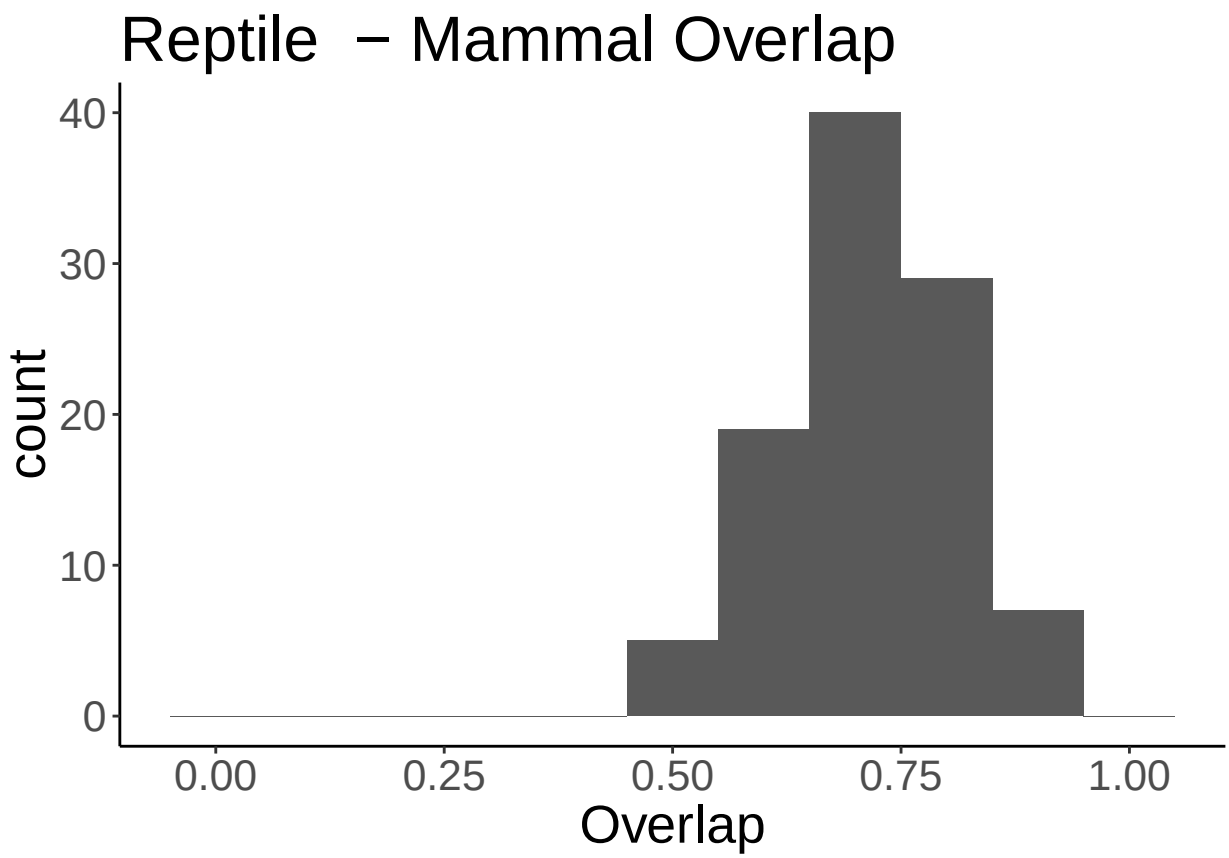

```
##### reptile gastropod

RG95.overlap <- bayesianOverlap(ellipse_Reptilia,
                                ellipse_Gastropoda,
                                ellipses.posterior,
                                draws = 100,
                                p.interval = 0.95,
                                n = 100)

RG95_overlap_prop <- vector()
for(i in 1:length(RG95.overlap$overlap)){

RG95_overlap_prop[i]  <- RG95.overlap$overlap[i]/min(RG95.overlap[i,1:2])

}

myplot_RG = ggplot(data.frame(Overlap =  RG95_overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Reptile  - Gastropod Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_RG + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



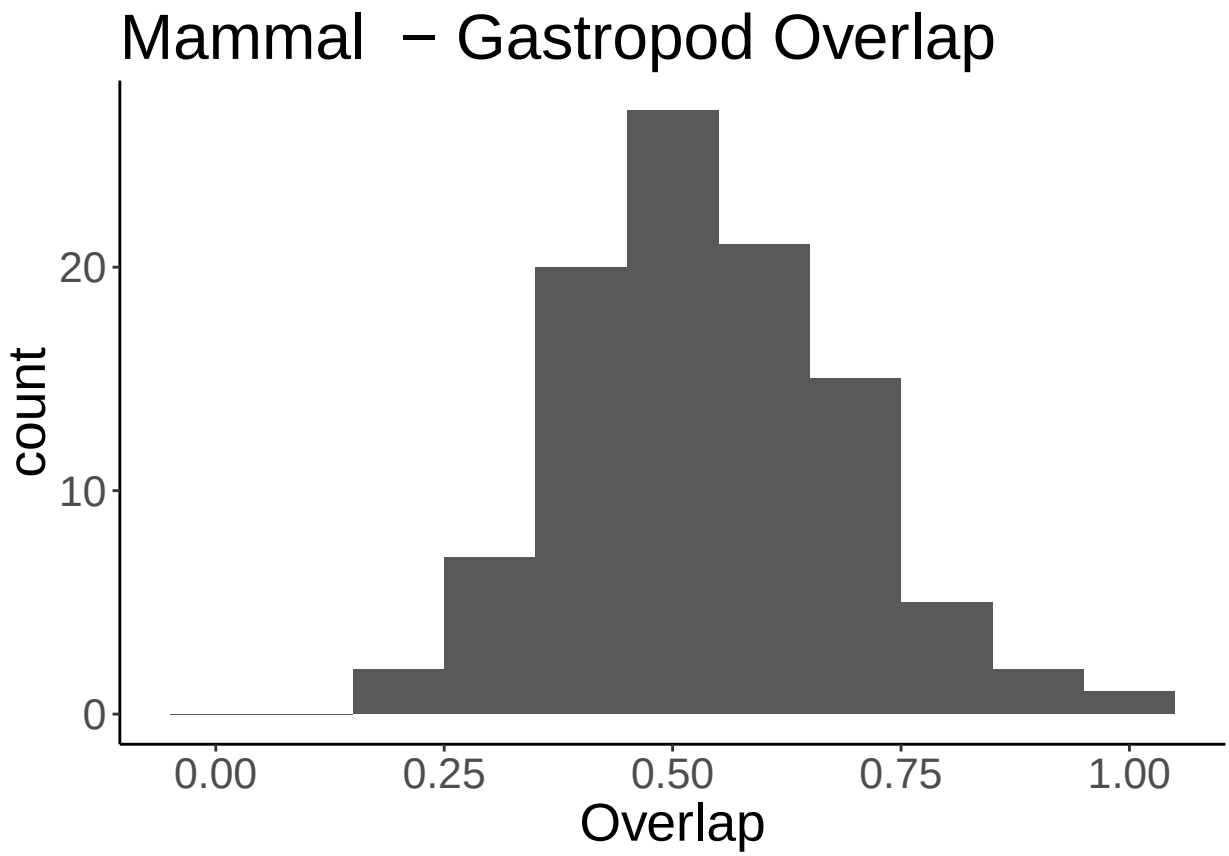

```
##### Aves coral

AvAn95.overlap <- bayesianOverlap(ellipse_Aves,
                                  ellipse_Anthozoa,
                                  ellipses.posterior,
                                  draws = 100,
                                  p.interval = 0.95,
                                  n = 100)

AvAn95_overlap_prop <- vector()
for(i in 1:length(AvAn95.overlap$overlap)){

AvAn95_overlap_prop[i]  <- AvAn95.overlap$overlap[i]/min(AvAn95.overlap[i,1:2])

}

myplot_AvAn = ggplot(data.frame(Overlap =  AvAn95_overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Aves  - Coral Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_AvAn + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```




```
##### reptile mammal

RM95.overlap <- bayesianOverlap(ellipse_Mammalia,
                                ellipse_Reptilia,
                                ellipses.posterior,
                                draws = 100,
                                p.interval = 0.95,
                                n = 100)

RM95_overlap_prop <- vector()
for(i in 1:length(RG95.overlap$overlap)){

RM95_overlap_prop[i]  <- RM95.overlap$overlap[i]/min(RM95.overlap[i,1:2])

}

myplot_RM = ggplot(data.frame(Overlap =  RM95_overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Reptile  - Mammal Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_RM + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```




```
#####  mammal gastropod

MG95.overlap <- bayesianOverlap(ellipse_Mammalia,
                                ellipse_Gastropoda,
                                ellipses.posterior,
                                draws = 100,
                                p.interval = 0.95,
                                n = 100)

MG95_overlap_prop <- vector()
for(i in 1:length(RG95.overlap$overlap)){

MG95_overlap_prop[i]  <- MG95.overlap$overlap[i]/min(MG95.overlap[i,1:2])

}

myplot_MG = ggplot(data.frame(Overlap =  MG95_overlap_prop),
                   aes(Overlap)) +
                   ggtitle("Mammal  - Gastropod Overlap") +
                   geom_histogram(binwidth=0.1) +
                   scale_x_continuous(limits = c(-0.05, 1.05))

myplot_MG + theme_bw() + theme(panel.border = element_blank(),
                               panel.grid.major = element_blank(),
                               panel.grid.minor = element_blank(),
                               axis.line = element_line(colour = "black"),
                               text = element_text(size=20))
```



IUCN overlap calculations for mode of life

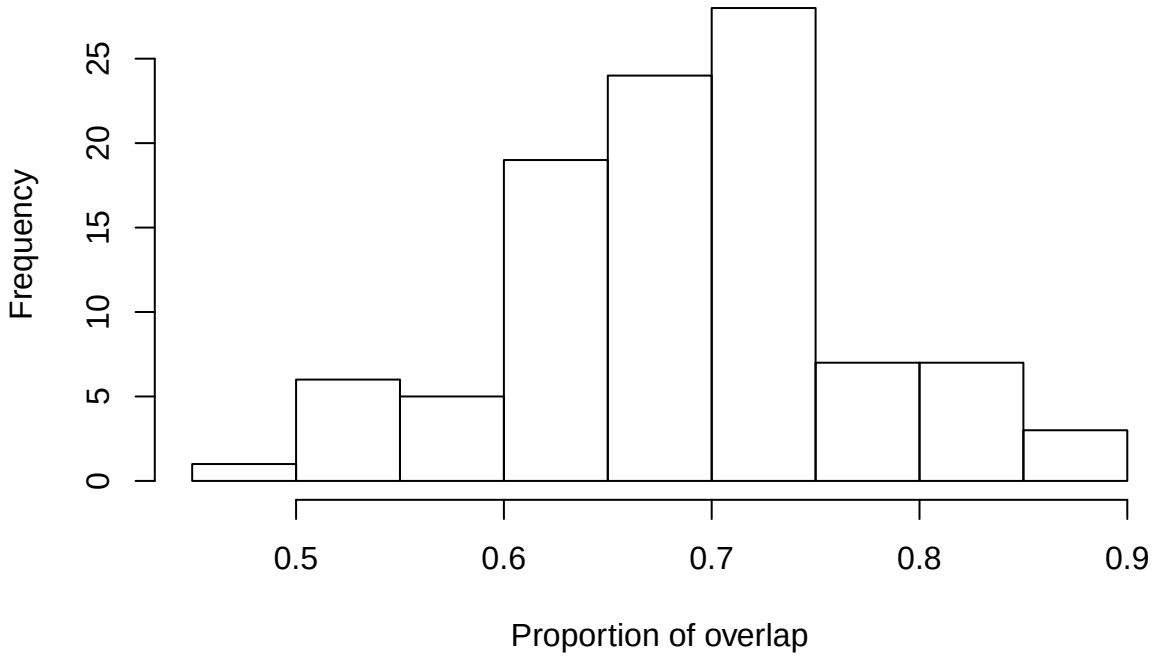
```
group.ML <- groupMetricsML(siber.plots)
group.ML_iucn <- groupMetricsML(siber.iucn)


# options for running jags
parms <- list()
parms$n.iter <- 2 * 10^4   # number of iterations to run the model for
parms$n.burnin <- 1 * 10^3 # discard the first set of values
parms$n.thin <- 10     # thin the posterior by this many
parms$n.chains <- 2        # run this many chains

# define the priors
priors <- list()
priors$R <- 1 * diag(2)
priors$k <- 2
priors$tau.mu <- 1.0E-3



ellipses.posterior_iucn <- siberMVN(siber.iucn, parms, priors)
```

```
## Compiling model graph
```

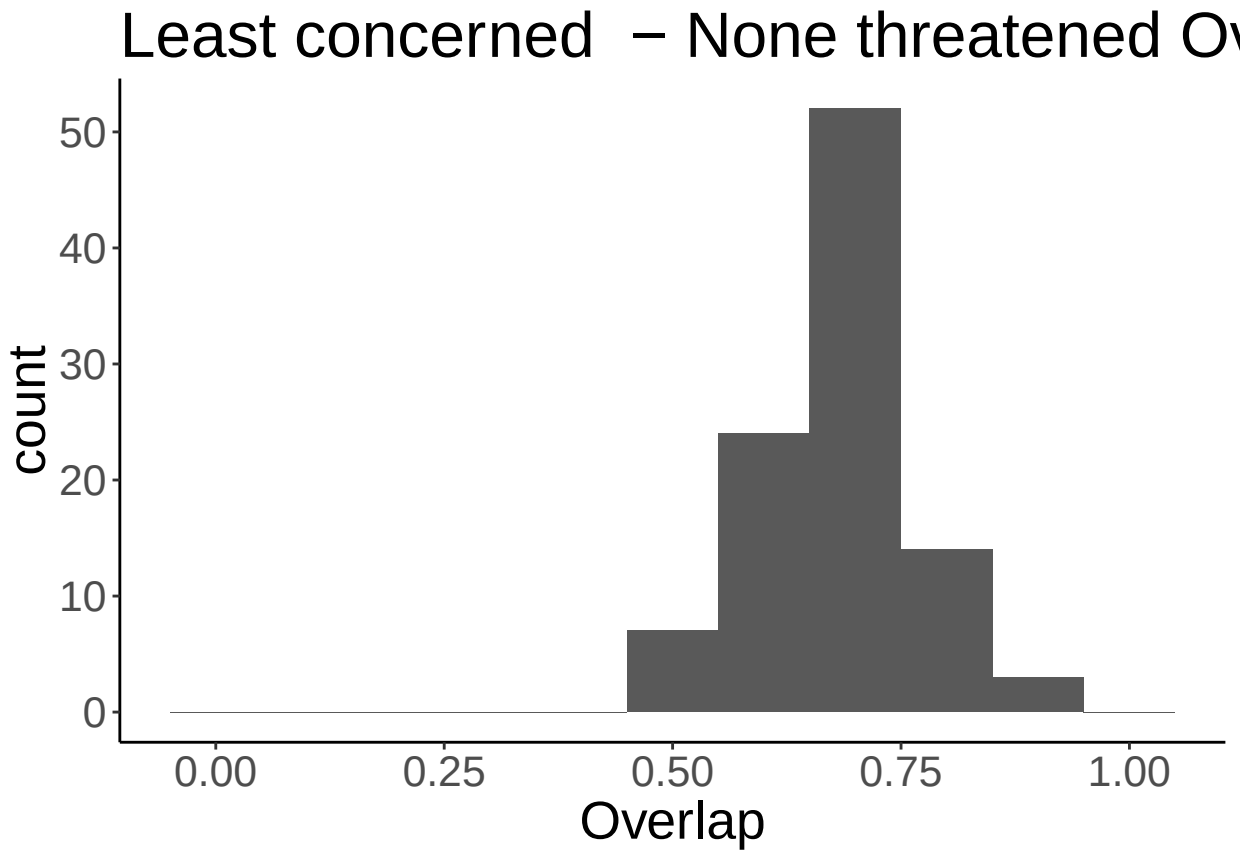
```
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 31
##    Unobserved stochastic nodes: 3
##    Total graph size: 46
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 191
##    Unobserved stochastic nodes: 3
##    Total graph size: 206
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 3
##    Total graph size: 25
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 3
##    Total graph size: 36
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 14
##    Unobserved stochastic nodes: 3
##    Total graph size: 29
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 16
```

```
##    Unobserved stochastic nodes: 3
##    Total graph size: 31
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2
##    Unobserved stochastic nodes: 3
##    Total graph size: 17
##
## Initializing model
```

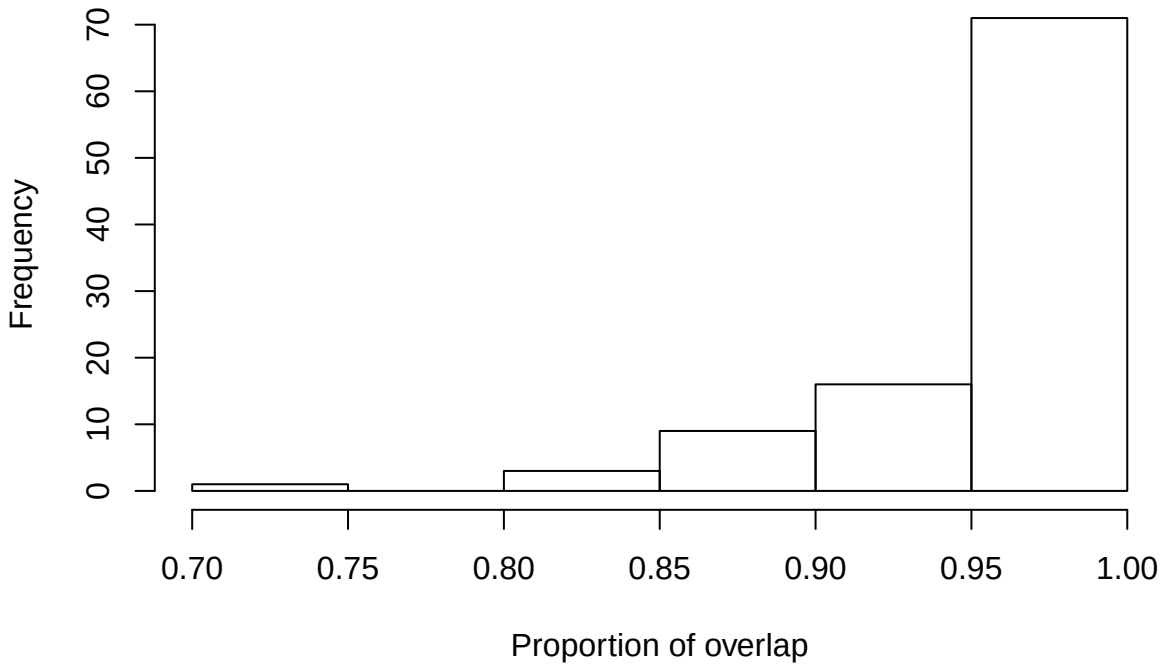
```r
# The first ellipse is referenced using a character string representation where
# in "x.y", "x" is the community, and "y" is the group within that community.
# So in this example: community 1, group 1


#ellipse group numbers
ellipse_NA <- "1.1"
ellipse_CE <- "1.2"
ellipse_E <- "1.3"
ellipse_LC <- "1.4"
ellipse_LR <- "1.5"
ellipse_NT <- "1.6"
ellipse_V <- "1.7"


#####LC  - NT
LC_NT_95.overlap <- bayesianOverlap(ellipse_LC,
                                    ellipse_E,
                                    ellipses.posterior_iucn,
                                    draws = 100,
                                    p.interval = 0.95,
                                    n = 100)
LC_NT_95.overlap_prop <- vector()
for(i in 1:length(LC_NT_95.overlap$overlap)){

LC_NT_95.overlap_prop[i]  <- LC_NT_95.overlap$overlap[i]/min(LC_NT_95.overlap[i,1:2])

}

hist(LC_NT_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



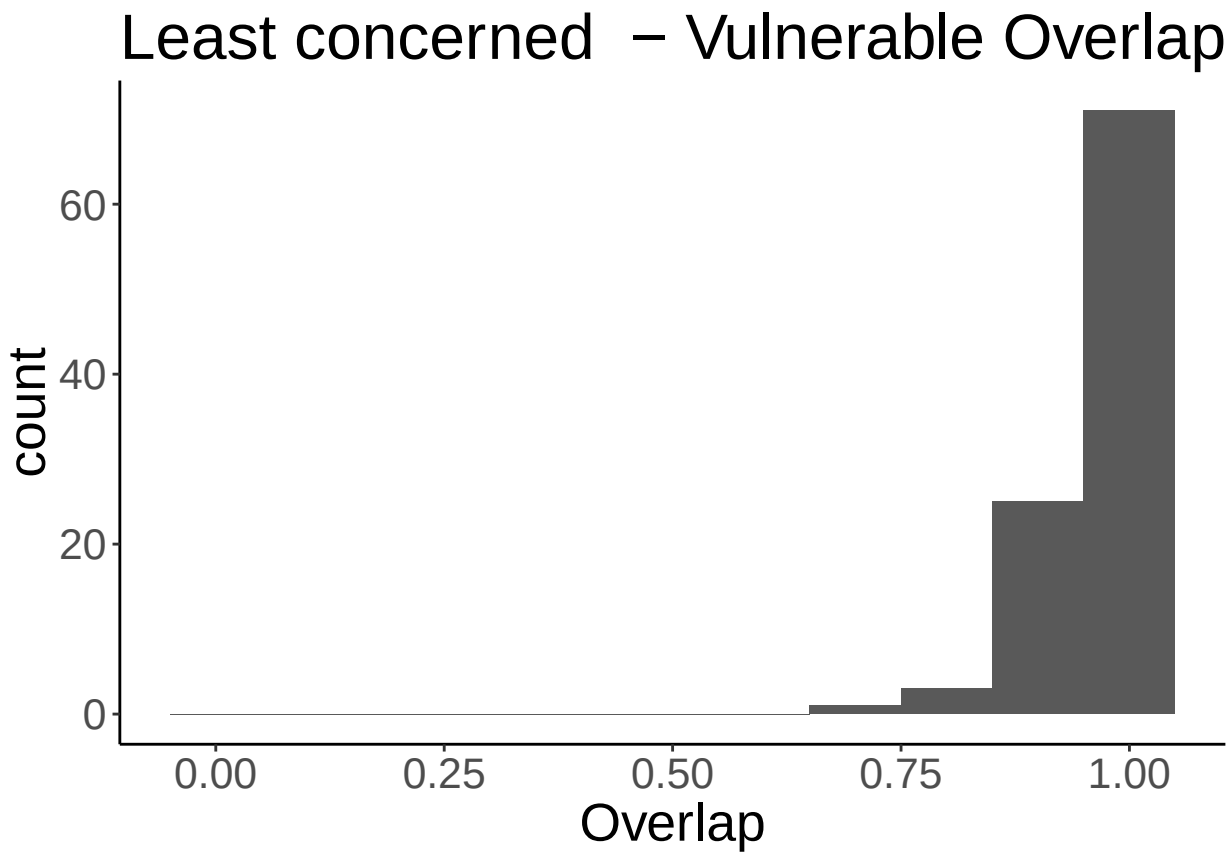

```
myplot_LC_NT = ggplot(data.frame(Overlap =  LC_NT_95.overlap_prop),
                      aes(Overlap)) +
                      ggtitle("Least concerned  - None threatened Overlap") +
                      geom_histogram(binwidth=0.1) +
                      scale_x_continuous(limits = c(-0.05, 1.05))


myplot_LC_NT + theme_bw() + theme(panel.border = element_blank(),
                                  panel.grid.major = element_blank(),
                                  panel.grid.minor = element_blank(),
                                  axis.line = element_line(colour = "black"),
                                  text = element_text(size=20))
```



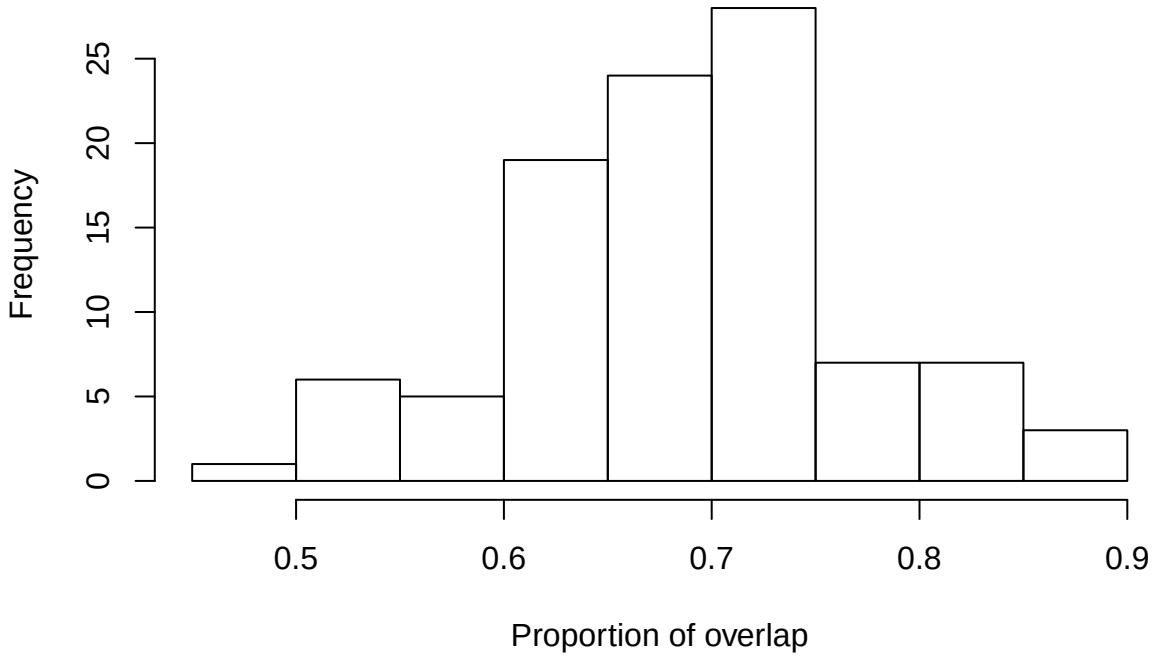

```
#####LC  - V
LC_V_95.overlap <- bayesianOverlap(ellipse_LC,
                                   ellipse_V,
                                   ellipses.posterior_iucn,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
LC_V_95.overlap_prop <- vector()
for(i in 1:length(LC_V_95.overlap$overlap)){

LC_V_95.overlap_prop[i]  <- LC_V_95.overlap$overlap[i]/min(LC_V_95.overlap[i,1:2])

}

hist(LC_V_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



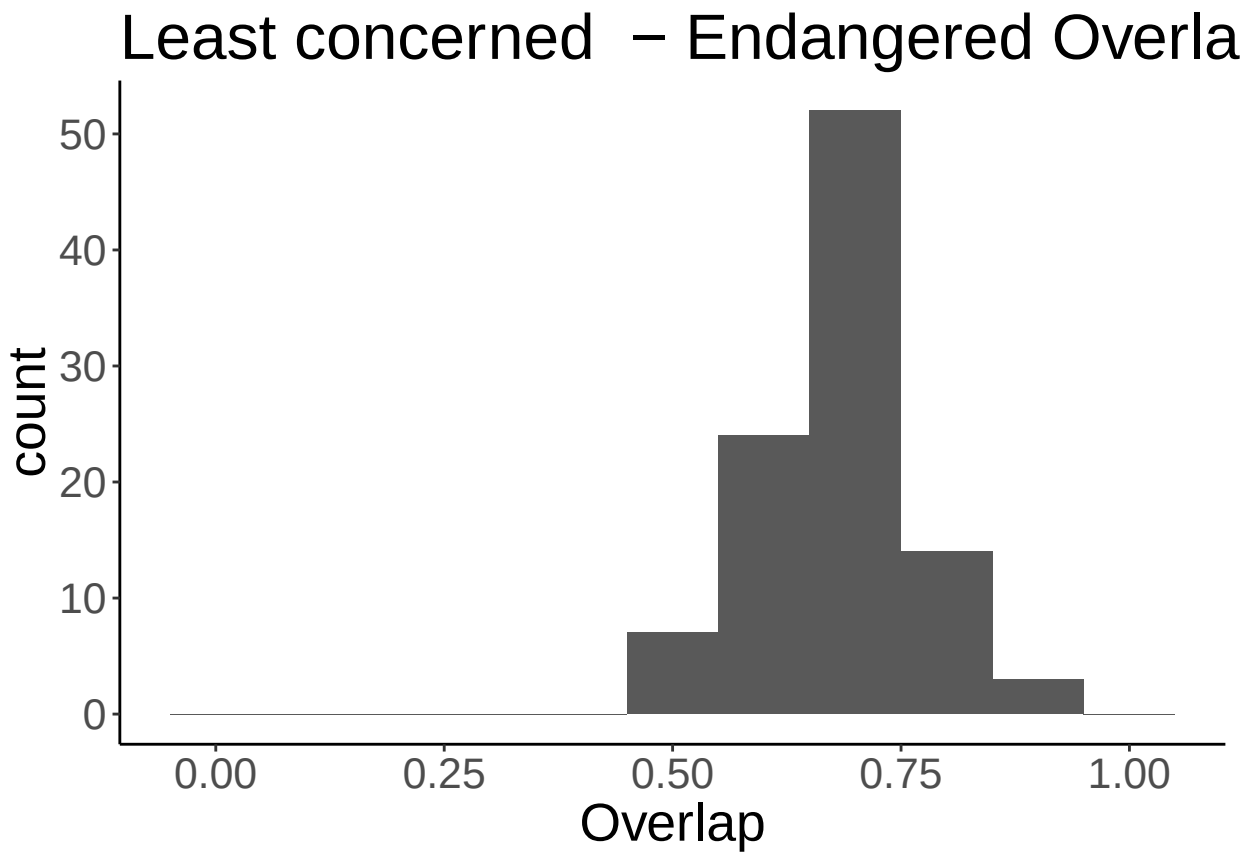

```
myplot_LC_V = ggplot(data.frame(Overlap =  LC_V_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Least concerned  - Vulnerable Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_LC_V + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



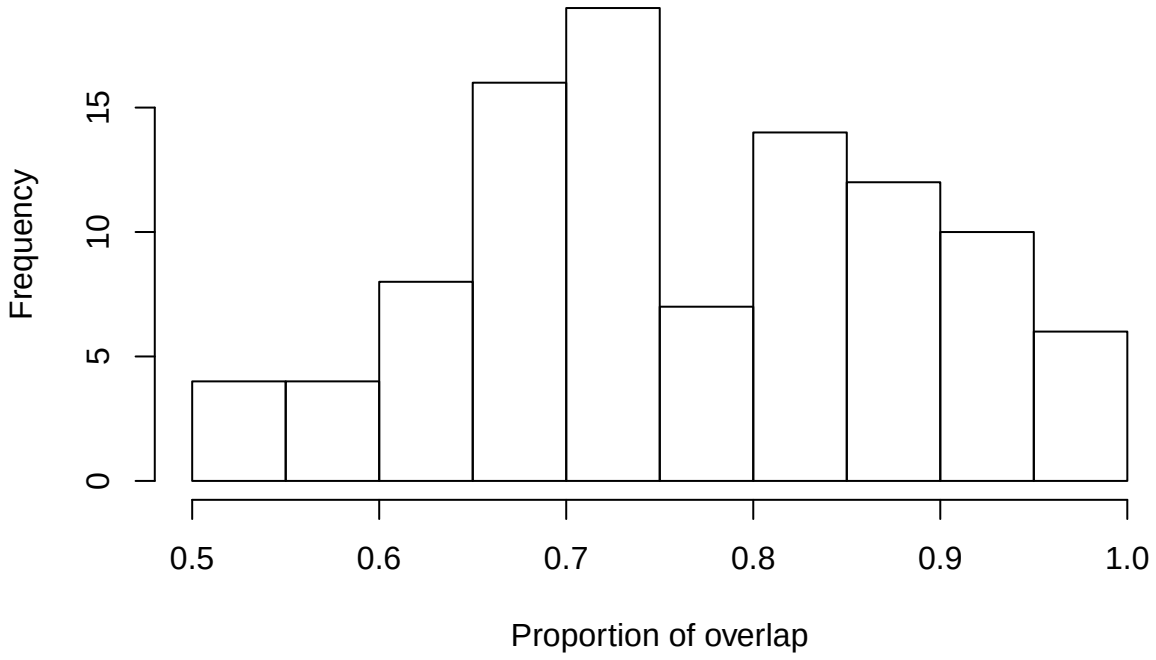

```
#####LC  - E
LC_E_95.overlap <- bayesianOverlap(ellipse_LC,
                                   ellipse_E,
                                   ellipses.posterior_iucn,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
LC_E_95.overlap_prop <- vector()
for(i in 1:length(LC_E_95.overlap$overlap)){

LC_E_95.overlap_prop[i]  <- LC_E_95.overlap$overlap[i]/min(LC_E_95.overlap[i,1:2])

}

hist(LC_E_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



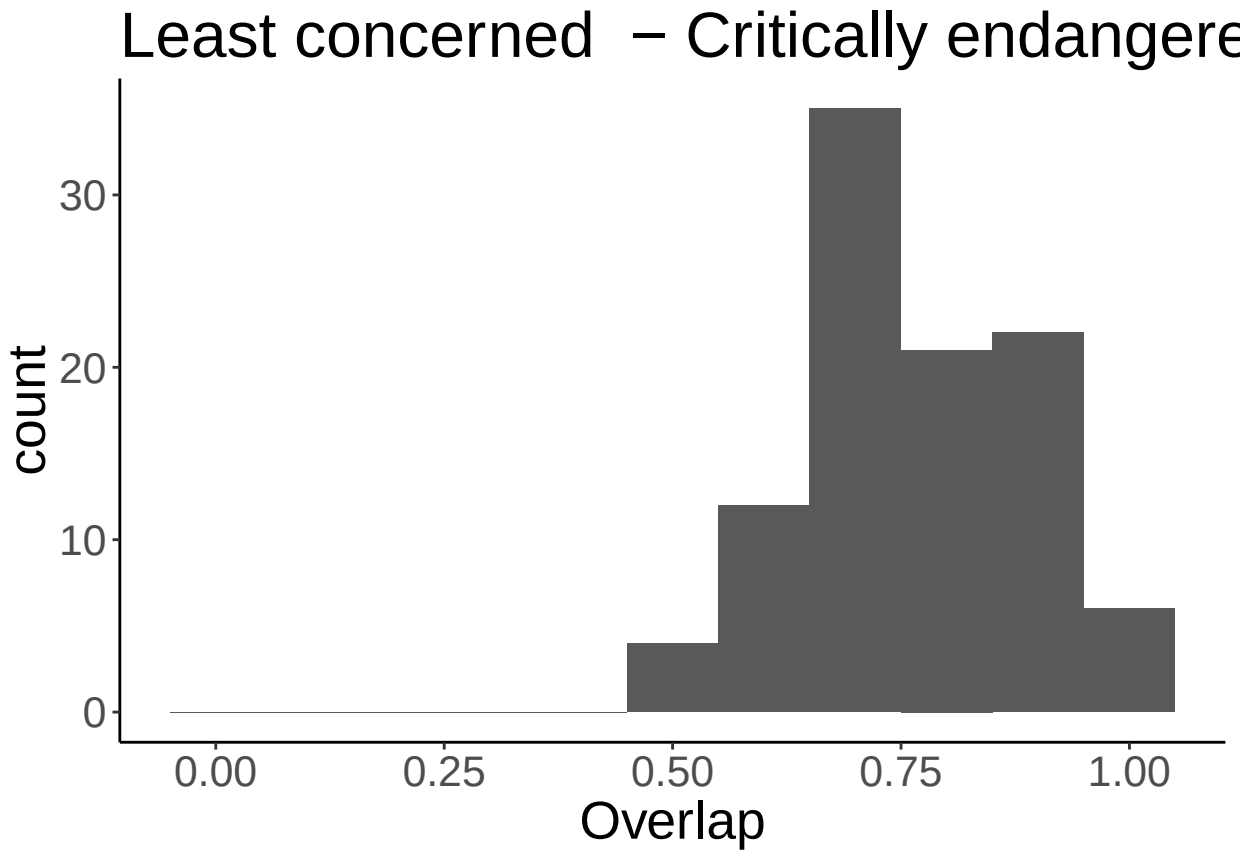

```
myplot_LC_E = ggplot(data.frame(Overlap =  LC_E_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Least concerned  - Endangered Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_LC_E + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



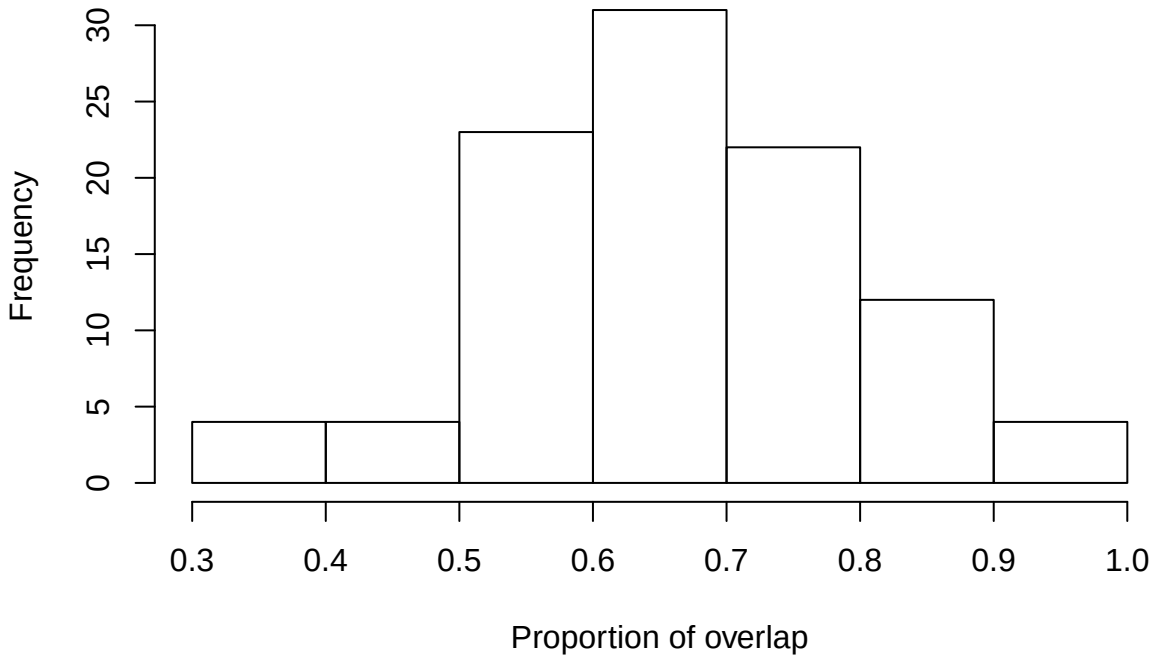

```
#####LC  - CE
LC_CE_95.overlap <- bayesianOverlap(ellipse_LC,
                                    ellipse_CE,
                                    ellipses.posterior_iucn,
                                    draws = 100,
                                    p.interval = 0.95,
                                    n = 100)
LC_CE_95.overlap_prop <- vector()
for(i in 1:length(LC_CE_95.overlap$overlap)){

LC_CE_95.overlap_prop[i]  <- LC_CE_95.overlap$overlap[i]/min(LC_CE_95.overlap[i,1:2])

}

hist(LC_CE_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



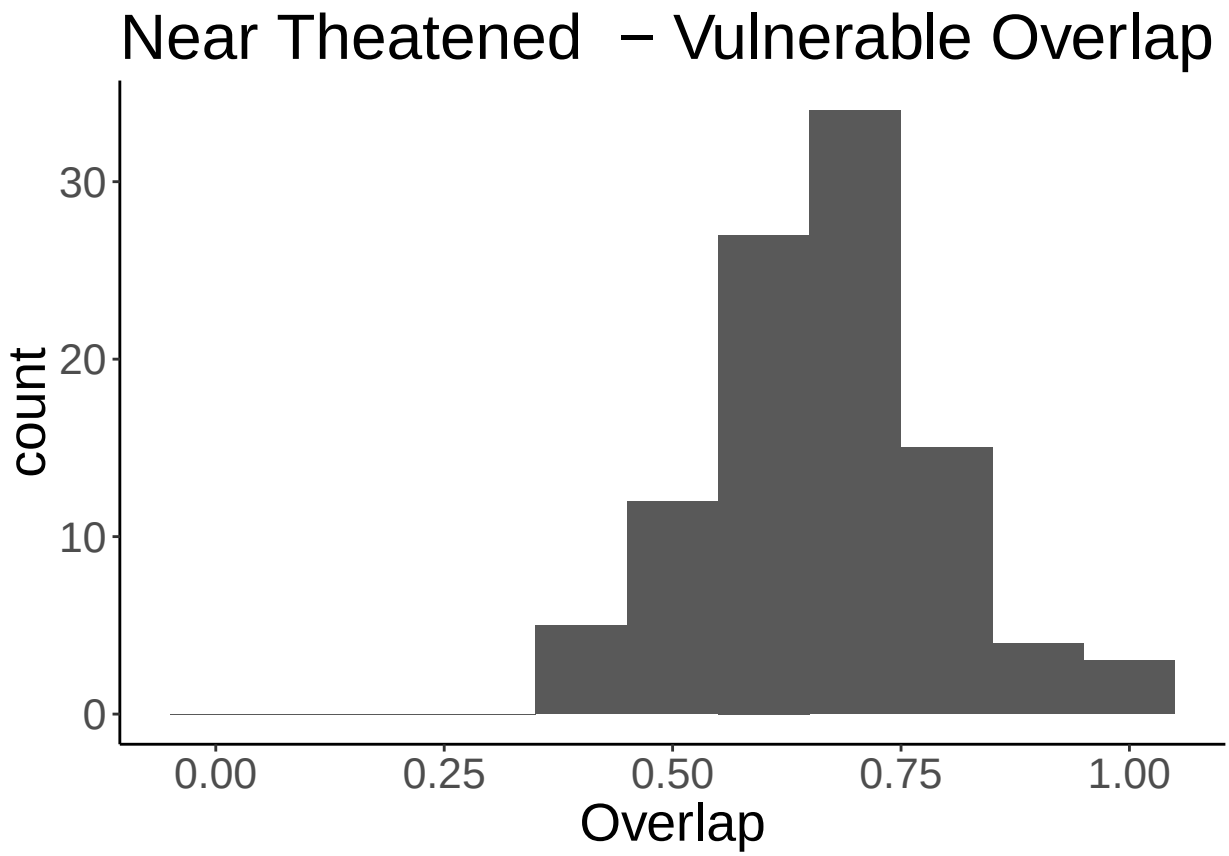

```
myplot_LC_CE = ggplot(data.frame(Overlap =  LC_CE_95.overlap_prop),
                      aes(Overlap)) +
                      ggtitle("Least concerned  - Critically endangered Overlap") +
                      geom_histogram(binwidth=0.1) +
                      scale_x_continuous(limits = c(-0.05, 1.05))

myplot_LC_CE + theme_bw() + theme(panel.border = element_blank(),
                                  panel.grid.major = element_blank(),
                                  panel.grid.minor = element_blank(),
                                  axis.line = element_line(colour = "black"),
                                  text = element_text(size=20))
```



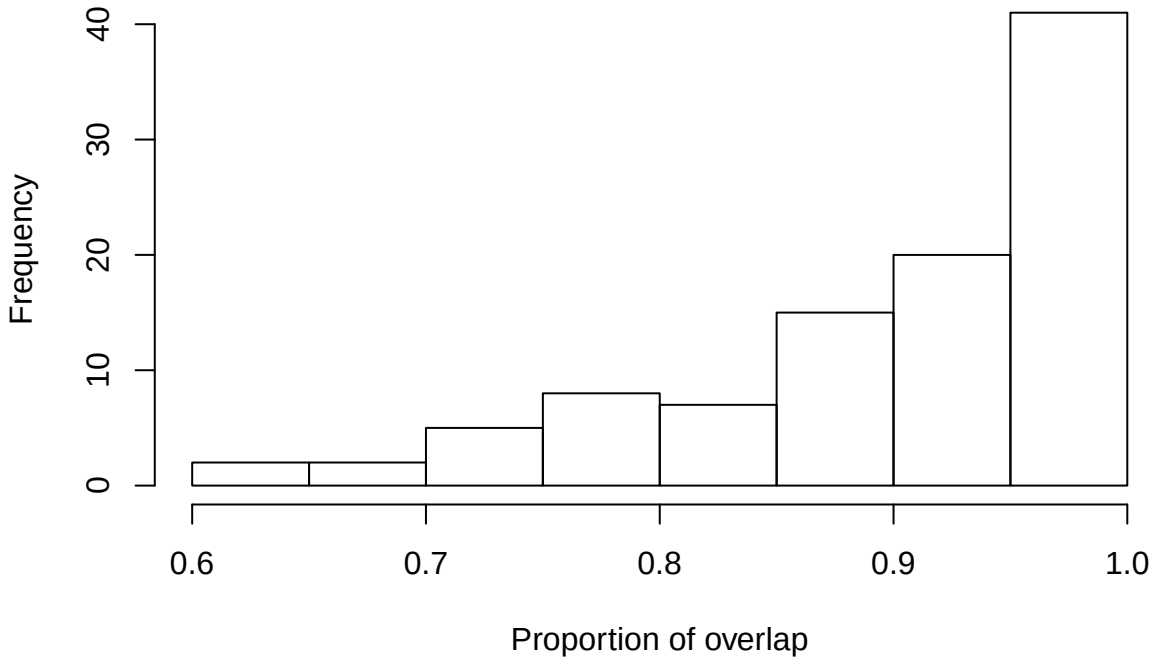

```
#####NT  - V
NT_V_95.overlap <- bayesianOverlap(ellipse_NT,
                                   ellipse_V,
                                   ellipses.posterior_iucn,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
NT_V_95.overlap_prop <- vector()
for(i in 1:length(NT_V_95.overlap$overlap)){

NT_V_95.overlap_prop[i]  <- NT_V_95.overlap$overlap[i]/min(NT_V_95.overlap[i,1:2])

}

hist(NT_V_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



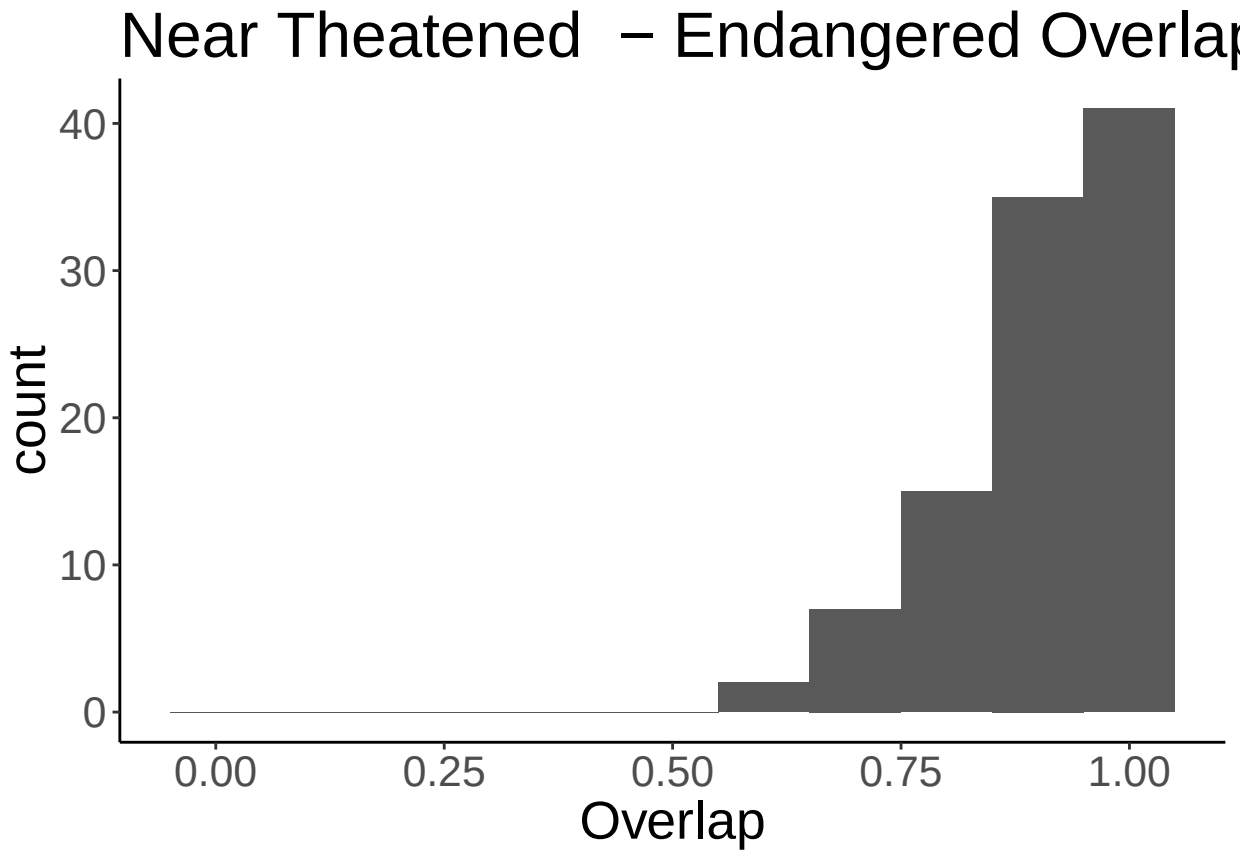

```
myplot_NT_V = ggplot(data.frame(Overlap =  NT_V_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Near Theatened  - Vulnerable Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_NT_V + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



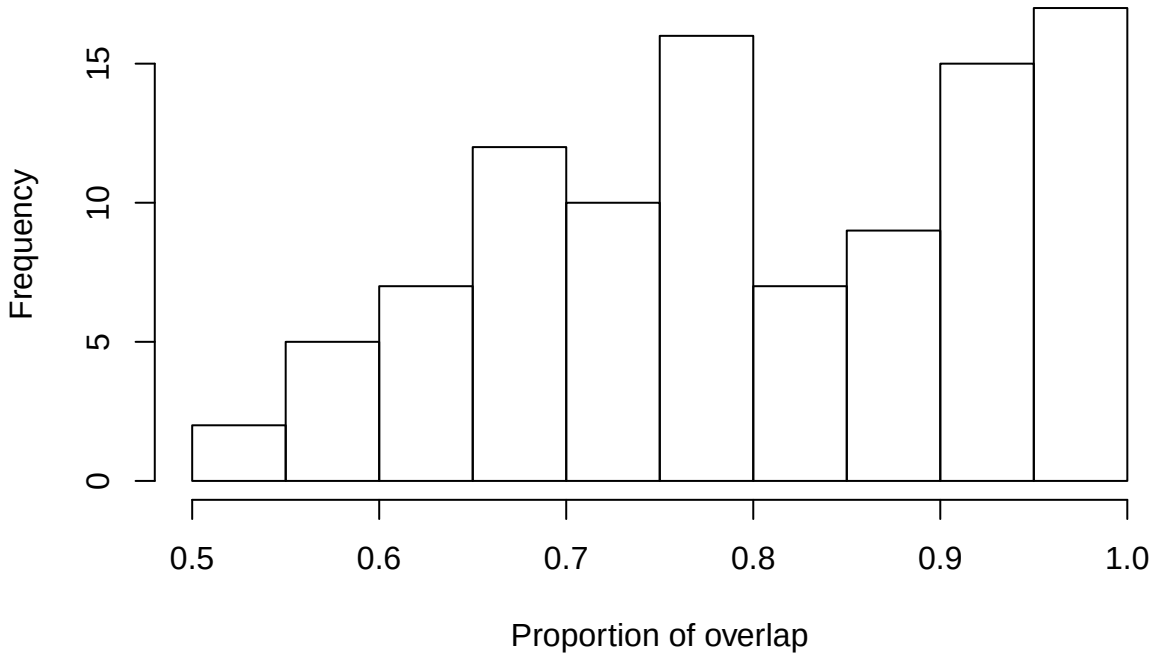

```
#####NT  - E
NT_E_95.overlap <- bayesianOverlap(ellipse_NT,
                                   ellipse_E,
                                   ellipses.posterior_iucn,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
NT_E_95.overlap_prop <- vector()
for(i in 1:length(NT_E_95.overlap$overlap)){

NT_E_95.overlap_prop[i]  <- NT_E_95.overlap$overlap[i]/min(NT_E_95.overlap[i,1:2])

}

hist(NT_E_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



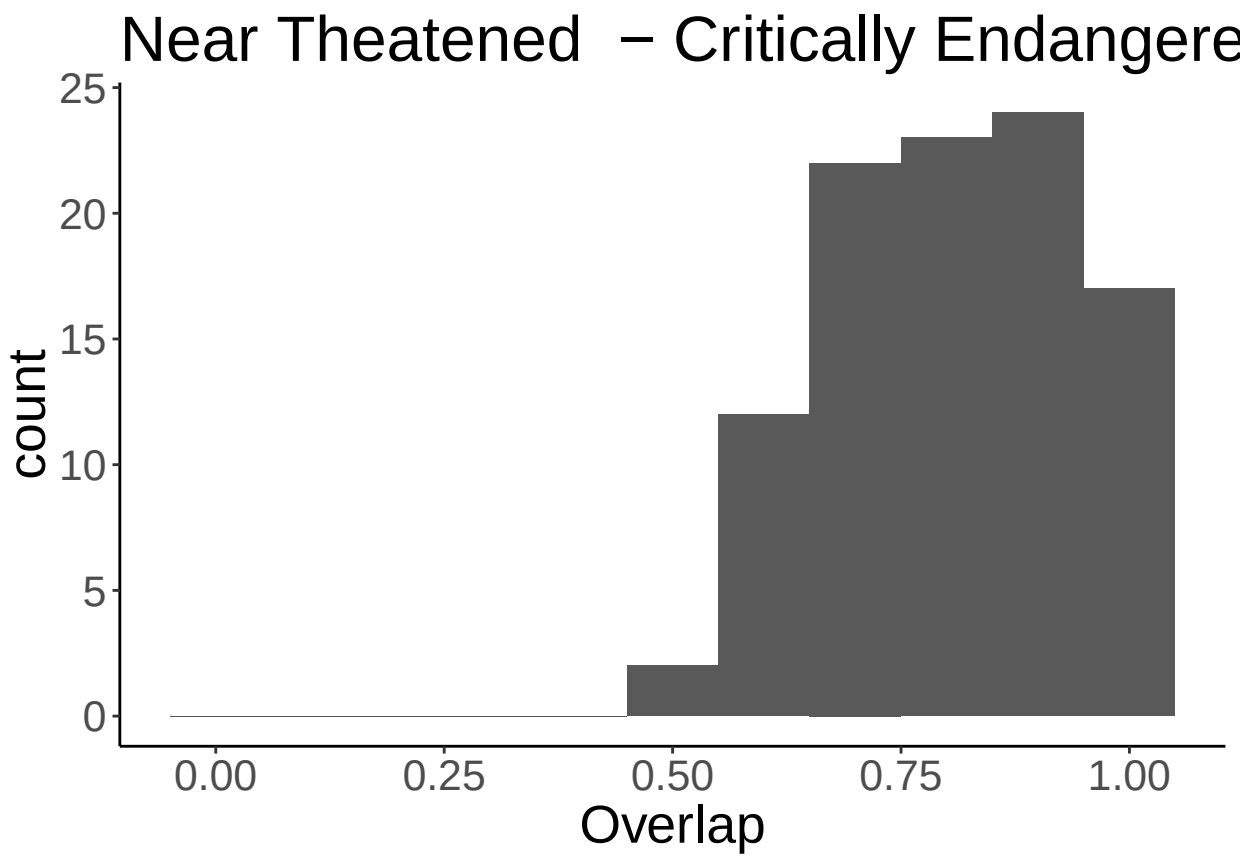

```
myplot_NT_E = ggplot(data.frame(Overlap =  NT_E_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Near Theatened  - Endangered Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_NT_E + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



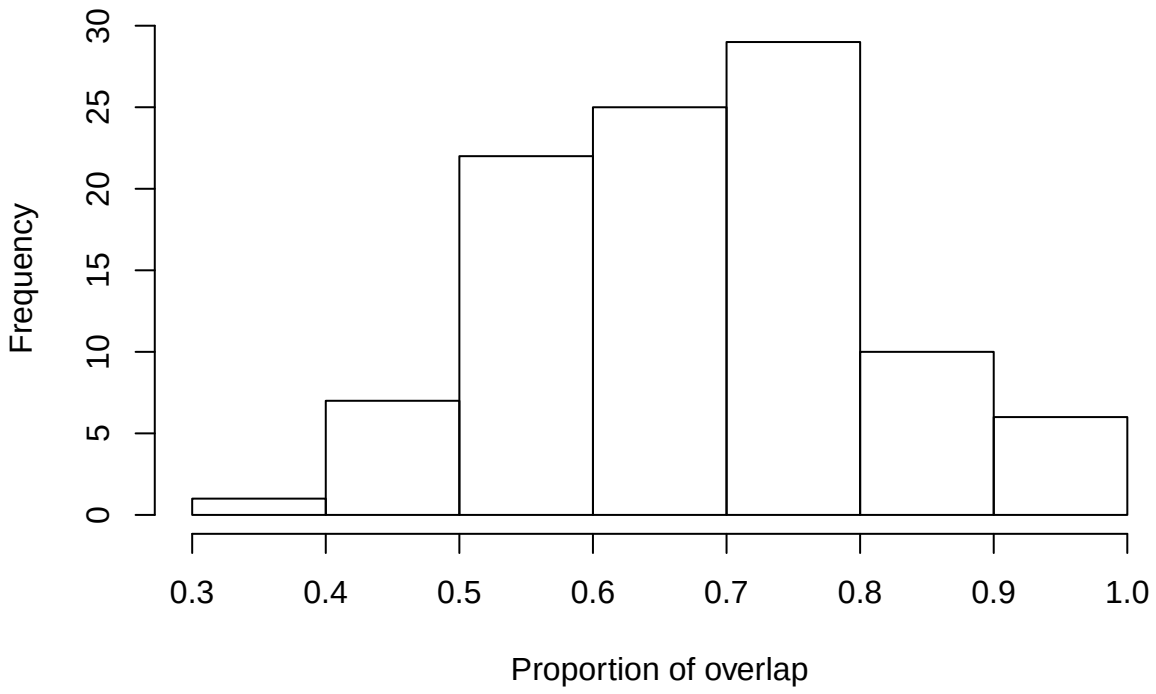

```
#####NT  - CE
NT_CE_95.overlap <- bayesianOverlap(ellipse_NT,
                                    ellipse_CE,
                                    ellipses.posterior_iucn,
                                    draws = 100,
                                    p.interval = 0.95,
                                    n = 100)
NT_CE_95.overlap_prop <- vector()
for(i in 1:length(NT_CE_95.overlap$overlap)){

NT_CE_95.overlap_prop[i]  <- NT_CE_95.overlap$overlap[i]/min(NT_CE_95.overlap[i,1:2])

}

hist(NT_CE_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



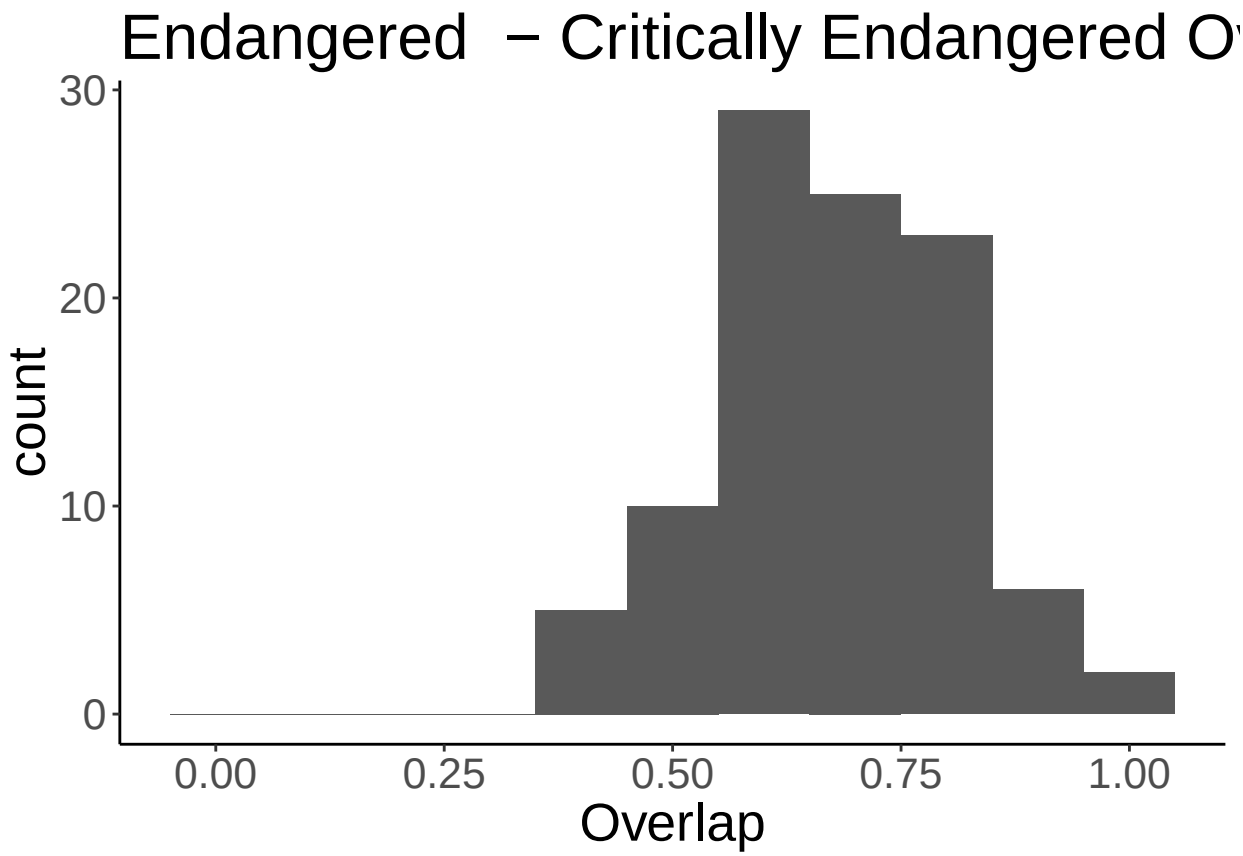

```
myplot_NT_CE = ggplot(data.frame(Overlap =  NT_CE_95.overlap_prop),
                      aes(Overlap)) +
                      ggtitle("Near Theatened  - Critically Endangered Overlap") +
                      geom_histogram(binwidth=0.1) +
                      scale_x_continuous(limits = c(-0.05, 1.05))

myplot_NT_CE + theme_bw() + theme(panel.border = element_blank(),
                                  panel.grid.major = element_blank(),
                                  panel.grid.minor = element_blank(),
                                  axis.line = element_line(colour = "black"),
                                  text = element_text(size=20))
```



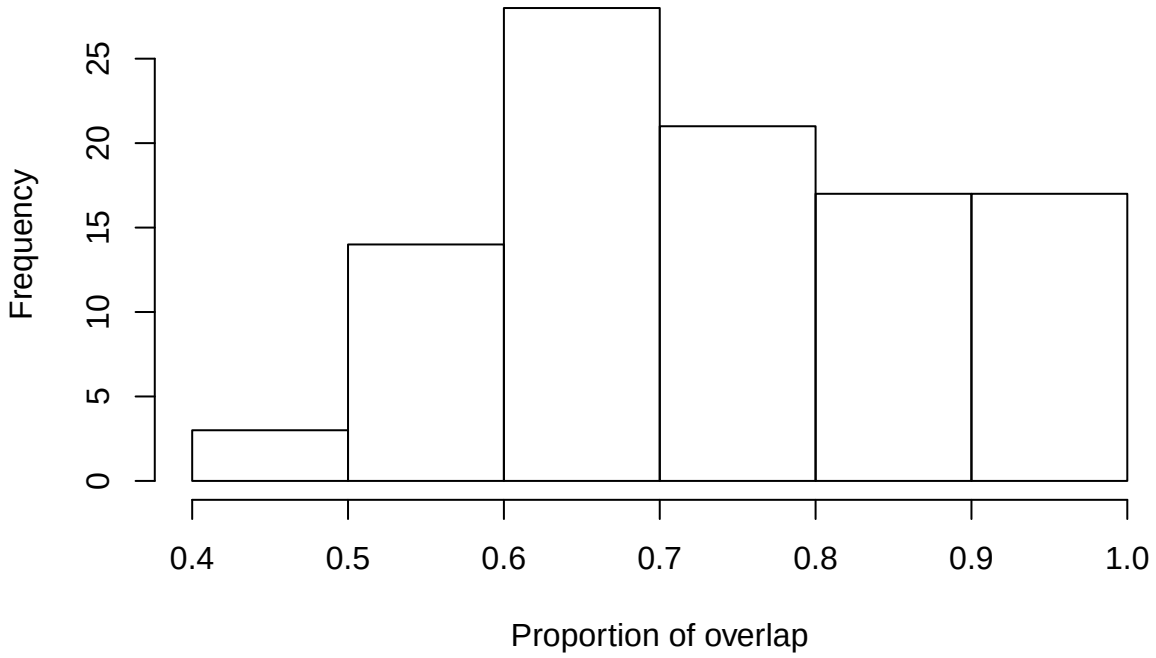

```
#####E  - CE
E_CE_95.overlap <- bayesianOverlap(ellipse_E,
                                   ellipse_CE,
                                   ellipses.posterior_iucn,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
E_CE_95.overlap_prop <- vector()
for(i in 1:length(E_CE_95.overlap$overlap)){

E_CE_95.overlap_prop[i]  <- E_CE_95.overlap$overlap[i]/min(E_CE_95.overlap[i,1:2])

}

hist(E_CE_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



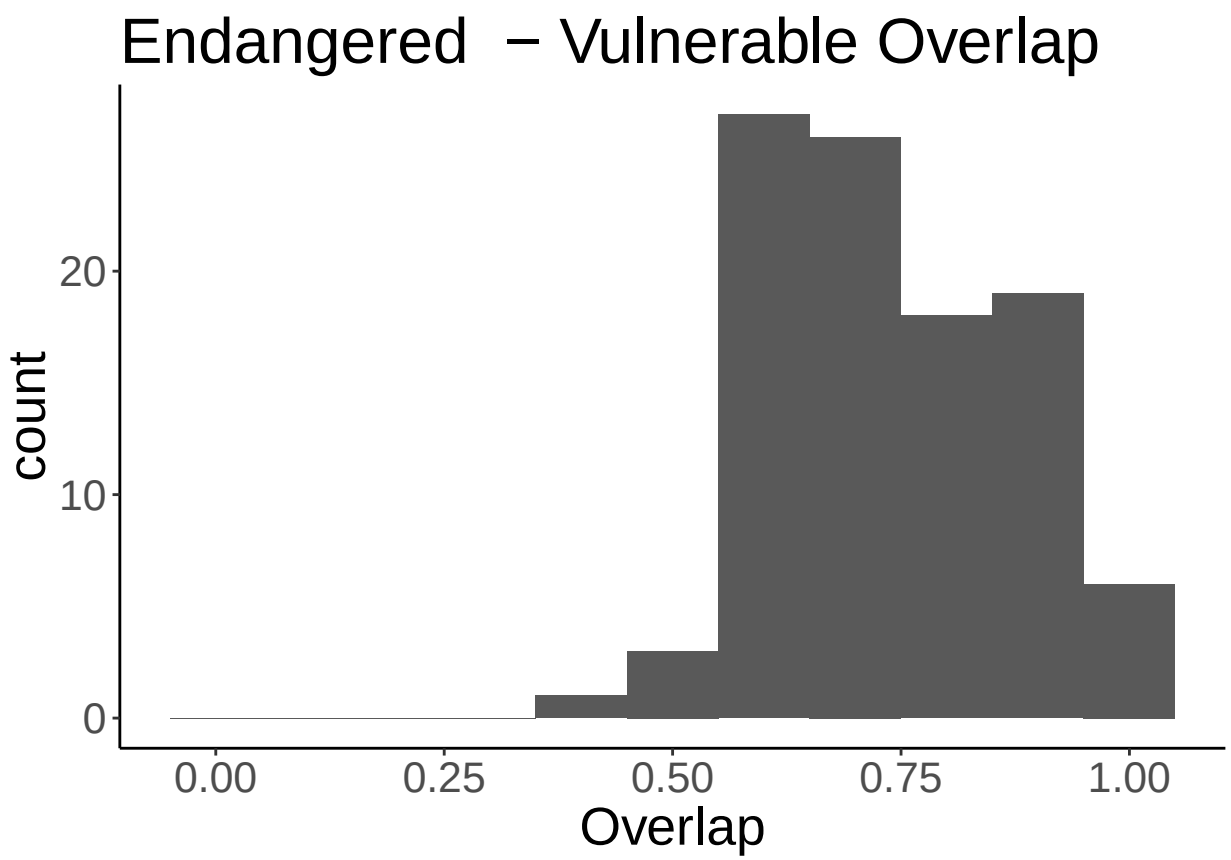

```
myplot_E_CE = ggplot(data.frame(Overlap =  E_CE_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Endangered  - Critically Endangered Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_E_CE + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



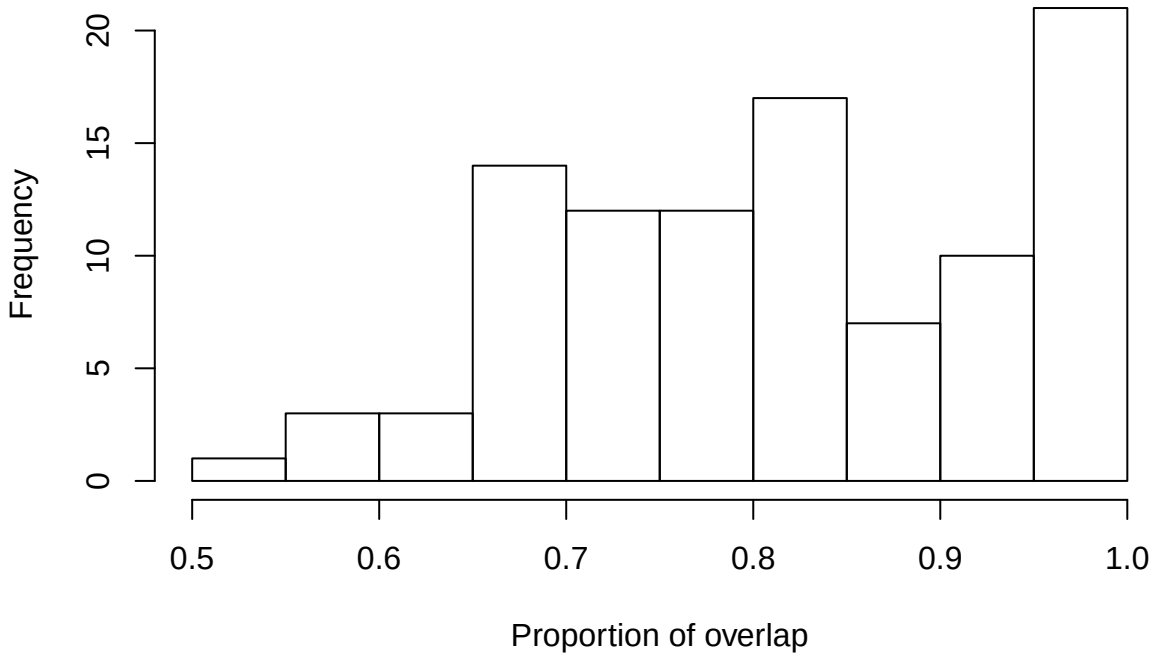

```
#####E  - V
E_V_95.overlap <- bayesianOverlap(ellipse_E,
                                  ellipse_V,
                                  ellipses.posterior_iucn,
                                  draws = 100,
                                  p.interval = 0.95,
                                  n = 100)
E_V_95.overlap_prop <- vector()
for(i in 1:length(E_V_95.overlap$overlap)){

E_V_95.overlap_prop[i]  <- E_V_95.overlap$overlap[i]/min(E_V_95.overlap[i,1:2])

}

hist(E_V_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```



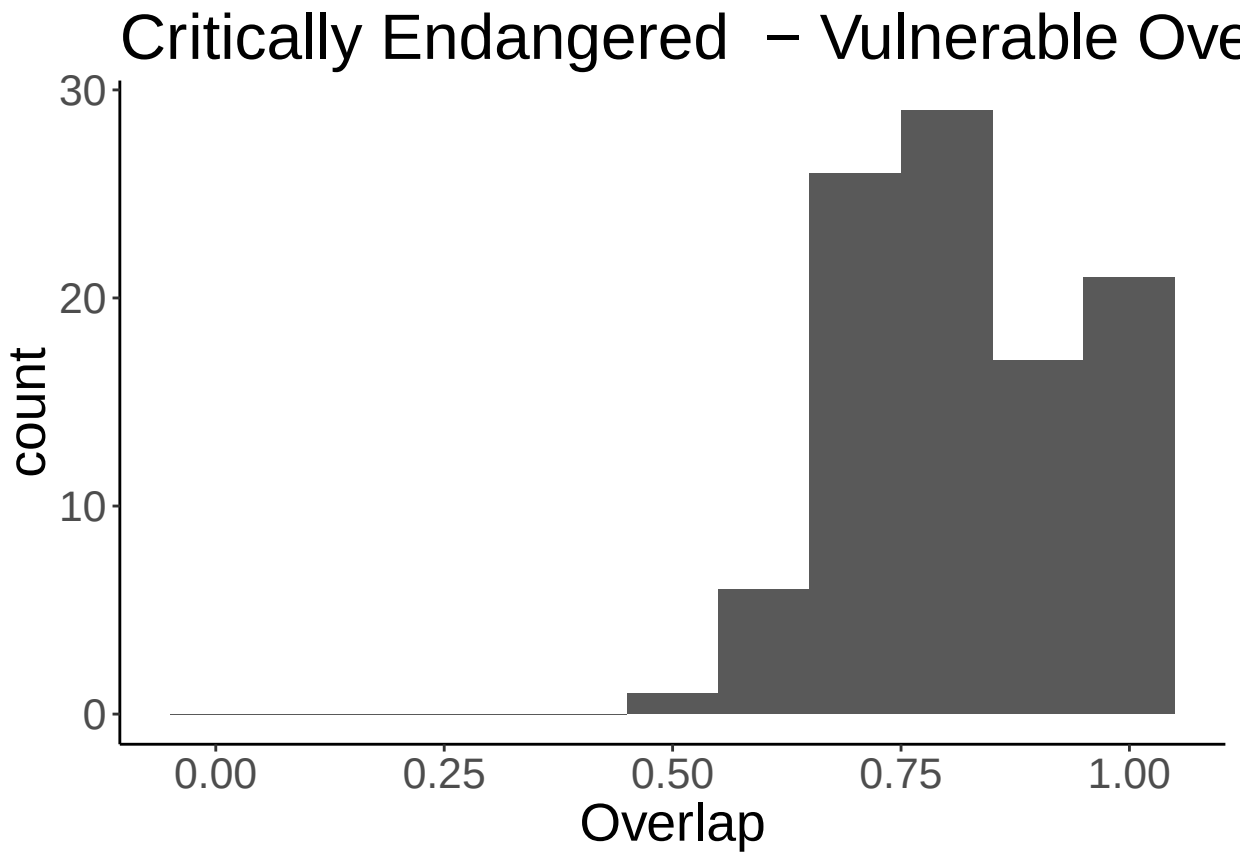

```
myplot_E_V = ggplot(data.frame(Overlap =  E_V_95.overlap_prop),
                    aes(Overlap)) +
                    ggtitle("Endangered  - Vulnerable Overlap") +
                    geom_histogram(binwidth=0.1) +
                    scale_x_continuous(limits = c(-0.05, 1.05))

myplot_E_V + theme_bw() + theme(panel.border = element_blank(),
                                panel.grid.major = element_blank(),
                                panel.grid.minor = element_blank(),
                                axis.line = element_line(colour = "black"),
                                text = element_text(size=20))
```




```
#####CE  - V
CE_V_95.overlap <- bayesianOverlap(ellipse_CE,
                                   ellipse_V,
                                   ellipses.posterior_iucn,
                                   draws = 100,
                                   p.interval = 0.95,
                                   n = 100)
CE_V_95.overlap_prop <- vector()
for(i in 1:length(CE_V_95.overlap$overlap)){

CE_V_95.overlap_prop[i]  <- CE_V_95.overlap$overlap[i]/min(CE_V_95.overlap[i,1:2])

}

hist(CE_V_95.overlap_prop, xlab = "Proportion of overlap", main = "")
```




```
myplot_CE_V = ggplot(data.frame(Overlap =  CE_V_95.overlap_prop),
                     aes(Overlap)) +
                     ggtitle("Critically Endangered  - Vulnerable Overlap") +
                     geom_histogram(binwidth=0.1) +
                     scale_x_continuous(limits = c(-0.05, 1.05))

myplot_CE_V + theme_bw() + theme(panel.border = element_blank(),
                                 panel.grid.major = element_blank(),
                                 panel.grid.minor = element_blank(),
                                 axis.line = element_line(colour = "black"),
                                 text = element_text(size=20))
```



Juvinal Survival versus SD of mortality

```
prior<-list(R = list(V = 1/2, nu=0.002),
            G = list(G1=list(V = 1/2,n = 1, alpha.mu=rep(0,1), alpha.V= diag(1)*10^3),
                     G1=list(V = 1/2,n = 1, alpha.mu=rep(0,1), alpha.V= diag(1)*10^3)))

juv_data <- data.frame(surv_sd = as.vector(pop_data$surv_sd),
                       prop_la = as.vector(pop_data$prop_la),
                       animal = pop_data$animal,
                       species = pop_data$species)



j_surv <- MCMCglmm(prop_la ~ surv_sd,
                   data = juv_data,
                   random=~animal + species,
                   pedigree = axis_trees[[1]],
                   prior = prior,
                   nitt = c(110000), burnin = 10000, thin = 50, verbose = F)

summary(j_surv)
```

```
##
```

```
##  Iterations = 10001:109951
##  Thinning interval  = 50
##  Sample size  = 2000
##
##  DIC: -386.2264
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal   0.08128  0.03129   0.1342     1491
##
##                ~species
##
##         post.mean l-95% CI u-95% CI eff.samp
## species   0.01015 0.003956  0.01644     1753
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units   0.01115 0.008687  0.01352     2000
##
##  Location effects: prop_la ~ surv_sd
##
##             post.mean l-95% CI u-95% CI eff.samp  pMCMC
## (Intercept)    0.4605   0.1300   0.7807     2000  0.005 **
## surv_sd       -4.6707  -6.2276  -2.9508     2000 <5e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
j_surv_vcv_phylo <- j_surv$VCV[,1]
j_surv_vcv_spec <- j_surv$VCV[,2]
j_surv_vcv_units <- j_surv$VCV[,3]

j_surv_phylo <- j_surv_vcv_phylo/c(j_surv_vcv_phylo + j_surv_vcv_spec + j_surv_vcv_units)
j_surv_spec <- j_surv_vcv_spec/c(j_surv_vcv_phylo + j_surv_vcv_spec + j_surv_vcv_units)
j_surv_units <- j_surv_vcv_units/c(j_surv_vcv_phylo + j_surv_vcv_spec + j_surv_vcv_units)

hdr(j_surv_phylo)$mode
```

```
## [1] 0.7940034
```

```
hdr(j_surv_spec)$mode
```

```
## [1] 0.07884778
```

```
hdr(j_surv_units)$mode
```

```
## [1] 0.107091
```