Axis_analysis_19_June

Kevin Healy

19 June 2017

Lets source our packages etc.

```r
library(popbio)
library(popdemo)
```

```
## 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
```

```r
library(ape)
library(caper)
```

```
## Loading required package: MASS

## Loading required package: mvtnorm
```

```r
library(phytools)
```

```
## Loading required package: maps

## Loading required package: rgl
```

```r
library(MCMCglmm)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

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

## Loading required package: coda
```

```r
library(mulTree)
```

```
## Loading required package: hdrcde

## This is hdrcde 3.3

## Loading required package: snow
```

```r
library(ineq)
library(pspline)
library(paran)
library(SIBER)
library(ggplot2)

#devtools to get the Mage package
library(devtools)
source("Demography_functions.R")
```

You should have calulated all the population matrics and created a dataset to uplaod for the analysis here.

```r
pop_data <- read.csv("axis_analysis_data_6_march_2019.csv",
                          sep = ",", header = T)
```

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

Ok now lets do the circular bar plot

```r
#First make a named vector of the mean trait value.

plot_gini <- vector()
plot_sd <- vector()
plot_T <- vector()
plot_repo <- vector()
plot_matrix <- vector()

for(i in 1:length(axis_trees[[1]]$tip.label)){

  plot_matrix[i] <- mean(pop_data[pop_data$species ==  axis_trees[[1]]$tip.label[i] ,"matrix_size"])

 plot_gini[i] <- mean(pop_data[pop_data$species ==  axis_trees[[1]]$tip.label[i] ,"gini"])

 plot_sd[i] <- mean(pop_data[pop_data$species ==  axis_trees[[1]]$tip.label[i] ,"surv_sd"])

 plot_T[i] <- mean(pop_data[pop_data$species ==  axis_trees[[1]]$tip.label[i] ,"gen_time"])

 plot_repo[i] <- mean(pop_data[pop_data$species ==  axis_trees[[1]]$tip.label[i] ,"mean_repo_rate_stabl

}

names(plot_gini) <- axis_trees[[1]]$tip.label
names(plot_sd) <- axis_trees[[1]]$tip.label
names(plot_T) <- axis_trees[[1]]$tip.label
names(plot_repo) <- axis_trees[[1]]$tip.label

#pdf("phylobar_gini.pdf")
#plotTree.wBars(tree = axis_trees[[1]], x = plot_gini, scale= 100, type="fan", cex = 0.2, tip.labels =
#dev.off()

#pdf("phylobar_sd_surv.pdf")
#plotTree.wBars(tree = axis_trees[[1]], x = plot_sd, scale= 500, type="fan", cex = 0.2, tip.labels = T,
#dev.off()

#pdf("phylobar_genT.pdf")
#plotTree.wBars(tree = axis_trees[[1]], x = plot_T, scale= 0.8, type="fan", cex = 0.2, tip.labels = T,
#dev.off()

#pdf("phylobar_repo.pdf")
#plotTree.wBars(tree = axis_trees[[1]], x = plot_repo, scale= 30, type="fan", cex = 0.2, tip.labels = T
#dev.off()
```

Log10 the non index based metrics

```r
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))
```

Now let mean center all the data

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

First we make a multree object so we can loop each of the models through the trees

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

All the diagnostices look good, lets read that back in.

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

```
##                         Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                      0.19719722   -1.50849226  -0.36690480
## mass_g                           0.59886150    0.38364714   0.52569761
## matrix_size                      0.06146704   -0.01051303   0.03715196
## phylogenetic.variance            2.03182521    0.92568227   1.57138252
## residual.variance                0.29105041    0.17437996   0.24779073
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)               0.76240368      1.9105662
## mass_g                    0.67352314      0.8166247
## matrix_size               0.08761245      0.1367521
## phylogenetic.variance     2.54299781      3.8375286
## residual.variance         0.34484386      0.4648104
## attr(,"class")
## [1] "matrix"  "mulTree"
```

La_models variance terms

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

la_phlyo <- list()
```

```
la_spec <- list()
la_unit <- list()

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

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

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

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

hdr(la_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## 99% 0.6406183 0.6434432 0.6503685 0.6568608 0.6704964 0.6970627 0.7004244
## 95% 0.7332508 0.7464076 0.7483361 0.9558327        NA        NA        NA
## 50% 0.8324944 0.8330940 0.8451230 0.9092764 0.9138267 0.9171321        NA
##          [,8]      [,9]     [,10]
## 99% 0.9619999 0.9624982 0.9644117
## 95%        NA        NA        NA
## 50%        NA        NA        NA
##
## $mode
## [1] 0.8937473
##
## $falpha
##        1%        5%       50%
## 0.2469906 0.9254727 5.6389922
```

```
hdr(la_prop_spec)
```

```
## $hdr
##           [,1]       [,2]       [,3]      [,4]      [,5]      [,6]
## 99% 0.03478366 0.26431852 0.26583073 0.2845250 0.2911740 0.3008965
## 95% 0.03738579 0.03901049 0.04092025 0.2137509 0.2151911 0.2294792
## 50% 0.07665461 0.08087969 0.08282138 0.1419366 0.1501532 0.1514604
##          [,7]      [,8]      [,9]     [,10]     [,11]     [,12]     [,13]
## 99% 0.3050153 0.3089759 0.3125572 0.3140130 0.3234217 0.3241761 0.3260958
## 95% 0.2321272 0.2371634 0.2399997 0.2467182 0.2484158 0.2530036 0.2707032
## 50%        NA        NA        NA        NA        NA        NA        NA
##         [,14]     [,15]     [,16]     [,17]     [,18]
## 99% 0.3293287 0.3405726 0.3429459 0.3702027 0.3705973
## 95% 0.2714517        NA        NA        NA        NA
## 50%        NA        NA        NA        NA        NA
##
## $mode
## [1] 0.09529242
##
## $falpha
##       1%       5%      50%
## 0.330040 1.073751 5.984917
```

```
hdr(la_prop_residuals)
```

```
## $hdr
##            [,1]        [,2]        [,3]        [,4]        [,5]
## 99% 0.004173126 0.004208015 0.004438841 0.004482253 0.004713095
## 95% 0.005449061 0.017740518 0.017926165 0.018072369 0.018290177
## 50% 0.007896610 0.007922972 0.008196460 0.008254788 0.008456136
##            [,6]        [,7]        [,8]        [,9]       [,10]      [,11]
## 99% 0.020977529 0.021066973 0.021428959 0.022145687 0.02245146 0.02297645
## 95% 0.018471928 0.018837176 0.019134845 0.019515459 0.01961413         NA
## 50% 0.008609794 0.008666226 0.008783359 0.008893645 0.01190254 0.01204487
##          [,12]
## 99% 0.02313278
## 95%         NA
## 50% 0.01285248
##
## $mode
## [1] 0.01031497
##
## $falpha
##        1%        5%       50%
##  4.607315 15.866655 95.350920
```

Now we need to calulate the residuals

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

All the diagnostices look good, lets read that back in.

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

```
##                         Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                      -0.3821457    -1.9569788   -0.8931508
## mass_g                           -0.3269210    -0.5526147   -0.4028722
## matrix_size                      -0.2602216    -0.3864657   -0.3039546
## phylogenetic.variance             1.3396644     0.4631444    0.9726397
## residual.variance                 0.3852071     0.2280802    0.3277389
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)               0.08567504     1.02958316
## mass_g                   -0.24661572    -0.09573343
## matrix_size              -0.21771048    -0.13614979
## phylogenetic.variance     1.80301263     2.99722644
## residual.variance         0.45870537     0.61795971
## attr(,"class")
## [1] "matrix"  "mulTree"
```

repo_models variance terms

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

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

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

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

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

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

hdr(repo_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## 99% 0.3304617 0.3459109 0.4338728 0.4455735 0.4634407 0.9298817
## 95% 0.5272801 0.9034334        NA        NA        NA        NA
## 50% 0.6979054 0.7100960 0.7214446 0.8228251 0.8236185 0.8337907
##
## $mode
## [1] 0.749791
##
## $falpha
##        1%        5%       50%
## 0.1606762 0.6014679 3.4860705
```

```r
hdr(repo_prop_spec)
```

```
## $hdr
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## 99% 0.03816652 0.4286204 0.4428340 0.4458482 0.4687746 0.4732355 0.4988469
## 95% 0.06328580 0.3678677 0.3764072 0.3789635        NA        NA        NA
## 50% 0.11387528 0.2159938        NA        NA        NA        NA        NA
##          [,8]      [,9]     [,10]
## 99% 0.5063282 0.5460005 0.5493642
## 95%        NA        NA        NA
## 50%        NA        NA        NA
##
## $mode
## [1] 0.1711385
##
## $falpha
##        1%        5%       50%
## 0.1619615 0.9029375 4.3120476
```

```
hdr(repo_prop_residuals)
```

```
## $hdr
##           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## 99% 0.02545950 0.12921964 0.13539426 0.13726320         NA         NA
## 95% 0.03275220 0.10838396 0.11150978 0.11345735 0.11495791 0.11545449
## 50% 0.04929815 0.04989975 0.05220181 0.07652539 0.08142521 0.08371029
##
## $mode
## [1] 0.06141387
##
## $falpha
##         1%         5%        50%
##  0.6010618  2.6031406 15.7228361
```

Now we need to calulate the residuals

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

Lets run it for mean reproductive rate that is just the mean of the positive F values (not stable state nst)

All the diagnostices look good, lets read that back in.

```
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)                    0.20592155    -1.9870386  -0.50275160
## mass_g                        -0.01381715    -0.2666515  -0.09754541
## matrix_size                   -0.18379369    -0.2820744  -0.21757472
## phylogenetic.variance          3.02559729     1.3125387   2.25238180
## residual.variance              0.35172675     0.1496779   0.27277813
##                       upper.CI(75) upper.CI(97.5)
## (Intercept)             0.90936349      2.3362077
## mass_g                  0.07771676      0.2478755
## matrix_size            -0.15030131     -0.0860611
## phylogenetic.variance   4.00089887      6.5208370
## residual.variance       0.42885374      0.5987622
## attr(,"class")
## [1] "matrix"  "mulTree"
```

repo_nst_models variance terms

```
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]
```

```r
  }

repo_nst_phlyo <- unlist(repo_nst_phlyo)
repo_nst_spec <- unlist(repo_nst_spec)
repo_nst_unit <- unlist(repo_nst_unit)

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

hdr(repo_nst_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## 99% 0.6889961 0.6989034 0.7008208 0.7041441 0.7078575 0.7125583 0.7137018
## 95% 0.7327639 0.7359139 0.7441847 0.7477584 0.7511318 0.7535076 0.7581548
## 50% 0.8668325 0.8694338 0.8699654 0.8811941 0.8926059 0.9511400        NA
##          [,8]      [,9]     [,10]
## 99% 0.9818997        NA        NA
## 95% 0.7667052 0.7729678 0.9760557
## 50%        NA        NA        NA
##
## $mode
## [1] 0.929044
##
## $falpha
##       1%       5%      50%
## 0.338572 0.978501 5.687038
```

```r
hdr(repo_nst_prop_spec)
```

```
## $hdr
##            [,1]       [,2]       [,3]      [,4]      [,5]      [,6]
## 99% 0.009671813 0.26809147 0.26811030 0.2816910        NA        NA
## 95% 0.013831219 0.20400063 0.21016320 0.2261061 0.2379580 0.2398576
## 50% 0.040367925 0.09278263 0.09979294 0.1084204 0.1097175 0.1146234
##
## $mode
## [1] 0.08173958
##
## $falpha
##        1%        5%       50%
## 0.3011703 0.9476453 6.3711450
```

```r
hdr(repo_nst_prop_residuals)
```

```
## $hdr
##            [,1]        [,2]        [,3]       [,4]       [,5]       [,6]
## 99% 0.005755433 0.005772303 0.006592023 0.02915650 0.02933019 0.02986175
## 95% 0.007131696 0.023125620 0.023273812 0.02584850 0.02619559 0.02642302
## 50% 0.009637023 0.010018068 0.010177768 0.01224453 0.01266264 0.01356451
##           [,7]       [,8]       [,9]      [,10]      [,11]      [,12]
## 99% 0.03040132 0.03147881 0.03304289 0.03326004 0.03597439 0.03603415
## 95%         NA         NA         NA         NA         NA         NA
## 50% 0.01365827 0.01483560 0.01501015 0.01652746 0.01672665 0.01720037
```

```
##
## $mode
## [1] 0.01408839
##
## $falpha
##        1%        5%       50%
##  3.244211 13.899985 65.475867
```

Now we need to calulate the residuals

```
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

All the diagnostices look good, lets read that back in.

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

```
##                        Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                     -0.1639072   -1.65326400  -0.63006258
## mass_g                          -0.1924591   -0.43282306  -0.27444884
## matrix_size                      0.1277970   -0.03112928   0.07252109
## phylogenetic.variance            1.1075544    0.30275387   0.75209887
## residual.variance                0.3782147    0.16402751   0.29921755
##                        upper.CI(75) upper.CI(97.5)
## (Intercept)               0.2940136     1.21802434
## mass_g                   -0.1085908     0.05379142
## matrix_size               0.1801878     0.28327663
## phylogenetic.variance     1.5471337     2.85791220
## residual.variance         0.4665206     0.66387348
## attr(,"class")
## [1] "matrix"  "mulTree"
```

gen_time_models variance terms

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

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

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

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

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

mxlxsd_prop_phlyo <- mxlxsd_phlyo/(mxlxsd_phlyo + mxlxsd_spec + mxlxsd_unit)
```

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

hdr(mxlxsd_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]
## 99% 0.2643258 0.8910551        NA        NA
## 95% 0.3640159 0.3750883 0.3814946 0.8586202
## 50% 0.5703276 0.5965176 0.6024163 0.7475046
##
## $mode
## [1] 0.6704041
##
## $falpha
##        1%        5%       50%
## 0.1378582 0.6119429 2.4023317
```

```
hdr(mxlxsd_prop_spec)
```

```
## $hdr
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## 99% 0.02268929 0.4579201 0.4628629 0.4792459 0.5072331 0.5102751
## 95% 0.04017092 0.3847158 0.3882358 0.3969752        NA        NA
## 50% 0.10300596 0.2050351 0.2088769 0.2214962        NA        NA
##
## $mode
## [1] 0.1697693
##
## $falpha
##        1%        5%       50%
## 0.1703277 0.7297996 3.5800693
```

```
hdr(mxlxsd_prop_residual)
```

```
## $hdr
##           [,1]      [,2]      [,3]      [,4]
## 99% 0.06980017 0.3120442 0.3422491 0.3432030
## 95% 0.07923663 0.2560268 0.2625259 0.2720485
## 50% 0.12536900 0.1850378 0.1884294 0.1941885
##
## $mode
## [1] 0.164099
##
## $falpha
##        1%        5%       50%
## 0.3983083 1.2794866 6.2070687
```

Now we need to calulate the residuals

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

All the diagnostices look good, lets read that back in.

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

```
##                         Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                      0.60230035  -1.421124979  -0.07966155
## mass_g                           0.58946186   0.383169243   0.51837202
## matrix_size                     -0.03398890  -0.121445851  -0.06351646
## phylogenetic.variance            3.02631924   1.685164647   2.50951070
## residual.variance                0.07253635  -0.001593354   0.04181541
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)              1.268497685     2.61019863
## mass_g                   0.658244045     0.79282154
## matrix_size             -0.002310072     0.05672399
## phylogenetic.variance    3.695077886     5.18098858
## residual.variance        0.105109650     0.17293323
## attr(,"class")
## [1] "matrix"  "mulTree"
```

gen_time_models variance terms

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

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

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

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

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

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

hdr(gen_time_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## 99% 0.8249987 0.8257493 0.8488786 0.8508666 0.8570096 0.8577666 0.8620359
## 95% 0.8914276 0.8928867 0.8947280 0.8966186 0.8977921 0.8983582 0.9014317
## 50% 0.9403439 0.9412924 0.9432609 0.9437884 0.9497649 0.9743616 0.9765054
##          [,8]      [,9]     [,10]     [,11]     [,12]    [,13]     [,14]
## 99% 0.8627734 0.8643483 0.8665584 0.8695865 0.8713434 0.873064 0.8741273
## 95% 0.9025613 0.9043517 0.9069714 0.9083974 0.9877568       NA        NA
## 50% 0.9772085        NA        NA        NA        NA       NA        NA
##         [,15]     [,16]     [,17]     [,18]
## 99% 0.8763387 0.8784535 0.8807454 0.9897361
## 95%        NA        NA        NA        NA
## 50%         NA         NA         NA         NA
##
## $mode
## [1] 0.9690783
##
## $falpha
##         1%          5%         50%
##  0.5662276  2.2701398 14.0133108
```

```
hdr(gen_time_prop_spec)
```

```
## $hdr
##                [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## 99% -0.0018585027 0.08395070 0.08544532 0.08693469 0.08866977 0.09434585
## 95% -0.0009924721 0.06700528 0.06891456 0.06988070 0.07837194 0.07912414
## 50%  0.0065698578 0.02883896         NA         NA         NA         NA
##            [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## 99% 0.09968784 0.1042189 0.1092902 0.1096494 0.1207809 0.1227524
## 95%         NA        NA        NA        NA        NA        NA
## 50%         NA        NA        NA        NA        NA        NA
##
## $mode
## [1] 0.01622625
##
## $falpha
##         1%          5%         50%
##  0.5012662  3.0375745 18.3937212
```

```
hdr(gen_time_prop_residual)
```

```
## $hdr
##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## 99% 0.00996279 0.03728400 0.03746791 0.03901980 0.03905797 0.04003358
## 95% 0.01189825 0.03351992 0.03379308 0.03546167         NA         NA
## 50% 0.01640808 0.02327199 0.02365737 0.02412280 0.02534696 0.02538581
##            [,7]       [,8]       [,9]      [,10]
## 99% 0.04197632 0.04230088 0.04409717 0.04451534
## 95%         NA         NA         NA         NA
## 50%         NA         NA         NA         NA
##
## $mode
## [1] 0.02177171
##
## $falpha
##        1%        5%       50%
##  2.731566 10.579572 54.341345
```

Now we need to calulate the residuals

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

Lets run it for life expectancy conditional on reaching sexual maturity

All the diagnostices look good, lets read that back in.

```
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)                     0.31576098  -1.439073720  -0.26028877
## mass_g                          0.58795591   0.386373547   0.51712000
## matrix_size                     0.05466011  -0.049173230   0.02086836
## phylogenetic.variance           2.24965729   1.100292031   1.78280700
## residual.variance               0.07459563  -0.005804658   0.03870326
##                        upper.CI(75) upper.CI(97.5)
## (Intercept)              0.90913886      2.0895082
## mass_g                   0.65293404      0.7840524
## matrix_size              0.09396632      0.1641846
## phylogenetic.variance    2.78690064      4.0508399
## residual.variance        0.11422760      0.1933473
## attr(,"class")
## [1] "matrix"  "mulTree"
```

M_rep_lif_exp_models variance terms

```
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

hdr(M_rep_lif_exp_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## 99% 0.7202008 0.7212668 0.7679105 0.7713399 0.7747669 0.7794336 0.7811104
## 95% 0.8127583 0.8193569 0.8206474 0.8237017 0.8260872 0.9663523        NA
## 50% 0.8899500 0.8911998 0.8923896 0.8953435 0.9016716 0.9443826        NA
##          [,8]     [,9]     [,10]     [,11]     [,12]
## 99% 0.7877129 0.789438 0.7924834 0.7985336 0.9735319
## 95%        NA       NA        NA        NA        NA
## 50%        NA       NA        NA        NA        NA
##
## $mode
## [1] 0.9303792
```

```
##
## $falpha
##        1%        5%       50%
## 0.4426789 1.4154873 8.4976864
```

```
hdr(M_rep_lif_exp_prop_spec)
```

```
## $hdr
##               [,1]        [,2]        [,3]       [,4]       [,5]       [,6]
## 99% -1.366250e-03 0.109124638 0.111073405 0.11837650 0.11972427 0.12135369
## 95% -9.676678e-04 0.089412696 0.092463889 0.09369499 0.09597386 0.09951742
## 50% -4.301442e-05 0.002933646 0.005060425 0.01037997 0.01285116 0.02535762
##           [,7]       [,8]       [,9]      [,10]    [,11]     [,12]
## 99% 0.12169578 0.12189725 0.12431492 0.12591678 0.127305 0.1302091
## 95%         NA         NA         NA         NA       NA        NA
## 50% 0.02768163 0.03657883 0.04016347 0.04088809       NA        NA
##        [,13]     [,14]   [,15]     [,16]     [,17]     [,18]
## 99% 0.143484 0.1450535 0.14733 0.1488654 0.1526898 0.1529817
## 95%       NA        NA      NA        NA        NA        NA
## 50%       NA        NA      NA        NA        NA        NA
##
## $mode
## [1] 0.01846302
##
## $falpha
##         1%         5%        50%
##  0.6167438  2.4424673 13.7450488
```

```
hdr(M_rep_lif_exp_prop_residuals)
```

```
## $hdr
##           [,1]       [,2]      [,3]       [,4]       [,5]       [,6]
## 99% 0.02452006 0.11055279 0.1115022 0.11287259 0.11596625 0.11605916
## 95% 0.02858911 0.02922202 0.0300296 0.09365835 0.09550034 0.09706253
## 50% 0.04125548 0.04202726 0.0431870 0.06162428 0.06348429 0.06527947
##          [,7]      [,8]
## 99% 0.1277325 0.1284555
## 95%        NA        NA
## 50%        NA        NA
##
## $mode
## [1] 0.05578845
##
## $falpha
##         1%         5%        50%
##  0.7355868  3.2571590 19.7733814
```

Now we need to calulate the residuals

```
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")
 )
```

Lets run it for the gini index

All the diagnostices look good, lets read that back in.

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

summary(gini_models)
```

```
##                         Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                     -0.42792607    -2.0464245   -0.9689465
## mass_g                          -0.19601910    -0.4244734   -0.2745834
## matrix_size                     -0.09997738    -0.2324761   -0.1452911
## phylogenetic.variance            1.79675072     0.9171706    1.4300623
## residual.variance                0.25784403     0.1276057    0.2086542
##                         upper.CI(75) upper.CI(97.5)
## (Intercept)               0.11276264     1.19790393
## mass_g                   -0.11832226     0.03280158
## matrix_size              -0.05474631     0.03184344
## phylogenetic.variance     2.19375023     3.26168320
## residual.variance         0.31350049     0.44078588
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Since the read in seems to miss the species variance term we need to read that in seperate.

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

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

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

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

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

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

hdr(gini_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## 99% 0.6014925 0.6029427 0.6221889 0.9106813 0.9162405 0.9182208        NA
## 95% 0.6456083 0.6483562 0.6543877 0.6557645 0.6656940 0.8893681 0.8922537
## 50% 0.7476243 0.7500067 0.7643972 0.7686894 0.7699635 0.8292471 0.8322060
##          [,8]      [,9]     [,10]
## 99%        NA        NA        NA
## 95% 0.8942774        NA        NA
## 50% 0.8383279 0.8402843 0.8453827
##
## $mode
## [1] 0.7922497
##
## $falpha
##        1%        5%       50%
## 0.1915535 1.1621993 5.1429723
```

```r
hdr(gini_prop_spec)
```

```
## $hdr
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## 99% 0.02806573 0.2233123 0.2255779 0.2487039 0.2528714 0.2532666
## 95% 0.04264019 0.2066662 0.2123950 0.2183107        NA        NA
## 50% 0.07666218 0.1326623 0.1401001 0.1410701        NA        NA
##
## $mode
## [1] 0.1046461
##
## $falpha
##        1%        5%       50%
## 0.3773721 1.5087452 6.9429816
```

```r
hdr(gini_prop_residuals)
```

```
## $hdr
##           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## 99% 0.03954057 0.04124459 0.04760719 0.15339161 0.15505207 0.15617927
## 95% 0.05207554 0.05331253 0.05379206 0.05529746 0.05595720 0.13625517
## 50% 0.06714741 0.06832185 0.07228855 0.08253943 0.08342508 0.09522928
##           [,7]      [,8]      [,9]     [,10]     [,11]     [,12]     [,13]
## 99% 0.15802543 0.1611068 0.1754038 0.1777230        NA        NA        NA
## 95% 0.13765449 0.1394366 0.1406215 0.1426684 0.1436330 0.1449316        NA
## 50% 0.09609703 0.1004976 0.1044944 0.1051261 0.1067461 0.1077285 0.1089811
##         [,14]
## 99%        NA
## 95%        NA
## 50% 0.1092551
##
## $mode
## [1] 0.09190319
##
## $falpha
##        1%        5%       50%
##  0.817346  2.895939 13.641542
```

Now we need to calulate the residuals

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

Lets run it for the standard deviation of mortality rates

All the diagnostices look good, lets read that back in.

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

summary(surv_sd_models)
```

```
##                          Estimates(mode hdr) lower.CI(2.5) lower.CI(25)
## (Intercept)                        0.2919404   -1.17657834   -0.1883169
## mass_g                             0.2633088    0.03469285    0.1864593
## matrix_size                       -0.3503768   -0.50536667   -0.4040565
## phylogenetic.variance              1.1714224    0.38878793    0.8386725
## residual.variance                  0.2431978    0.09919949    0.1882855
##                          upper.CI(75) upper.CI(97.5)
## (Intercept)                 0.7439635      1.7092974
## mass_g                      0.3445202      0.5007707
## matrix_size                -0.2995984     -0.1995481
## phylogenetic.variance       1.6285572      2.8397630
## residual.variance           0.3083809      0.4565594
## attr(,"class")
## [1] "matrix"  "mulTree"
```

Since the read in seems to miss the species variance term we need to read that in seperate.

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

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

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

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

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

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

hdr(surv_sd_prop_phlyo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]
## 99% 0.3335133 0.8827036        NA        NA
## 95% 0.4348327 0.8563945        NA        NA
## 50% 0.5956603 0.6192499 0.6318059 0.7555922
##
## $mode
## [1] 0.7194289
##
## $falpha
##        1%        5%       50%
## 0.1542353 0.6181539 2.9152680
```

```
hdr(surv_sd_prop_spec)
```

```
## $hdr
##            [,1]       [,2]      [,3]      [,4]      [,5]      [,6]
## 99% 0.02104198 0.29423689 0.2950760 0.3136595 0.3196814 0.3358598
## 95% 0.03162066 0.25883345 0.2590323 0.2638300 0.2700172 0.2750848
## 50% 0.06736595 0.09728116 0.1002710 0.1462048 0.1493806 0.1515445
##
## $mode
## [1] 0.1070557
##
## $falpha
##        1%        5%       50%
## 0.3454369 0.9256044 5.2555032
```

```
hdr(surv_sd_prop_residuals)
```

```
## $hdr
##            [,1]       [,2]       [,3]      [,4]      [,5]      [,6]
## 99% 0.07410689 0.07412768 0.08114927 0.3632304 0.3804389 0.3810536
## 95% 0.09362128 0.09870303 0.10104871 0.3168509 0.3355439 0.3385396
## 50% 0.14705875 0.14858537 0.15431778 0.1721416 0.1752069 0.1937691
##           [,7]      [,8]      [,9]     [,10]
## 99% 0.4394676 0.4413945        NA        NA
## 95%        NA        NA        NA        NA
## 50% 0.1966278 0.2164779 0.2182223 0.2363943
##
## $mode
## [1] 0.2247078
##
## $falpha
##        1%        5%       50%
## 0.2207477 1.1501733 5.5529351
```

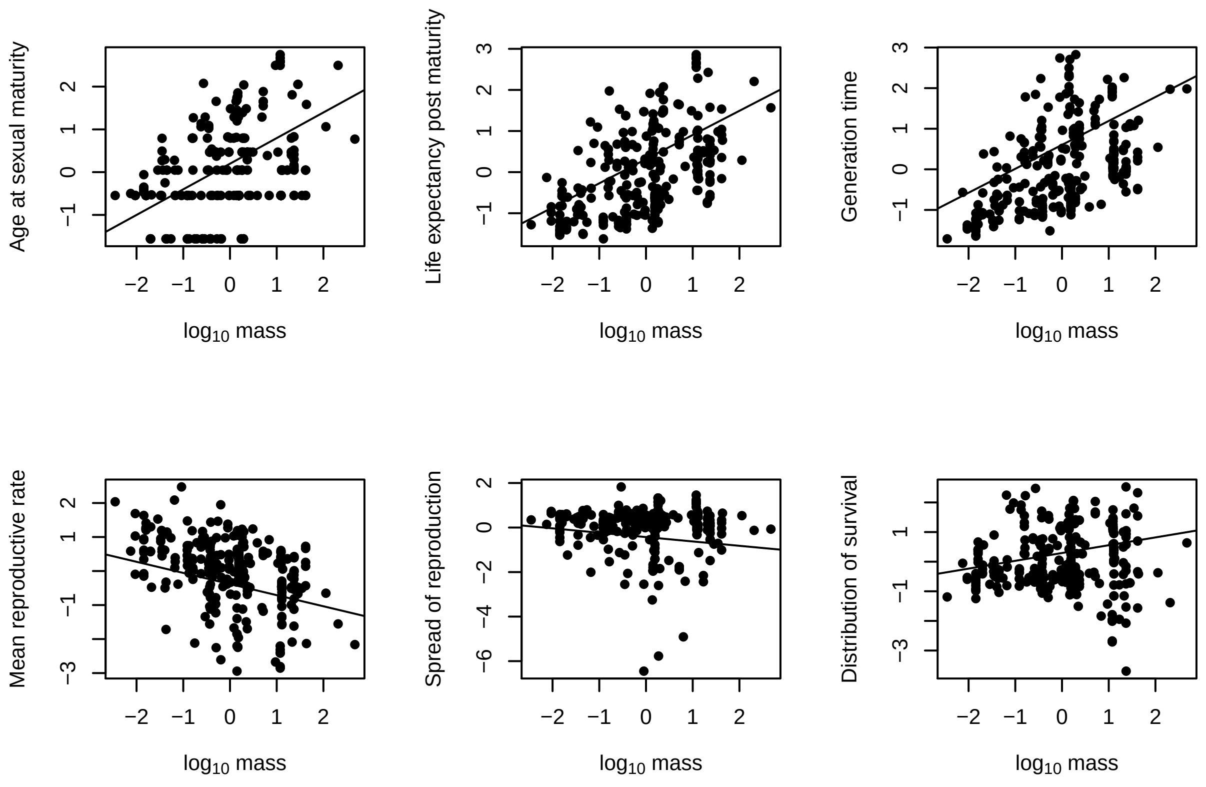
Now we need to calulate the residuals

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

Lets plot out these allometries.

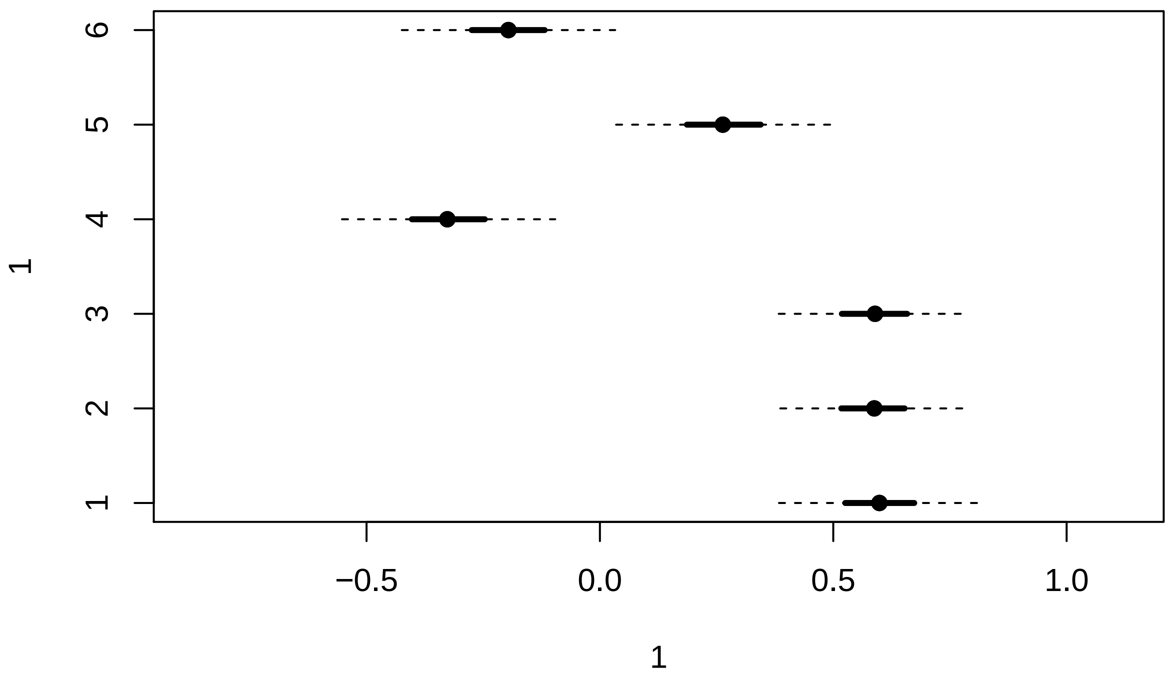
```
par(mfrow=c(2,3))

##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 plot out these model cooficents into a table

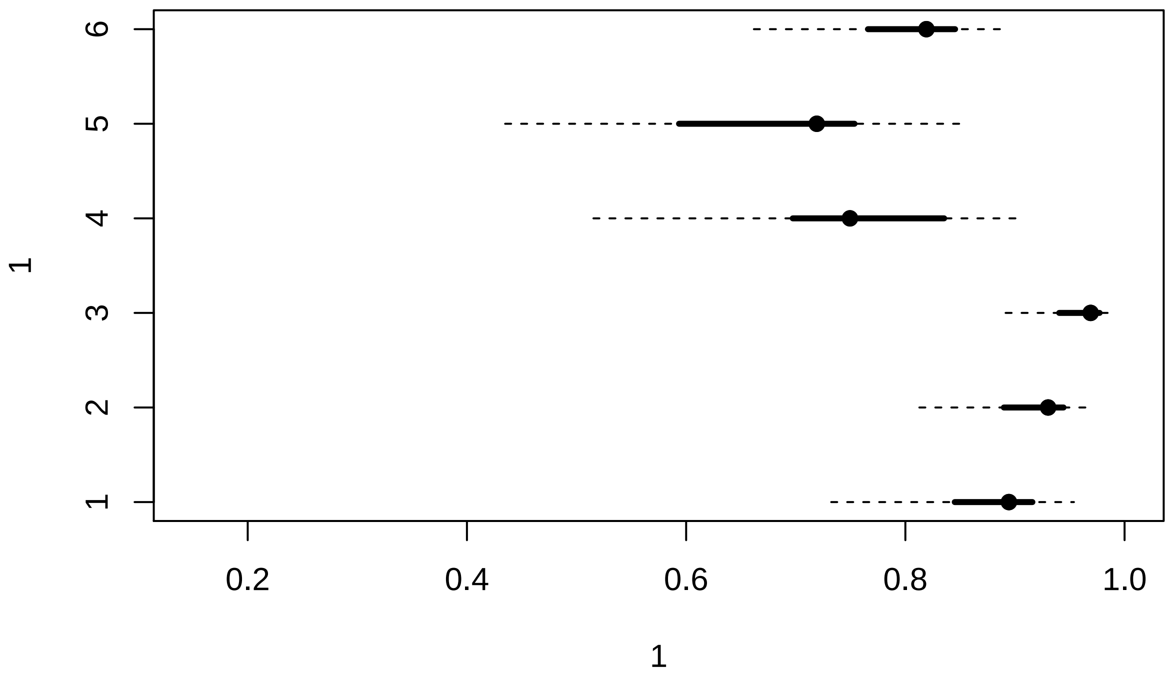
```r
##Allometric scaling
#pdf("scaling_bar_plots.pdf")
scaling_list <- list( La_B = La_models$mass_g,
                      Sur_B = M_rep_lif_exp_models$mass_g,
                      T_B = gen_time_models$mass_g,
                      Repo_B = repo_models$mass_g,
                      life_shape_B =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



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

And the variance terms

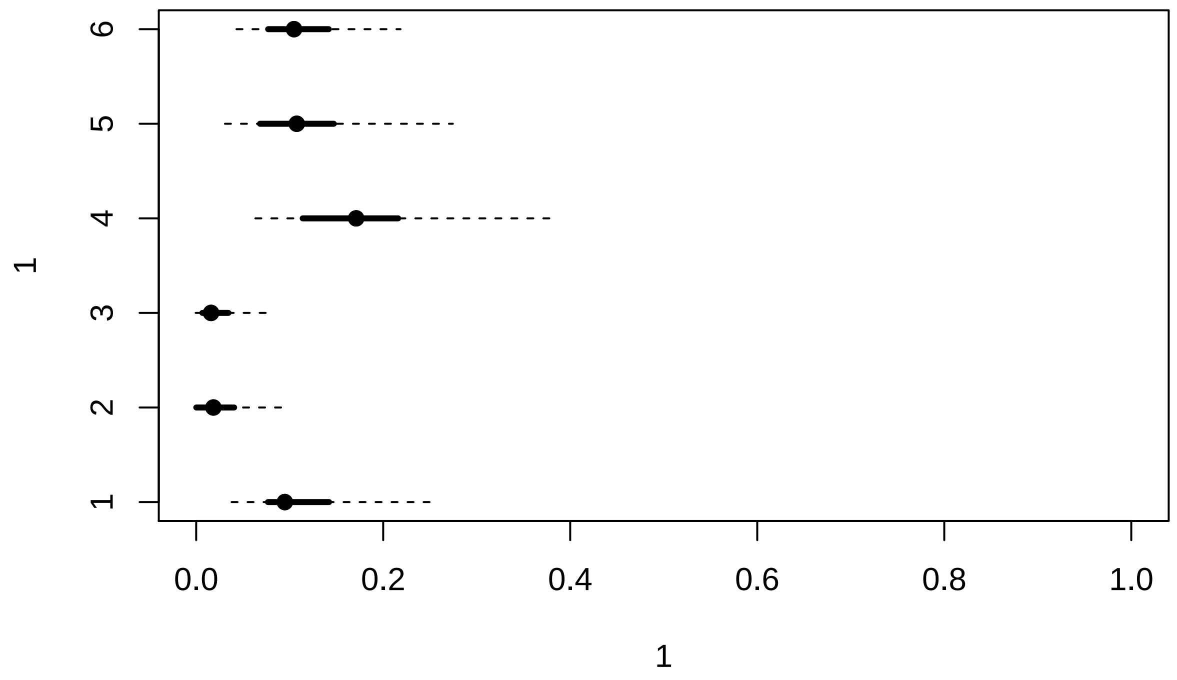
```r
#pdf("phy_var_plots.pdf")
phy_var_list <- list( La_B = la_prop_phlyo,
                      Sur_B = M_rep_lif_exp_prop_phlyo,
                      T_B = gen_time_prop_phlyo,
                      Repo_B = repo_prop_phlyo,
                      life_shape_B = surv_sd_prop_phlyo,
               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



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

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

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



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

Now lets creata a new dataset of these residuals

```
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
                        )


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
                        )


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
                        )
```

```
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
                       )


predicted_data_justT <- data.frame(
                        SD_mort = surv_sd_resids,
                       M_repo = repo_resids,
                       gen_r = gen_time_resids,
                       gini = gini_resids
                       )
```

And run a PCA

```
pca_res <- prcomp(predicted_data)

pca_nst <- prcomp(predicted_data_M_repo_nst)

pca_mxlxsd <- prcomp(predicted_data_mxlxsd)

pca_noT <- prcomp(predicted_data_noT)
```

```
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           2.614513    2.804697      0.190183
## 2           1.383173    1.481387      0.098213
## --------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

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

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 180 iterations, using the mean estimate
##
## --------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## --------------------------------------------------
## 1           2.561541    2.759154      0.197612
## 2           1.312938    1.410411      0.097472
## --------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

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

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 180 iterations, using the mean estimate
##
## --------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## --------------------------------------------------
## 1           2.710939    2.911046      0.200107
## 2           1.332993    1.437757      0.104764
## --------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

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

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 150 iterations, using the mean estimate
##
## --------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## --------------------------------------------------
## 1           1.768713    1.934080      0.165367
## 2           1.396090    1.466976      0.070885
## --------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (2 components retained)
```

PLOTS

Lets make some beter plots. First we can flip the PCA axis so that it reads with increases towards slow lifestyles on the right and increasing towards tye 1 survorship curves upwards

```r
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"]
```

Now lets make a nicer looking PCA graph

```r
result <- results

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

```r
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"])

   }


pca_data <- cbind(loads_taxa,class_match, species_match)



##rename some of the mode of life levels
PCA1 <- pca_data[,1]
PCA2 <- pca_data[,2]

mobility_PCA <- pop_multree$data$mobility
mobility_PCA <- as.vector(mobility_PCA)
mobility_PCA[mobility_PCA == "fw_benthic"] <- "benthic"
mobility_PCA[mobility_PCA == "m_benthic"] <- "benthic"
mobility_PCA[mobility_PCA == "fw_pelagic"] <- "pelagic"
mobility_PCA[mobility_PCA == "m_pelagic"] <- "pelagic"
mobility_PCA[mobility_PCA == "fw_river"] <- "benthic"

mobility_PCA <- factor(mobility_PCA, levels = c("sessile", "arboreal", "benthic", "volant",
                                          "semiaquatic", "terrestrial" ,"pelagic",  "semifossorial"))

pca_data$mobility_PCA <- mobility_PCA


PCA_moblist <- list(semifossorial = pca_data[pca_data$mobility_PCA == "semifossorial",1],
                    pelagic = pca_data[pca_data$mobility_PCA == "pelagic",1],
                    terrestrial = pca_data[pca_data$mobility_PCA == "terrestrial",1],
                    semiaquatic = pca_data[pca_data$mobility_PCA == "semiaquatic",1],
                    volant = pca_data[pca_data$mobility_PCA == "volant",1],
                    benthic = pca_data[pca_data$mobility_PCA == "benthic",1],
                    arboreal = pca_data[pca_data$mobility_PCA == "arboreal",1],
                    sessile = pca_data[pca_data$mobility_PCA == "sessile",1]
                    )






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)
```

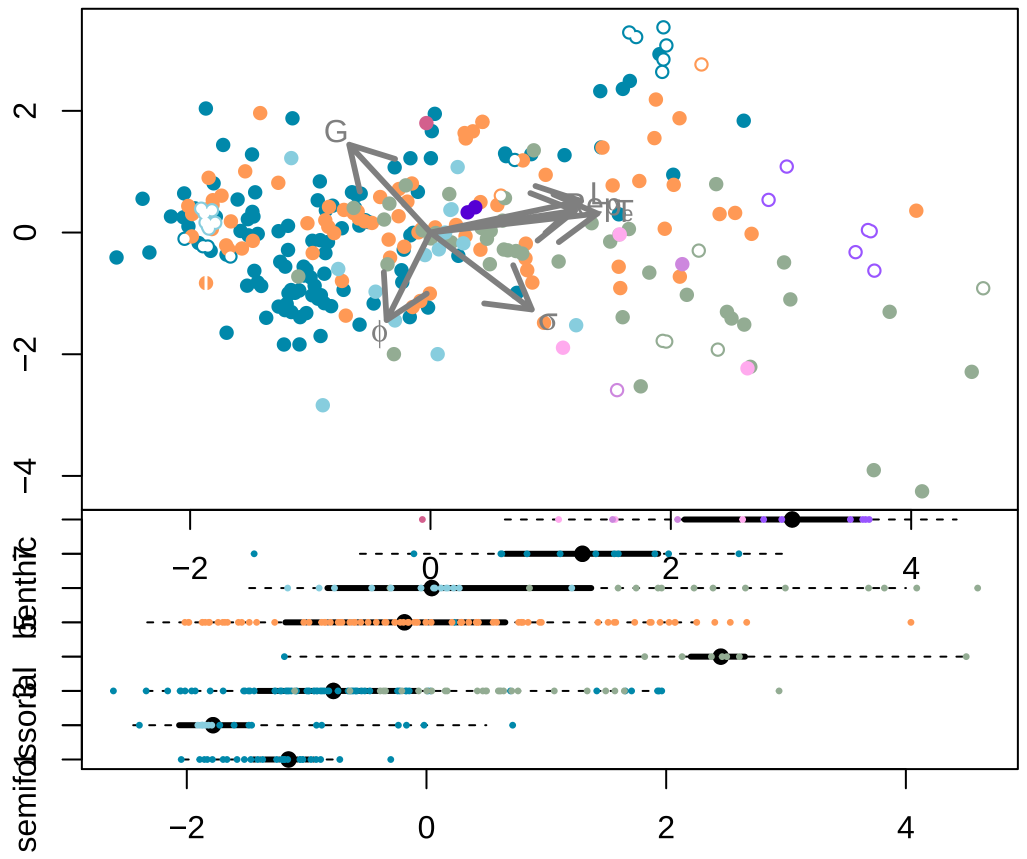
```r
shark_col <- rgb(85,0,212, max= 255)
```

Now let do the actual plot

```r
#pdf("Figure2_PCA.pdf")
split <- rbind(c(0.15,0.9, 0.4,0.98), c(0.15,0.9, 0.1, 0.4))
split.screen(split)
```

```
## [1] 1 2
```

```r
screen(1)
par(mar = c(0, 0, 0, 0))
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], p

points(pca_data[pca_data$class_match == "Aves",1], pca_data[pca_data$class_match == "Aves",2], pch=16, c

points(pca_data[pca_data$class_match == "Reptilia",1], pca_data[pca_data$class_match == "Reptilia",2], p

points(pca_data[pca_data$class_match == "Actinopterygii",1], pca_data[pca_data$class_match == "Actinopte

points(pca_data[pca_data$class_match == "Gastropoda",1], pca_data[pca_data$class_match == "Gastropoda",2

points(pca_data[pca_data$class_match == "Demospongiae",1], pca_data[pca_data$class_match == "Demospongia

points(pca_data[pca_data$class_match == "Anthozoa",1], pca_data[pca_data$class_match == "Anthozoa",2], p

points(pca_data[pca_data$class_match == "Bivalvia",1], pca_data[pca_data$class_match == "Bivalvia",2], p

points(pca_data[pca_data$class_match == "Elasmobranchii",1], pca_data[pca_data$class_match == "Elasmobra

###And lets add Humans
points(pca_data[pca_data$species_match == "Homo_sapiens",1], pca_data[pca_data$species_match == "Homo_sa

##and other points
points(pca_data[pca_data$species_match == "Elephas_maximus",1], pca_data[pca_data$species_match == "Elep

points(pca_data[pca_data$species_match == "Fulmarus_glacialis",1], pca_data[pca_data$species_match == "F

points(pca_data[pca_data$species_match == "Tympanuchus_cupido",1], pca_data[pca_data$species_match == "T

points(pca_data[pca_data$species_match == "Gyps_coprotheres",1], pca_data[pca_data$species_match == "Gyp

points(pca_data[pca_data$species_match == "Crocodylus_johnsoni",1], pca_data[pca_data$species_match ==

points(pca_data[pca_data$species_match == "Urocitellus_armatus",1], pca_data[pca_data$species_match ==

points(pca_data[pca_data$species_match == "Paramuricea_clavata",1], pca_data[pca_data$species_match ==

points(pca_data[pca_data$species_match == "Oncorhynchus_tshawytscha",1], pca_data[pca_data$species_match

points(pca_data[pca_data$species_match == "Mya_arenaria",1], pca_data[pca_data$species_match == "Mya_are
points(pca_data[pca_data$species_match == "Clemmys_guttata",1], pca_data[pca_data$species_match == "Clem


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$c

#arrows(x0=0,y0=0,x1=loadings[,1]*scalingArrows,y1=loadings[,2]*scalingArrows,col="black", lwd=arrowThi
#arrows(x0=0,y0=0,x1=loadings[,1]*scalingArrows,y1=loadings[,2]*scalingArrows,col=as.character(loadings$

text(loadings[1,"PC1"]*scalingLetters-.0,loadings[1,"PC2"]*scalingLetters,loadings$LHTexpr[[1]],col = lo
text(loadings[2,"PC1"]*scalingLetters-.0,loadings[2,"PC2"]*scalingLetters,loadings$LHTexpr[[2]],col = lo
text(loadings[3,"PC1"]*scalingLetters+.0,loadings[3,"PC2"]*scalingLetters,loadings$LHTexpr[[3]],col = lo
text(loadings[4,"PC1"]*scalingLetters-.0,loadings[4,"PC2"]*scalingLetters,loadings$LHTexpr[[4]],col = lo
text(loadings[5,"PC1"]*scalingLetters-.0,loadings[5,"PC2"]*scalingLetters,loadings$LHTexpr[[5]],col = lo
text(loadings[6,"PC1"]*scalingLetters+.0,loadings[6,"PC2"]*scalingLetters,loadings$LHTexpr[[6]],col = lo
#text(loadings[7,"PC1"]*scalingLetters+.0,loadings[7,"PC2"]*scalingLetters,loadings$LHTexpr[[7]],col =


screen(2)
par(mar = c(0, 0, 0, 0))

MultiDisPlot(PCA_moblist)
tick_lables <- names(PCA_moblist)
axis(2, 1:length(tick_lables), labels = tick_lables)

mob_match <- as.vector(mobility_PCA)

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

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

points(pca_data[pca_data$class_match == "Aves","mob_match"] ~ pca_data[pca_data$class_match == "Aves",1]

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

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
```



```
#close.screen(all.screens = TRUE)
#dev.off()
```

Now lets do some more plots

```
habitat_PCA <- pop_multree$data$habitat
habitat_PCA[habitat_PCA == "marine"] <- "marine"
habitat_PCA[habitat_PCA == "freshwater"] <- "marine"
habitat_PCA[habitat_PCA == "marine-freshwater"] <- "marine"
habitat_PCA <- factor(habitat_PCA, levels = c("terrestrial", "marine"))

pca_data$habitat_PCA <- habitat_PCA

pca_data$animal <- pca_data$species_match

##aqutic species

aquatic_res <- pca_data[pca_data$mobility_PCA == "sessile" | pca_data$mobility_PCA == "benthic"
                        | pca_data$mobility_PCA == "pelagic",]

aquatic_res$mobility_PCA <- factor(aquatic_res$mobility_PCA, levels = c("pelagic", "sessile", "benthic"


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)))
```

```
##needs to run longer
aquatic_pc1 <- MCMCglmm(PC1 ~ mobility_PCA,
                     data = aquatic_res,
                     random=~animal + species_match,
                     pedigree = axis_trees[[1]],
                     prior = prior,
                     nitt = c(1100000), burnin = 100000, thin = 500, verbose = F)
summary(aquatic_pc1)
```

```
##
##  Iterations = 100001:1099501
##  Thinning interval  = 500
##  Sample size  = 2000
##
##  DIC: 142.0543
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal     2.604 0.001246    6.407     2000
##
##                ~species_match
##
##               post.mean  l-95% CI u-95% CI eff.samp
## species_match    0.7562 2.027e-05     1.66     2000
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.4102   0.2044    0.645     2000
##
##  Location effects: PC1 ~ mobility_PCA
##
##                      post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)           -1.03391 -3.78979  1.48484     2000 0.363
## mobility_PCAsessile   2.66368  0.12352  5.50703     1870 0.062 .
## mobility_PCAbenthic   1.66644  0.09585  3.01360     2000 0.030 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aquatic_pc1_vcv_phylo <- aquatic_pc1$VCV[,1]
aquatic_pc1_vcv_spec <- aquatic_pc1$VCV[,2]
aquatic_pc1_vcv_units <- aquatic_pc1$VCV[,3]

aquatic_pc1_phylo <- aquatic_pc1_vcv_phylo/c(aquatic_pc1_vcv_phylo + aquatic_pc1_vcv_spec + aquatic_pc1_
aquatic_pc1_spec <- aquatic_pc1_vcv_spec/c(aquatic_pc1_vcv_phylo + aquatic_pc1_vcv_spec + aquatic_pc1_vc
aquatic_pc1_units <- aquatic_pc1_vcv_units/c(aquatic_pc1_vcv_phylo + aquatic_pc1_vcv_spec + aquatic_pc1_

hdr(aquatic_pc1_phylo)
```

```
## $hdr
##           [,1]      [,2]
## 99% 0.05957729 1.0423730
## 95% 0.26955221 0.9798612
## 50% 0.57278798 0.8243079
##
## $mode
## [1] 0.7089488
##
## $falpha
##        1%        5%       50%
## 0.1138648 0.3942820 1.6427459
```

```
hdr(aquatic_pc1_spec)
```

```
## $hdr
##            [,1]      [,2]
## 99% -0.06273573 0.7629974
## 95% -0.03255971 0.5562057
## 50%  0.04895875 0.2486875
##
## $mode
## [1] 0.1622204
##
## $falpha
##        1%        5%       50%
## 0.1118308 0.4249094 2.0653048
```

```
hdr(aquatic_pc1_units)
```

```
## $hdr
##            [,1]      [,2]
## 99% 0.001403337 0.3368995
## 95% 0.019487762 0.2506065
## 50% 0.071472859 0.1425920
##
## $mode
## [1] 0.1065099
##
## $falpha
##        1%        5%       50%
## 0.2161613 0.9167200 5.6960532
```

```
##terrestiral species

ter_res <- pca_data[pca_data$mobility_PCA == "terrestrial" | pca_data$mobility_PCA == "arboreal"
                       | pca_data$mobility_PCA == "volant" | pca_data$mobility_PCA == "semiaquatic" | p

ter_res$mobility_PCA <- factor(ter_res$mobility_PCA, levels = c("terrestrial", "arboreal", "volant", "s


ter_pc1 <- MCMCglmm(PC1 ~ mobility_PCA,
                     data = ter_res,
                     random=~animal + species_match,
                     pedigree = axis_trees[[1]],
                     prior = prior,
                     nitt = c(1100000), burnin = 100000, thin = 500, verbose = F)
summary(ter_pc1)
```

```
##
```

```
##  Iterations = 100001:1099501
##  Thinning interval  = 500
##  Sample size  = 2000
##
##  DIC: 319.4125
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal     10.57    5.382    15.75     1671
##
##                ~species_match
##
##               post.mean l-95% CI u-95% CI eff.samp
## species_match    0.1377 1.37e-06   0.3342     1721
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.1753   0.1353   0.2183     2000
##
##  Location effects: PC1 ~ mobility_PCA
##
##                           post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)                 0.30402 -5.04768  5.47766     2000 0.885
## mobility_PCAarboreal        0.22312 -0.69394  1.05826     2000 0.610
## mobility_PCAvolant          0.08002 -1.72926  2.47692     2000 0.962
## mobility_PCAsemiaquatic     0.02463 -1.62026  1.77493     2000 0.966
## mobility_PCAsemifossorial  -0.40542 -1.78345  0.87149     2000 0.560
```

```
ter_pc1_vcv_phylo <- ter_pc1$VCV[,1]
ter_pc1_vcv_spec <- ter_pc1$VCV[,2]
ter_pc1_vcv_units <- ter_pc1$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)
```

```
## $hdr
##          [,1]      [,2]     [,3]     [,4]     [,5]      [,6]
## 99% 0.9062223 0.9073376 0.911247 0.916007 0.919754 0.9937795
## 95% 0.9371836 0.9914065       NA       NA       NA        NA
## 50% 0.9694404 0.9848373       NA       NA       NA        NA
##
## $mode
## [1] 0.9791207
##
## $falpha
##         1%         5%        50%
##  0.5745123  3.4329950 25.4838910
```

```r
hdr(ter_pc1_spec)
```

```
## $hdr
##             [,1]        [,2]       [,3]       [,4]       [,5]       [,6]
## 99% -0.001719918 0.054466799 0.05514227 0.05893179 0.07841291 0.07902547
## 95% -0.001319937 0.038627601         NA         NA         NA         NA
## 50%  0.000112596 0.009962863 0.01055440 0.01217742         NA         NA
##
## $mode
## [1] 0.005006542
##
## $falpha
##        1%        5%       50%
##  1.055593  3.667939 37.374994
```

```r
hdr(ter_pc1_units)
```

```
## $hdr
##            [,1]       [,2]       [,3]       [,4]
## 99% 0.007000206 0.03085296 0.03306523 0.03351056
## 95% 0.008719491 0.02648578         NA         NA
## 50% 0.012772645 0.01838083         NA         NA
##
## $mode
## [1] 0.01482833
##
## $falpha
##        1%        5%       50%
##  2.104418 11.940251 71.139024
```

Metabolic rate analysis

```r
met_PCA <- pop_multree$data$met_rate_Wg
pca_data$met_PCA <- met_PCA

pca_data$animal <- pca_data$species_match
pca_data$met_rate <- as.numeric(as.vector(pca_data$met_PCA))

pca_data_met <- na.omit(pca_data[,c("species_match","animal","PC1","class_match","met_rate")])




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

summary(met_mod)
```

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

```
##
##  DIC: 220.4934
##
##  G-structure:  ~animal
##
##        post.mean l-95% CI u-95% CI eff.samp
## animal     7.331    3.354     11.5     1869
##
##                ~species_match
##
##               post.mean  l-95% CI u-95% CI eff.samp
## species_match   0.05729 2.672e-08   0.2166     2000
##
##  R-structure:  ~units
##
##       post.mean l-95% CI u-95% CI eff.samp
## units    0.2063   0.1587   0.2729     2000
##
##  Location effects: PC1 ~ log10(met_rate)
##
##                 post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)       -2.5216  -7.4268   2.2588     2000 0.315
## log10(met_rate)   -1.0381  -1.8828  -0.1241     2000 0.024 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
met_mod_phylo <- met_mod$VCV[,1]
met_mod_spec <- met_mod$VCV[,2]
met_mod_units <- met_mod$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)
```

```
## $hdr
##          [,1]      [,2]
## 99% 0.8816459 0.9947336
## 95% 0.9253472 0.9903859
## 50% 0.9616506 0.9773326
##
## $mode
## [1] 0.9696589
##
## $falpha
##         1%         5%        50%
##  0.3646195  2.0960558 25.4189342
```

```
hdr(met_mod_spec_H)
```

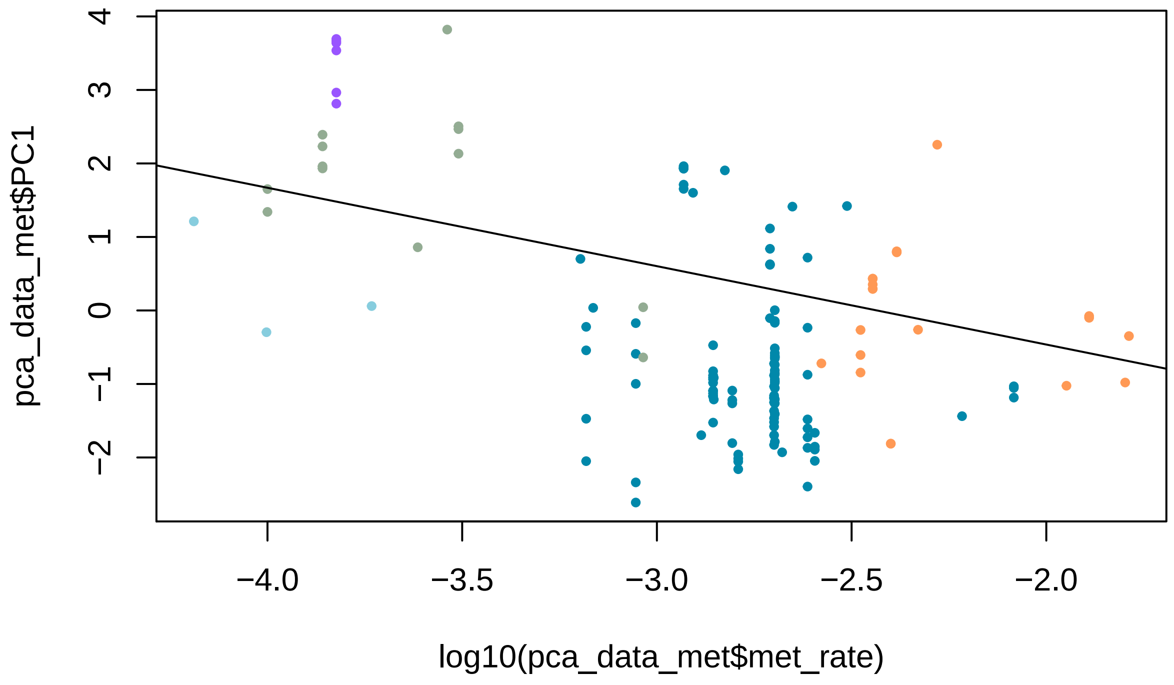
```
## $hdr
##              [,1]        [,2]        [,3]        [,4]       [,5]
## 99%            NA 0.033308755 0.034383345 0.039792336 0.04144077
## 95%            NA 0.026763158 0.027103884 0.028248672 0.02877469
## 50% -0.0003043433 0.003089078 0.003591603 0.004062234          NA
##           [,6]       [,7]       [,8]       [,9]      [,10]      [,11]
## 99% 0.04407108 0.04408625 0.04490320 0.04636180 0.04710839 0.04942646
## 95% 0.02902732 0.02994748 0.02996134 0.03037664 0.03126559 0.03156201
## 50%         NA         NA         NA         NA         NA         NA
##          [,12]      [,13]      [,14]      [,15]      [,16]      [,17]
## 99% 0.05011109 0.05025058 0.05115680 0.05265059 0.05281274 0.05313354
## 95% 0.03286971 0.03589365 0.03630552 0.03697813 0.03743899 0.03920592
## 50%         NA         NA         NA         NA         NA         NA
##          [,18]      [,19]      [,20]      [,21]      [,22]      [,23]
## 99% 0.05316924 0.05413300 0.05463529 0.05703074 0.05775910 0.05780200
## 95% 0.03945526 0.04431918 0.04467045 0.04976583 0.04978382 0.05064565
## 50%         NA         NA         NA         NA         NA         NA
##          [,24]      [,25]      [,26]      [,27]      [,28]      [,29]
## 99% 0.05891637 0.05923707 0.05977589 0.06104226 0.06172311 0.06222158
## 95% 0.05092723         NA         NA         NA         NA         NA
## 50%         NA         NA         NA         NA         NA         NA
##          [,30]      [,31]      [,32]      [,33]      [,34]      [,35]
## 99% 0.06324565 0.06388036 0.06396064 0.06496085 0.06496638 0.06555822
## 95%         NA         NA         NA         NA         NA         NA
## 50%         NA         NA         NA         NA         NA         NA
##         [,36]      [,37]      [,38]      [,39]     [,40]      [,41]
## 99% 0.0656272 0.06657396 0.06703433 0.06789169 0.0679228 0.06976292
## 95%        NA         NA         NA         NA        NA         NA
## 50%        NA         NA         NA         NA        NA         NA
##          [,42]      [,43]      [,44]      [,45]      [,46]      [,47]
## 99% 0.07077877 0.07131389 0.07236617 0.07591353 0.07596655 0.07710538
## 95%         NA         NA         NA         NA         NA         NA
## 50%         NA         NA         NA         NA         NA         NA
##          [,48]      [,49]      [,50]      [,51]      [,52]      [,53]
## 99% 0.07735182 0.07746284 0.07766832 0.08349839 0.08355331 0.08643088
## 95%         NA         NA         NA         NA         NA         NA
## 50%         NA         NA         NA         NA         NA         NA
##          [,54]     [,55]     [,56]     [,57]     [,58]
## 99% 0.08649149 0.1127854 0.1128587 0.2100523 0.2100712
## 95%         NA        NA        NA        NA        NA
## 50%         NA        NA        NA        NA        NA
##
## $mode
## [1] 0.0002158479
##
## $falpha
##        1%        5%       50%
##  1.013906  3.153052 59.871183
```

```r
hdr(met_mod_units_H)
```

```
## $hdr
##            [,1]       [,2]
## 99% 0.009775394 0.05859375
## 95% 0.012647033 0.04893915
## 50% 0.020942363 0.03193002
##
## $mode
## [1] 0.02613693
```

```
##
## $falpha
##        1%        5%       50%
##  1.786117  6.419047 37.081706
```

```
plot(log10(pca_data_met$met_rate), pca_data_met$PC1, pch = 16, col = "white", cex = 0.1)

points(log10(pca_data_met[pca_data_met$class_match == "Mammalia","met_rate"]), pca_data_met[pca_data_me

points(log10(pca_data_met[pca_data_met$class_match == "Aves","met_rate"]), pca_data_met[pca_data_met$cla

points(log10(pca_data_met[pca_data_met$class_match == "Reptilia","met_rate"]), pca_data_met[pca_data_me

points(log10(pca_data_met[pca_data_met$class_match == "Actinopterygii","met_rate"]), pca_data_met[pca_da

points(log10(pca_data_met[pca_data_met$class_match == "Gastropoda","met_rate"]), pca_data_met[pca_data_

points(log10(pca_data_met[pca_data_met$class_match == "Demospongiae","met_rate"]), pca_data_met[pca_data

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

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

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



reproduction productivity on PC2 axis Need to allow this data and the met date flow through the scripts properly.

```
r_size_g <- vector()
notes <- vector()
trophic_again <- read.csv("Trophic_data_June16_2017.csv")

for(k in 1:length(pca_data$species_match)){
```

```
r_size_g[k] <- ((trophic_again[trophic_again$species  == as.vector(pca_data$species_match[k]),"repo_siz

notes[k] <- as.character((trophic_again[trophic_again$species  == as.vector(pca_data$species_match[k]),

}
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

```
egg_size_data0 <- data.frame(PC1 = pca_data$PC1,
                             r_size_g,
                             species = pca_data$species_match,
                             animal = pca_data$species_match,
                             taxa = pca_data$class_match,
                             notes = notes)

egg_size_data <- na.omit(egg_size_data0)


egg_size <- MCMCglmm(PC1 ~ r_size_g,
                      data = egg_size_data,
                      random=~animal + species,
                      pedigree = axis_trees[[1]],
                      prior = prior,
                      nitt = c(1100000), burnin = 10000, thin = 50, verbose = F)


egg_size_phylo <- egg_size$VCV[,1]/(egg_size$VCV[,1] +
                                    egg_size$VCV[,2] +
                                    egg_size$VCV[,3])
egg_size_species <- egg_size$VCV[,2]/(egg_size$VCV[,1] +
                                    egg_size$VCV[,2] +
                                    egg_size$VCV[,3])
egg_size_units <- egg_size$VCV[,3]/(egg_size$VCV[,1] +
                                    egg_size$VCV[,2] +
                                    egg_size$VCV[,3])

hdr(egg_size_phylo)
```

```
## $hdr
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## 99% 0.8951514 0.8972206 0.8987438 0.9015902 0.9047617 0.9908595
## 95% 0.9264072 0.9291886 0.9294333 0.9892499        NA        NA
## 50% 0.9660125 0.9835624        NA        NA        NA        NA
##
## $mode
## [1] 0.9780624
##
## $falpha
##         1%         5%        50%
##  0.5945475  2.8591756 22.7179454
```

```
hdr(egg_size_species)
```

```
## $hdr
##              [,1]        [,2]        [,3]       [,4]       [,5]       [,6]
## 99%            NA 0.062819057 0.063162674 0.06485098 0.06625850 0.06771723
## 95% -0.0006862196 0.042476363 0.043181378 0.04429806 0.04493563 0.04565082
## 50% -0.0003194662 0.008894888 0.009047336 0.01021854 0.01049438 0.01115119
##           [,7]       [,8]       [,9]      [,10]
## 99% 0.06890145 0.06924265 0.07144036 0.07174806
## 95%         NA         NA         NA         NA
## 50%         NA         NA         NA         NA
##
## $mode
## [1] 0.0001587444
##
## $falpha
##         1%         5%        50%
##  0.6843343  3.5753145 33.1714577
```

```
hdr(egg_size_units)
```

```
## $hdr
##            [,1]       [,2]
## 99% 0.009300005 0.03930224
## 95% 0.010616915 0.03263813
## 50% 0.015133330 0.02228768
##
## $mode
## [1] 0.01818734
##
## $falpha
##        1%        5%       50%
##  2.674841  9.018205 58.498888
```

Next up plot the ellipses.

Adding some ellipses for taxinomic groups

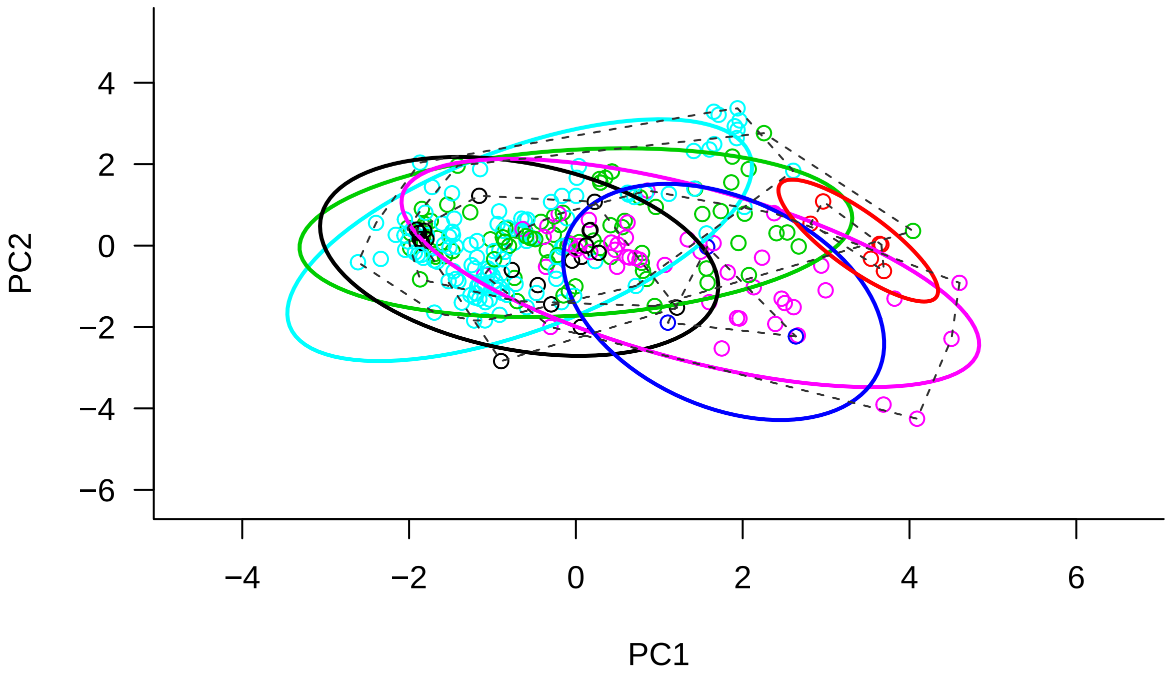
```
###add some ellipses by just using the SIBER stuff


##Set up the data
iucn_statue <- pop_multree$data$iucn_statues
pca_data$iucn_statue <- iucn_statue

therm_PCA <- pop_multree$data$met_type
pca_data$therm_PCA <- therm_PCA


##first we need to set up the data tp be read in as if its an isotope.

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",]

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)))


##we need to get rid of the

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


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



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

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


siber.plots <- createSiberObject(sidpca)
```

```
## Warning in createSiberObject(sidpca): At least one of your groups has less than 5 observations.
##           The absolute minimum sample size for each group is 3 in order
##           for the various ellipses and corresponding metrics to be
##           calculated. More reasonably though, a minimum of 5 data points
##           are required to calculate the two means and the 2x2 covariance
##           matrix and not run out of degrees of freedom. Check the item
##           named 'sample.sizes' in the object returned by this function
##           in order to locate the offending group. Bear in mind that NAs in
##           the sample.size matrix simply indicate groups that are not
##           present in that community, and is an acceptable data structure
##           for these analyses.
```

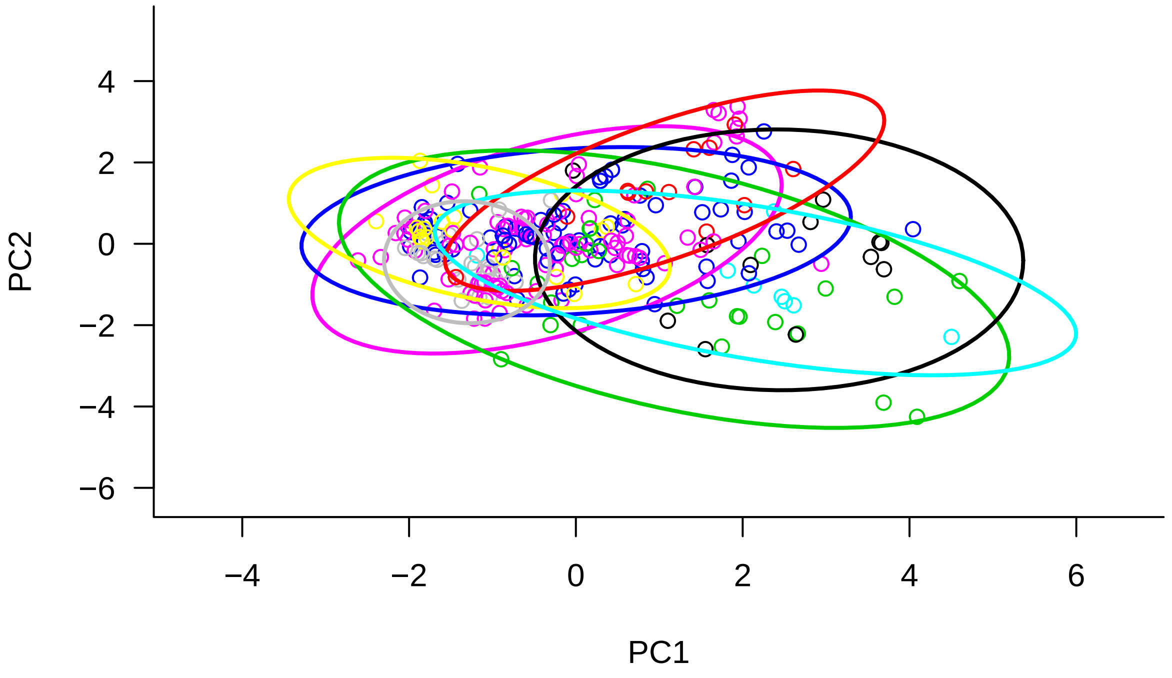
```
siber.mob <- createSiberObject(sidmob)
siber.therm <- createSiberObject(sidtherm)
siber.iucn<- createSiberObject(sired)
```

```
## Warning in createSiberObject(sired): At least one of your groups has less than 5 observations.
##           The absolute minimum sample size for each group is 3 in order
##           for the various ellipses and corresponding metrics to be
##           calculated. More reasonably though, a minimum of 5 data points
##            are required to calculate the two means and the 2x2 covariance
##            matrix and not run out of degrees of freedom. Check the item
##            named 'sample.sizes' in the object returned by this function
##            in order to locate the offending group. Bear in mind that NAs in
##            the sample.size matrix simply indicate groups that are not
##            present in that community, and is an acceptable data structure
##            for these analyses.
```

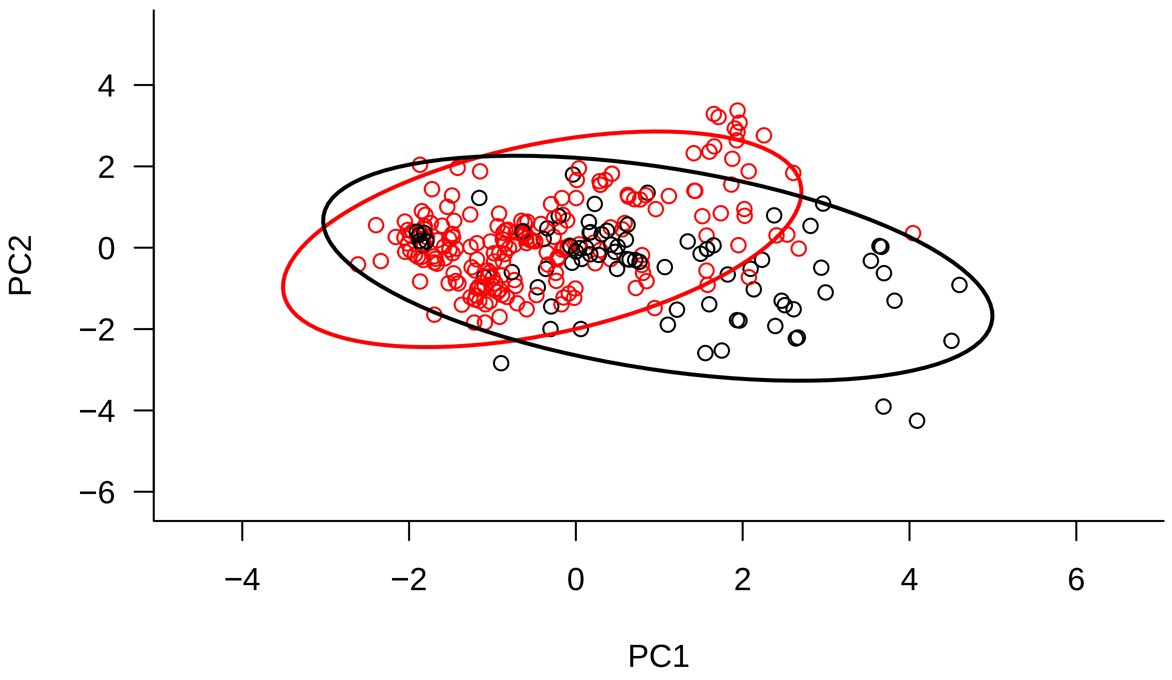
```
# Create lists of plotting arguments to be passed onwards to each
# of the three plotting functions.
community.hulls.args <- list(col = 1, lty = 1, lwd = 1)

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

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

#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"
                  )
```



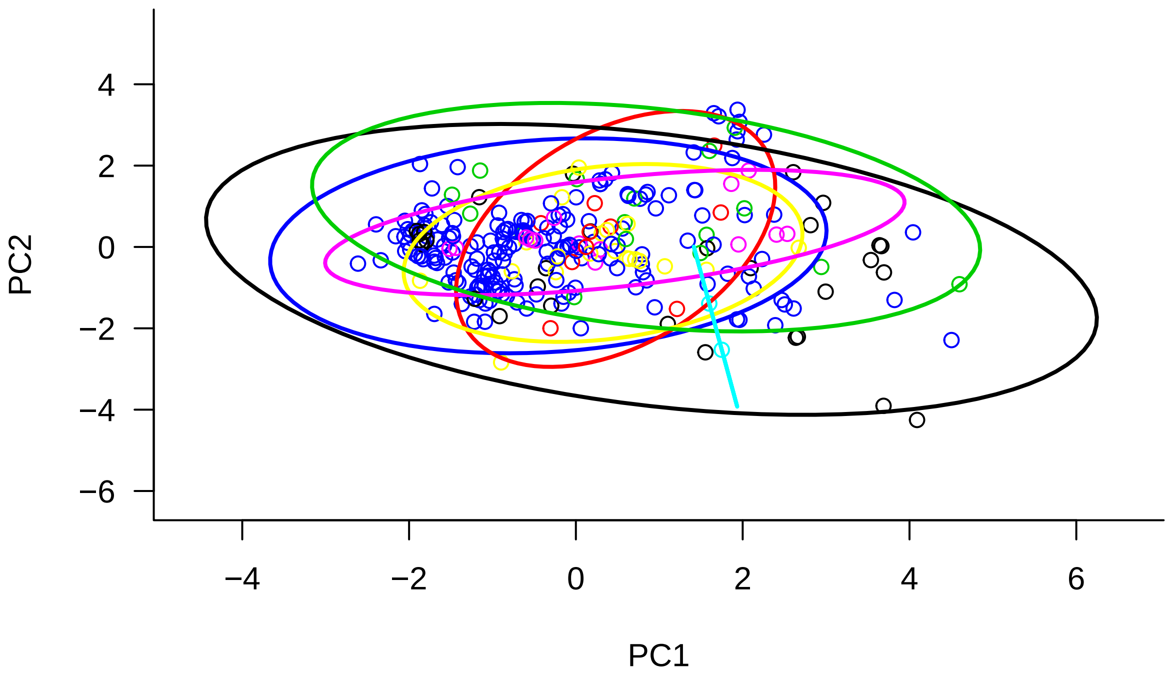
```
#plot for mode-of-life
plotSiberObject(siber.mob,
                  ax.pad = 2,
                  hulls = F, community.hulls.args,
                  ellipses = T, group.ellipses.args,
                  group.hulls = F, group.hull.args,
                  bty = "L",
                  iso.order = c(1,2),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



```
#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"
                   )
```



```r
#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"
                  )
```

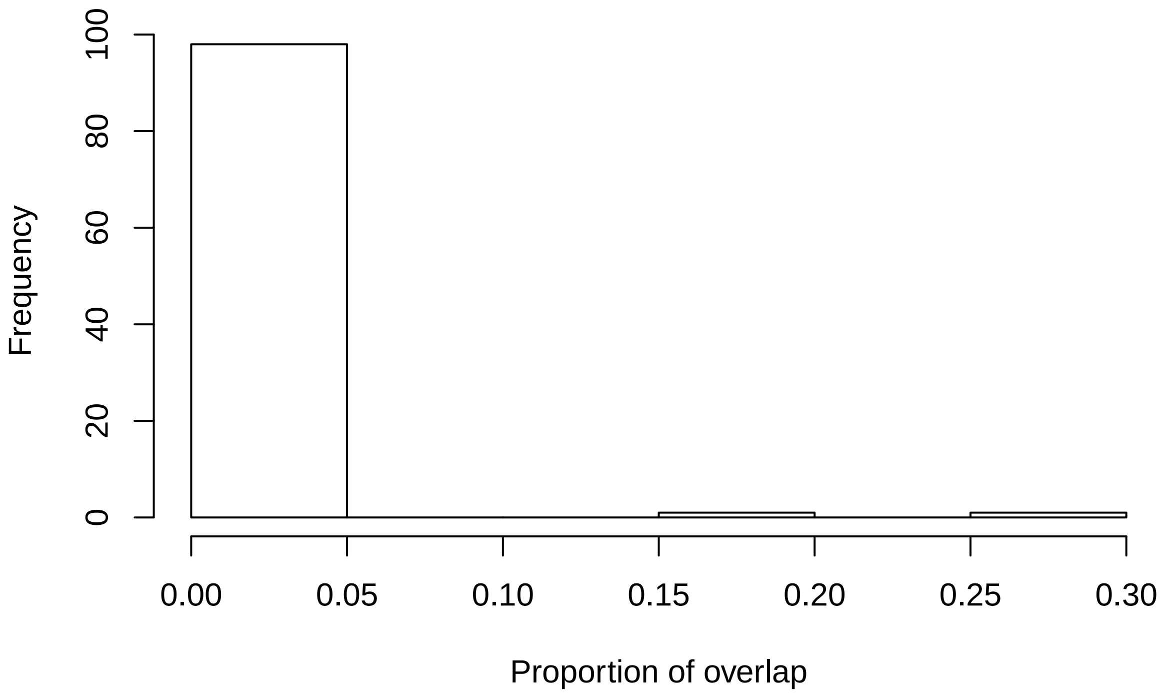


Ellipse overlap calculations for taxa

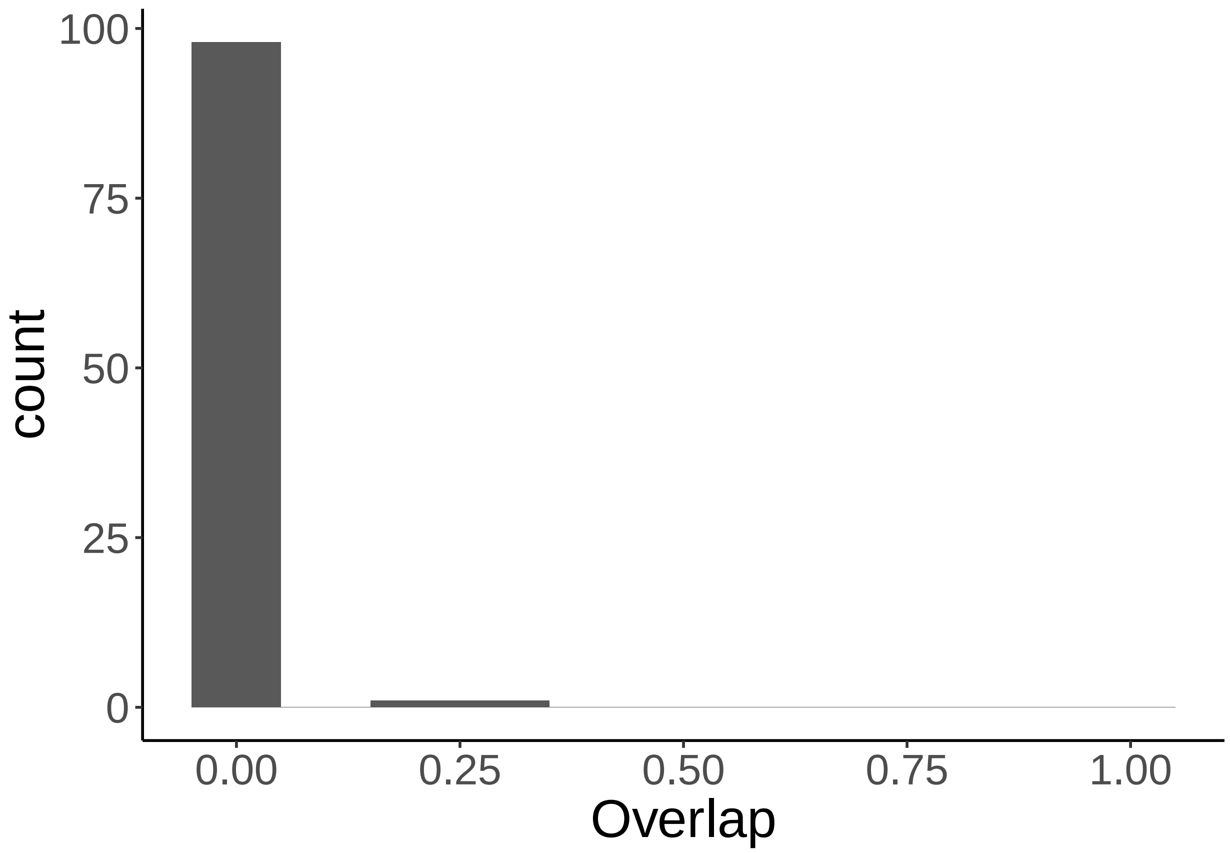
```r
group.MLtaxa <- groupMetricsML(siber.plots)
group.MLmob <- groupMetricsML(siber.mob)


# 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: 121
##    Unobserved stochastic nodes: 3
##    Total graph size: 136
##
## 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: 21
##    Unobserved stochastic nodes: 3
##    Total graph size: 36
##
## 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: 3
##    Unobserved stochastic nodes: 3
##    Total graph size: 18
##
## 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
```

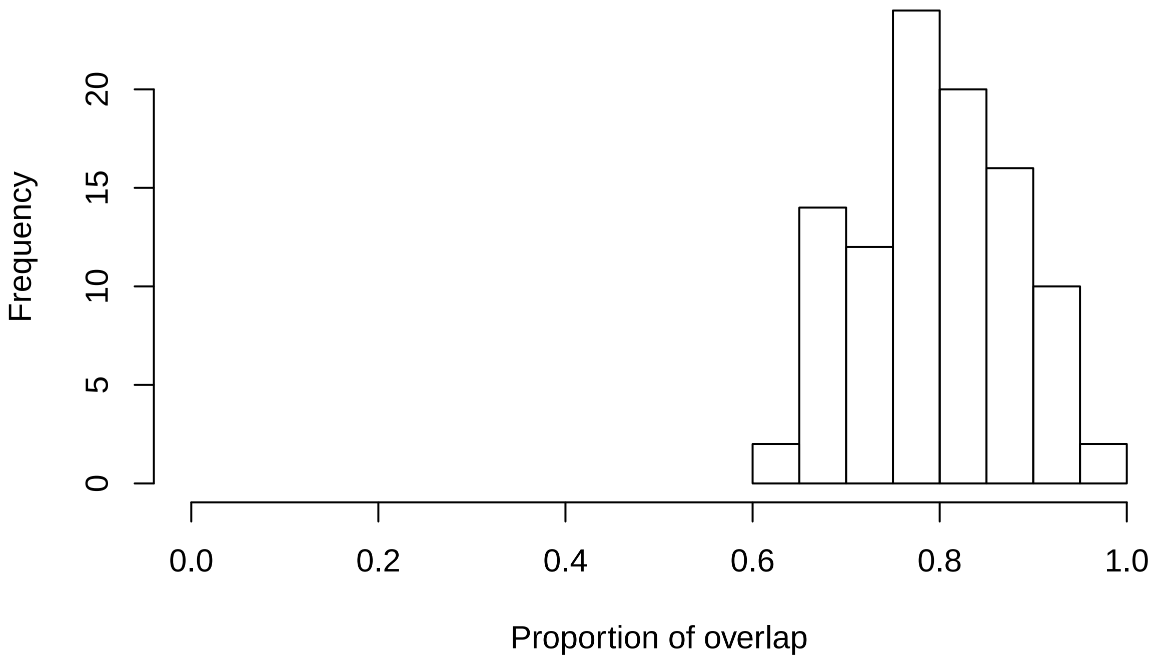
```
# The first ellipse is referenced using a character string representation where
# in "x.y", "x" is the community, and "y" is the group within that community.
# So in this example: community 1, group 1

#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)) +
  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))
```



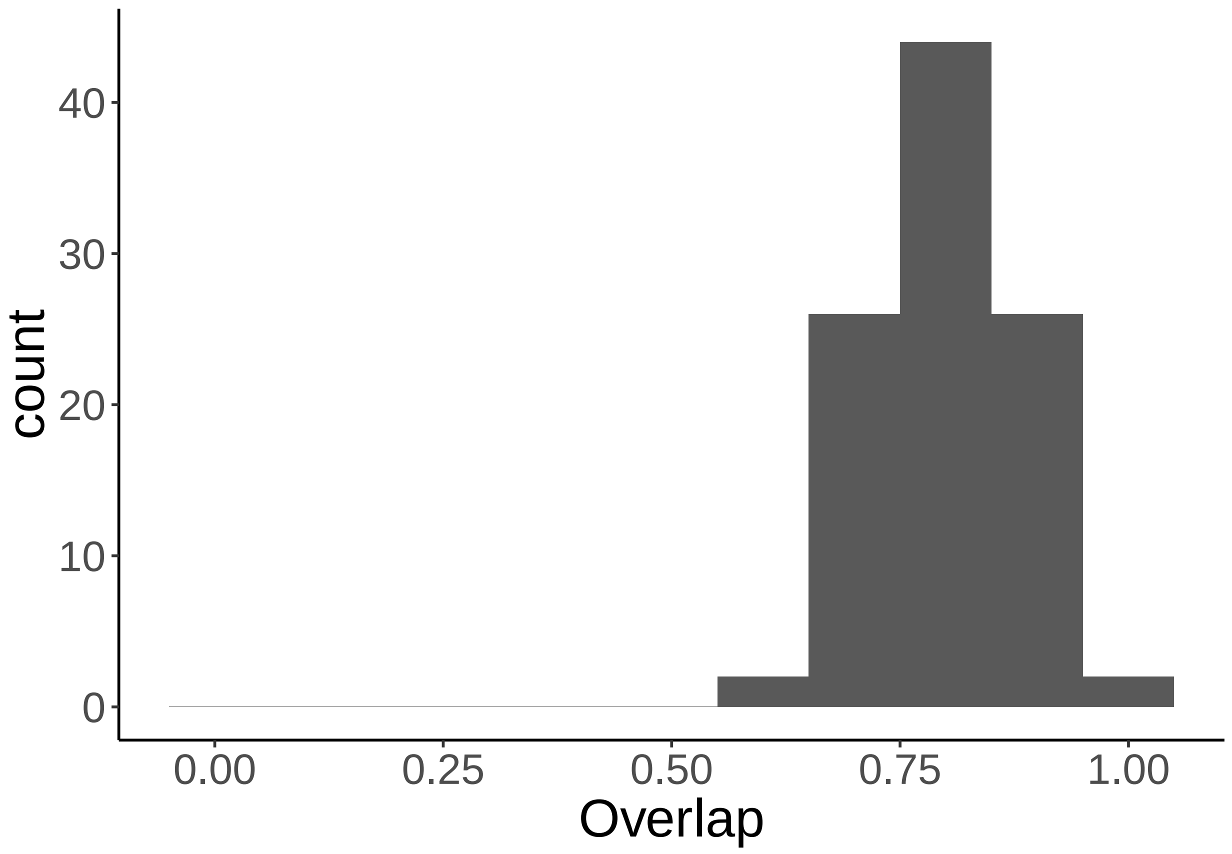
```
ggsave("fish_coral.png", dpi=300, width=4, height=3)

#hdr(AAn_95_overlap_prop, h = 10)


#####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))
```



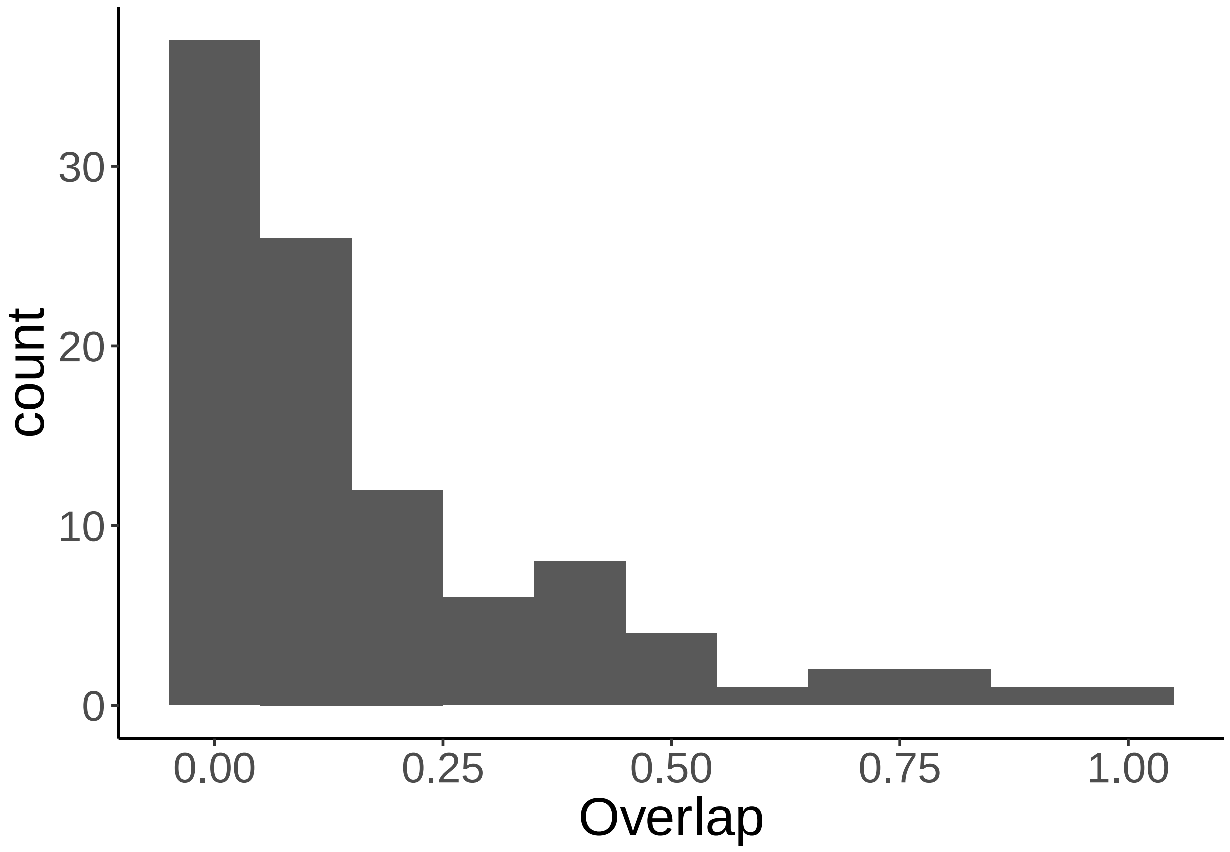
```r
hdr(AAv_95_overlap_prop)
```

```
## $hdr
##          [,1]      [,2]
## 99% 0.6351274 0.9531334
## 95% 0.6466299 0.9409581
## 50% 0.7479998 0.8574179
##
## $mode
## [1] 0.8015085
##
## $falpha
##       1%       5%      50%
## 1.136424 1.411099 3.481389
```

```r
myplot_Aav = ggplot(data.frame(Overlap =  AAv_95_overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  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))
```

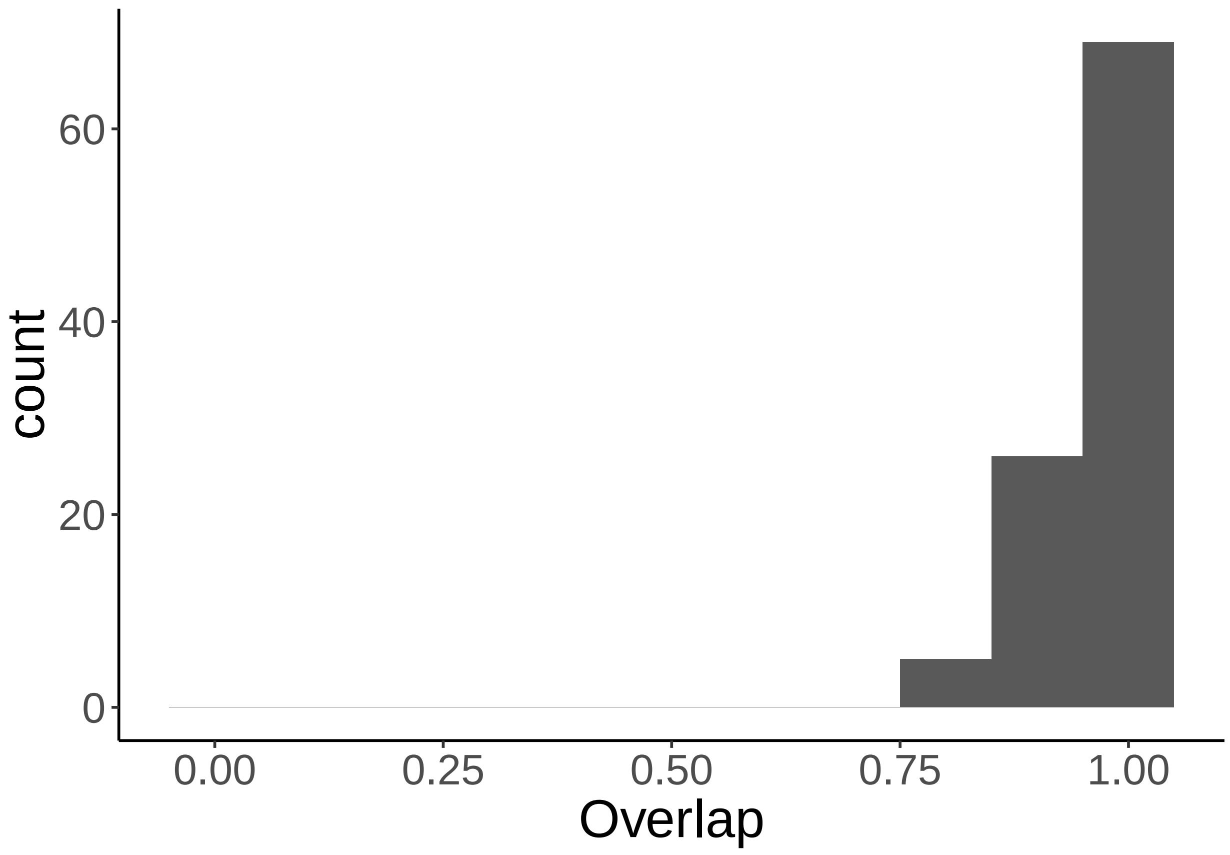


```
ggsave("fish_aves.png", dpi=300, width=4, height=3)




#####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)) +
  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))
```



```
ggsave("fish_gastropod.png", dpi=300, width=4, height=3)

#hdr(AG_95_overlap_prop)


##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)) +
  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))
```

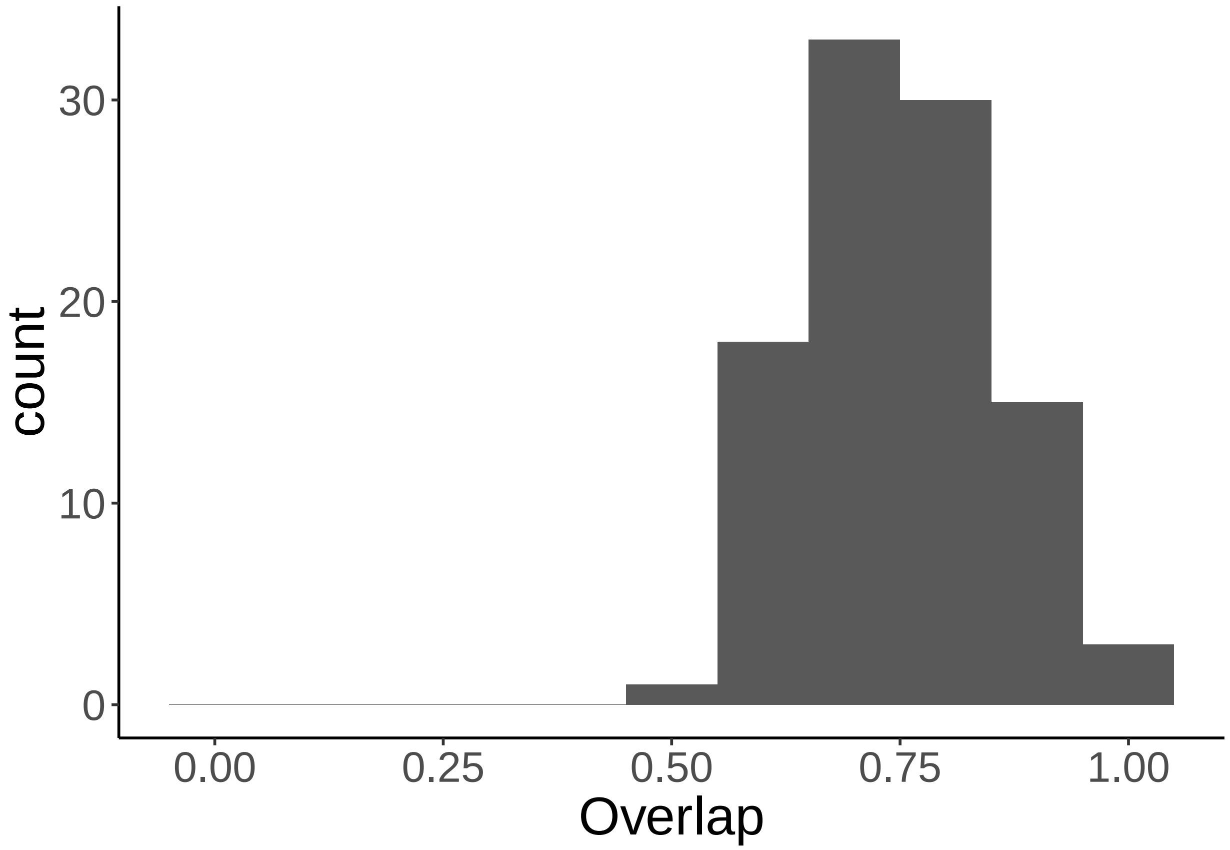


```
ggsave("fish_mammal.png", dpi=300, width=4, height=3)

hdr(Am_95_overlap_prop)
```

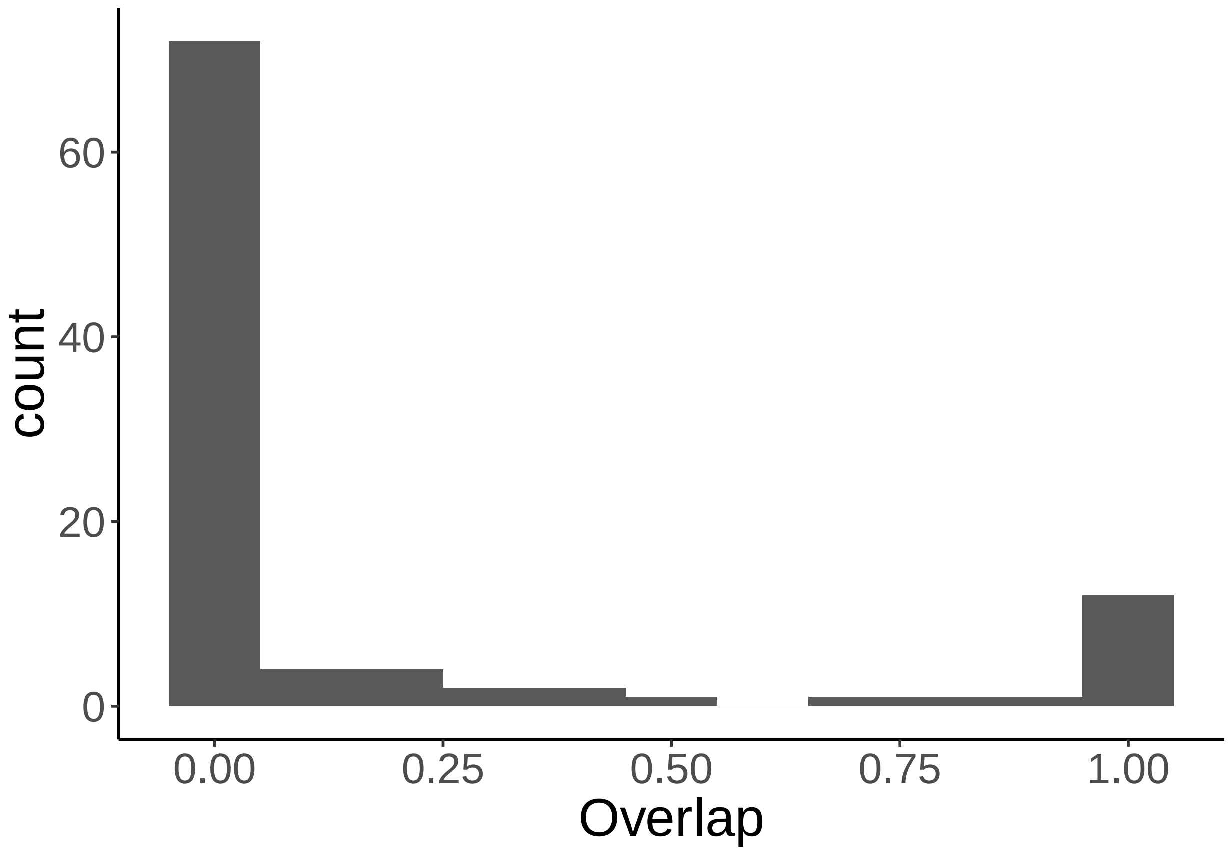
```
## $hdr
##          [,1]      [,2]      [,3]      [,4]    [,5]     [,6]
## 99% 0.8375788 1.0171641        NA        NA      NA       NA
## 95% 0.8446865 0.8538227 0.8781512 0.8871702 0.89052 1.015364
## 50% 0.9814695 1.0111432        NA        NA      NA       NA
##
## $mode
## [1] 0.9991794
##
## $falpha
##       1%       5%      50%
## 1.184232 2.145753 6.716224
```

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

```r
                                        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)) +
  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))
```

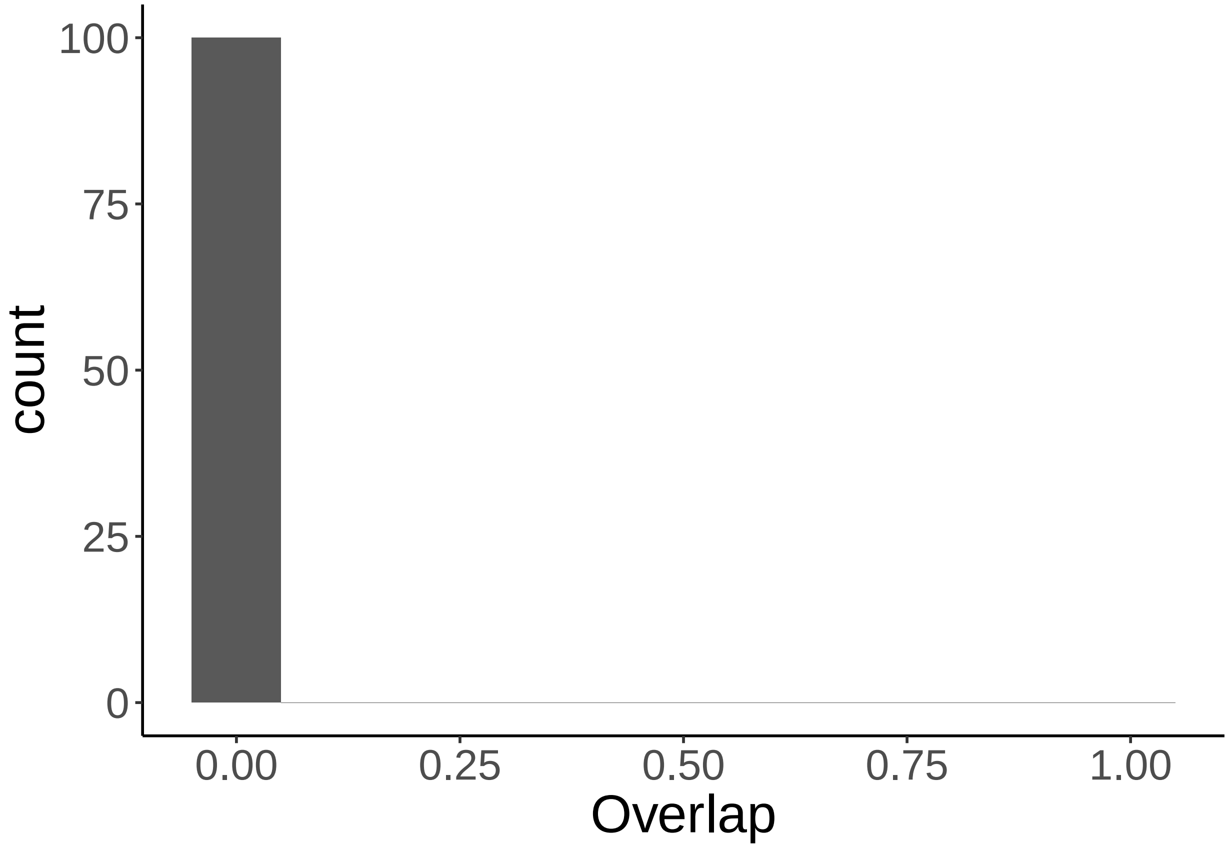


```r
ggsave("fish_reptile.png", dpi=300, width=4, height=3)

#hdr(AR_95_overlap_prop)



###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)) +
  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))
```



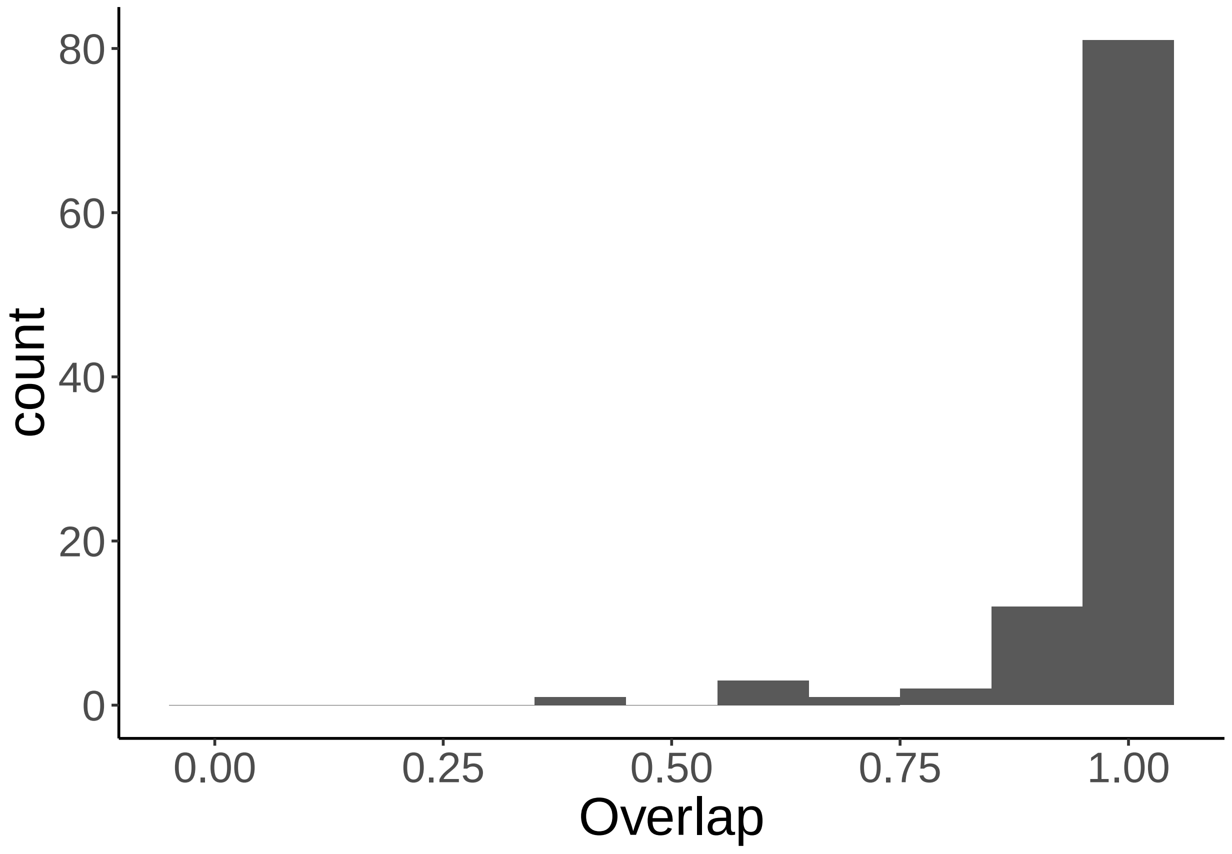
```
ggsave("coral_gastropod.png", dpi=300, width=4, height=3)

#hdr(AnG95_overlap_prop)


##coral mammal
```

```r
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)) +
  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))
```

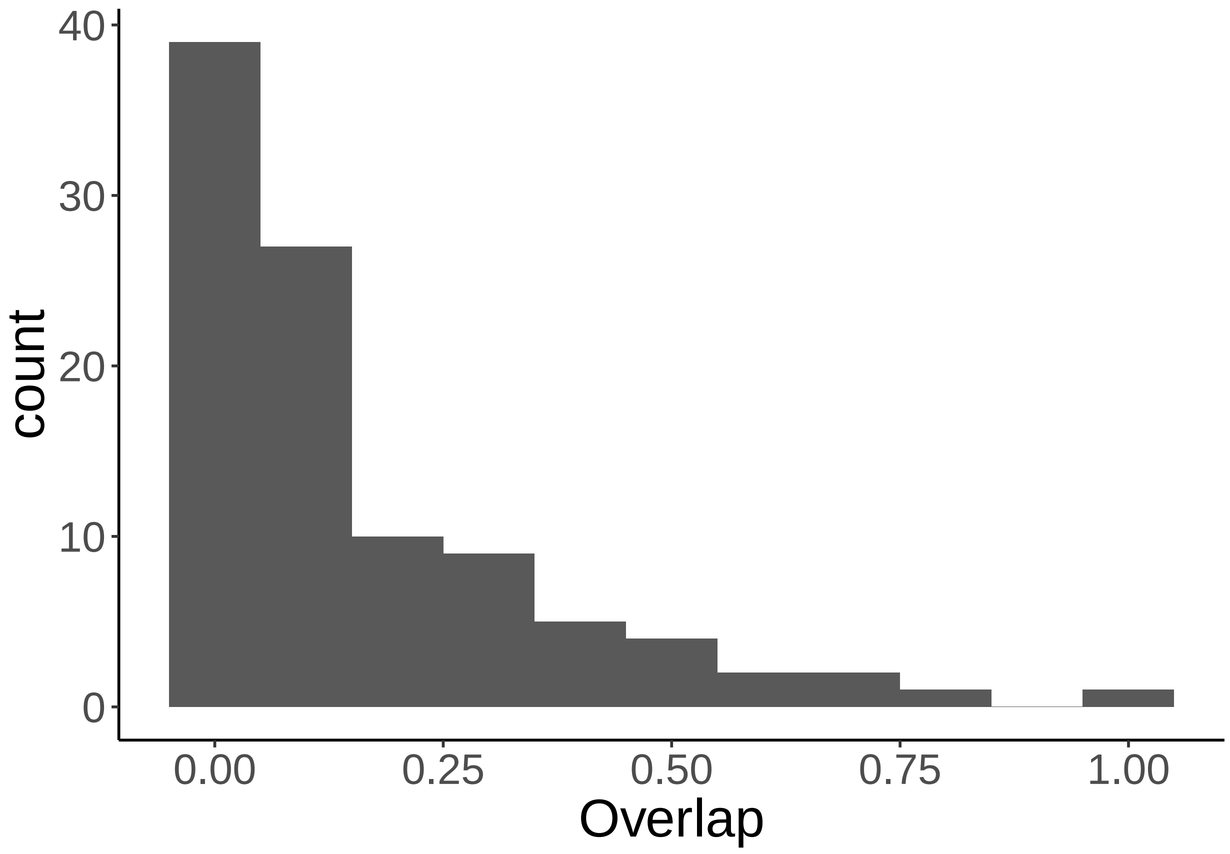


```r
ggsave("coral_mammal.png", dpi=300, width=4, height=3)

#hdr(AnM95_overlap_prop, h = 100)
```

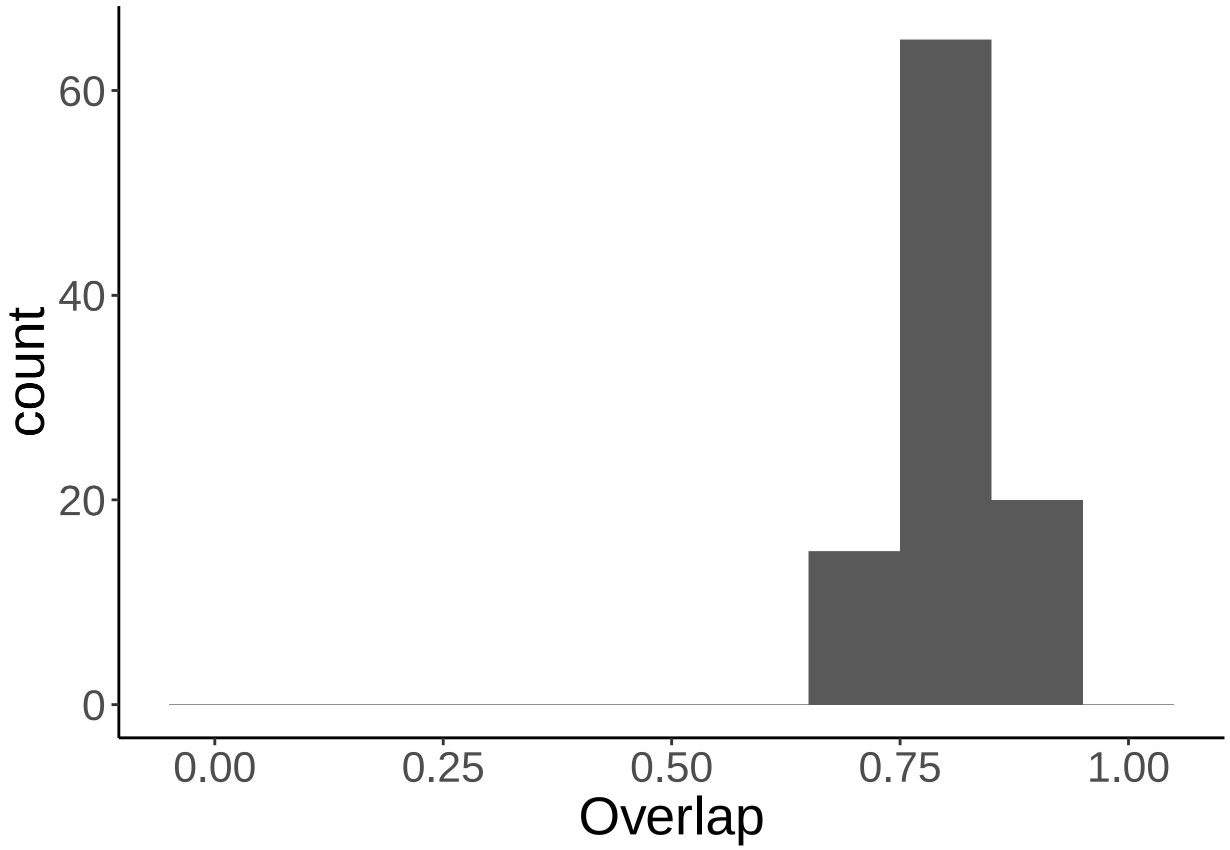
```r
##coral reptile
AnR95.overlap <- bayesianOverlap(ellipse_Anthozoa,
                                  ellipse_Reptilia,
                                  ellipses.posterior,
                                  draws = 100,
                                  p.interval = 0.95,
                      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



```r
ggsave("coral_reptile.png", dpi=300, width=4, height=3)

#hdr(AnR95_overlap_prop)


###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)) +
  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))
```

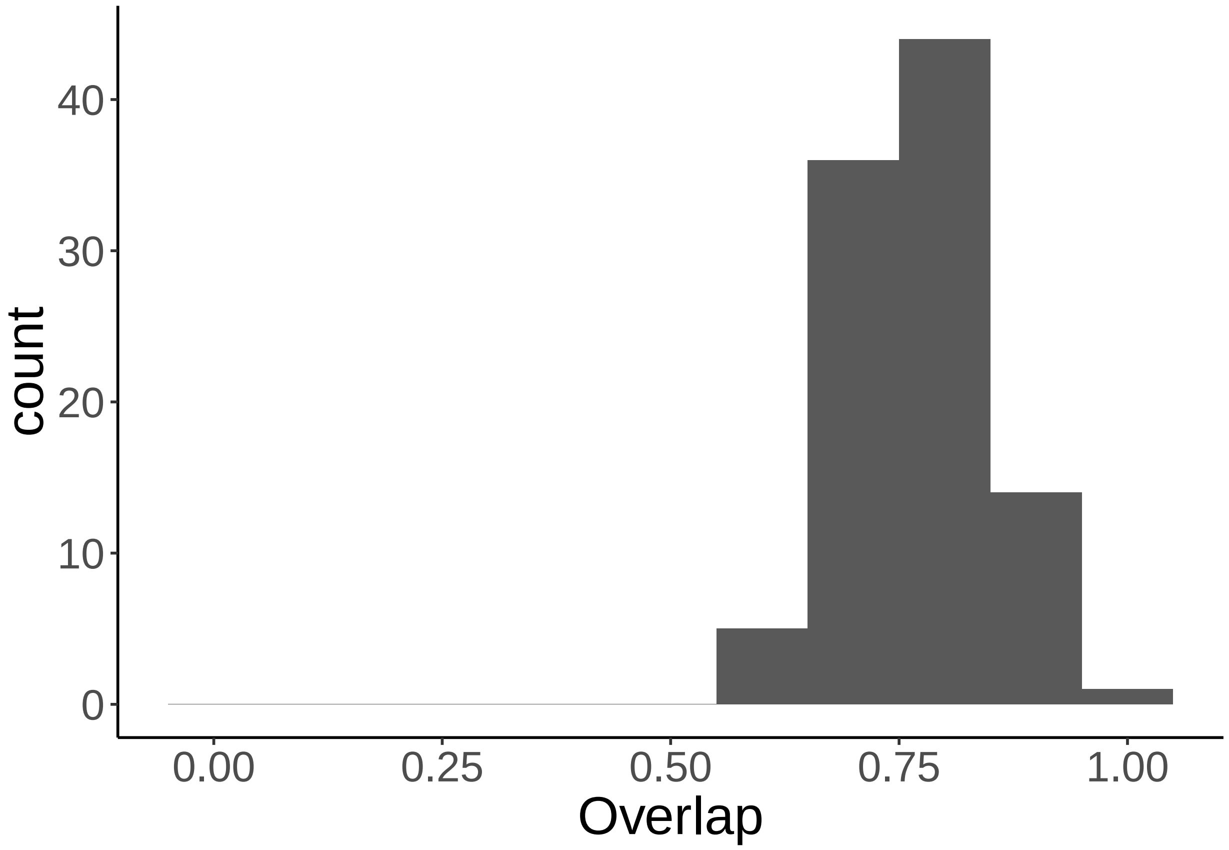


```
ggsave("aves_gastorpod.png", dpi=300, width=4, height=3)

#hdr(AvG95_overlap_prop)


####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)) +
  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))
```



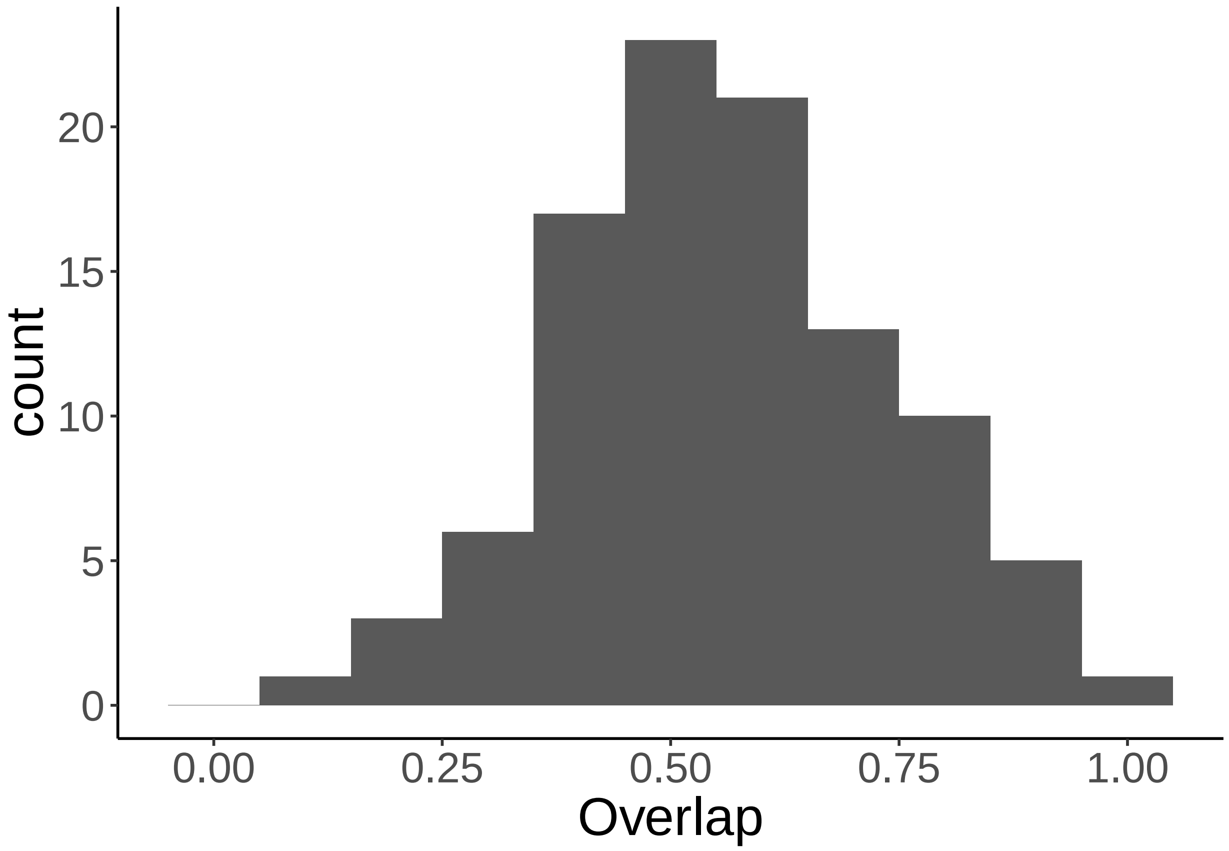
```
ggsave("aves_mammal.png", dpi=300, width=4, height=3)

#hdr(AvM95_overlap_prop)


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

```r
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))
```



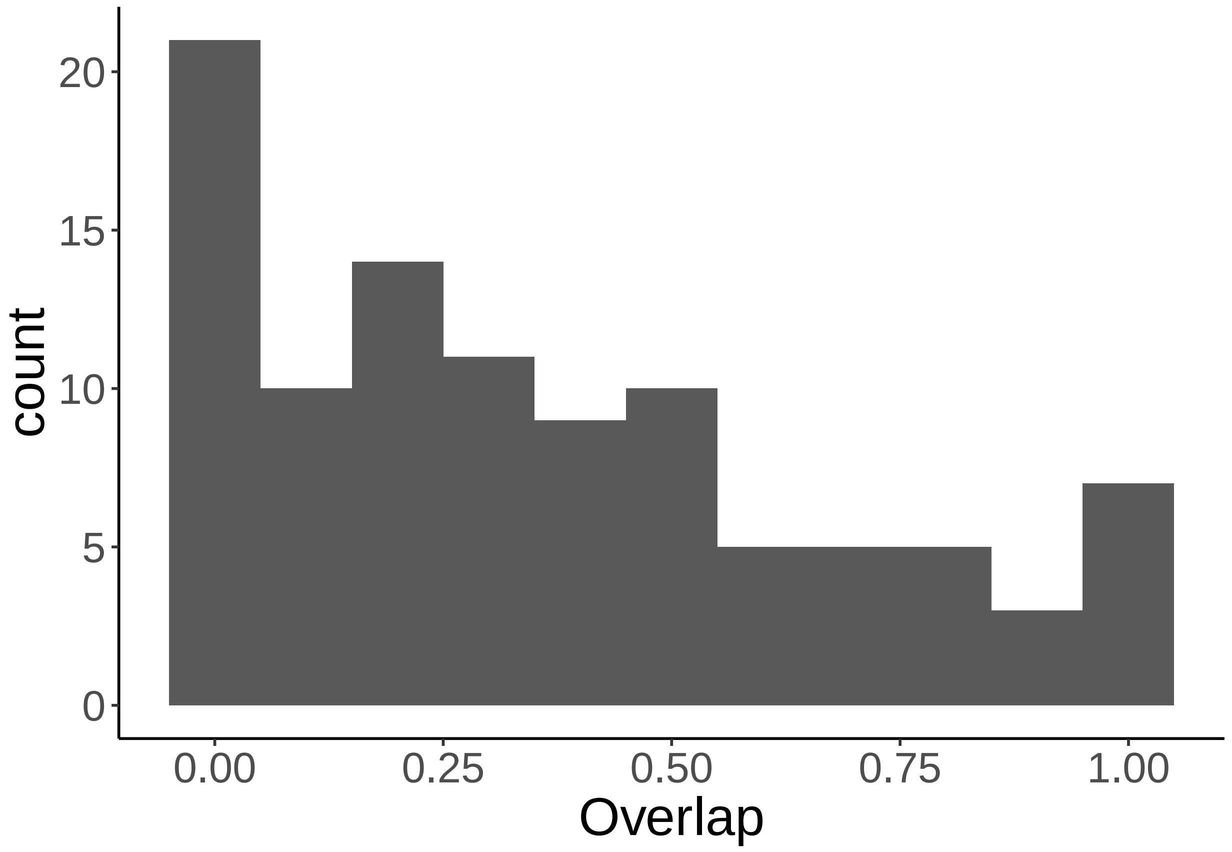
```r
ggsave("aves_reptile.png", dpi=300, width=4, height=3)

#hdr(AvR95_overlap_prop)



##### 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])

}
```

```r
myplot_RG = ggplot(data.frame(Overlap =  RG95_overlap_prop),
                    aes(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))
```



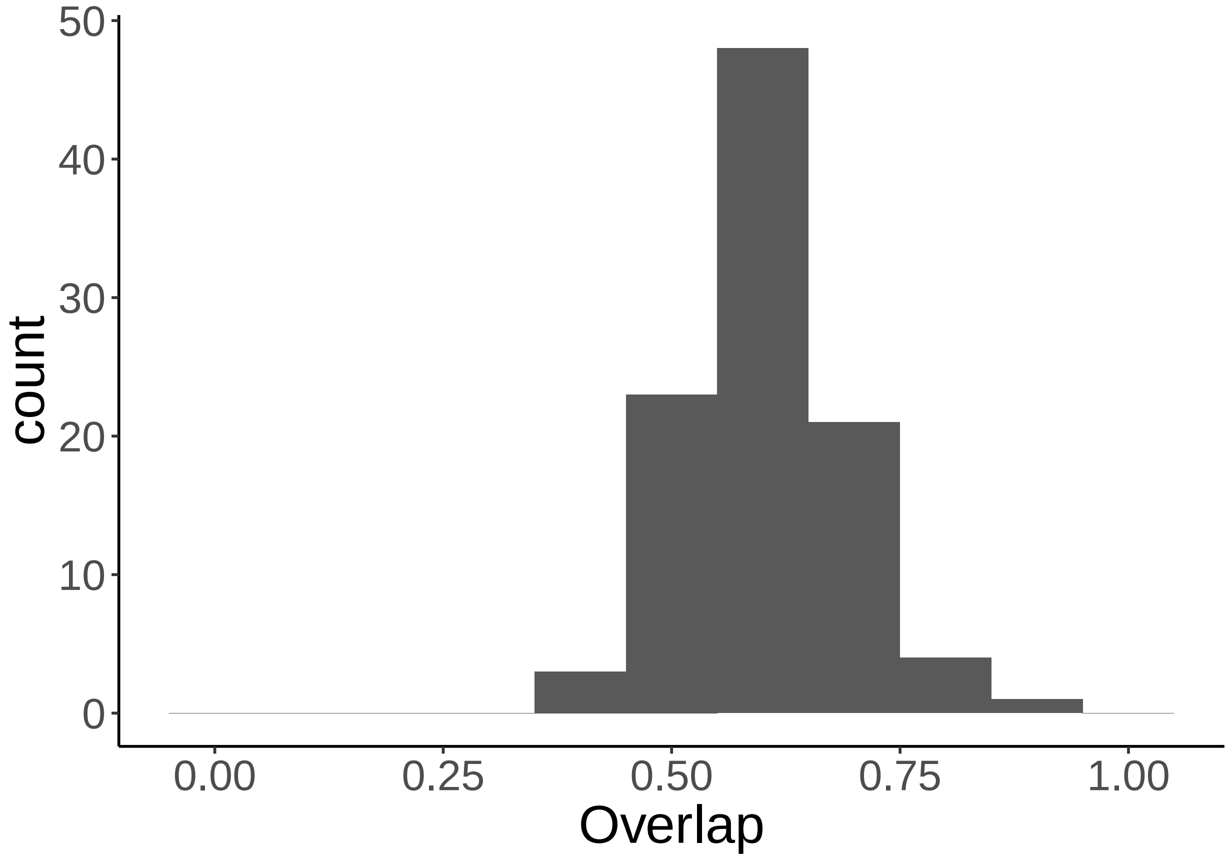
```r
ggsave("reptile_gastropod.png", dpi=300, width=4, height=3)

#hdr(RG95_overlap_prop)



##### 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)){
```

```r
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)) +
  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))
```

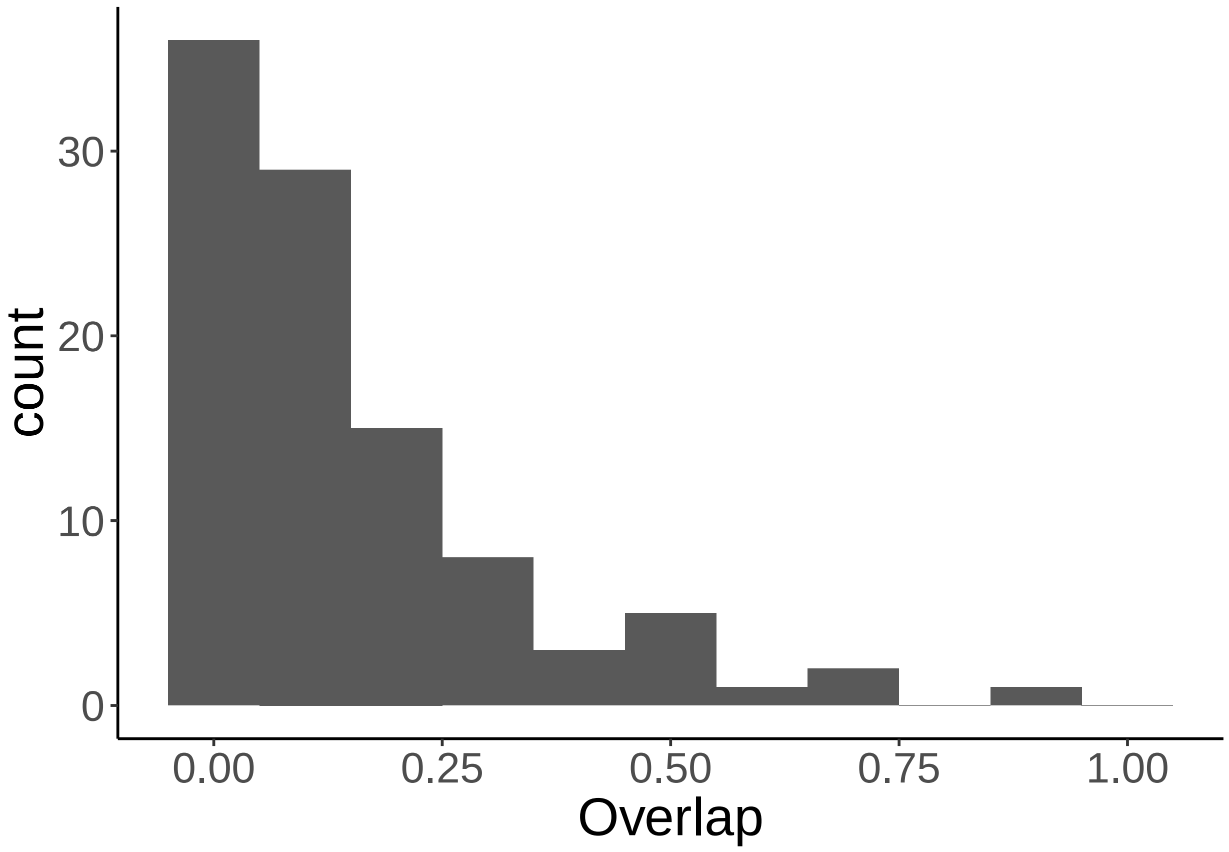


```r
ggsave("aves_coral.png", dpi=300, width=4, height=3)

#hdr(AvAn95_overlap_prop, h = 100)



##### reptile mammal

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

```r
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)) +
  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))
```



```r
ggsave("reptile_mammal.png", dpi=300, width=4, height=3)

#hdr(RM95_overlap_prop)


#####  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)) +
  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))
```



```
ggsave("mammal_gastropod.png", dpi=300, width=4, height=3)

hdr(MG95_overlap_prop)
```

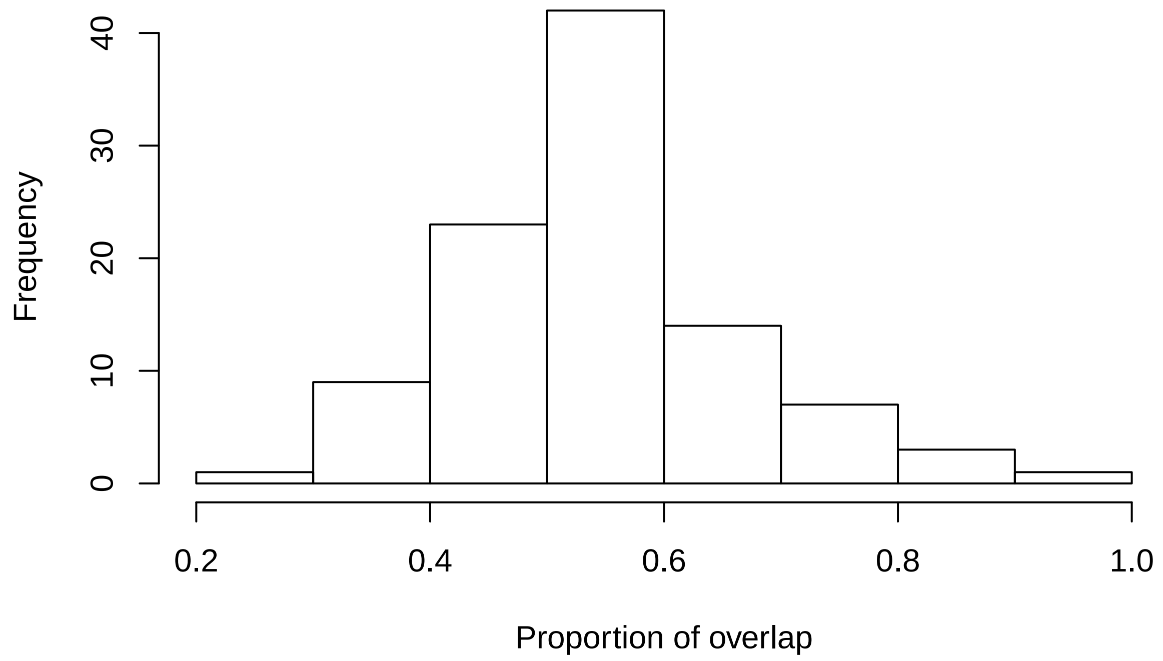
```
## $hdr
##             [,1]      [,2]      [,3]      [,4]
## 99% -0.08654210 0.5607515 0.6224771 0.6749137
## 95% -0.08073216 0.4201050 0.4506731 0.5324460
## 50% -0.01683704 0.1000328        NA        NA
##
## $mode
## [1] 0.0209193
##
## $falpha
##        1%        5%       50%
## 0.2394809 0.3297246 2.9250461
```

Ellipse overlap calculations for mode of life

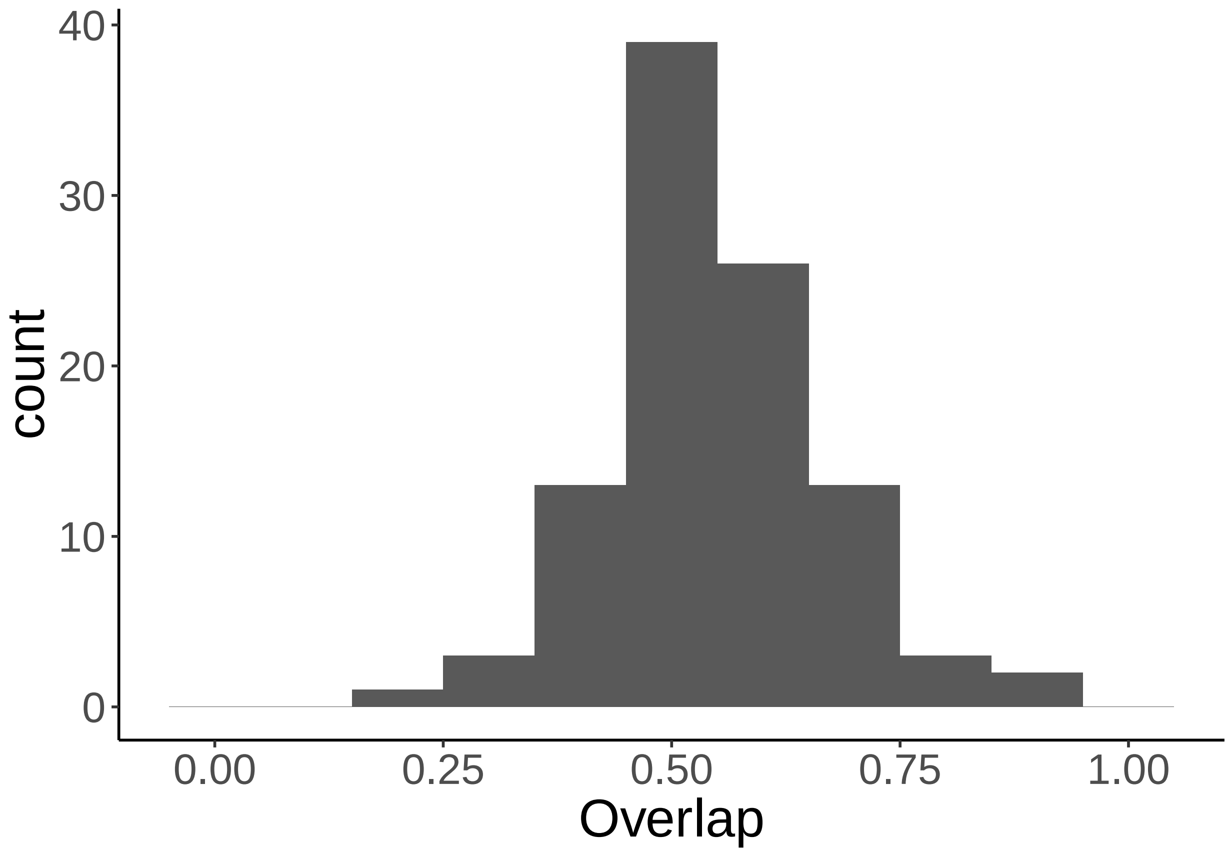
```
group.ML <- groupMetricsML(siber.plots)
group.MLmob <- groupMetricsML(siber.mob)


# 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: 95
##    Unobserved stochastic nodes: 3
##    Total graph size: 110
##
## 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: 28
##    Unobserved stochastic nodes: 3
##    Total graph size: 43
##
## 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: 21
##    Unobserved stochastic nodes: 3
##    Total graph size: 36
##
## 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: 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
```

```
# 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])

}

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



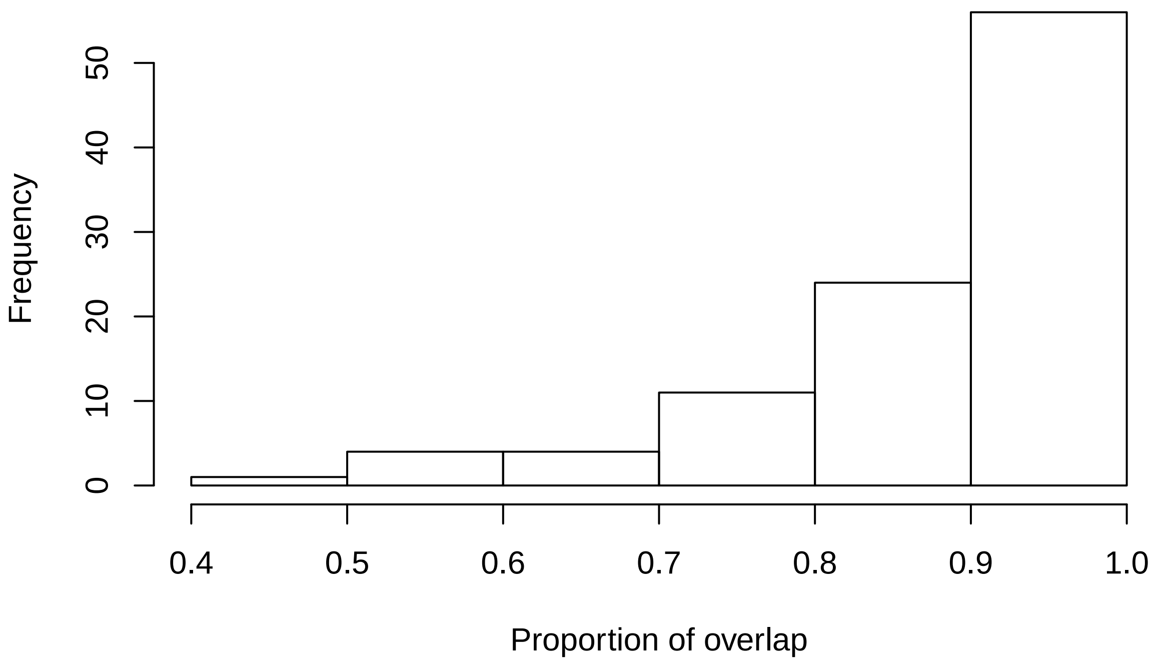
```
myplot_SA = ggplot(data.frame(Overlap =  SA_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



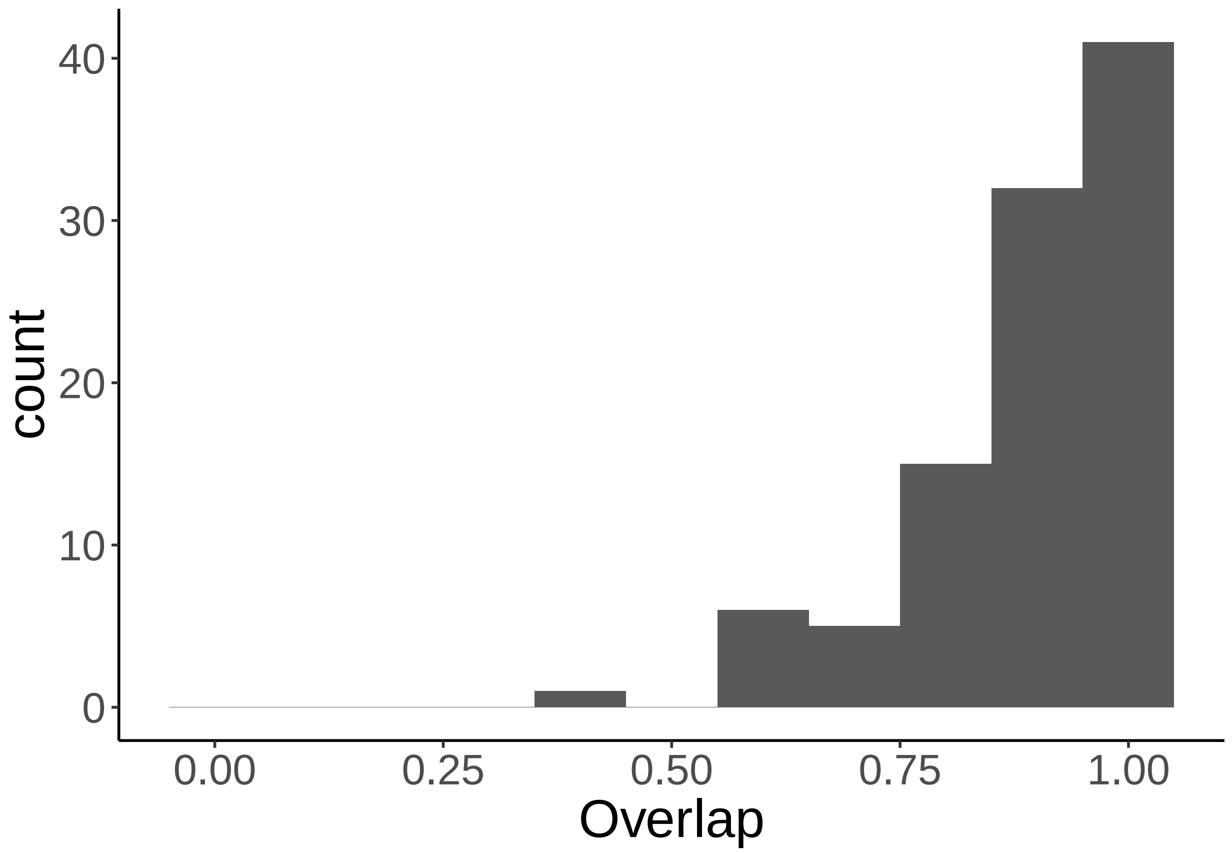
```
ggsave("sessil_arboreal.png", dpi=300, width=4, height=3)

hdr(SA_95.overlap_prop, h = 10)
```

```
## $hdr
##          [,1]      [,2]
## 99% 0.2066101 0.8871577
## 95% 0.3013192 0.7942679
## 50% 0.4858807 0.6159243
##
## $mode
## [1] 0.494468
##
## $falpha
##         1%         5%        50%
## 0.03986693 0.03987784 0.03988960
```

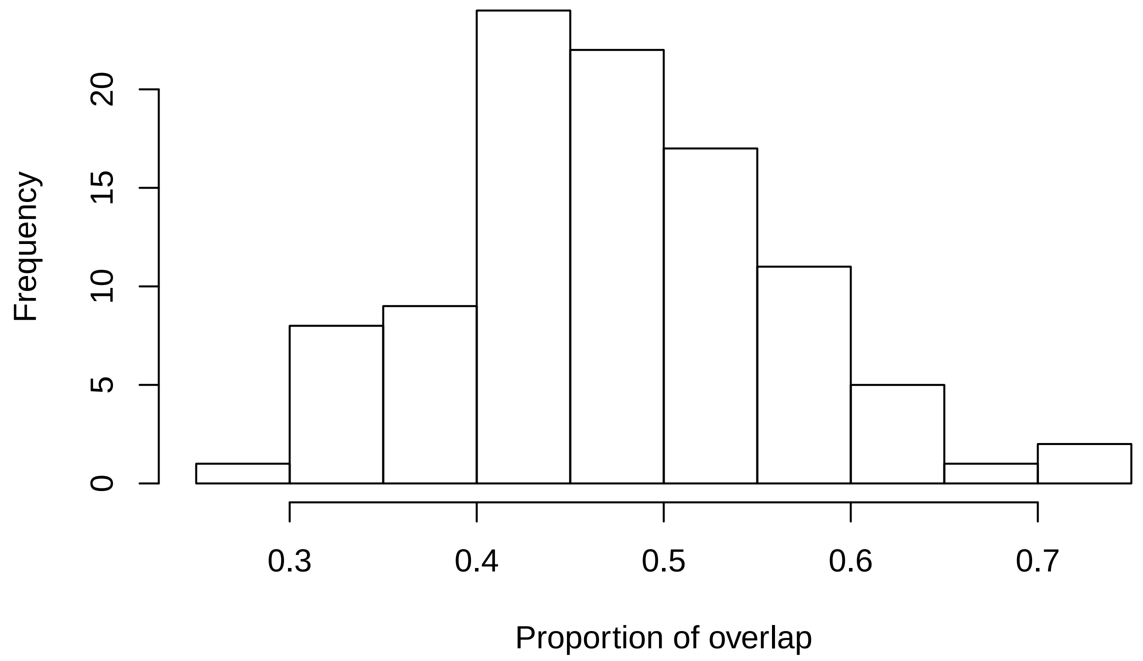
```
#####sessile  - benthic
SB_95.overlap <- bayesianOverlap(ellipse_sessile,
                                   ellipse_benthic,
                                   ellipses.posterior_mob,
                                   draws = 100,
                                   p.interval = 0.95,
         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



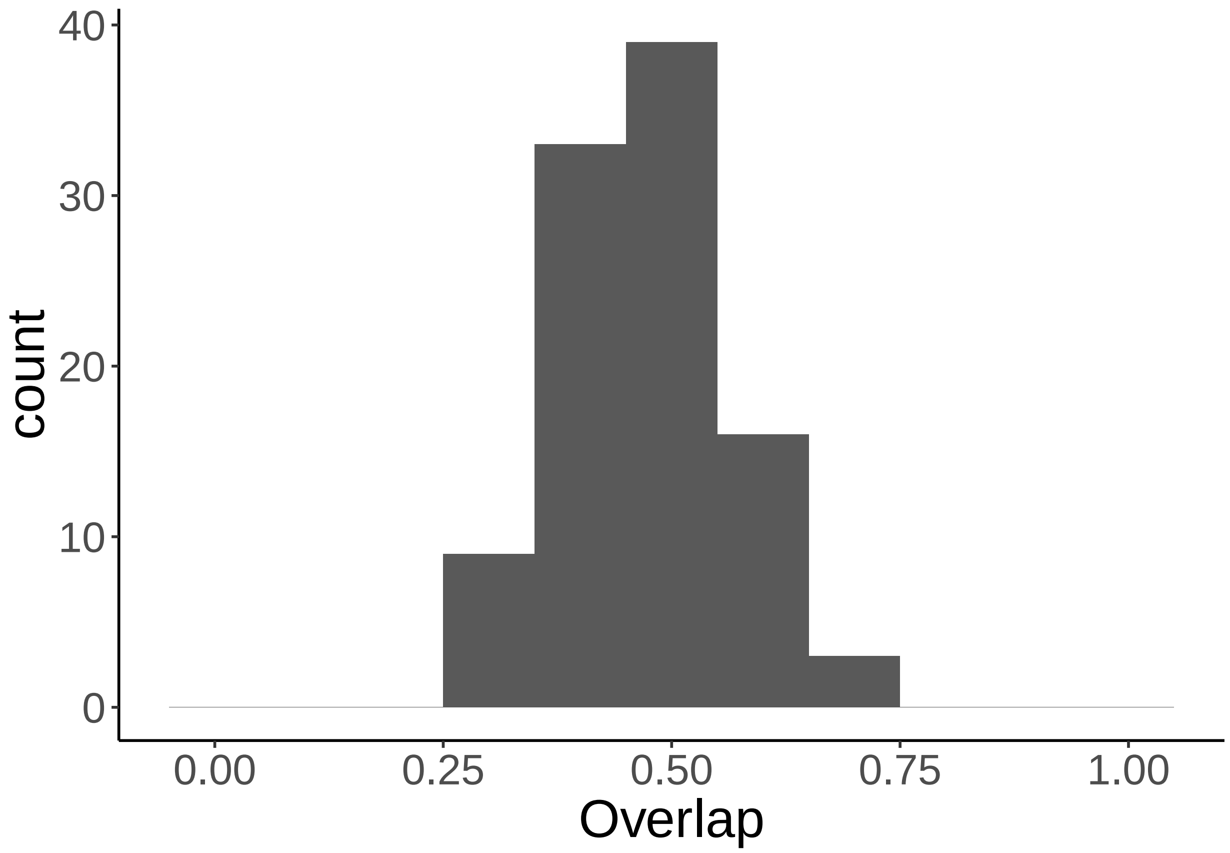
```
myplot_SB = ggplot(data.frame(Overlap =  SB_95.overlap_prop),
                   aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



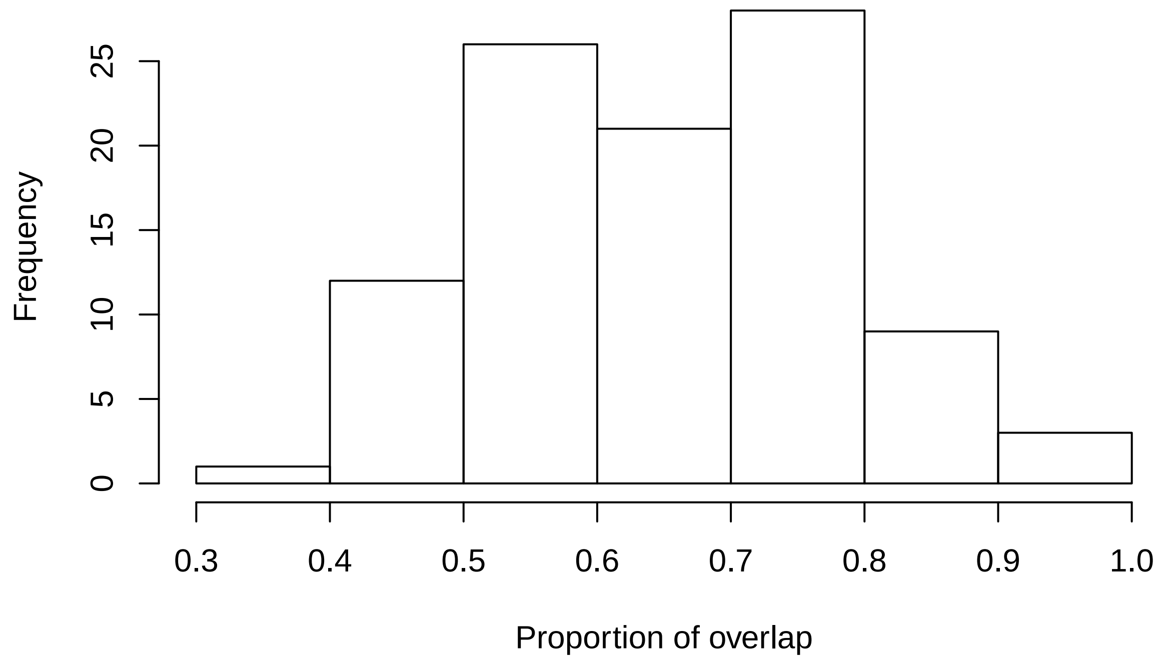
```
ggsave("sessil_benthic.png", dpi=300, width=4, height=3)

#hdr(SB_95.overlap_prop, h = 10)


#####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 = "")
```



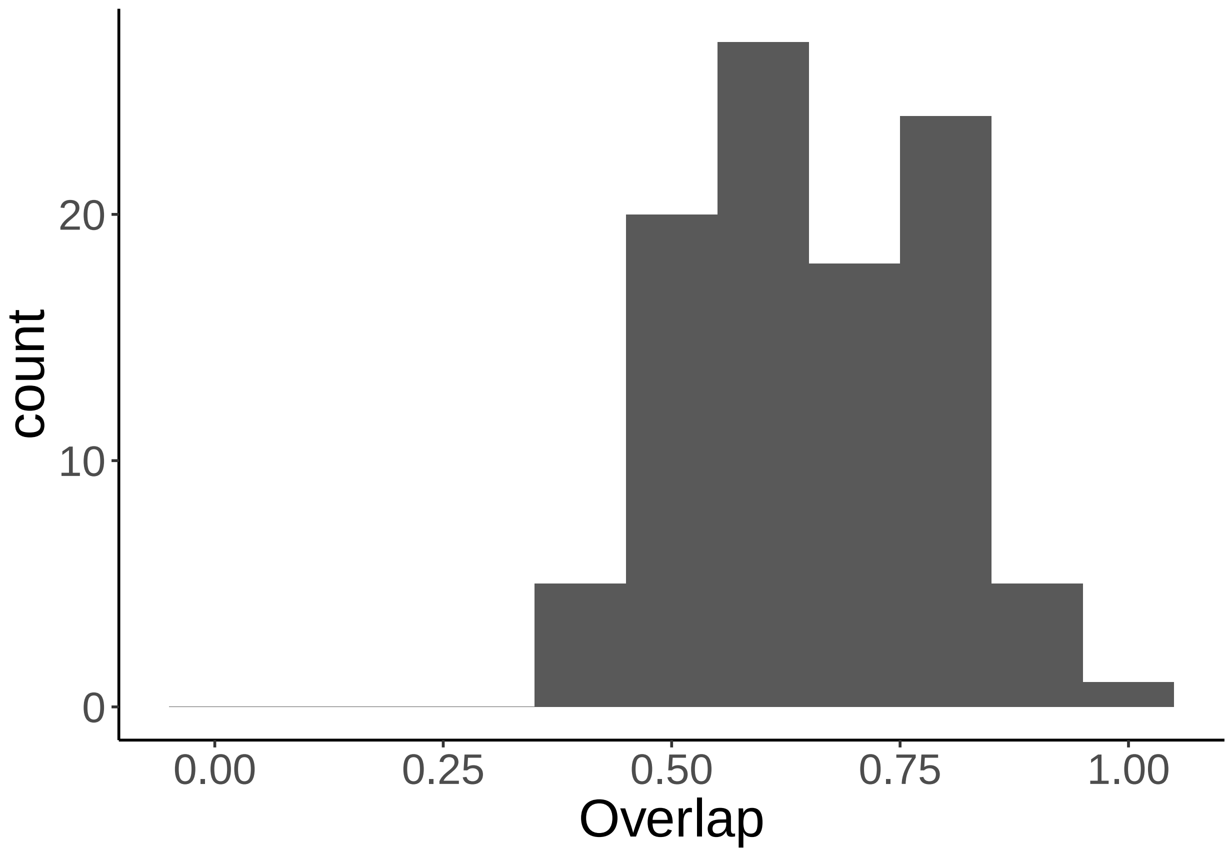
```
myplot_Sv = ggplot(data.frame(Overlap =  Sv_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



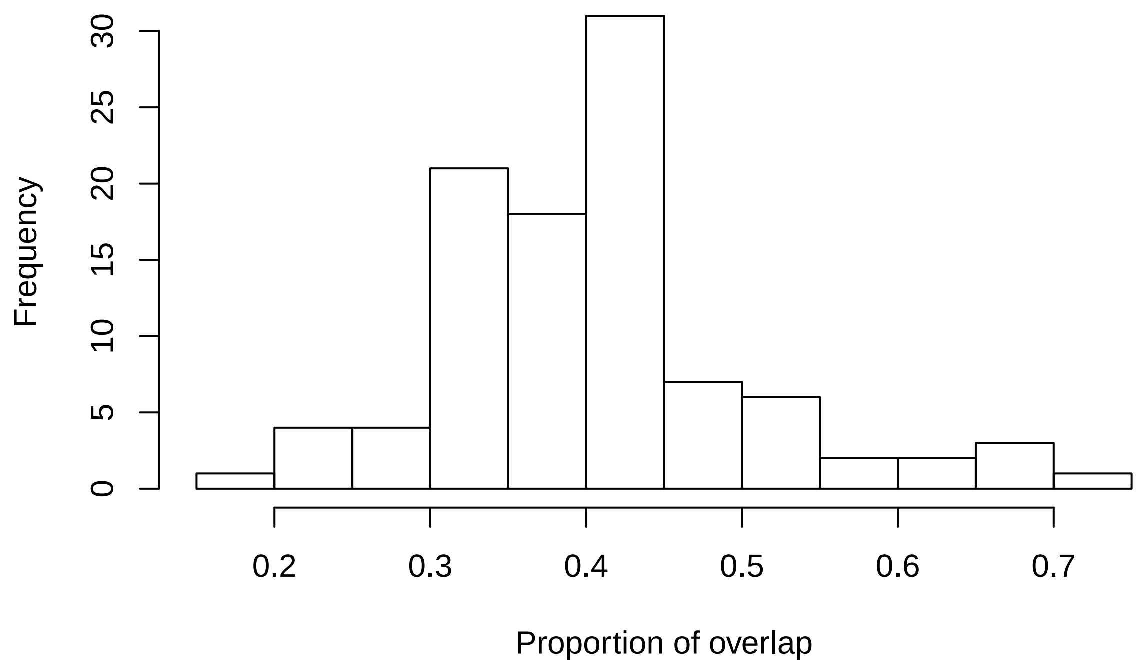
```
ggsave("sessil_volant.png", dpi=300, width=4, height=3)

#hdr(Sv_95.overlap_prop, h = 10)



#####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 = "")
```



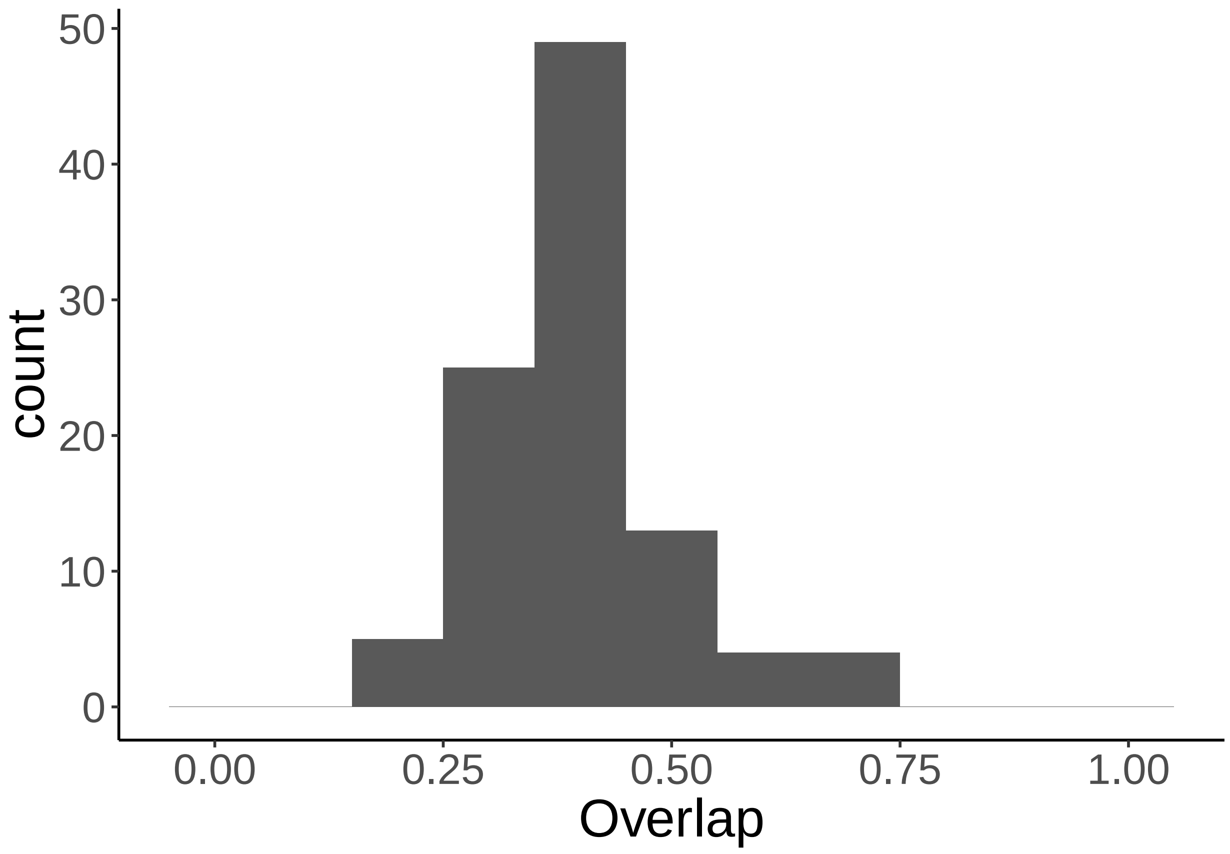
```
myplot_Ssem = ggplot(data.frame(Overlap =  Ssem_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



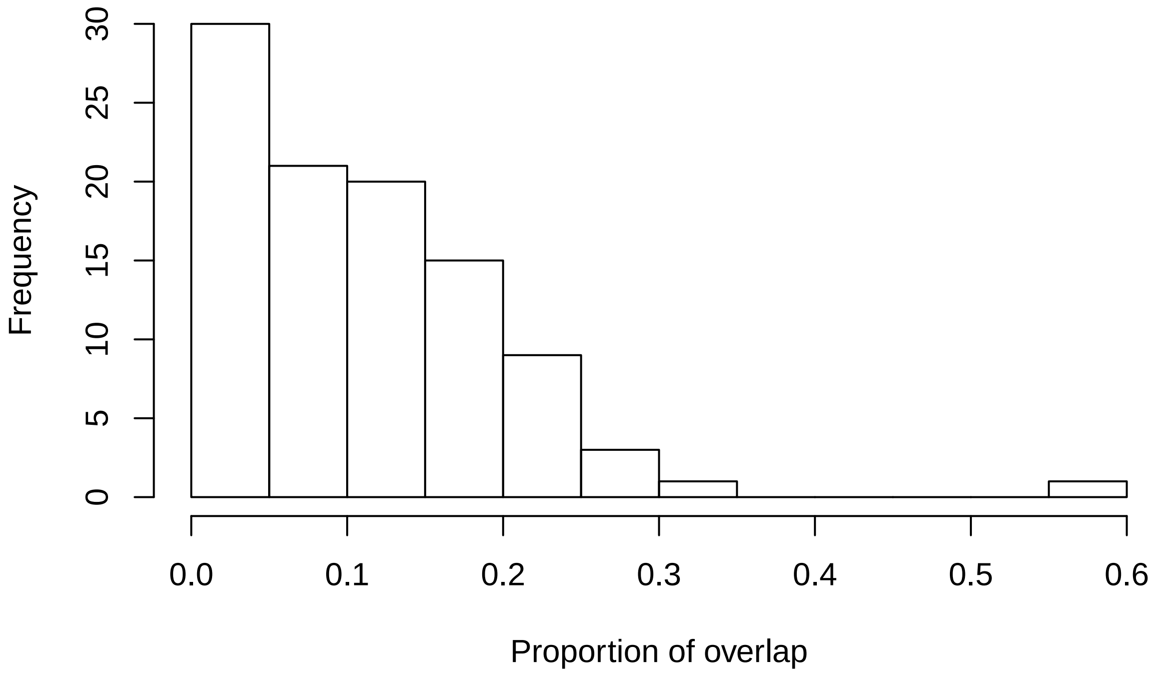
```
ggsave("sessil_semiaquativ.png", dpi=300, width=4, height=3)

#hdr(Ssem_95.overlap_prop, h = 10)


#####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 = "")
```



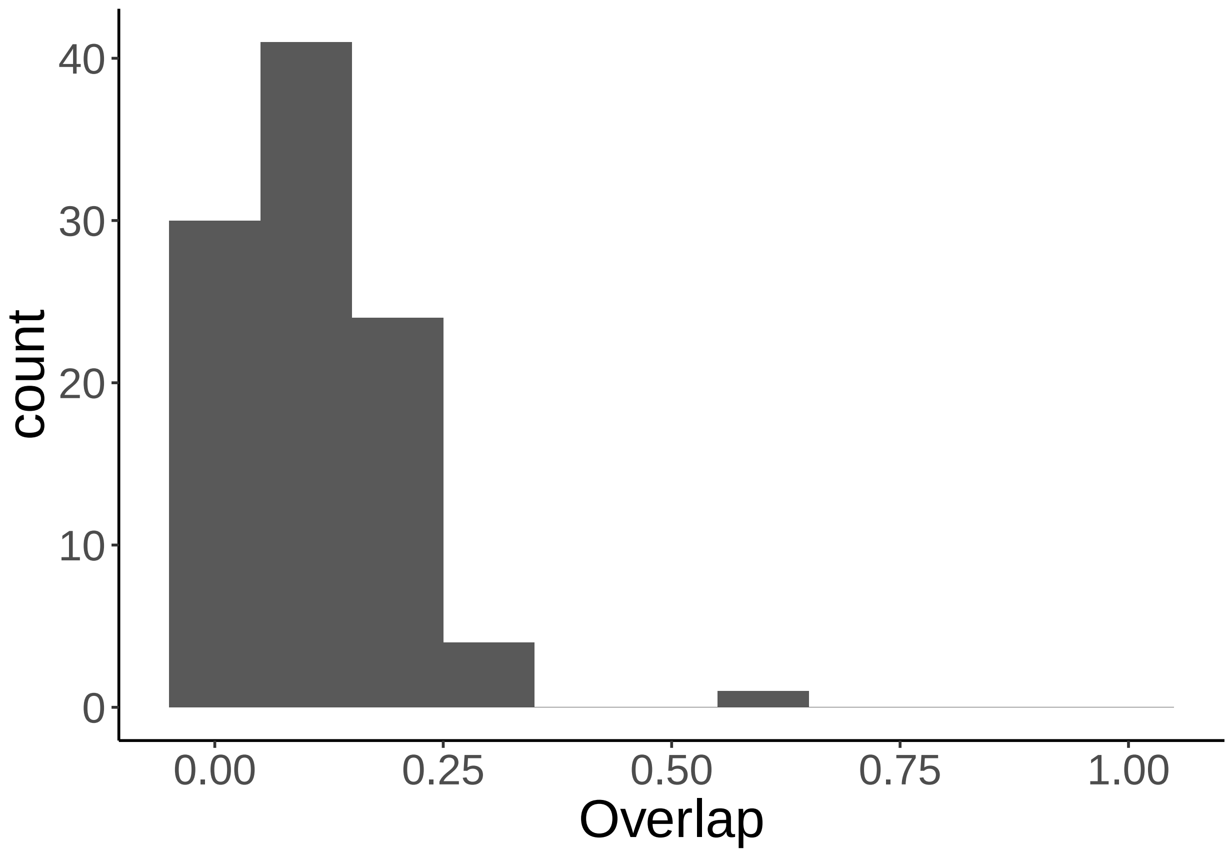
```
myplot_St = ggplot(data.frame(Overlap =  St_95.overlap_prop),
                   aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



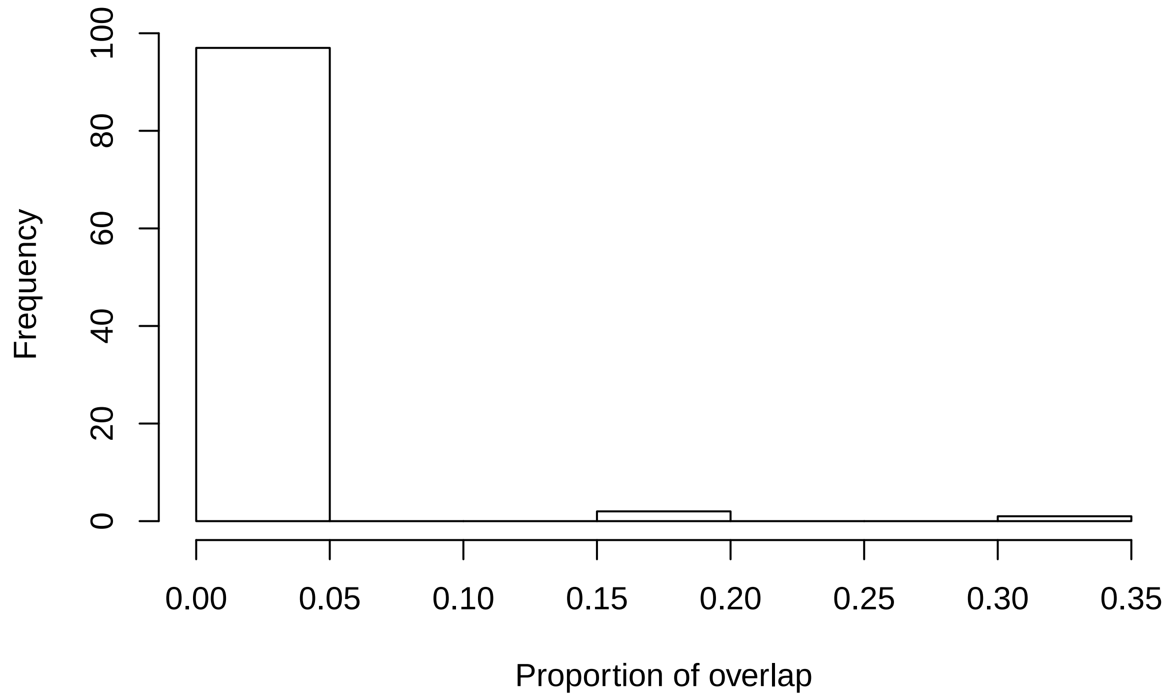
```r
ggsave("sessil_terrestrial.png", dpi=300, width=4, height=3)

#hdr(St_95.overlap_prop, h = 10)


#####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 = "")
```



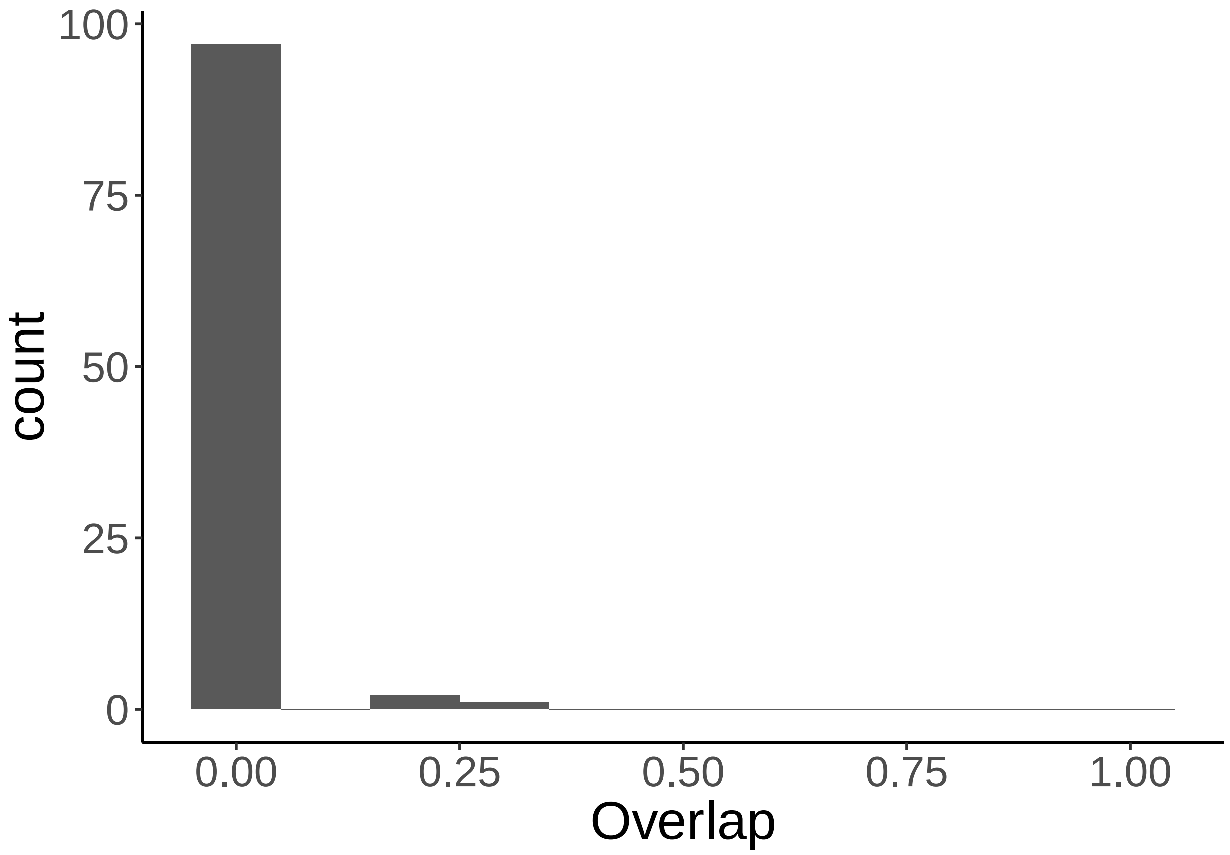
```
myplot_Sp = ggplot(data.frame(Overlap =  Sp_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



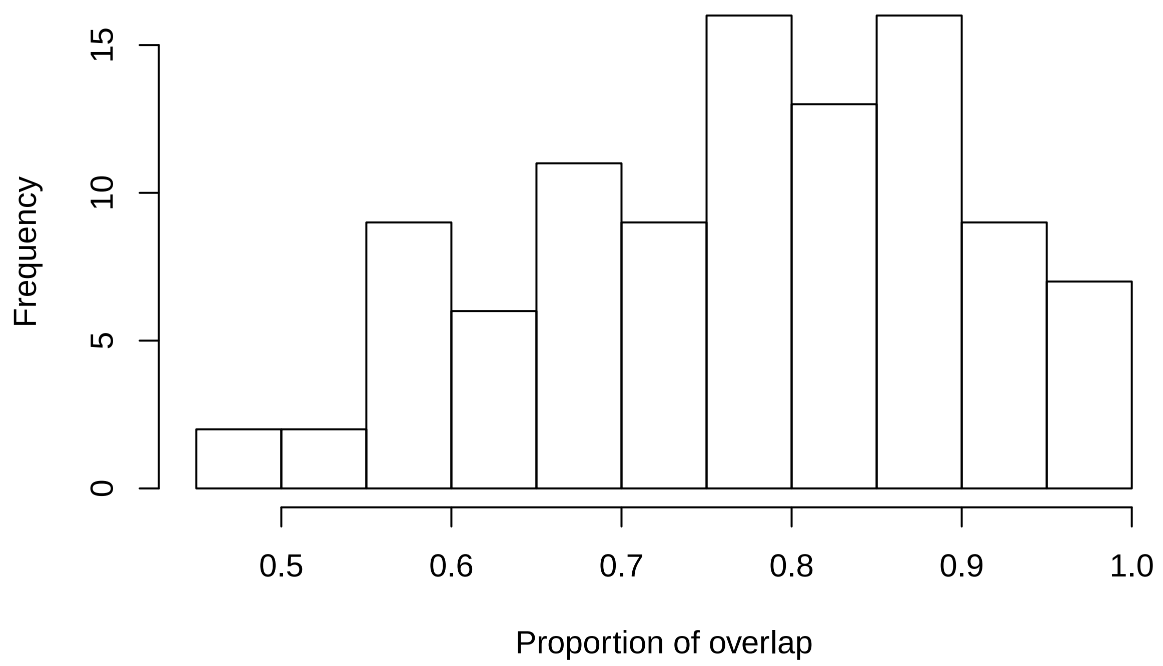
```
ggsave("sessil_pelagic.png", dpi=300, width=4, height=3)

#hdr(Sp_95.overlap_prop, h = 10)


#####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 = "")
```



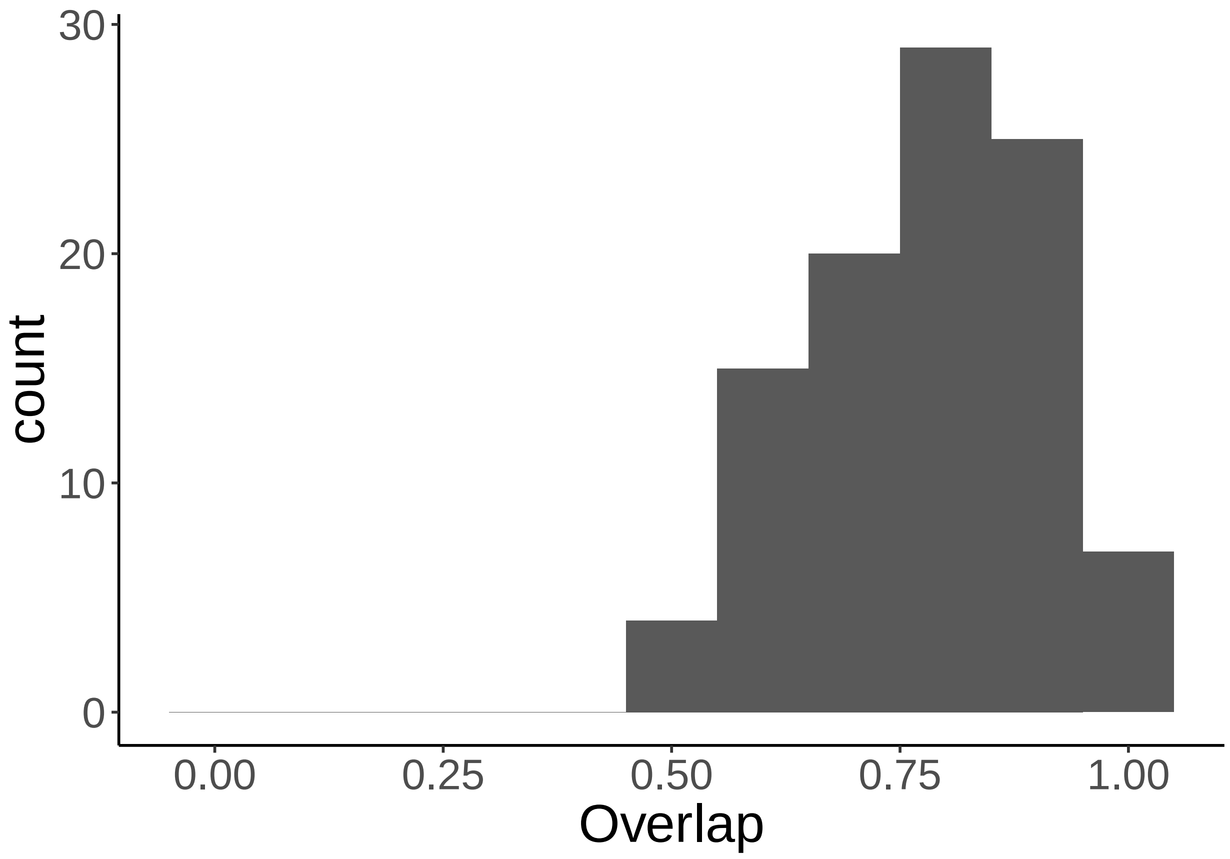
```
myplot_Ssf = ggplot(data.frame(Overlap =  Ssf_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



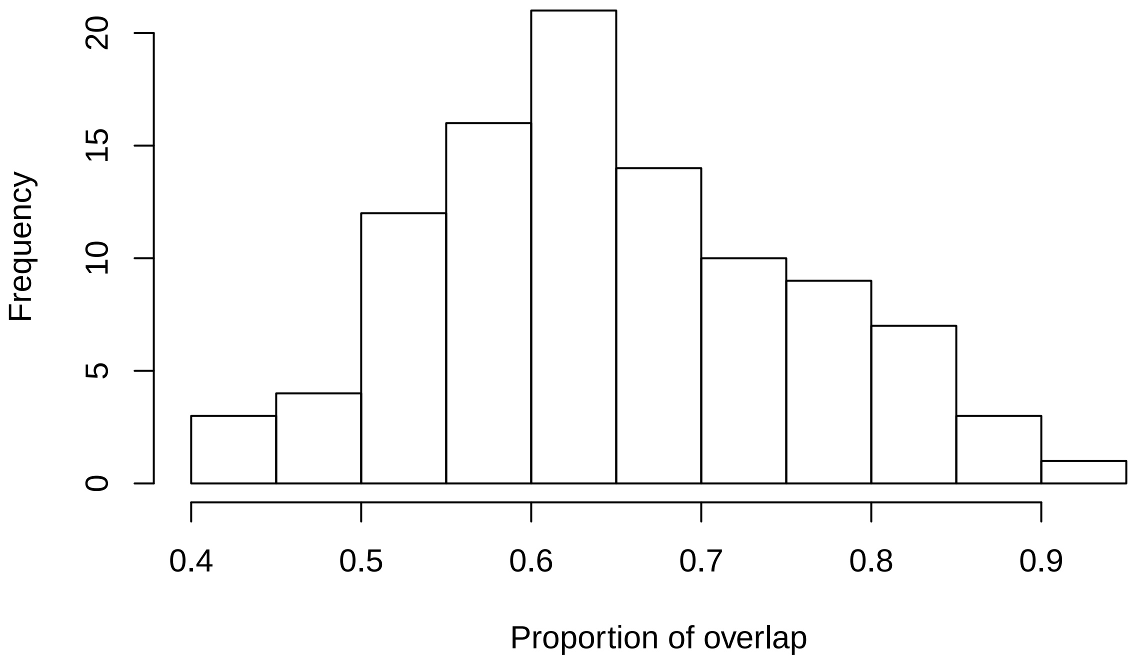
```
ggsave("sessil_semifoss.png", dpi=300, width=4, height=3)

#hdr(Ssf_95.overlap_prop, h = 10)


#####arboreal  - benthic
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 = "")
```



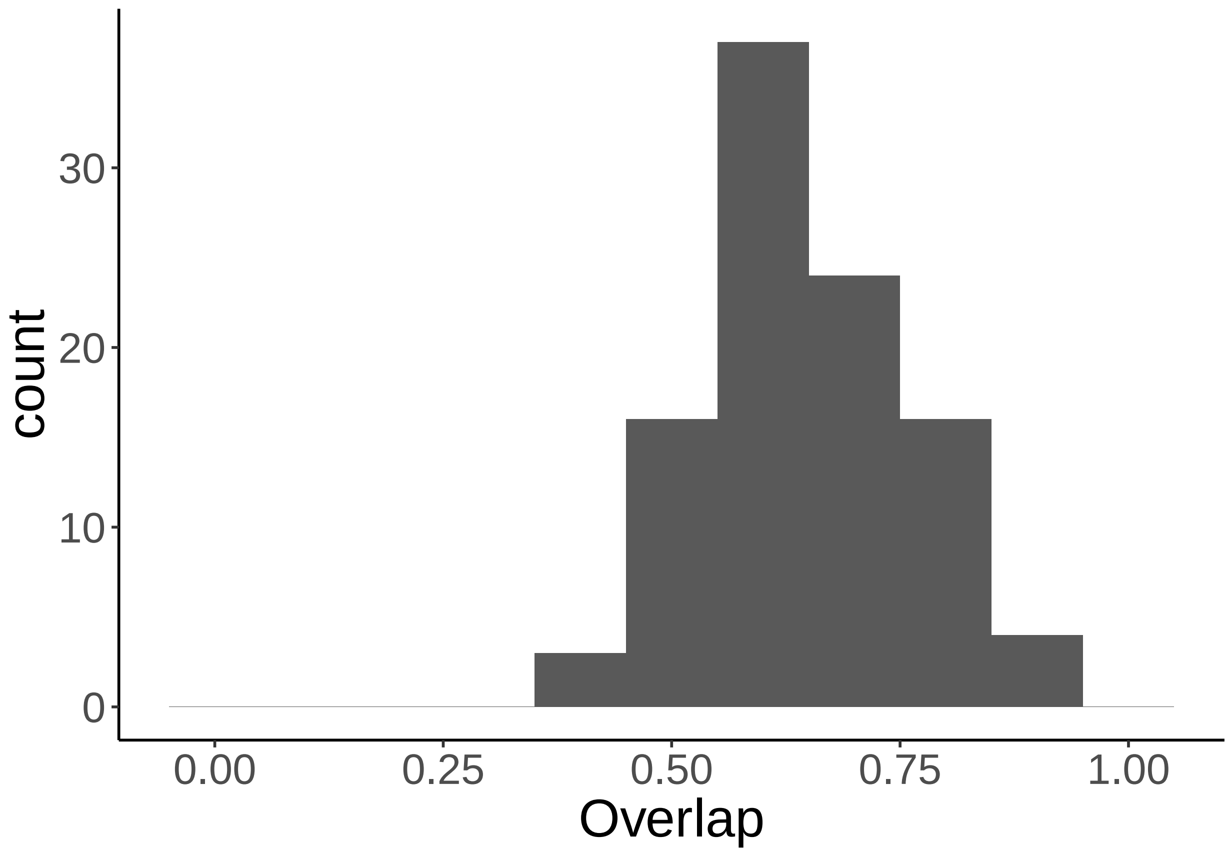
```
myplot_Ab = ggplot(data.frame(Overlap =  Ab_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



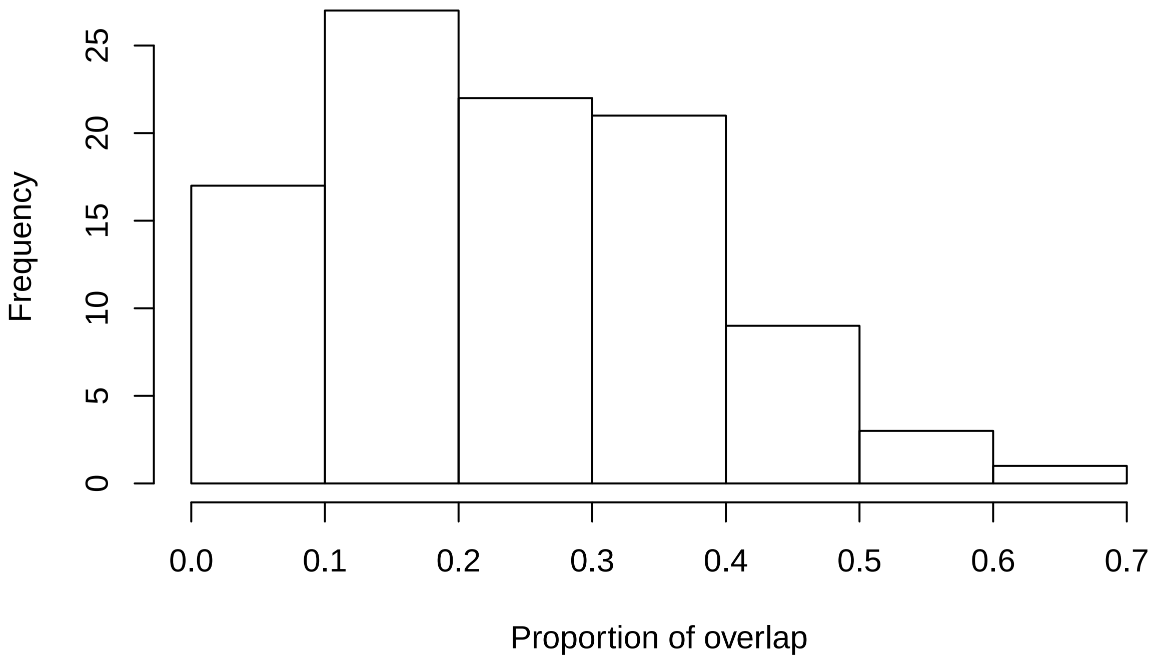
```
ggsave("arboreal_benthic.png", dpi=300, width=4, height=3)

#hdr(Ab_95.overlap_prop, h = 10)



#####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 = "")
```



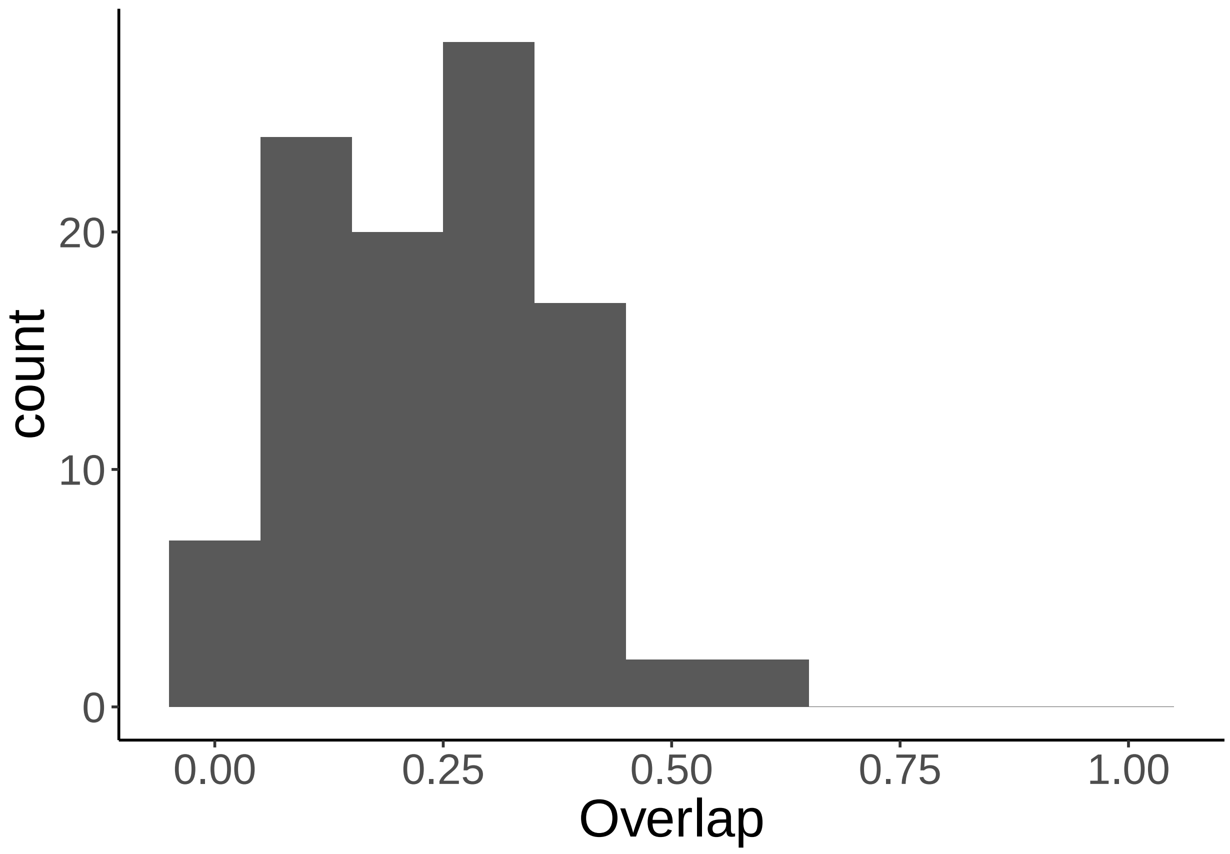
```
myplot_Av = ggplot(data.frame(Overlap =  Av_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



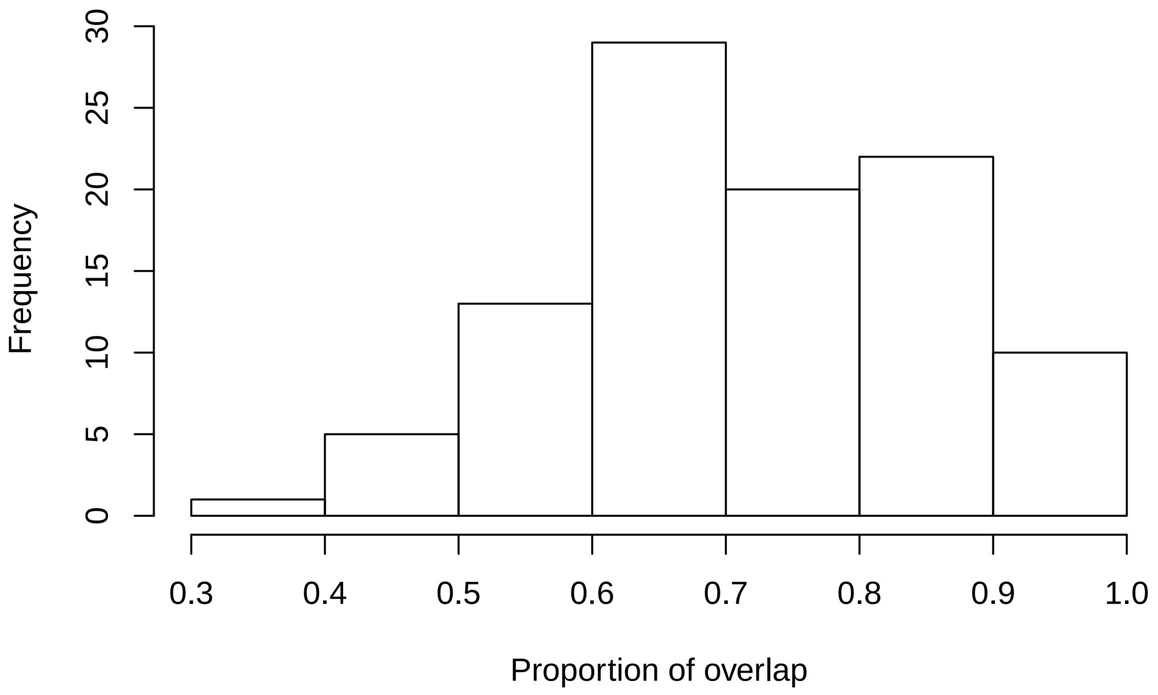
```
ggsave("arboreal_volant.png", dpi=300, width=4, height=3)

#hdr(Av_95.overlap_prop, h = 10)

#####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 = "")
```



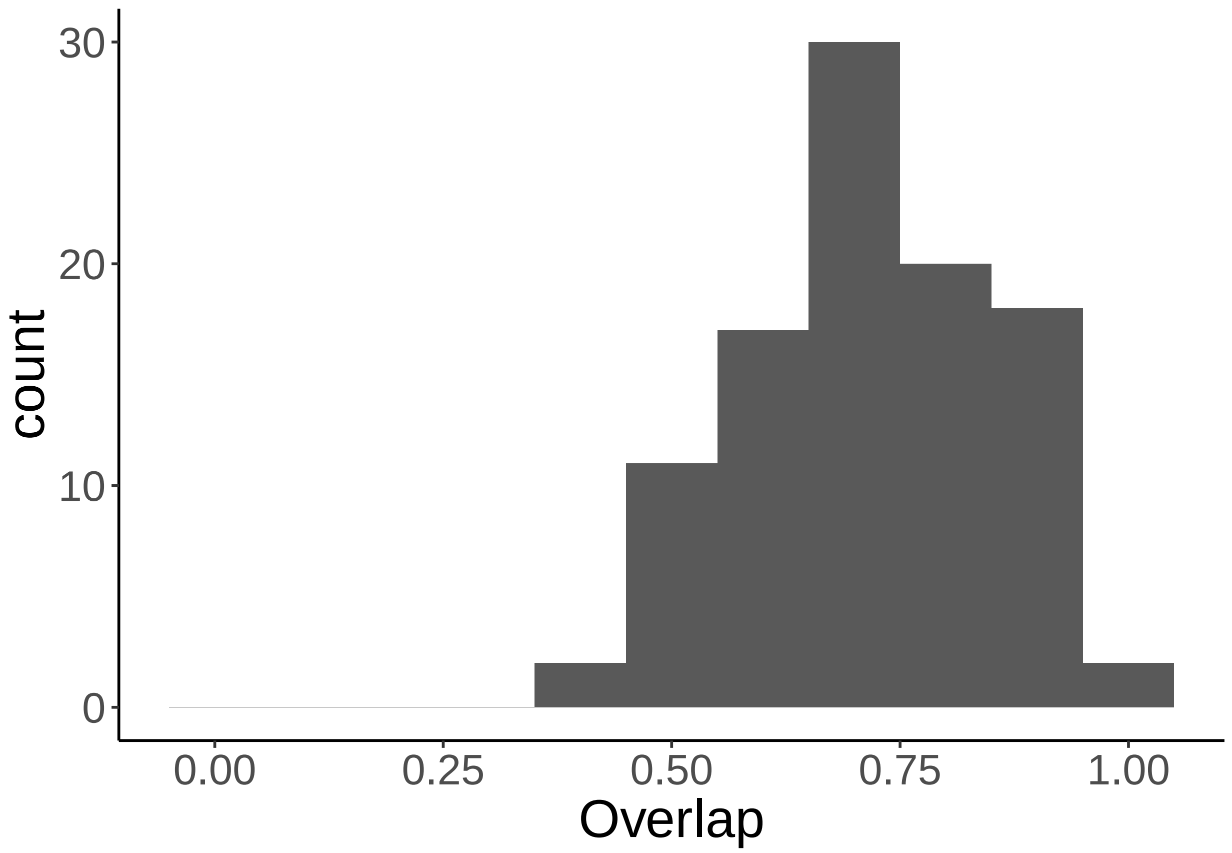
```r
myplot_Asemi = ggplot(data.frame(Overlap =  Asemi_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



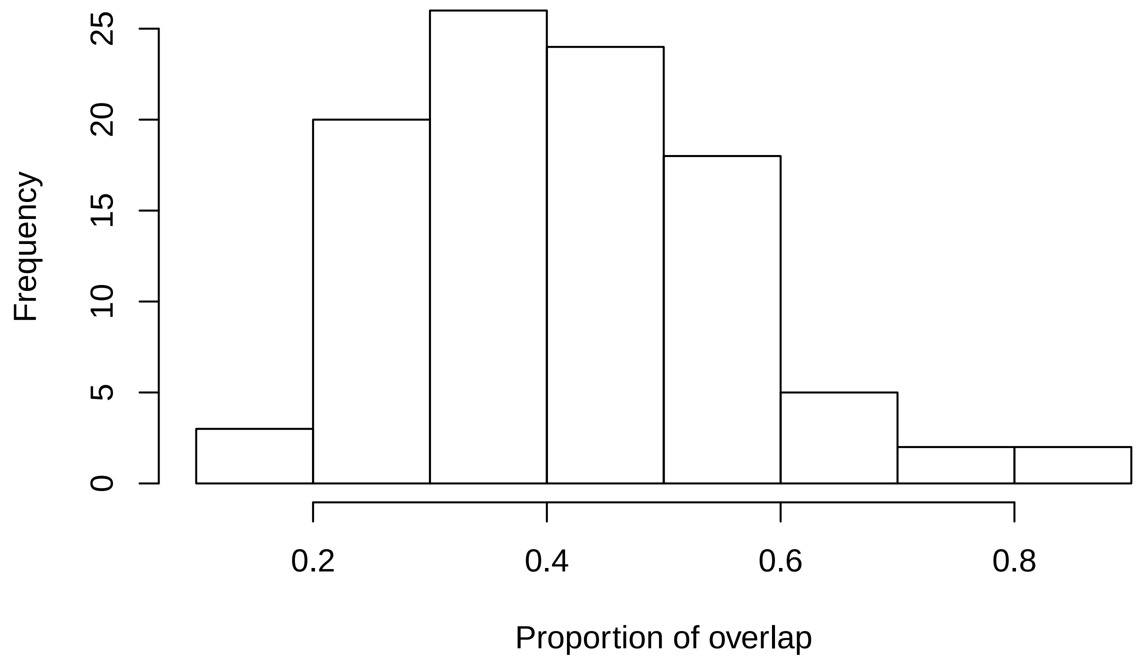
```r
ggsave("arboreal_semiaquatic.png", dpi=300, width=4, height=3)

#hdr(Asemi_95.overlap_prop, h = 10)

#####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 = "")
```



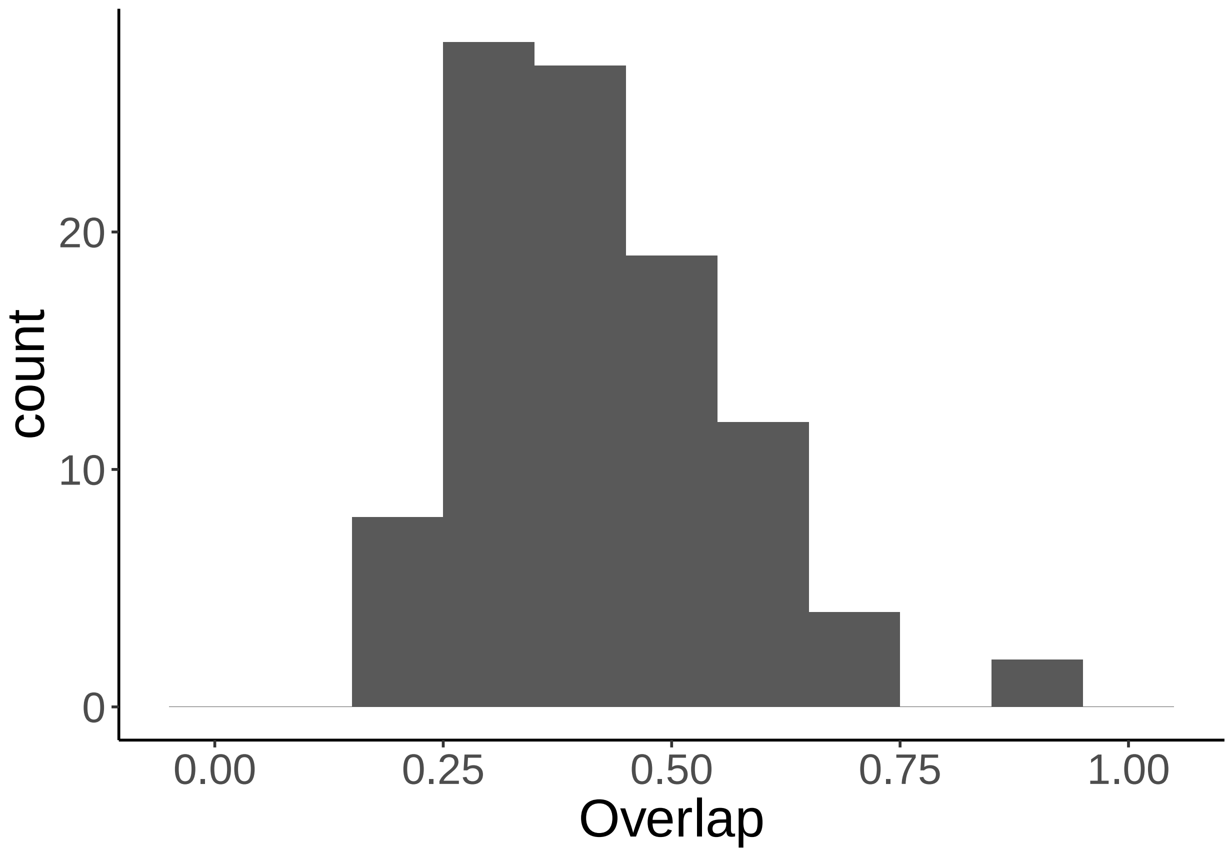
```r
myplot_At = ggplot(data.frame(Overlap =  At_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



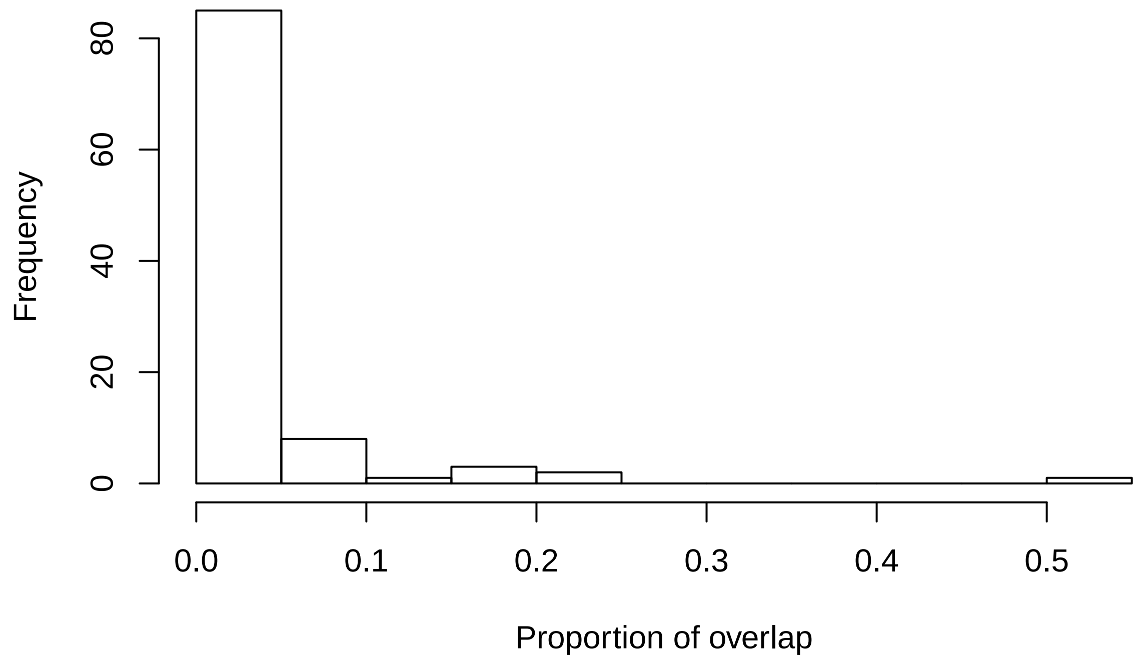
```
ggsave("arboreal_terrestrial.png", dpi=300, width=4, height=3)

#hdr(At_95.overlap_prop, h = 10)

#####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 = "")
```



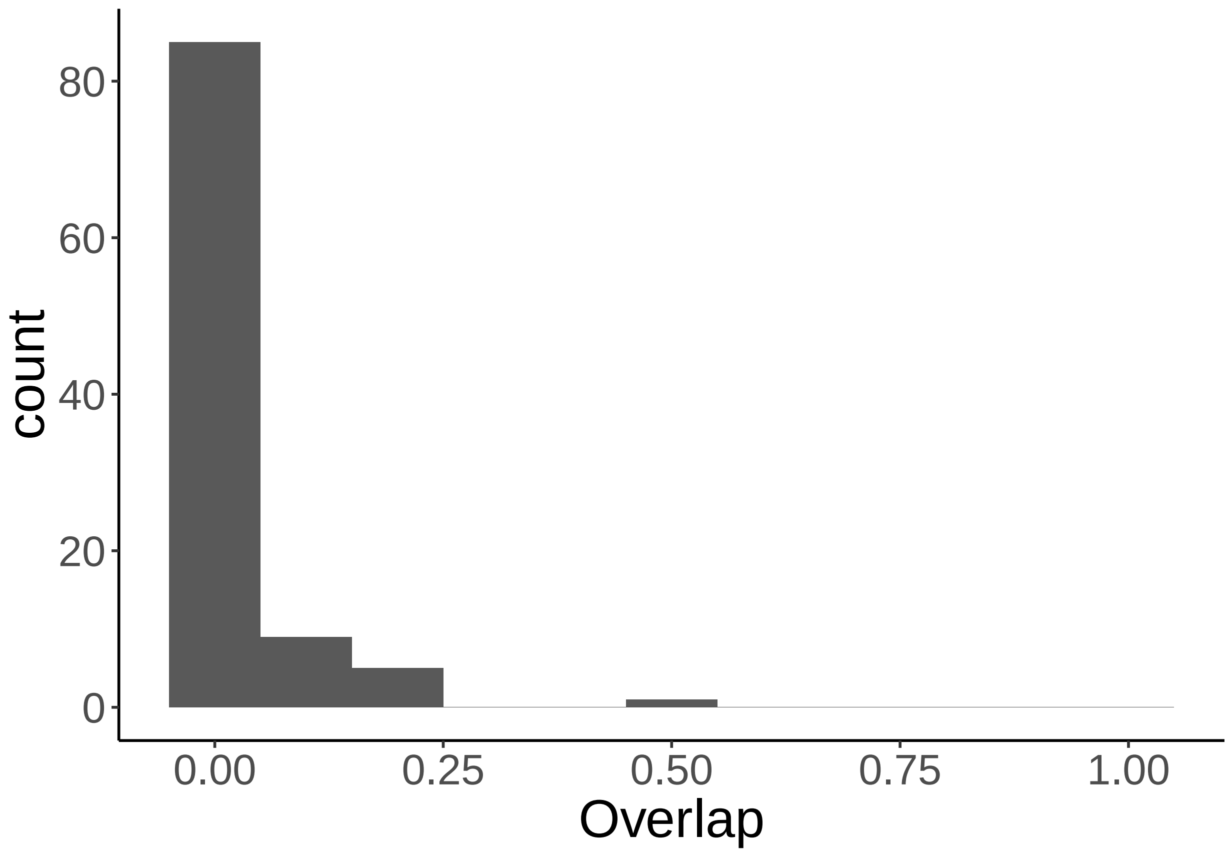
```
myplot_Ap = ggplot(data.frame(Overlap =  Ap_95.overlap_prop),
                   aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



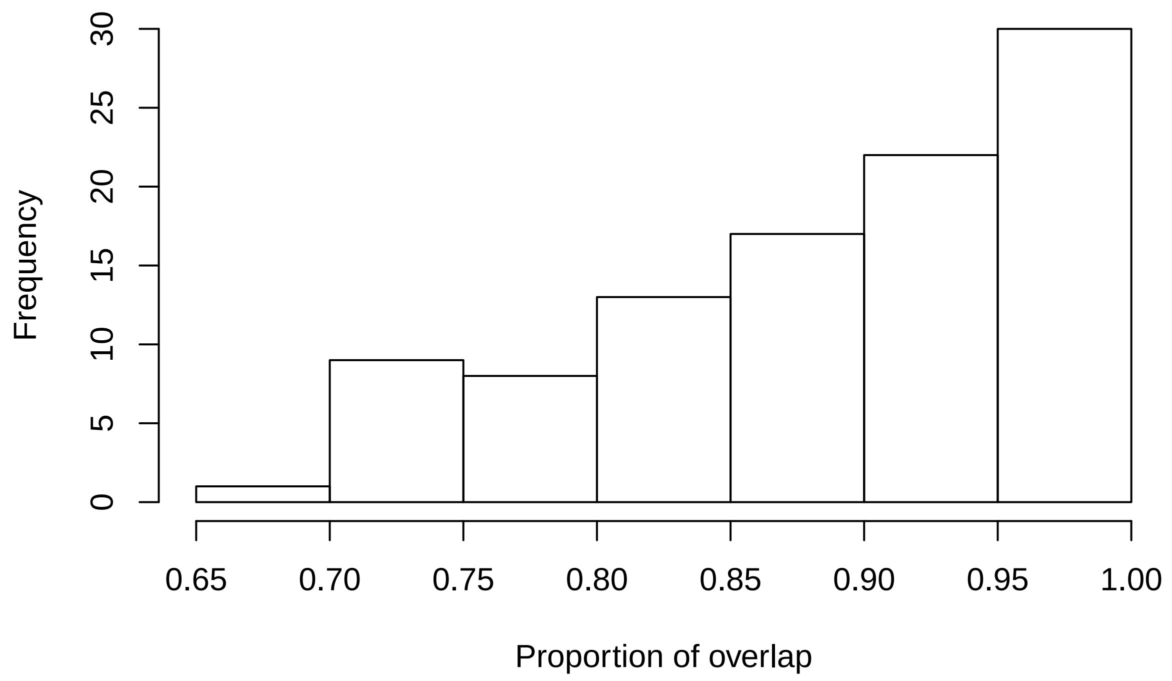
```r
ggsave("arboreal_pelegic.png", dpi=300, width=4, height=3)

#hdr(Ap_95.overlap_prop, h = 10)

#####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 = "")
```



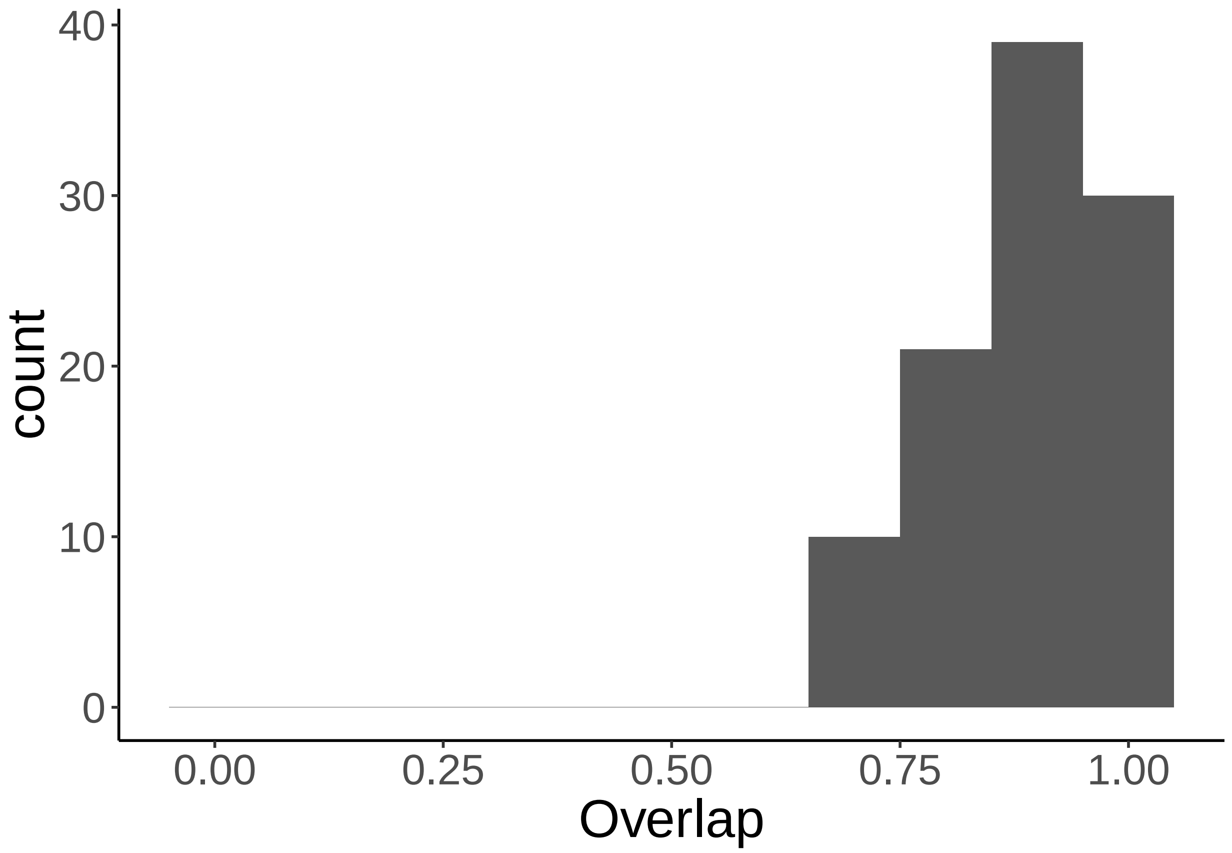
```
myplot_Afoss = ggplot(data.frame(Overlap =  Afoss_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



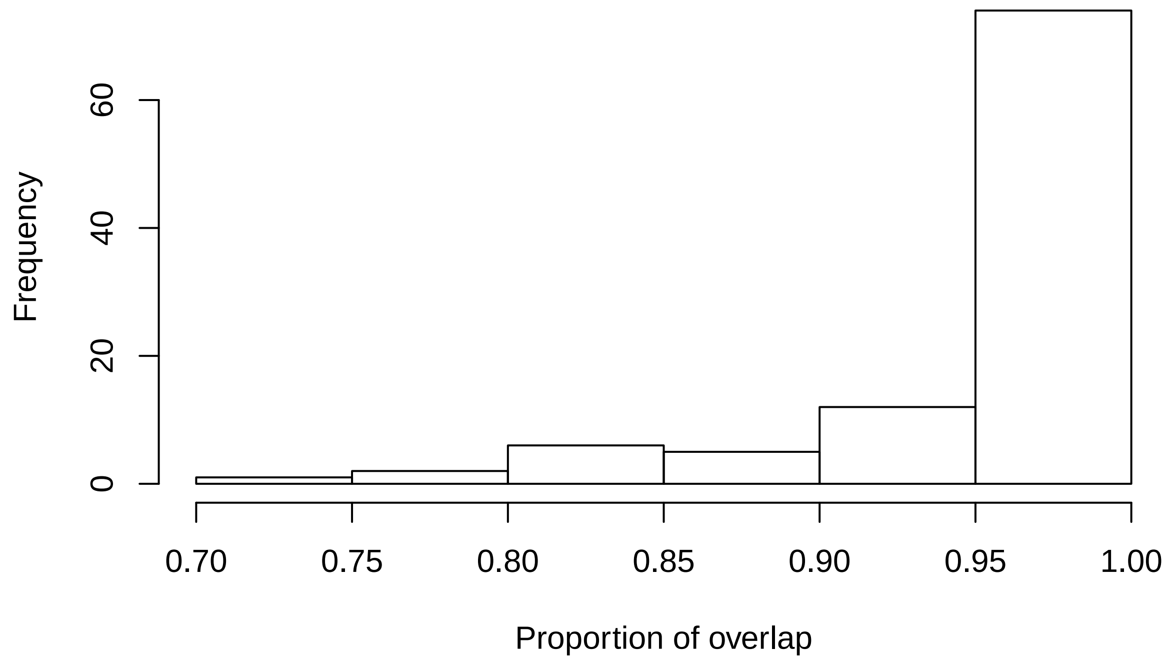
```r
ggsave("arboreal_semifossorial.png", dpi=300, width=4, height=3)

#hdr(Afoss_95.overlap_prop, h = 10)


#####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 = "")
```



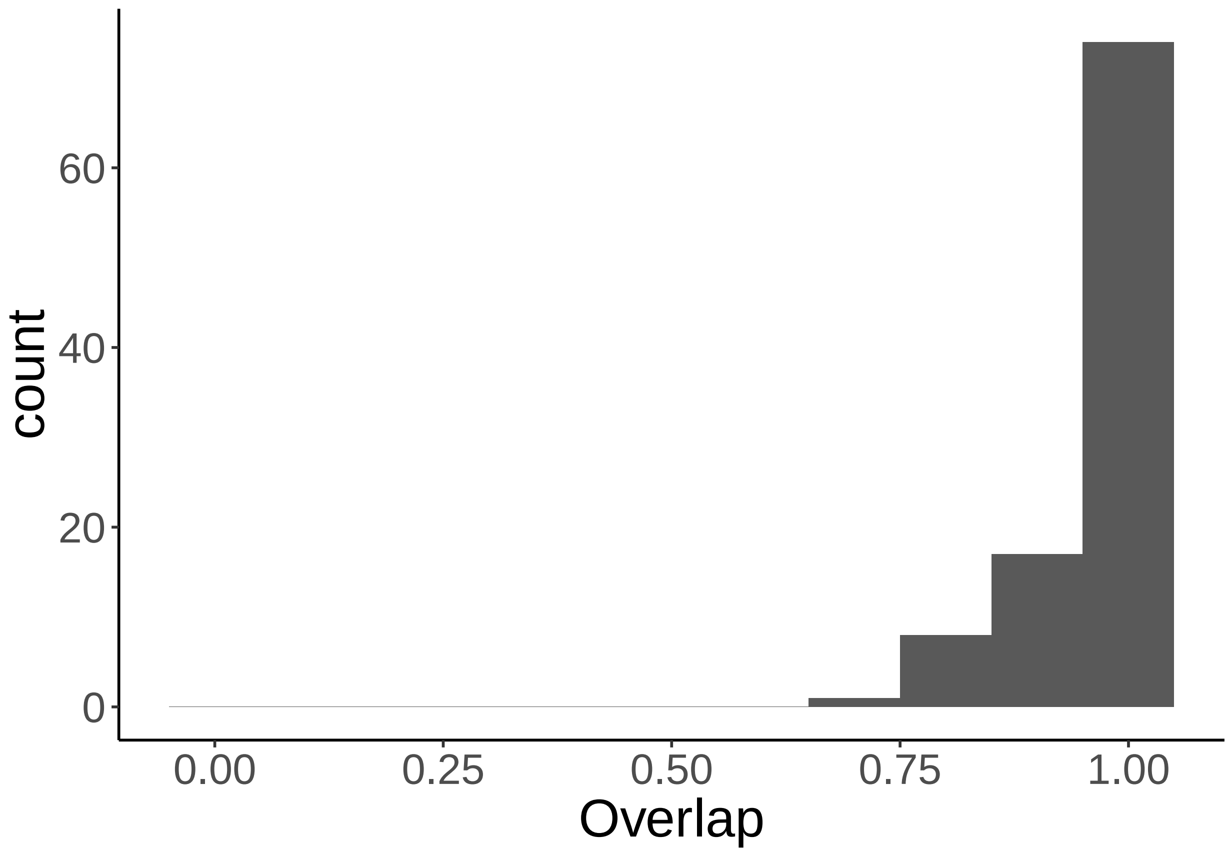
```
myplot_Bv = ggplot(data.frame(Overlap =  Bv_95.overlap_prop),
                   aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



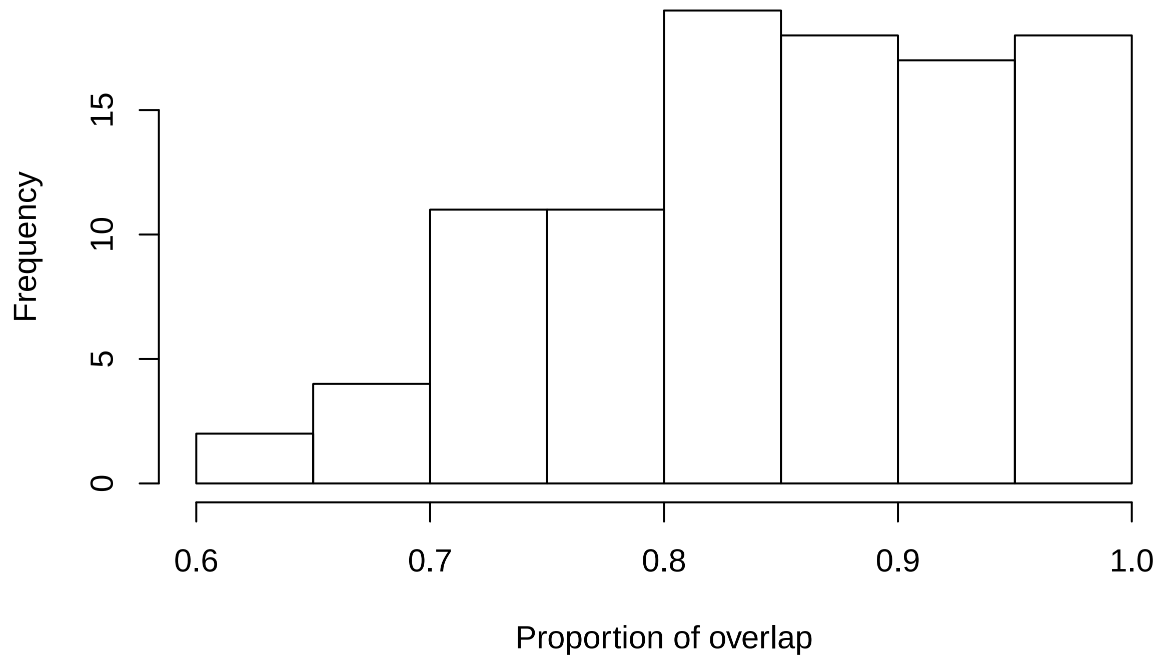
```
ggsave("benthic_volant.png", dpi=300, width=4, height=3)

#hdr(Bv_95.overlap_prop, h = 10)


#####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 = "")
```



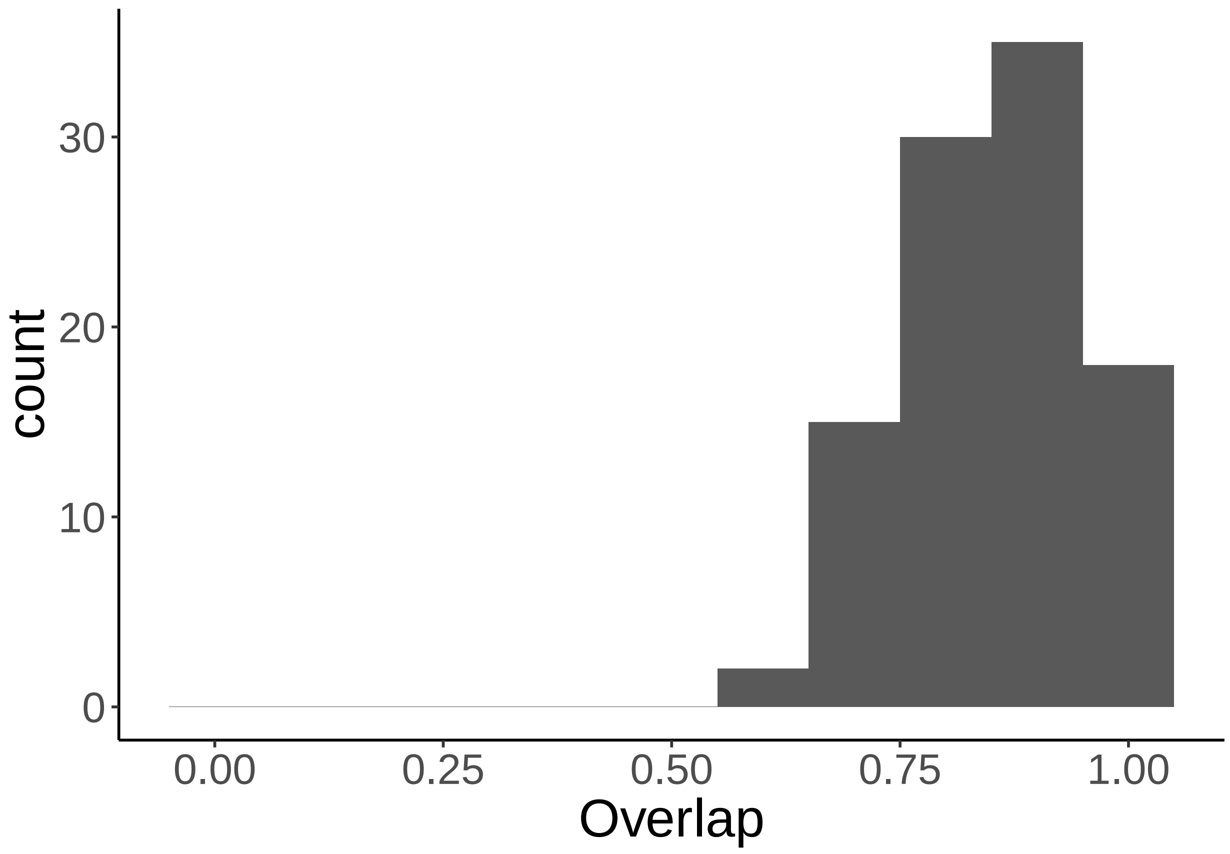
```
myplot_Bsemia = ggplot(data.frame(Overlap =  Bsemia_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



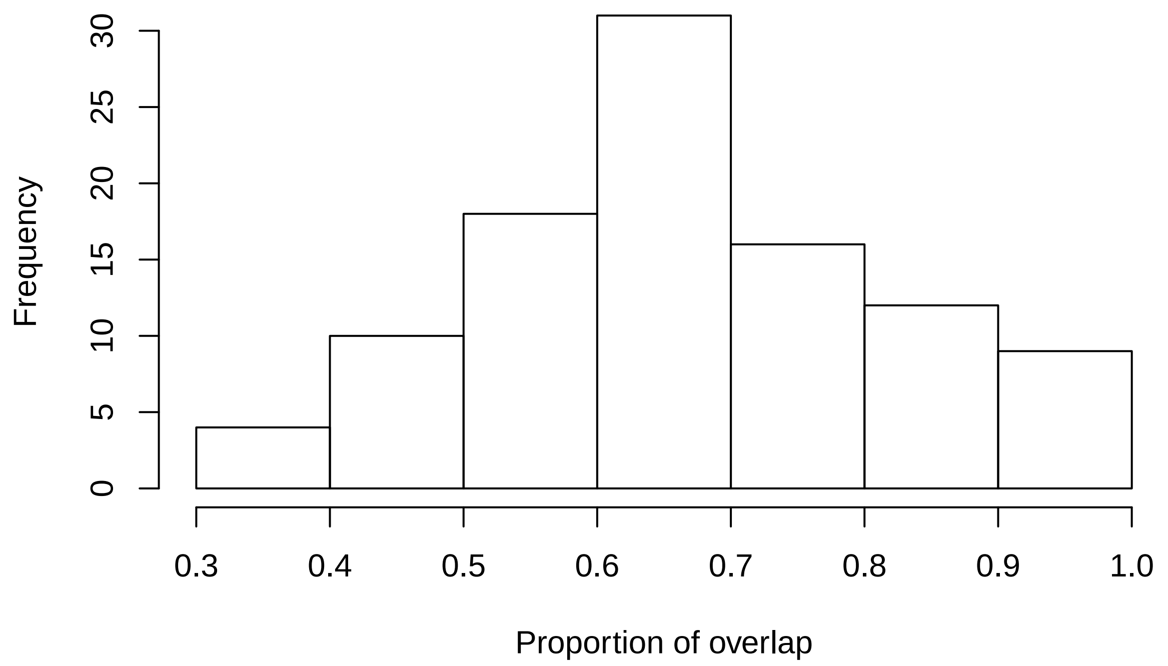
```r
ggsave("benthic_semiaquatic.png", dpi=300, width=4, height=3)

#hdr(Bsemia_95.overlap_prop, h = 10)



#####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 = "")
```



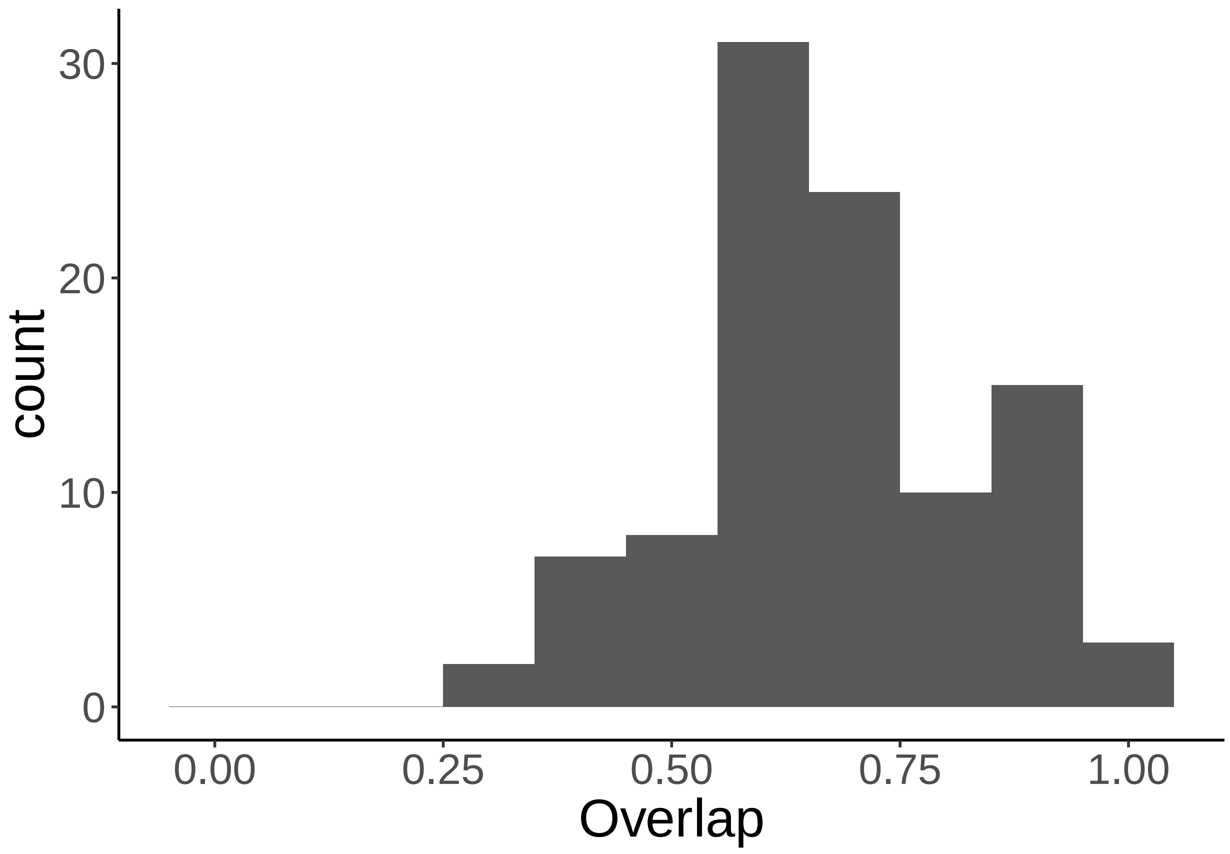
```
myplot_Bt = ggplot(data.frame(Overlap =  Bt_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



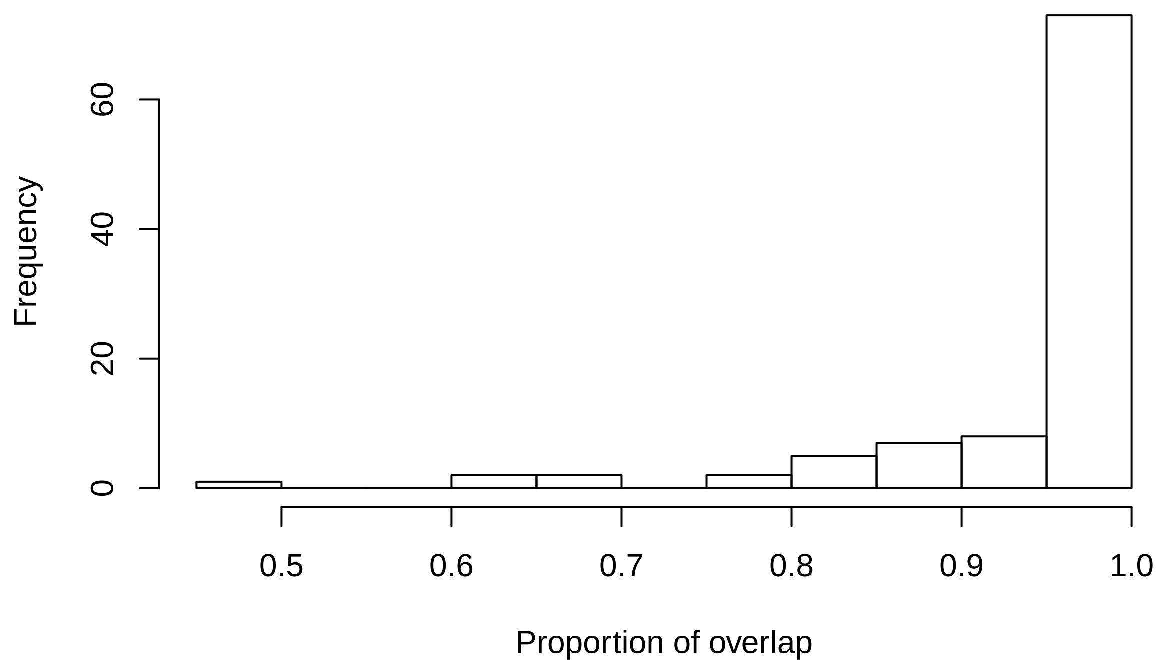
```
ggsave("benthic_terrestrial.png", dpi=300, width=4, height=3)

#hdr(Bt_95.overlap_prop, h = 10)




#####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 = "")
```



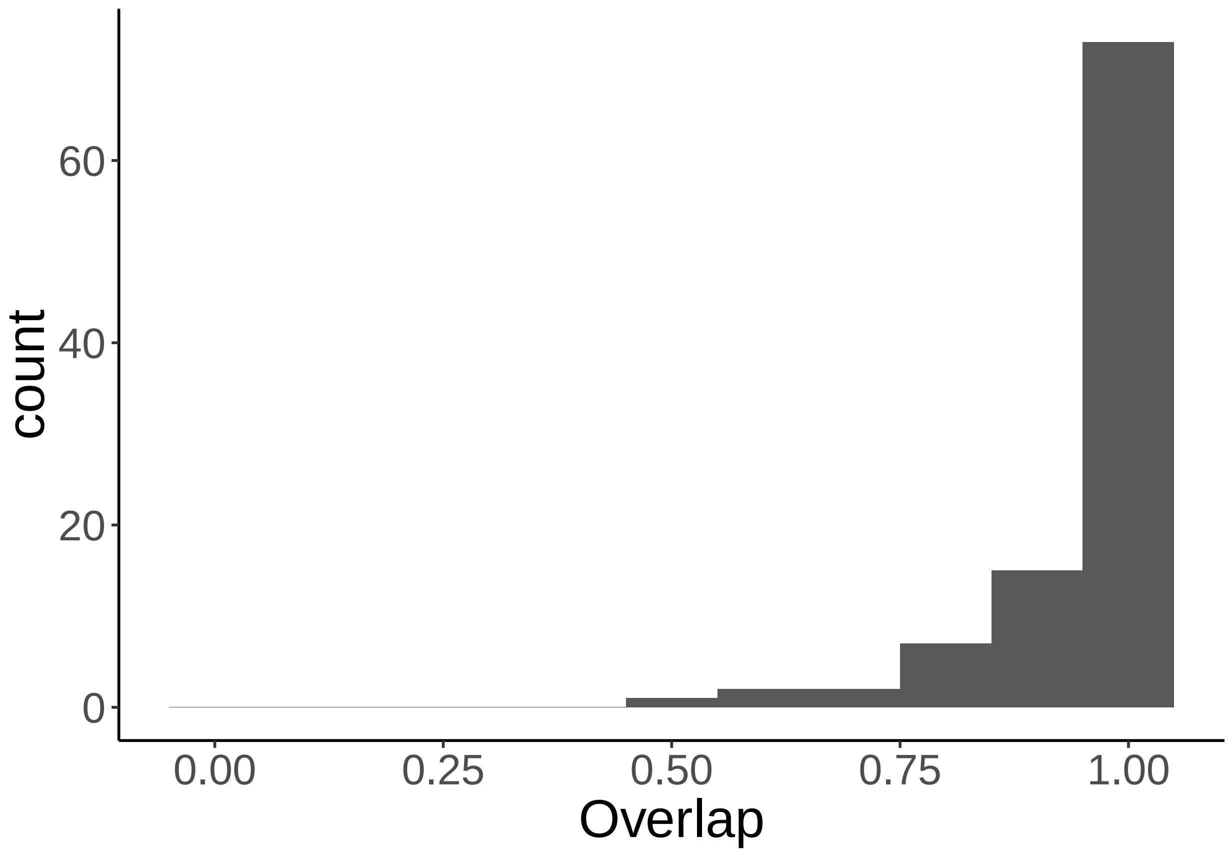
```
myplot_Bp = ggplot(data.frame(Overlap =  Bp_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



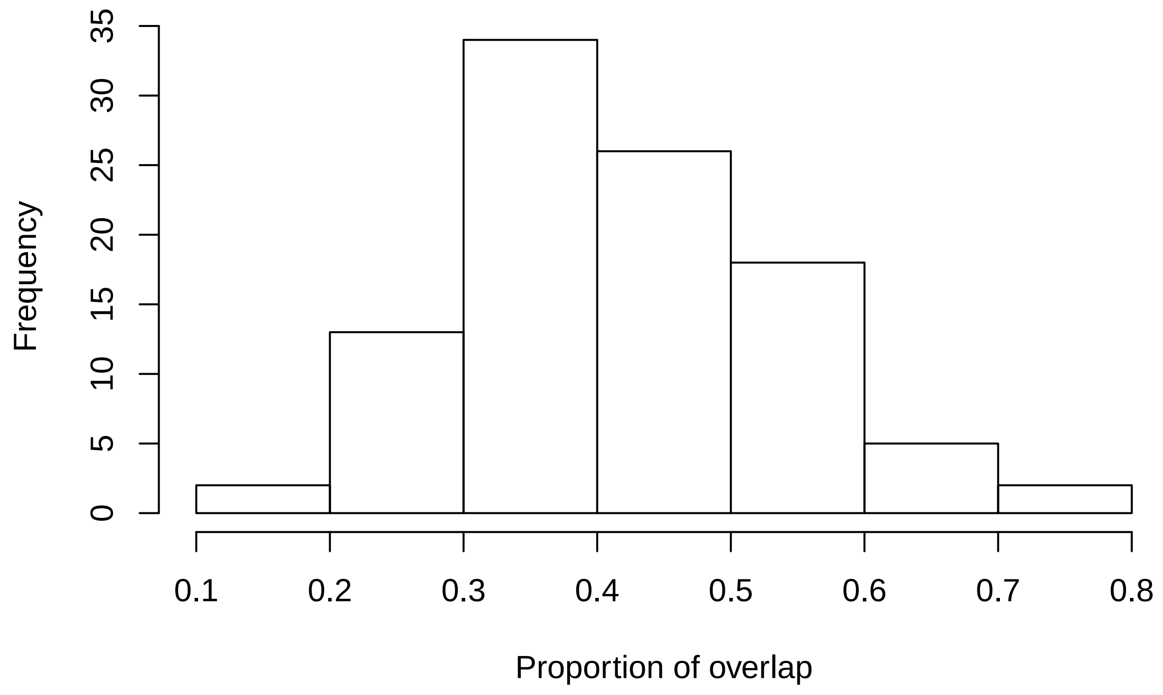
```
ggsave("benthic_pelagic.png", dpi=300, width=4, height=3)

#hdr(Bp_95.overlap_prop, h = 10)



#####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 = "")
```



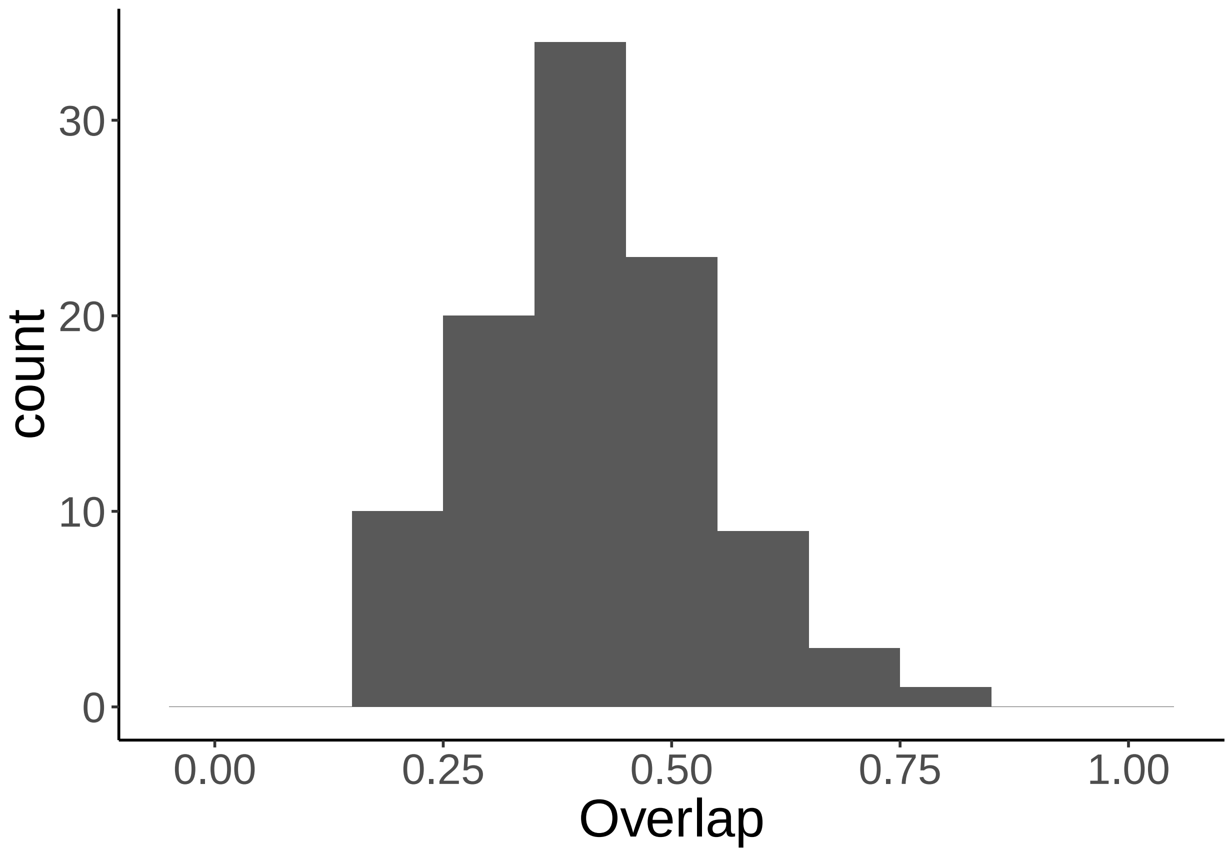
```
myplot_Bfoss = ggplot(data.frame(Overlap =  Bfoss_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



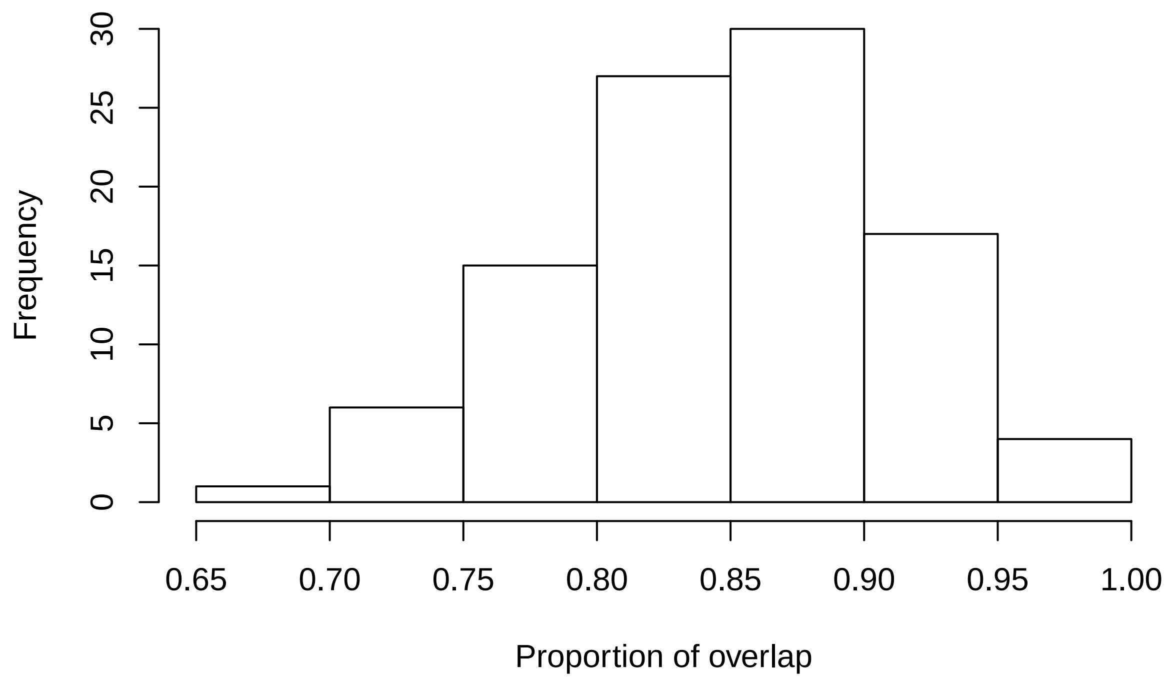
```r
ggsave("benthic_semifoss.png", dpi=300, width=4, height=3)

#hdr(Bfoss_95.overlap_prop, h = 10)



#####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 = "")
```



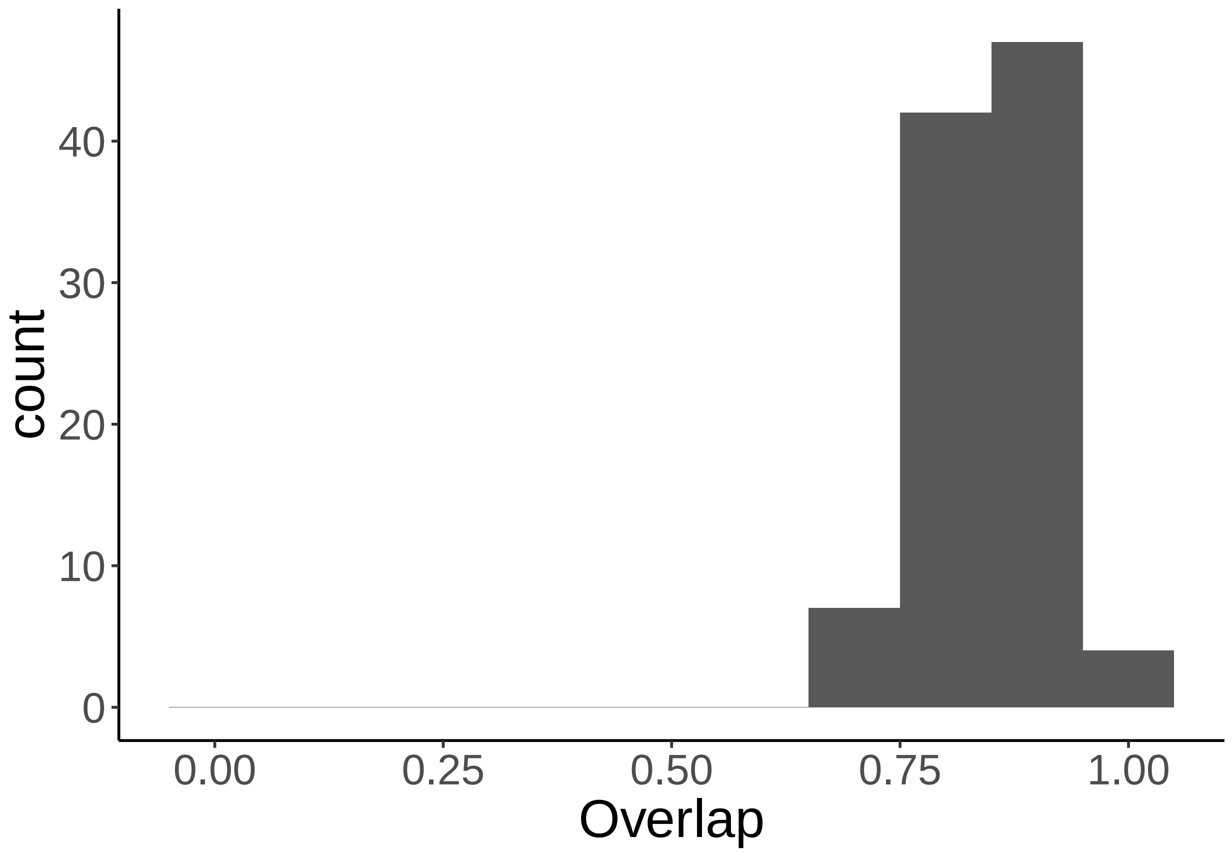
```
myplot_Vsem = ggplot(data.frame(Overlap =  Vsem_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



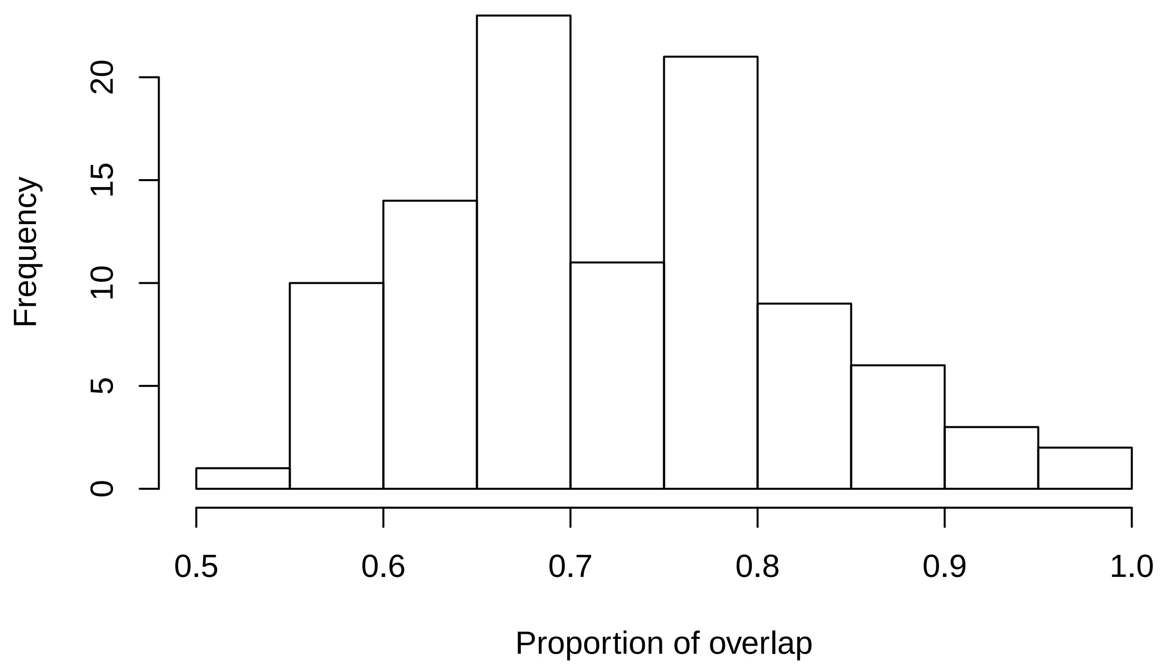
```
ggsave("volant_semiaquatic.png", dpi=300, width=4, height=3)

#hdr(Vsem_95.overlap_prop, h = 10)



#####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 = "")
```



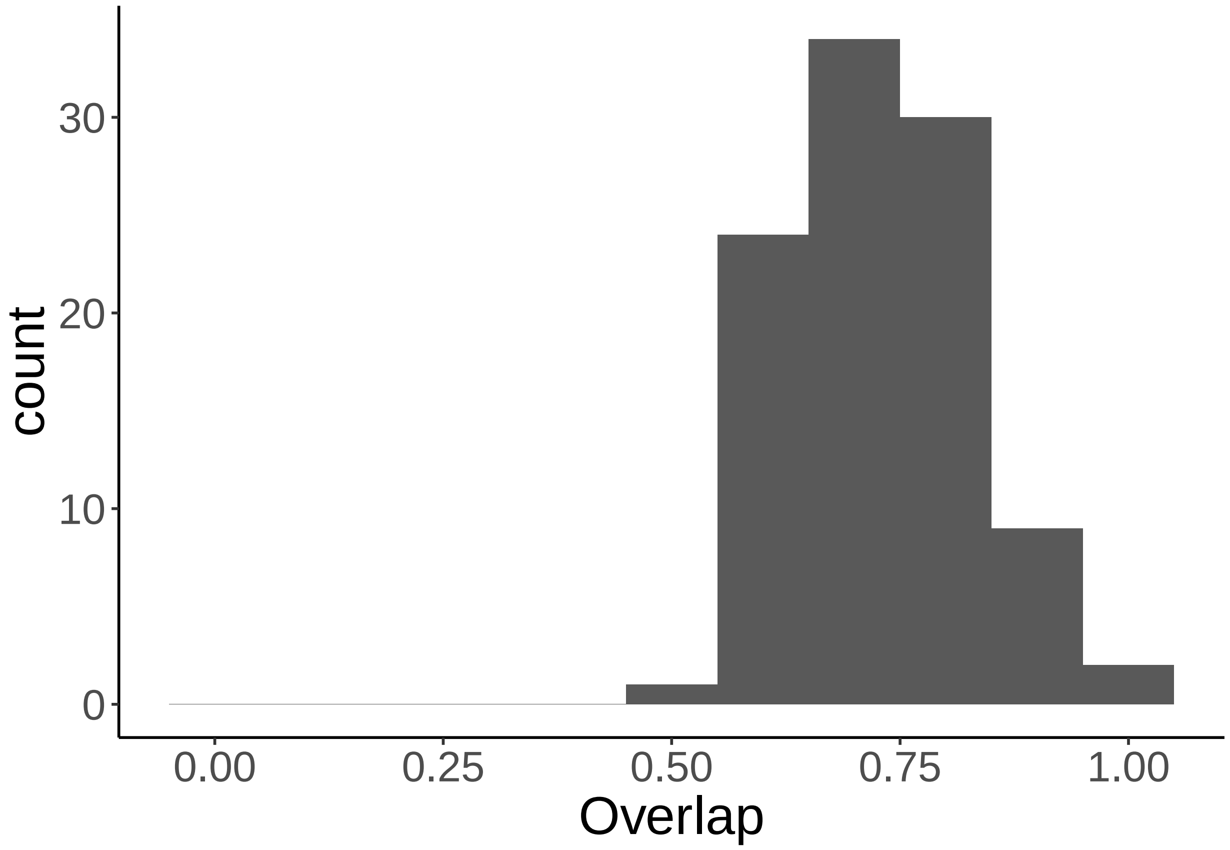
```r
myplot_Vt = ggplot(data.frame(Overlap =  Vt_95.overlap_prop),
                   aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



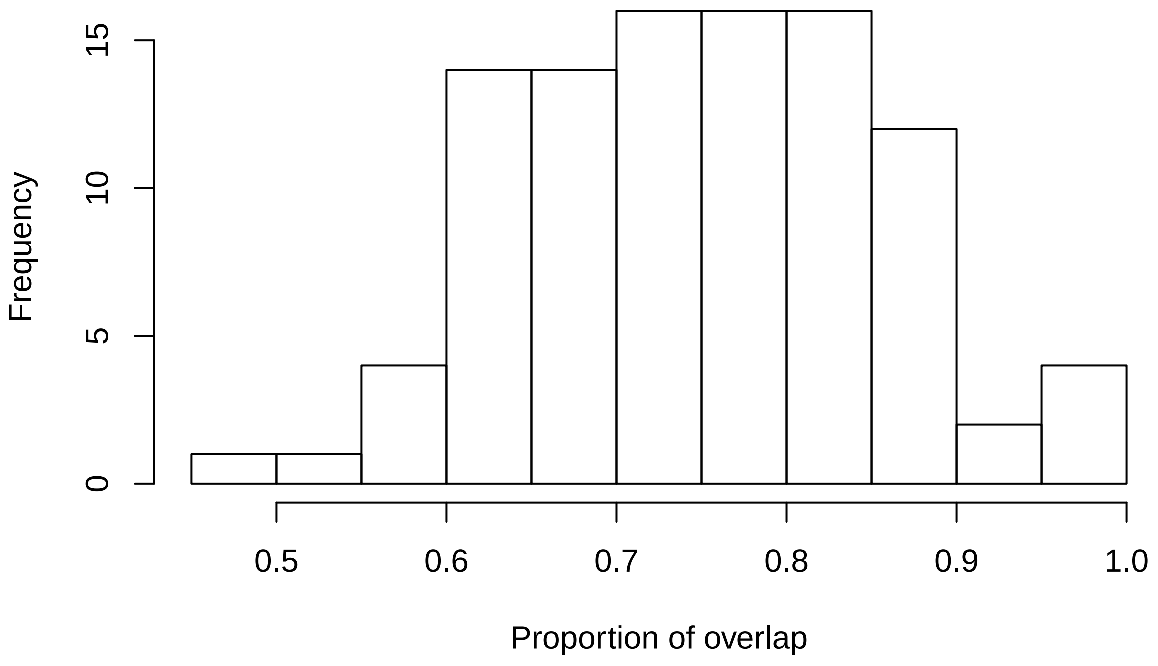
```r
ggsave("volant_terrestrial.png", dpi=300, width=4, height=3)

#hdr(Vt_95.overlap_prop, h = 10)



#####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 = "")
```



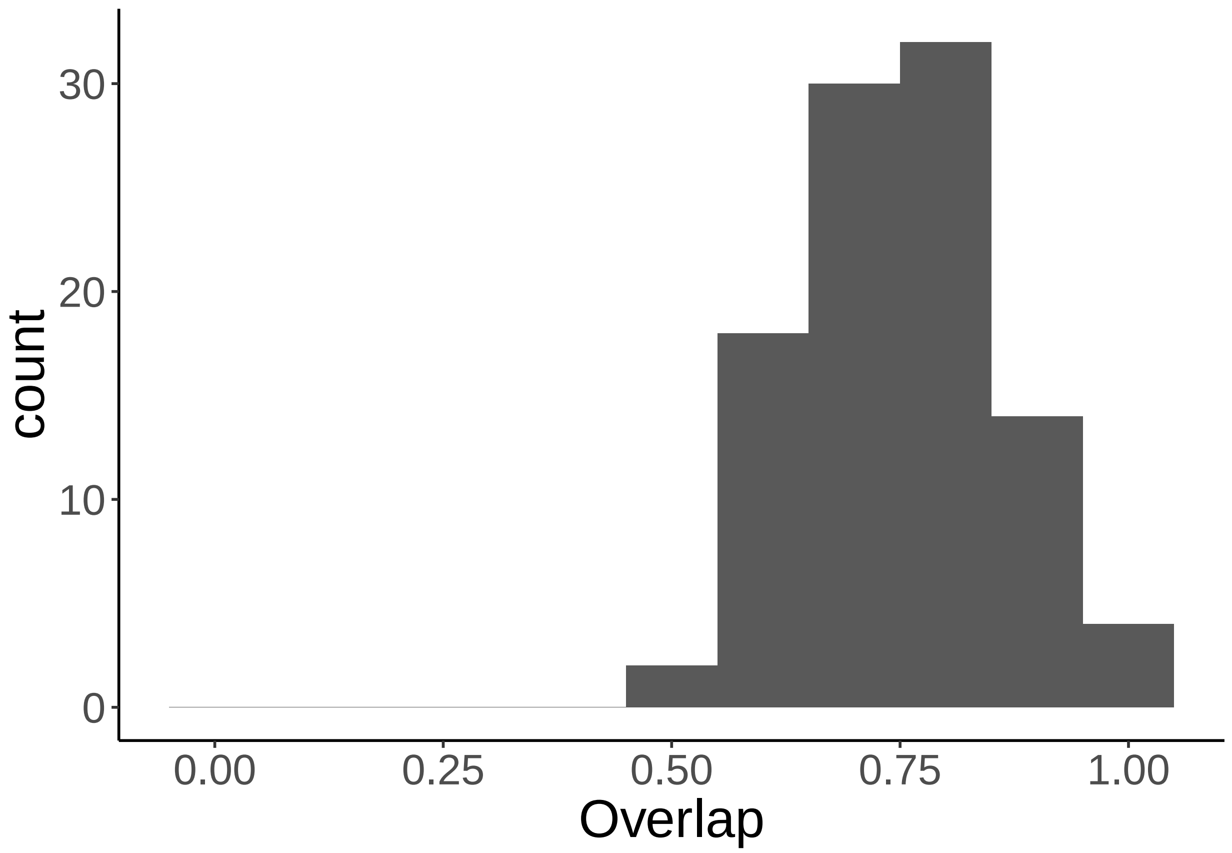
```
myplot_Vp = ggplot(data.frame(Overlap =  Vp_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



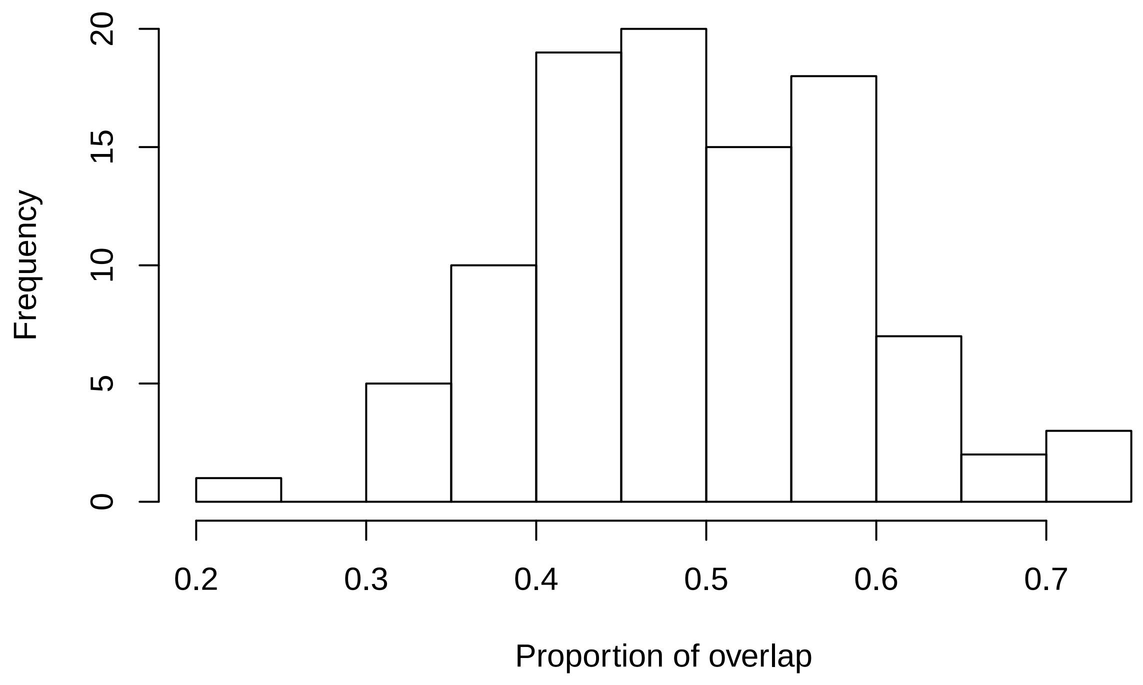
```
ggsave("volant_pelegic.png", dpi=300, width=4, height=3)

#hdr(Vp_95.overlap_prop, h = 10)



#####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 = "")
```



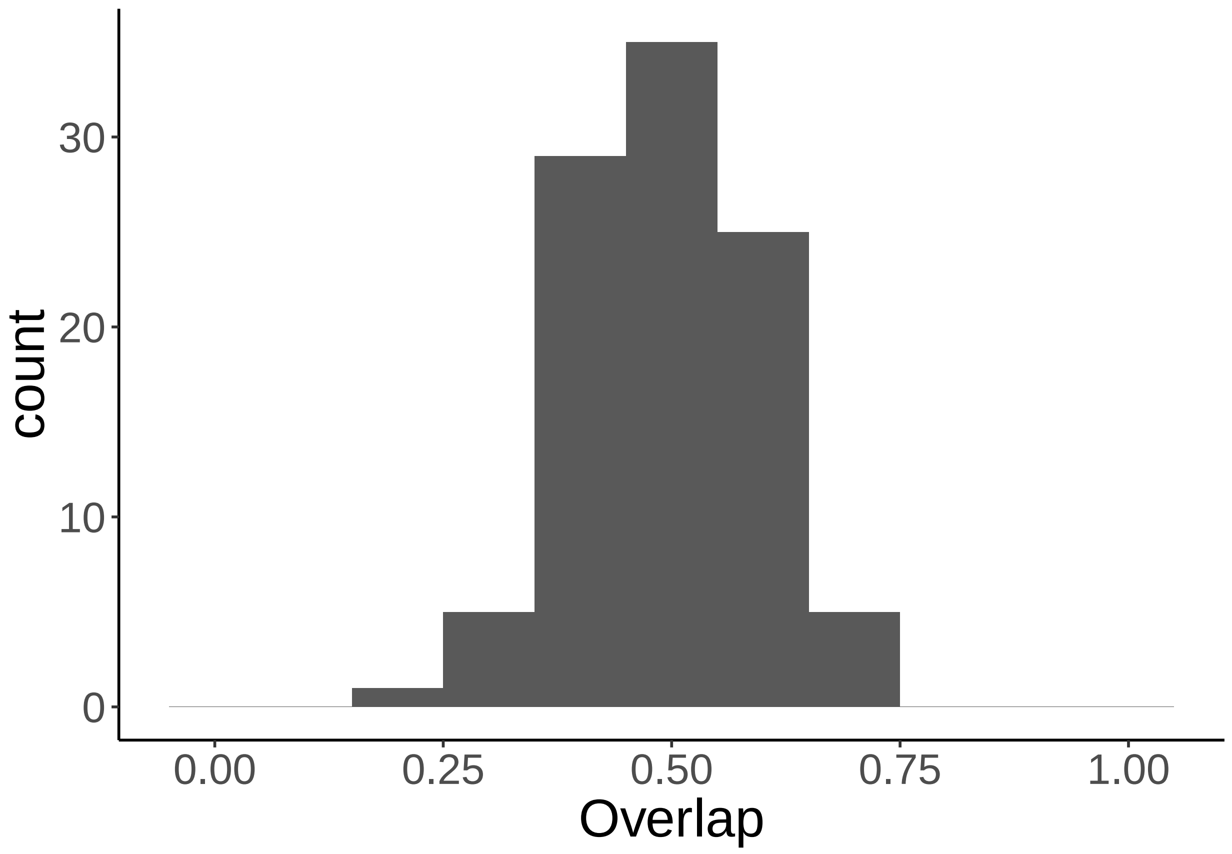
```
myplot_Vsfoss = ggplot(data.frame(Overlap = Vsfoss_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



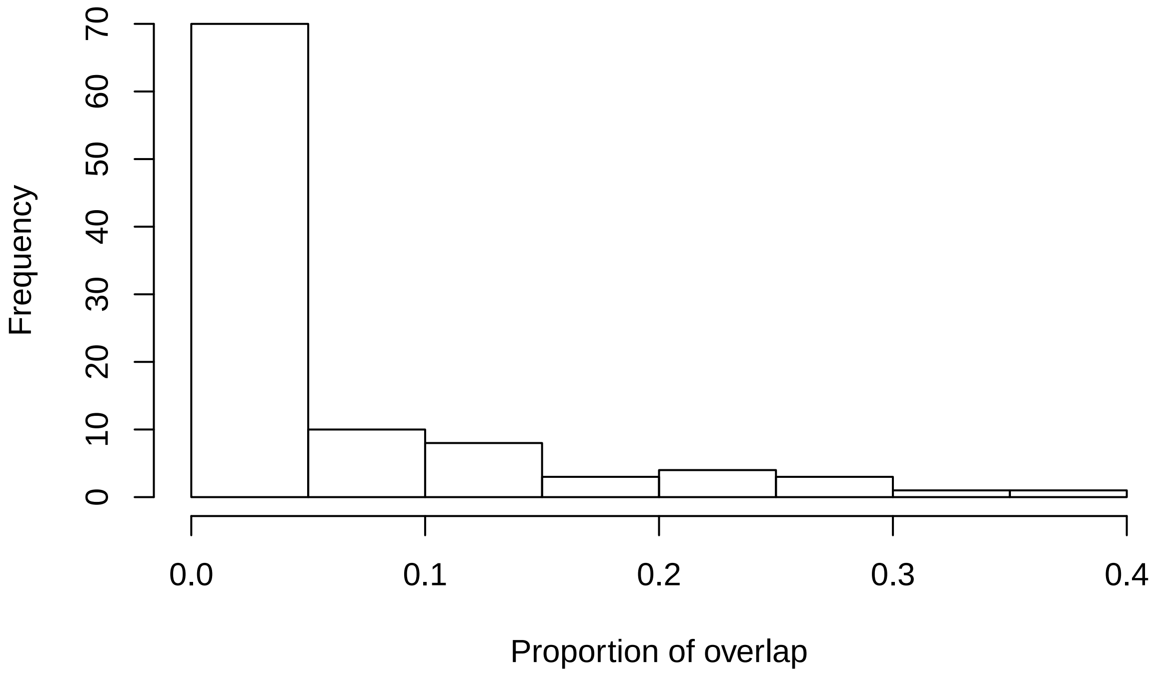
```r
ggsave("volant_semifoss.png", dpi=300, width=4, height=3)

#hdr(Vsfoss_95.overlap_prop, h = 10)


#####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 = "")
```



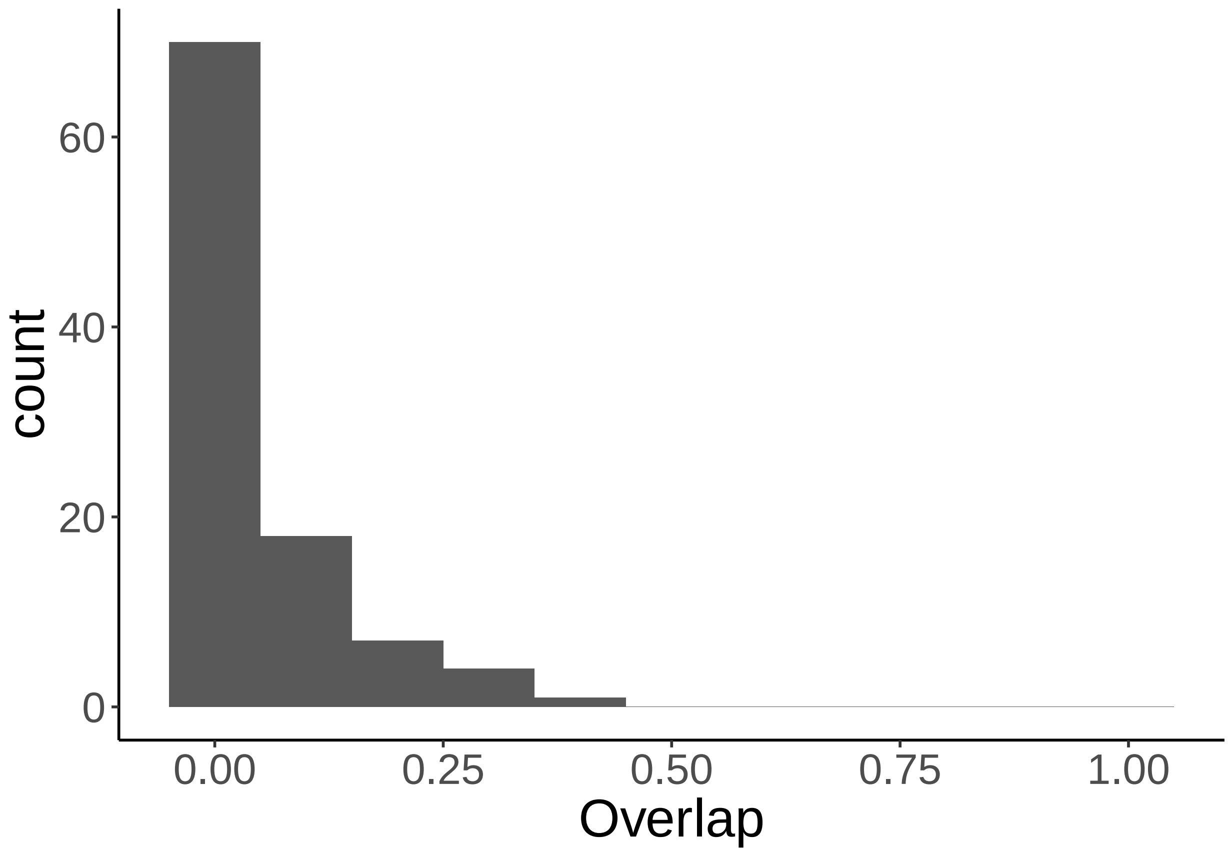
```r
myplot_SAt = ggplot(data.frame(Overlap = SAt_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



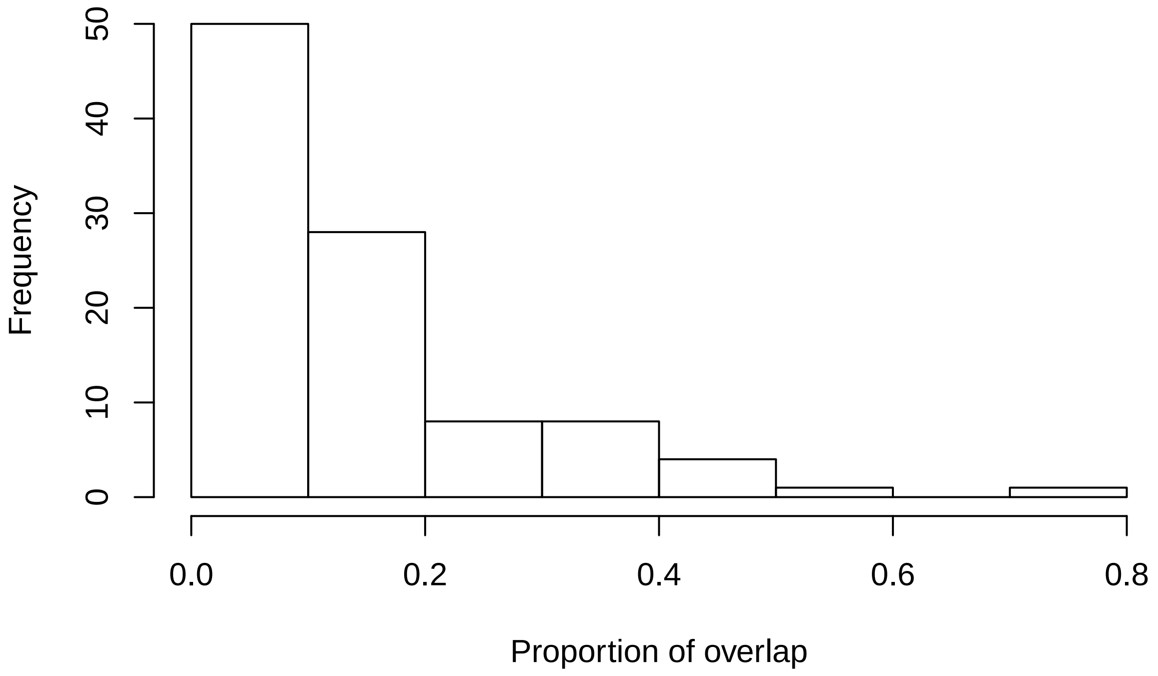
```
ggsave("semiaqu_terrestrial.png", dpi=300, width=4, height=3)

#hdr(SAt_95.overlap_prop, h = 10)


#####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 = "")
```



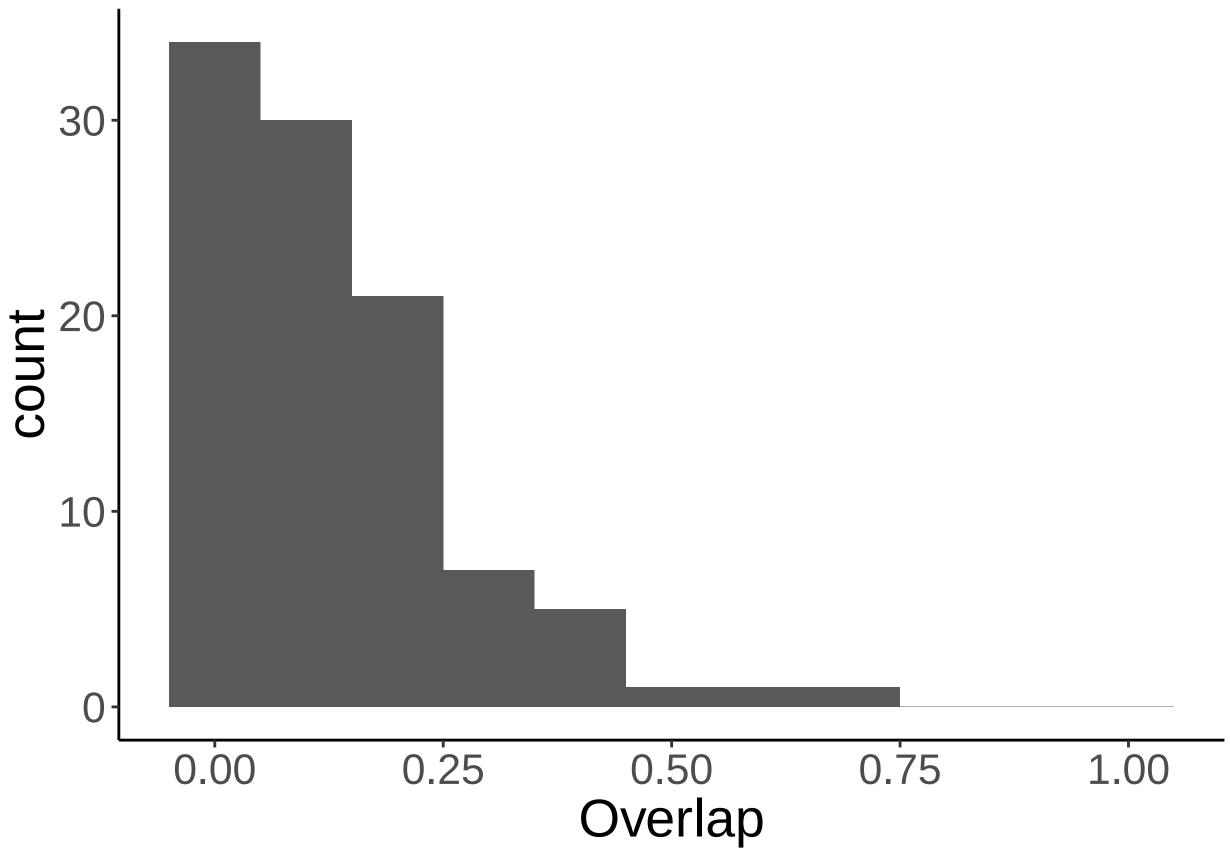
```
myplot_SAp = ggplot(data.frame(Overlap = SAp_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



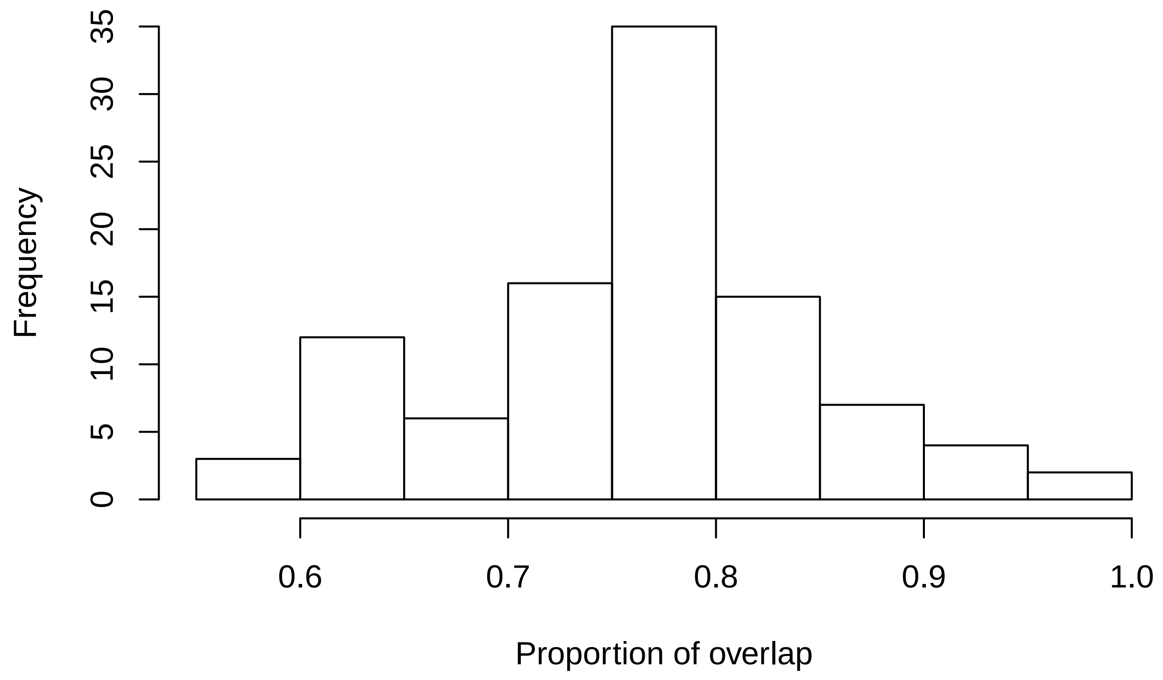
```r
ggsave("semiaqu_pelegic.png", dpi=300, width=4, height=3)

#hdr(SAp_95.overlap_prop, h = 10)



#####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 = "")
```



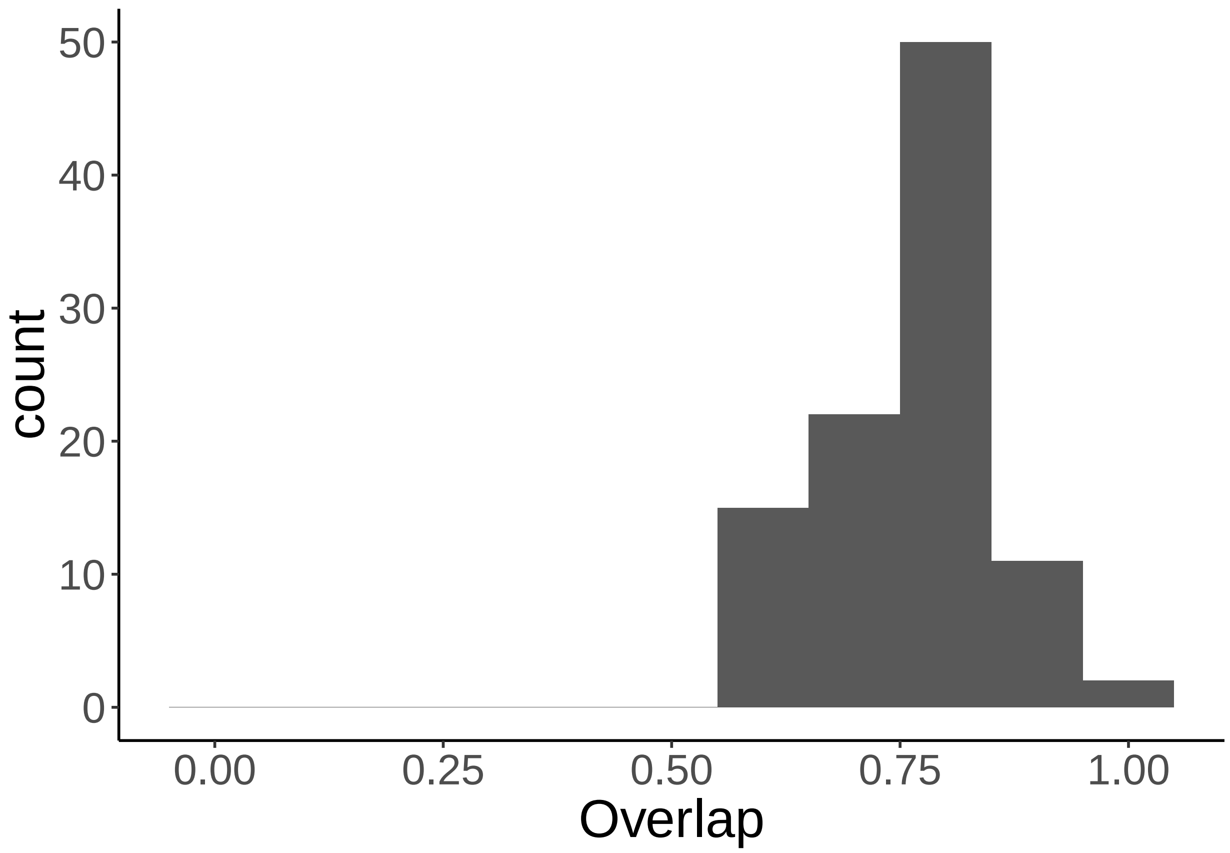
```
myplot_SAsfoss = ggplot(data.frame(Overlap = SAsfoss_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



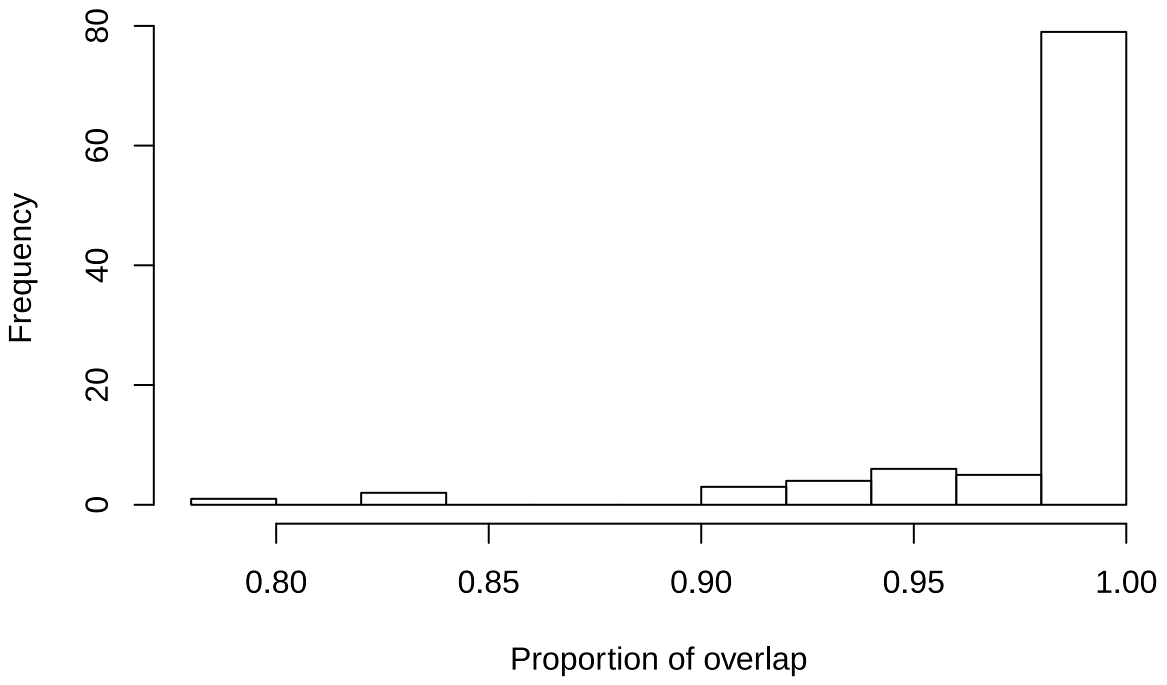
```r
ggsave("semiaqu_semifoss.png", dpi=300, width=4, height=3)

#hdr(SAsfoss_95.overlap_prop, h = 10)



#####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 = "")
```



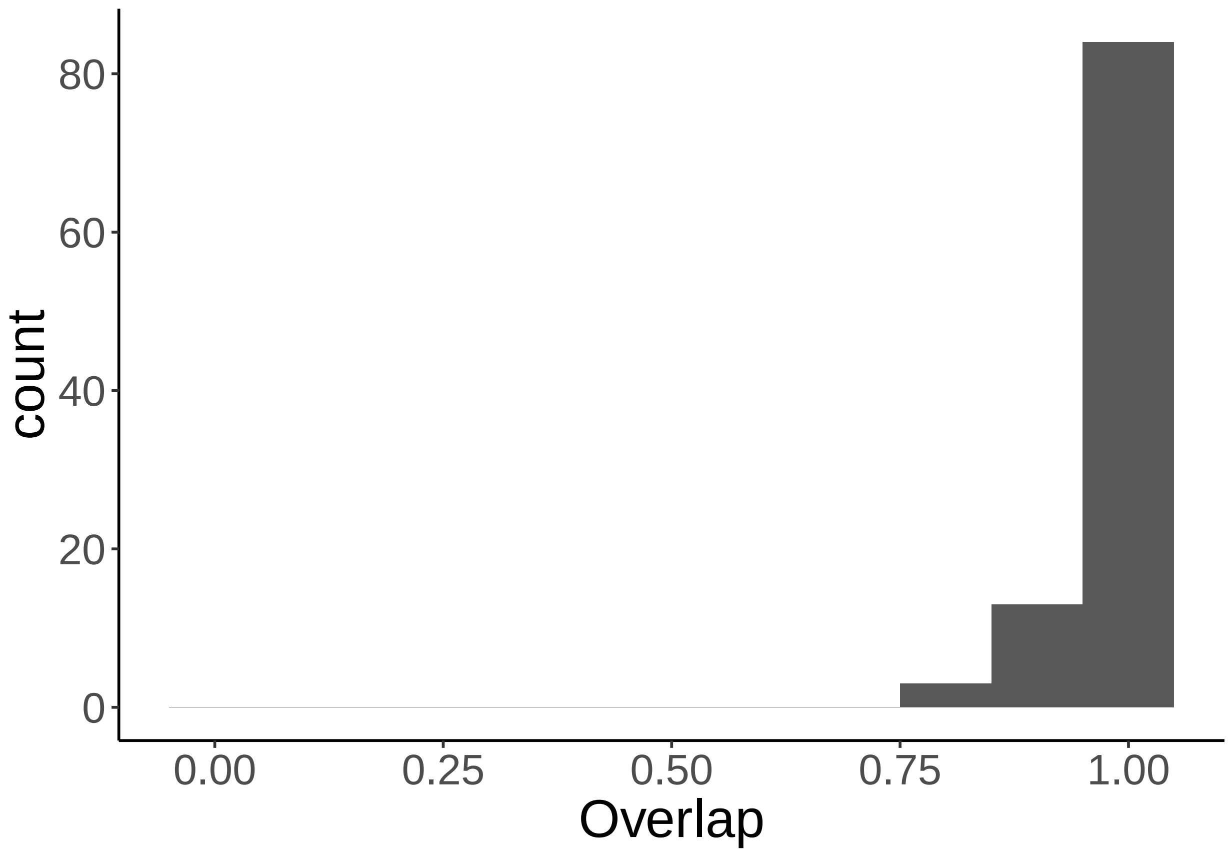
```
myplot_Tp = ggplot(data.frame(Overlap = Tp_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



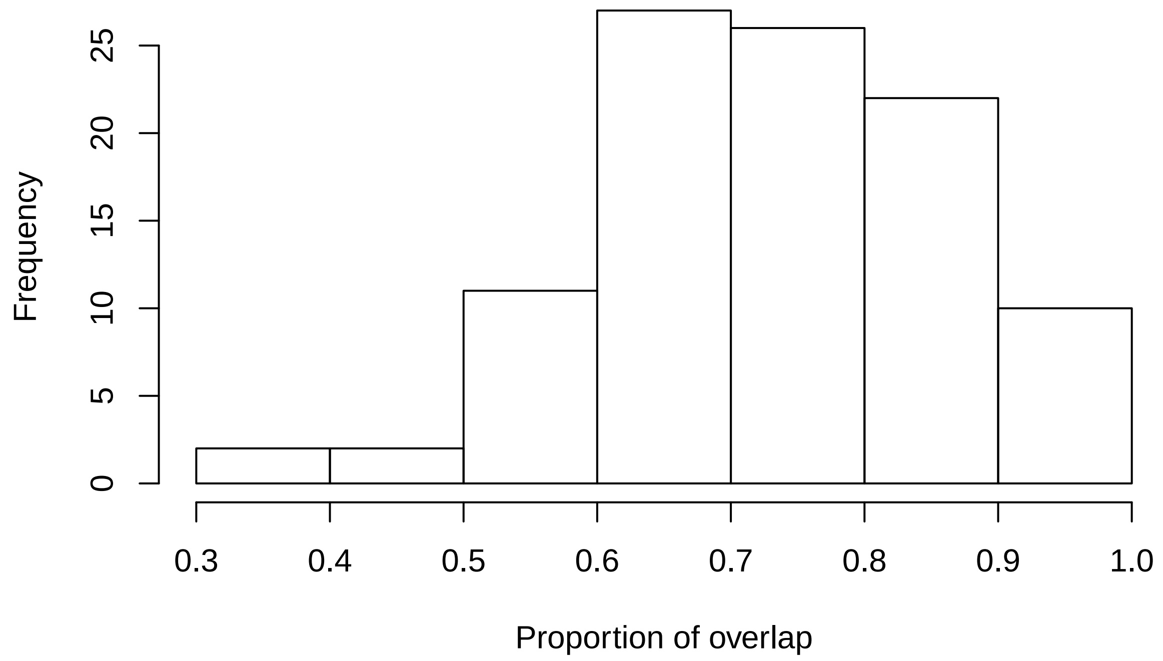
```
ggsave("terrestrial_pelegic.png", dpi=300, width=4, height=3)

#hdr(Tp_95.overlap_prop, h = 10)


#####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 = "")
```



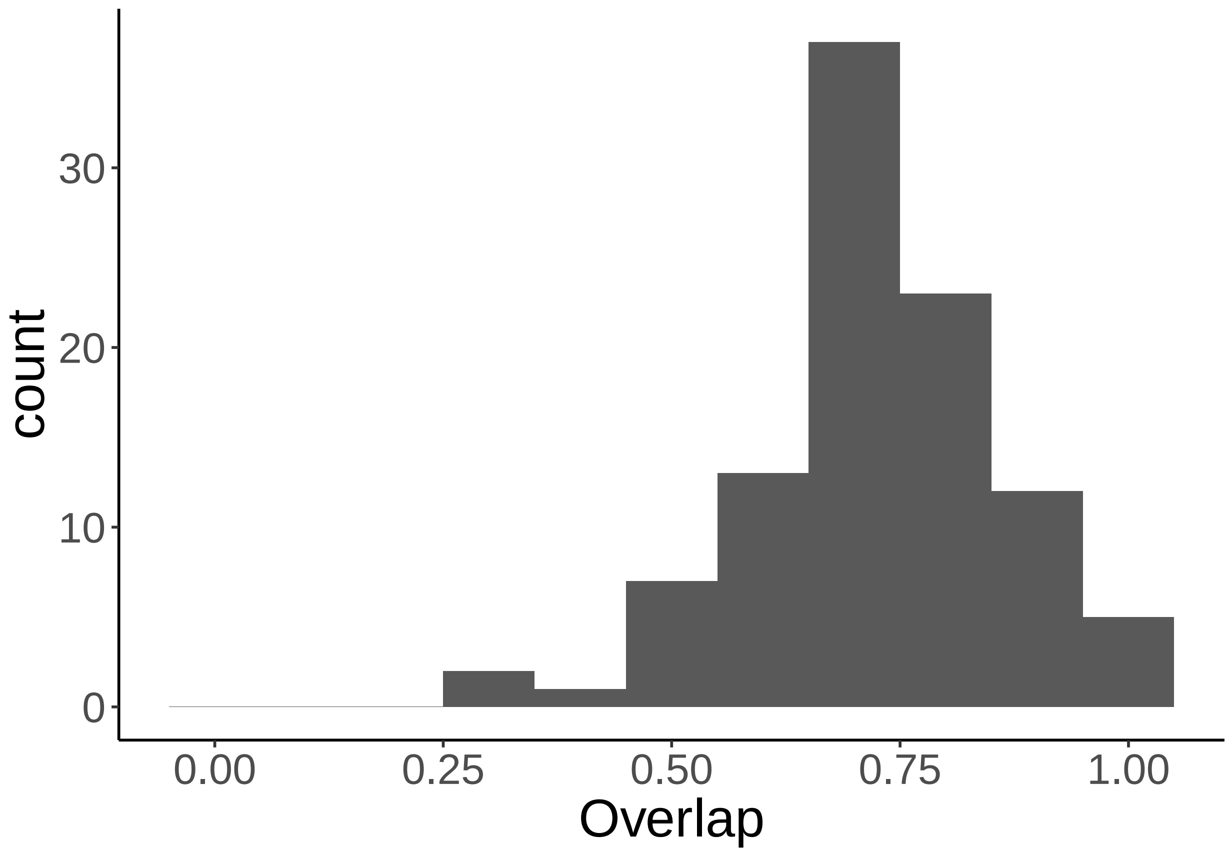
```
myplot_Tfoss = ggplot(data.frame(Overlap = Tfoss_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



```
ggsave("terrestrial_semifoss.png", dpi=300, width=4, height=3)

#hdr(Tfoss_95.overlap_prop, h = 10)



#####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)) +
  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))
```



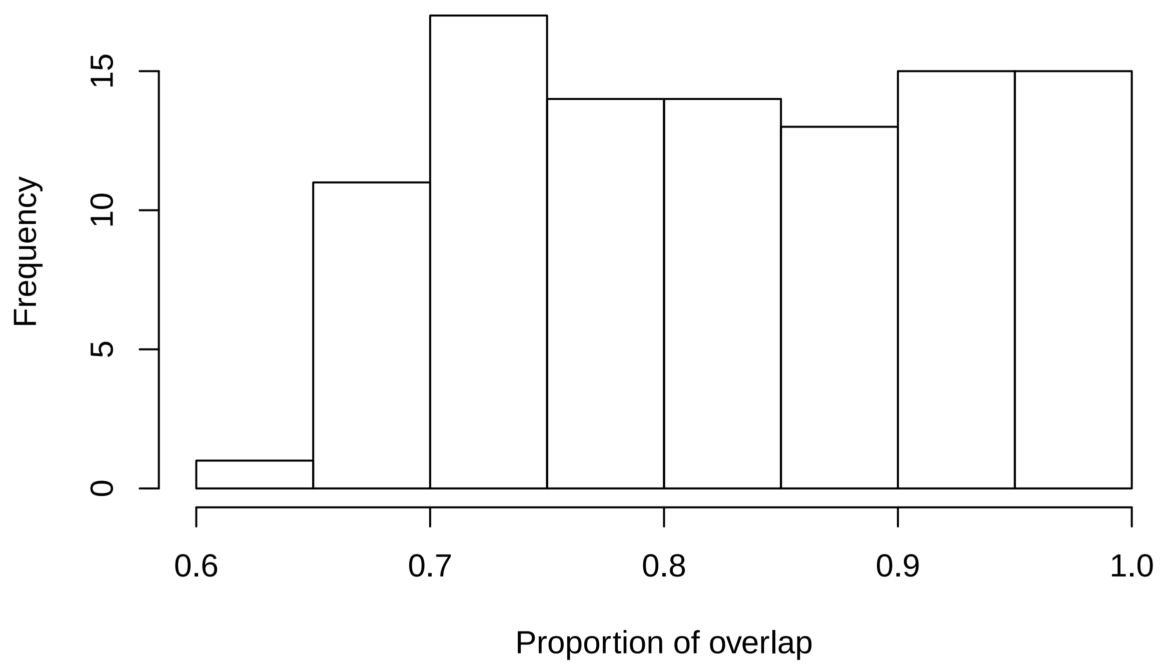
```
ggsave("pelegic_semifoss.png", dpi=300, width=4, height=3)

#hdr(Psfoss_95.overlap_prop, h = 10)
```

IUCN overlap calculations for mode of life

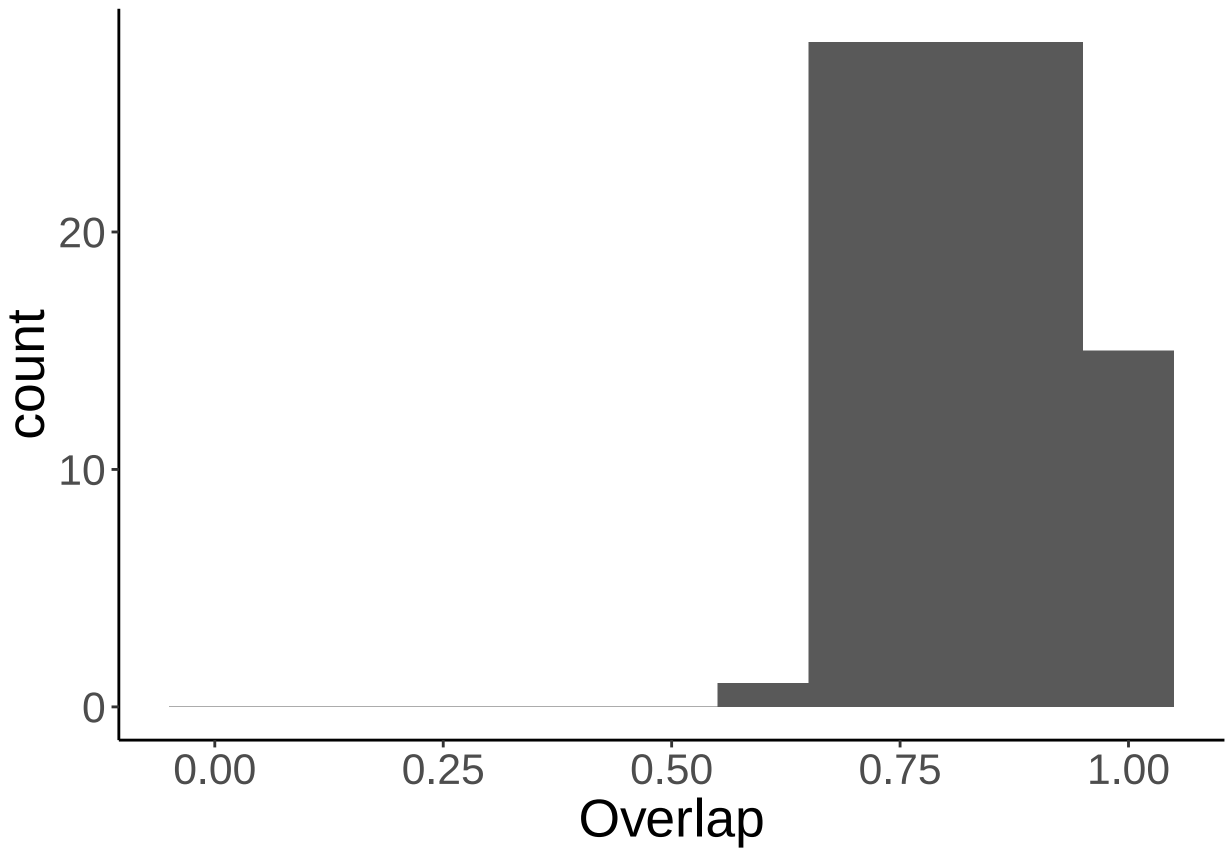
```
group.ML <- groupMetricsML(siber.plots)
group.ML_iucn <- groupMetricsML(siber.iucn)


# 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: 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: 31
##    Unobserved stochastic nodes: 3
##    Total graph size: 46
##
## 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
```

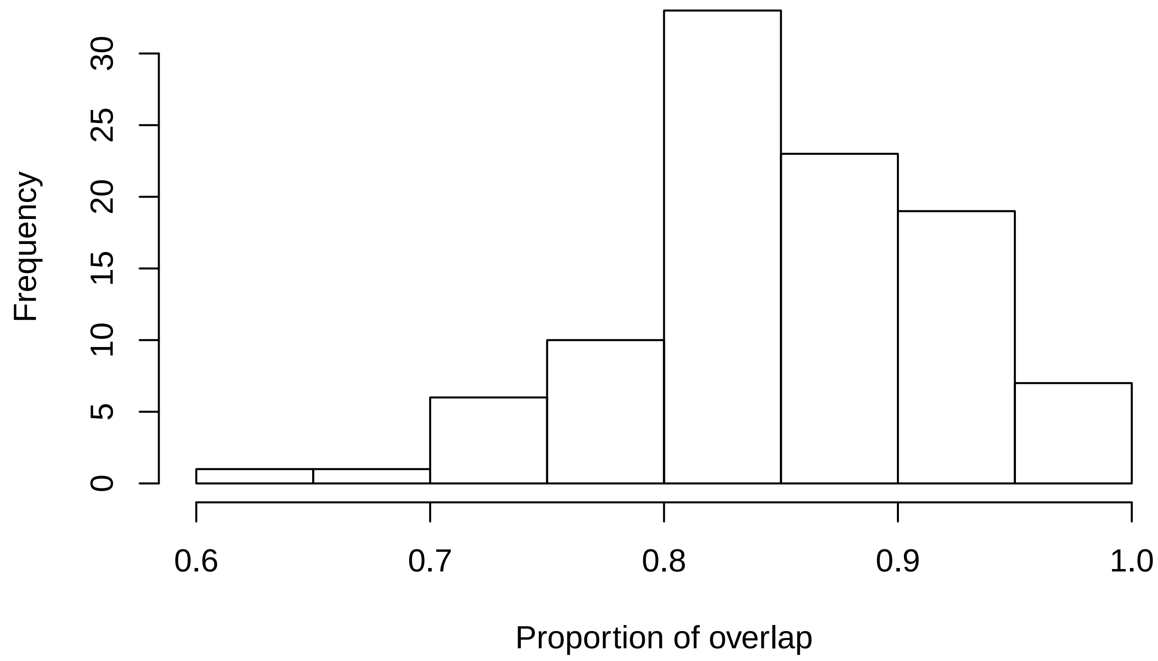
```
# The first ellipse is referenced using a character string representation where
# in "x.y", "x" is the community, and "y" is the group within that community.
# So in this example: community 1, group 1


#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 = "")
```



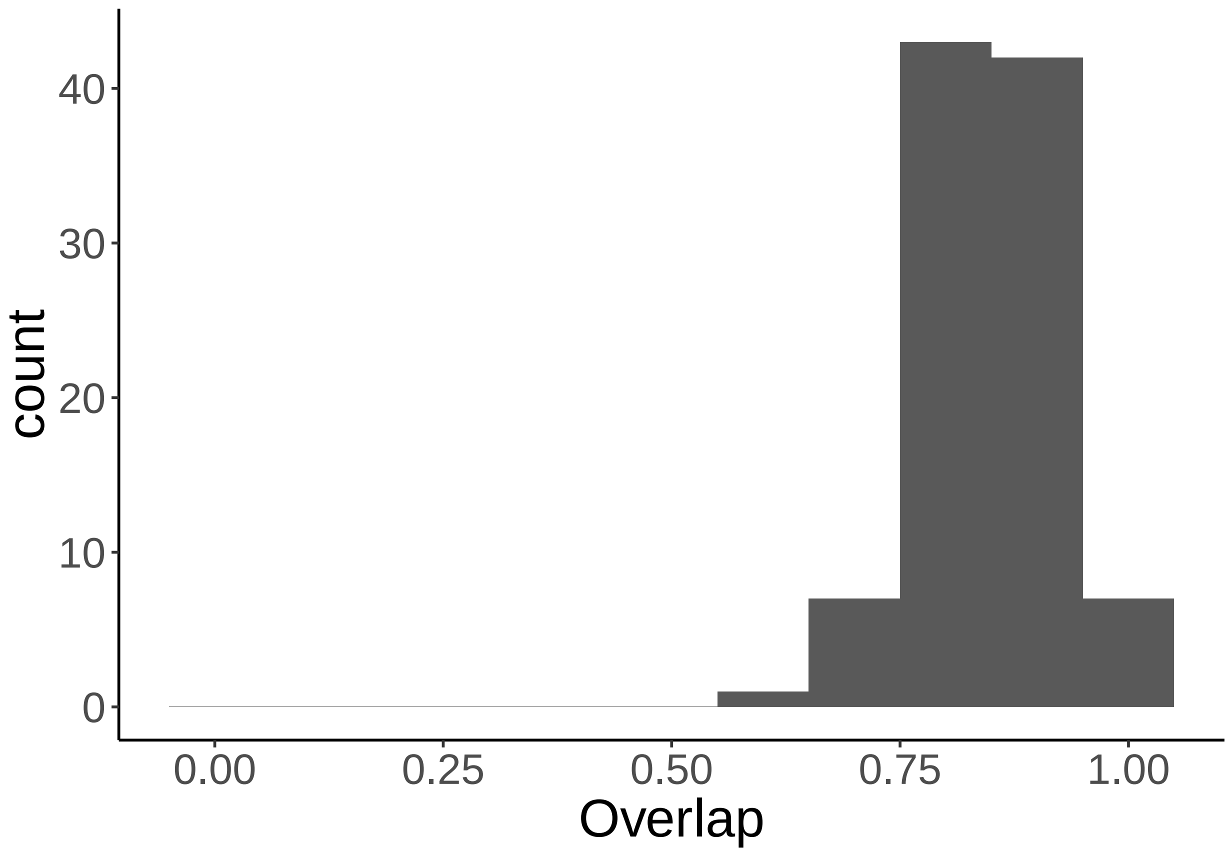
```
myplot_LC_NT = ggplot(data.frame(Overlap =  LC_NT_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



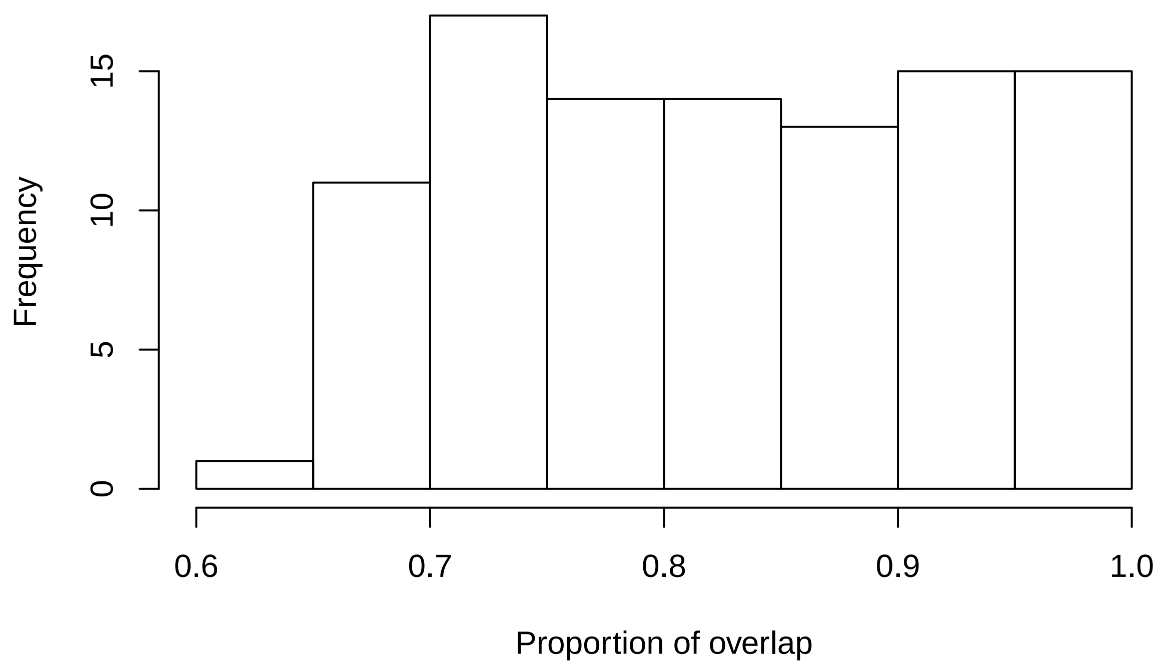
```
ggsave("LC_NT.png", dpi=300, width=4, height=3)

#hdr(LC_NT_95.overlap_prop, h = 10)







#####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 = "")
```



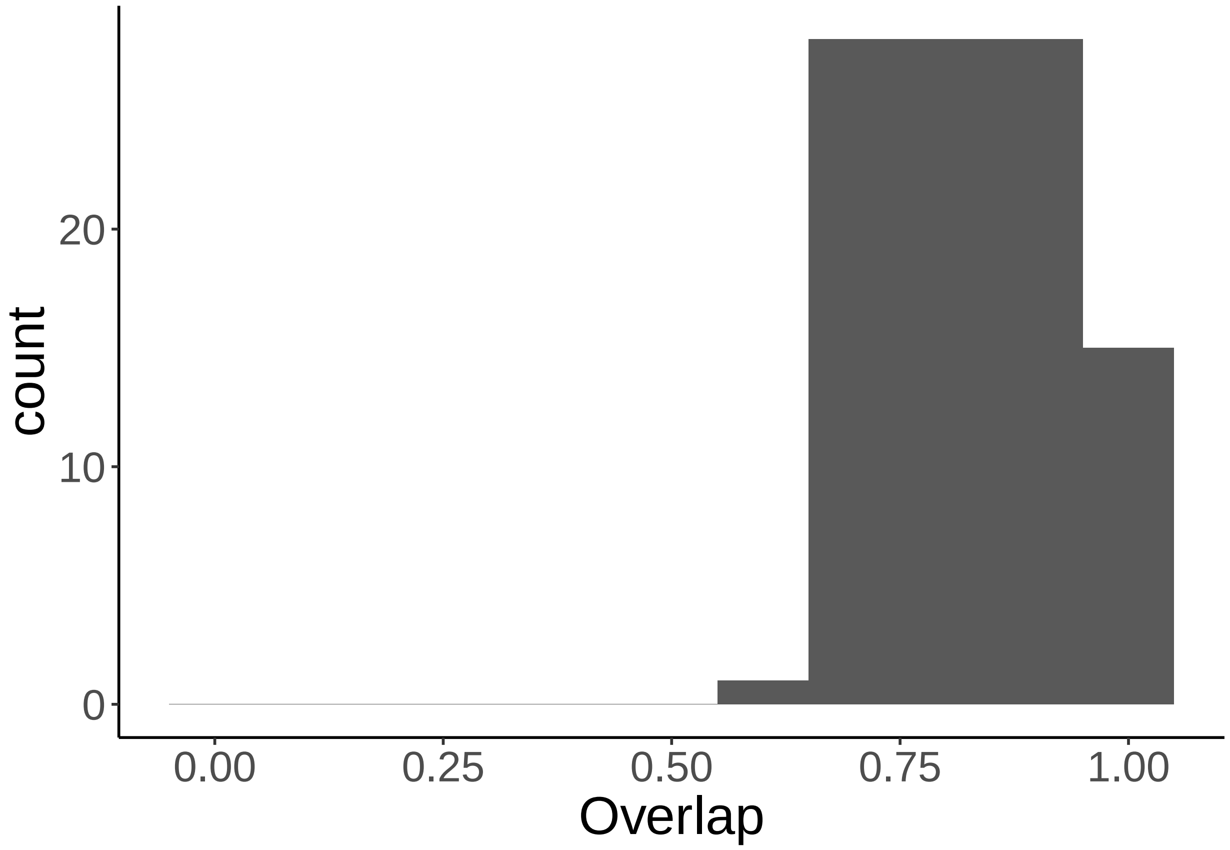
```
myplot_LC_V = ggplot(data.frame(Overlap =  LC_V_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



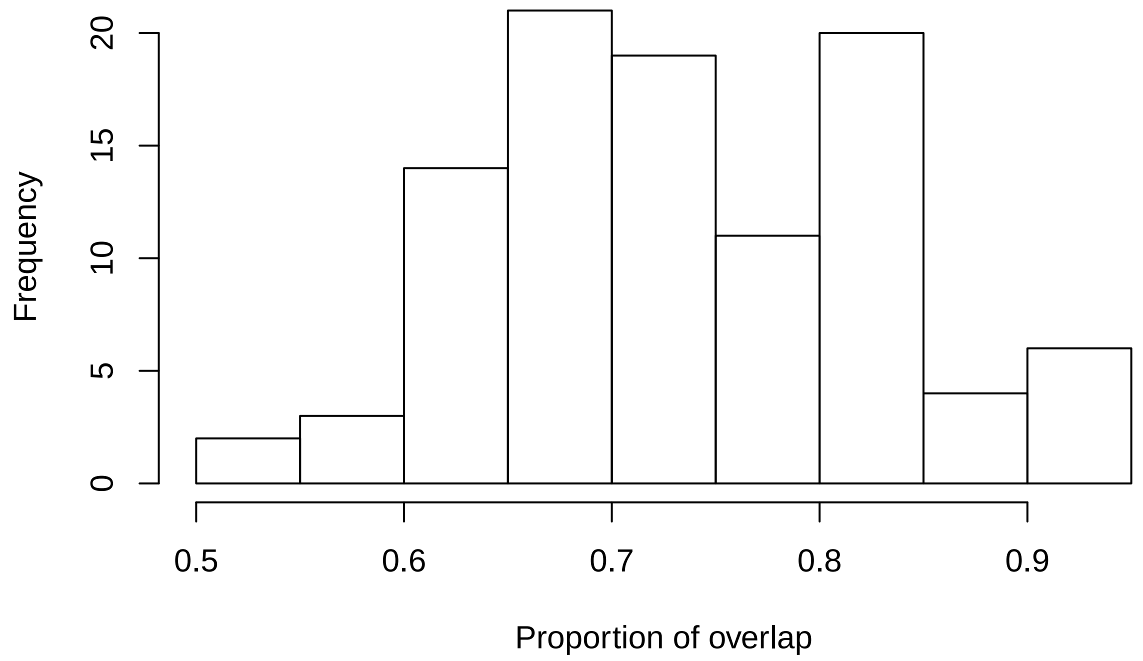
```
ggsave("LC_V.png", dpi=300, width=4, height=3)

#hdr(LC_V_95.overlap_prop, h = 10)






#####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 = "")
```



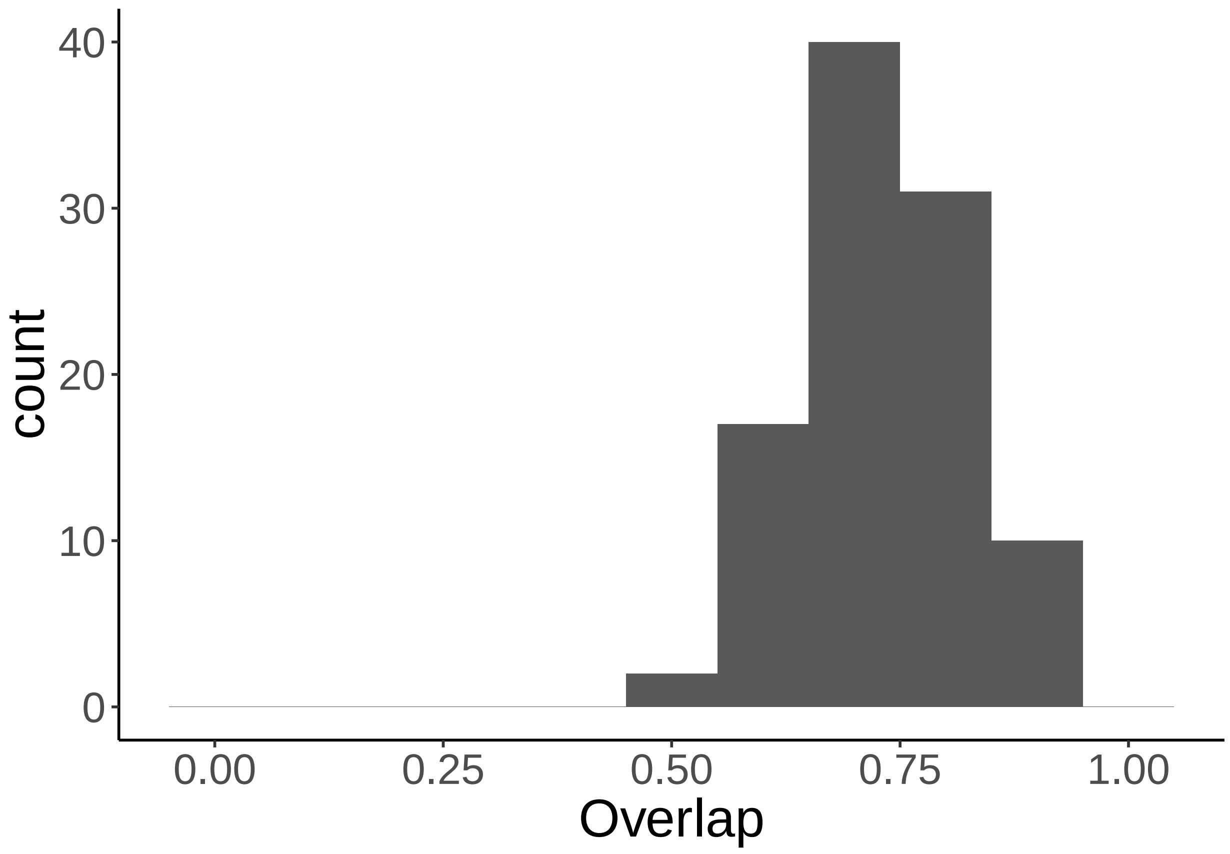
```r
myplot_LC_E = ggplot(data.frame(Overlap =  LC_E_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



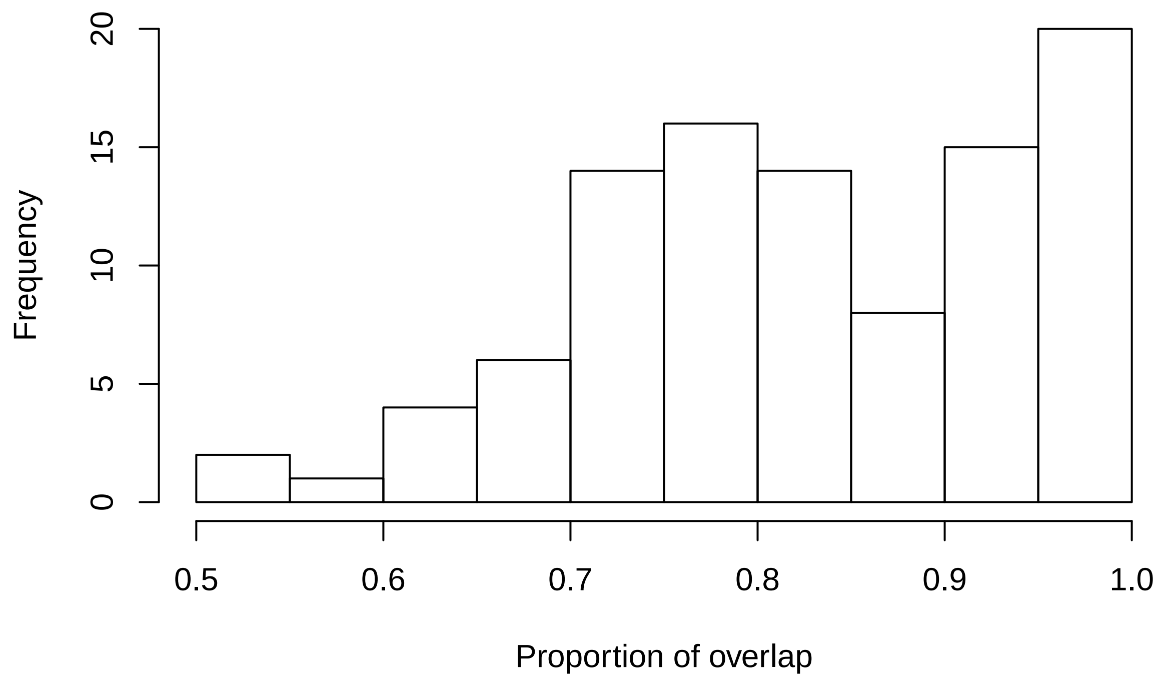
```r
ggsave("LC_E.png", dpi=300, width=4, height=3)

#hdr(LC_E_95.overlap_prop, h = 10)






#####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 = "")
```



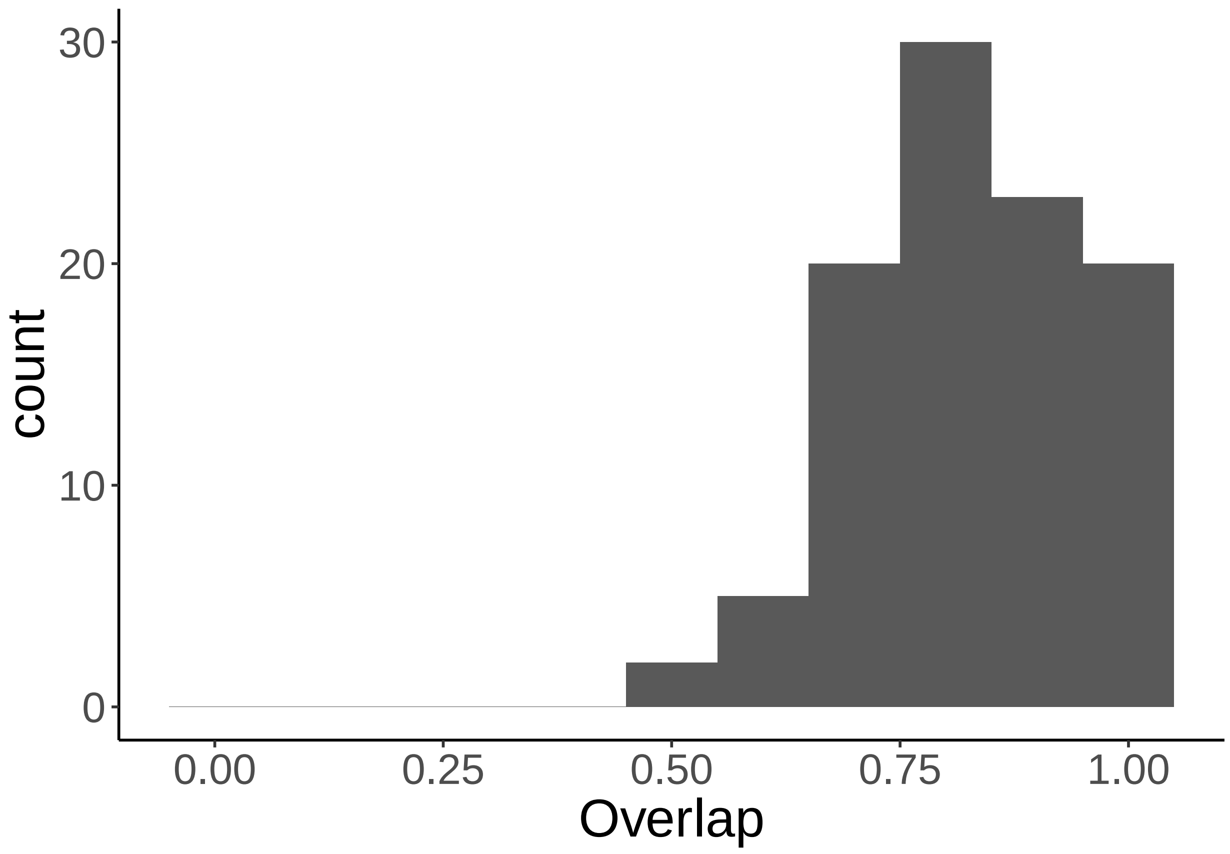
```
myplot_LC_CE = ggplot(data.frame(Overlap =  LC_CE_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



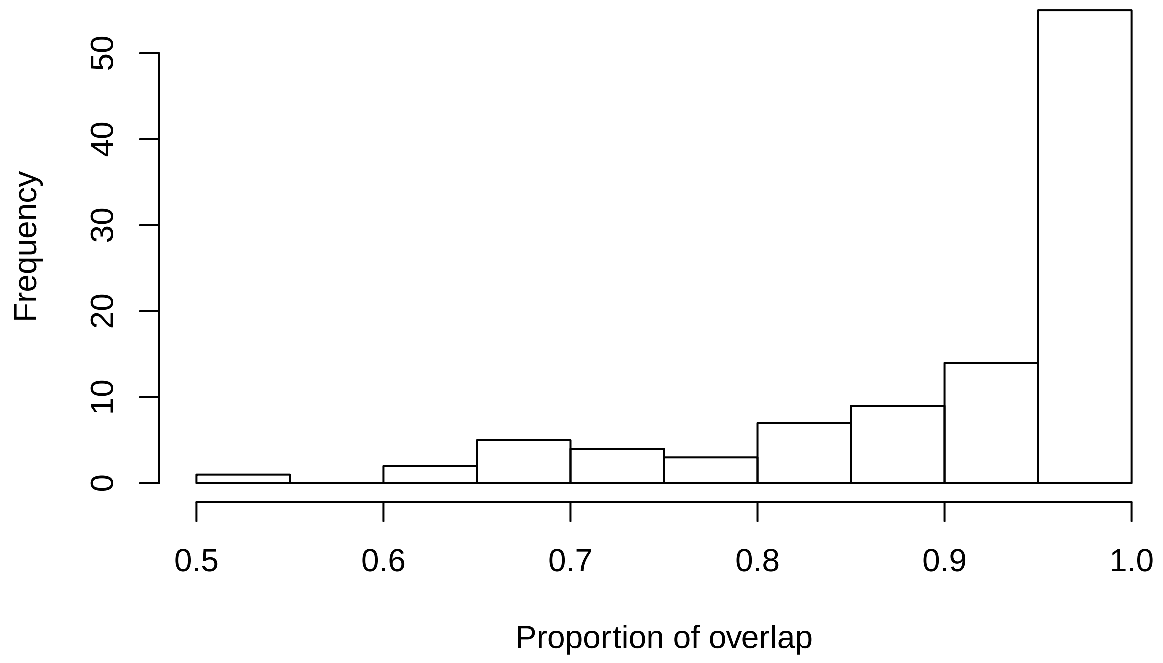
```
ggsave("LC_CE.png", dpi=300, width=4, height=3)

#hdr(LC_CE_95.overlap_prop, h = 10)






#####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 = "")
```



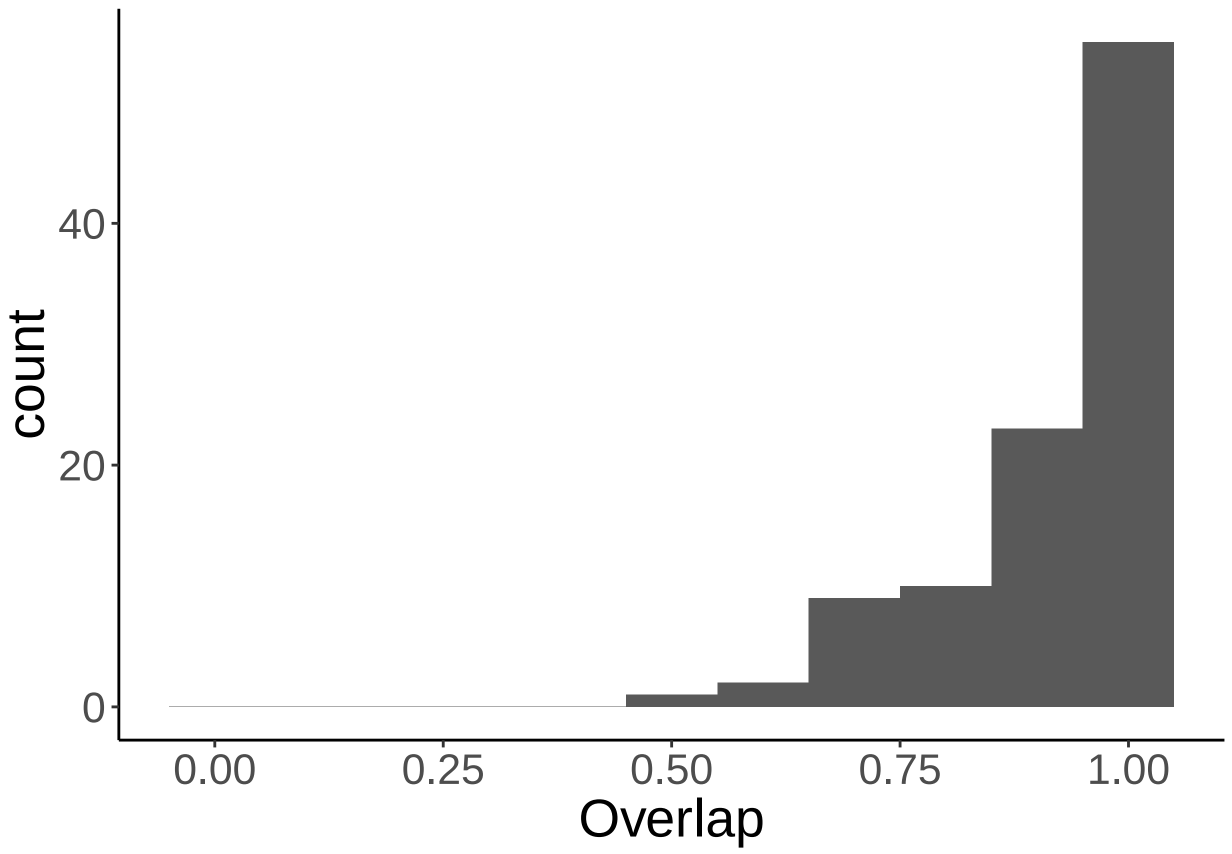
```
myplot_NT_V = ggplot(data.frame(Overlap =  NT_V_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



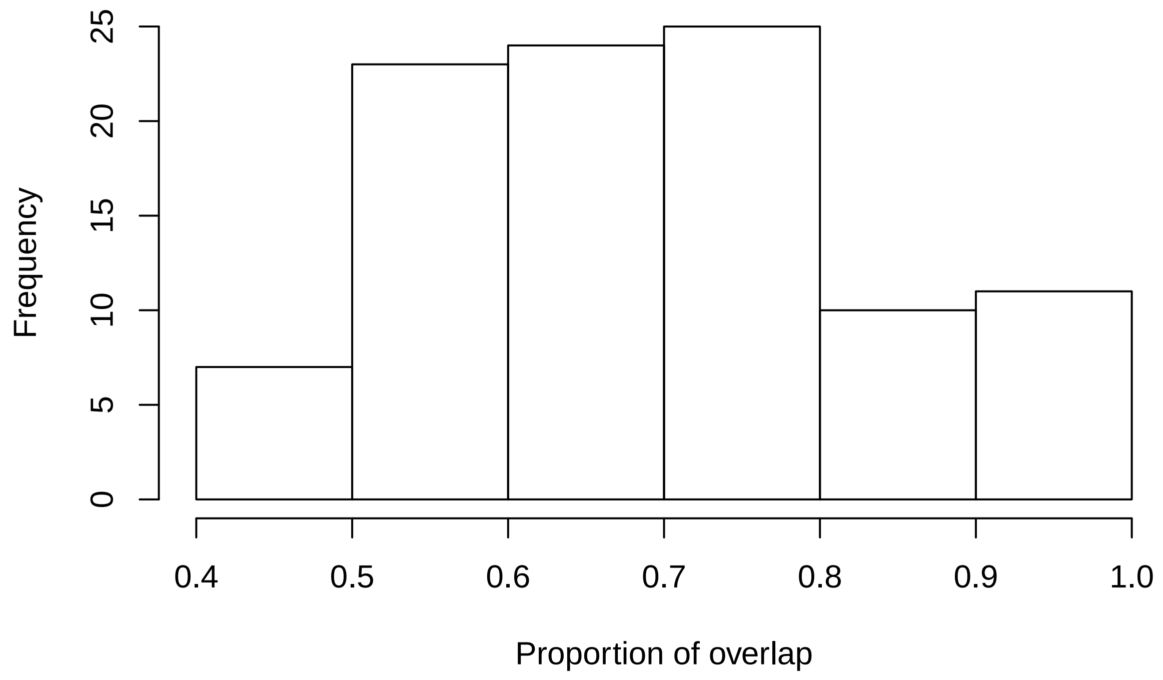
```
ggsave("NT_V.png", dpi=300, width=4, height=3)

#hdr(NT_V_95.overlap_prop, h = 10)







#####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 = "")
```



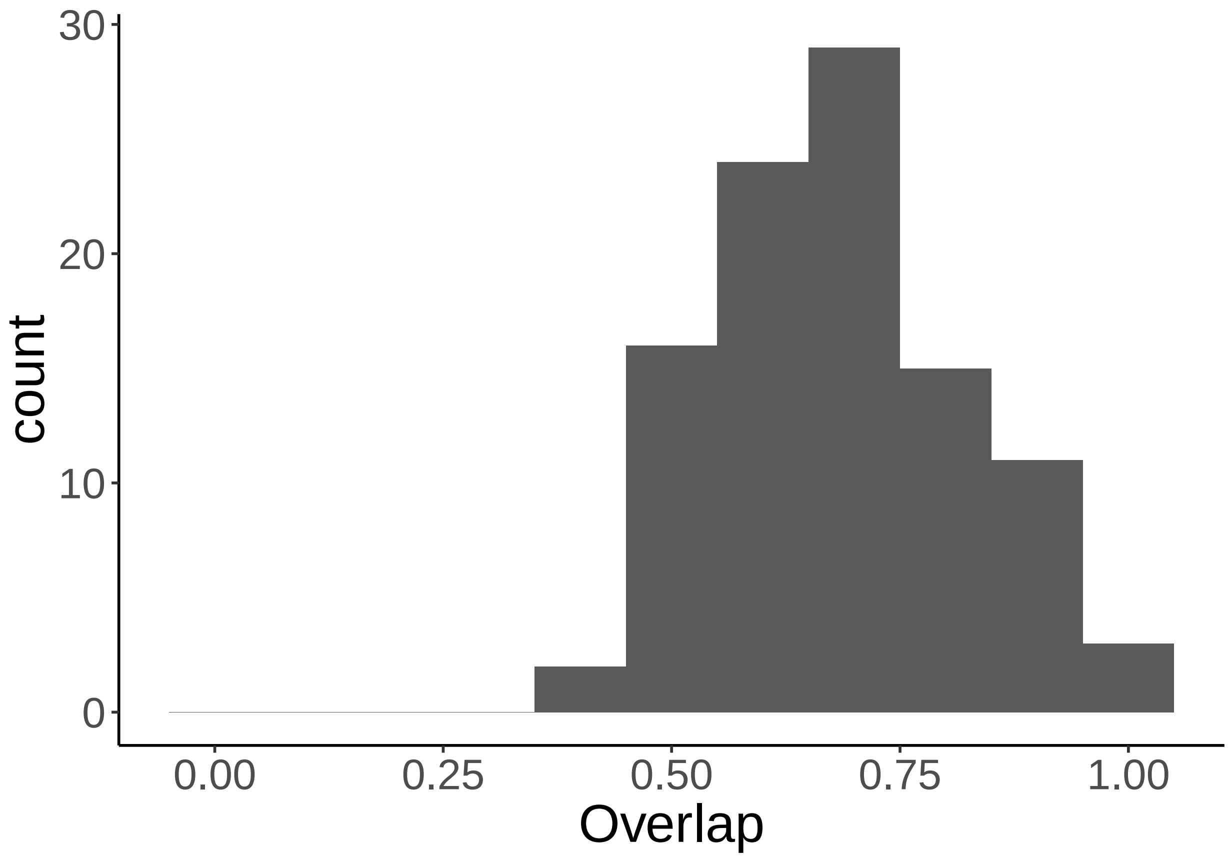
```
myplot_NT_E = ggplot(data.frame(Overlap =  NT_E_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



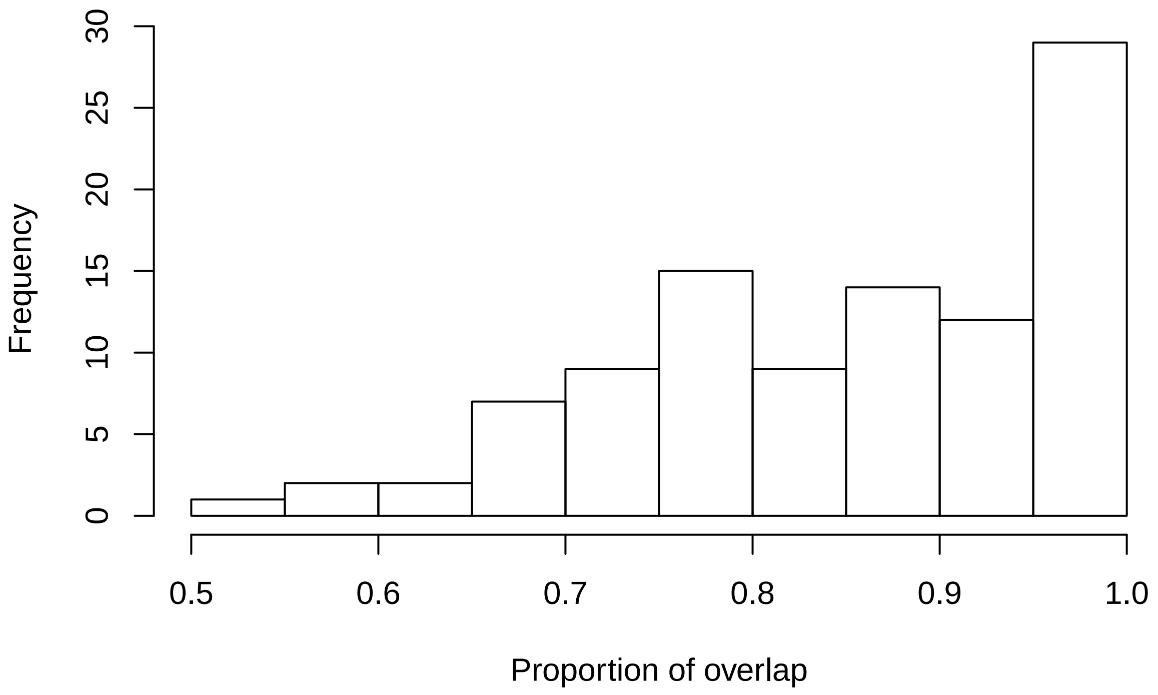
```
ggsave("NT_E.png", dpi=300, width=4, height=3)

#hdr(NT_E_95.overlap_prop)






#####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 = "")
```



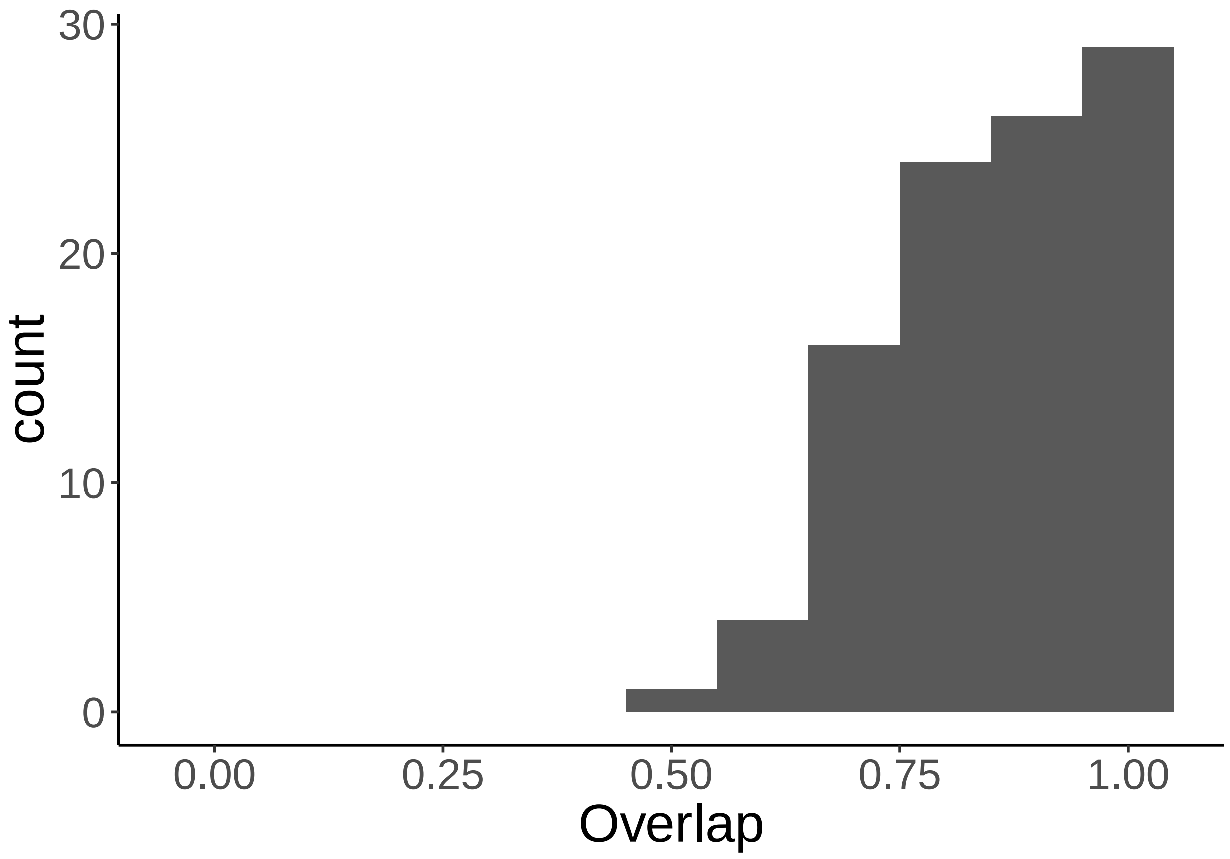
```
myplot_NT_CE = ggplot(data.frame(Overlap =  NT_CE_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



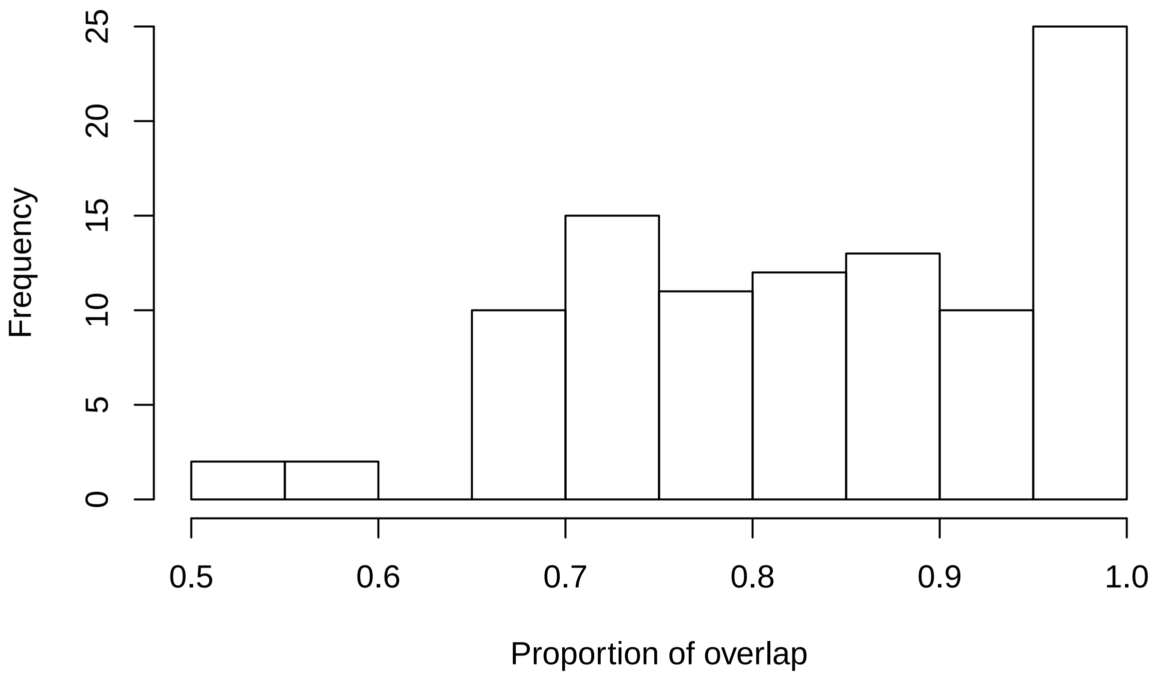
```
ggsave("NT_CE.png", dpi=300, width=4, height=3)

#hdr(NT_CE_95.overlap_prop, h = 10)






#####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 = "")
```



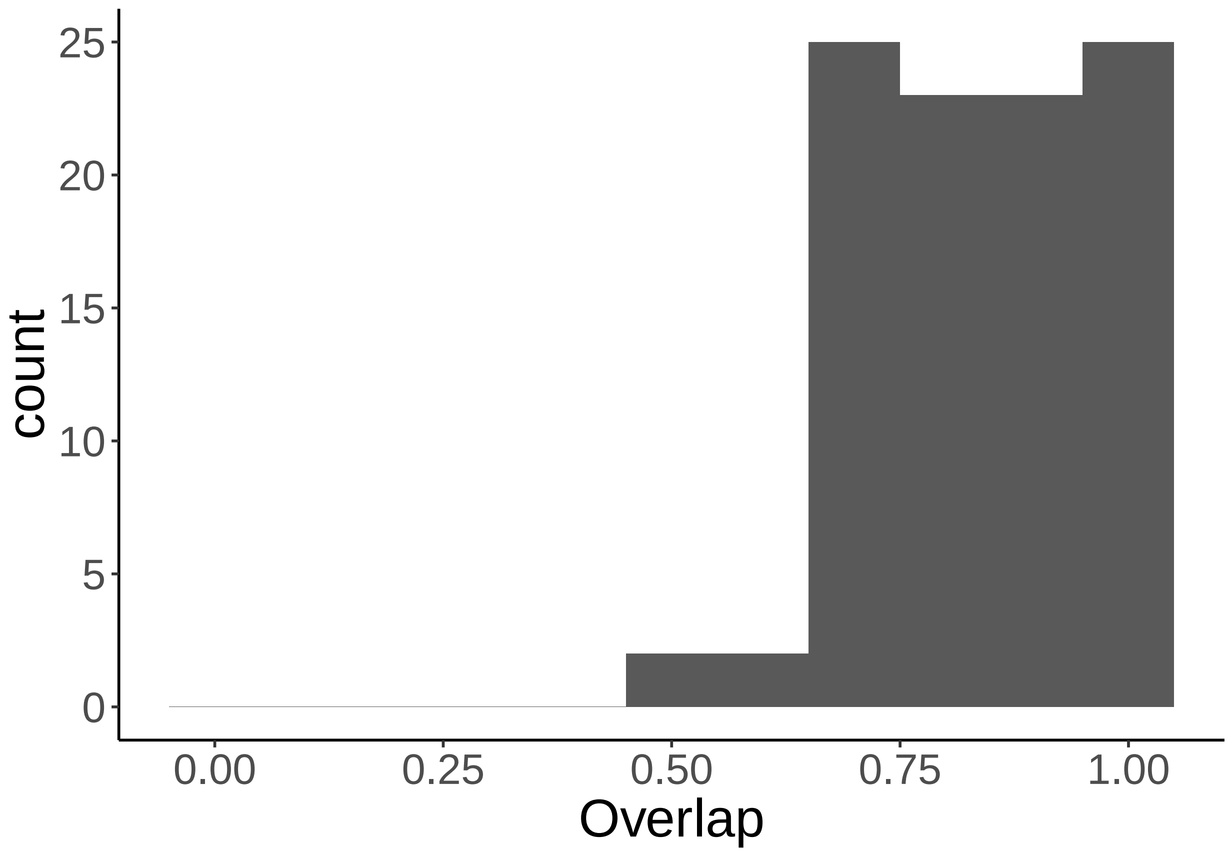
```
myplot_E_CE = ggplot(data.frame(Overlap =  E_CE_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



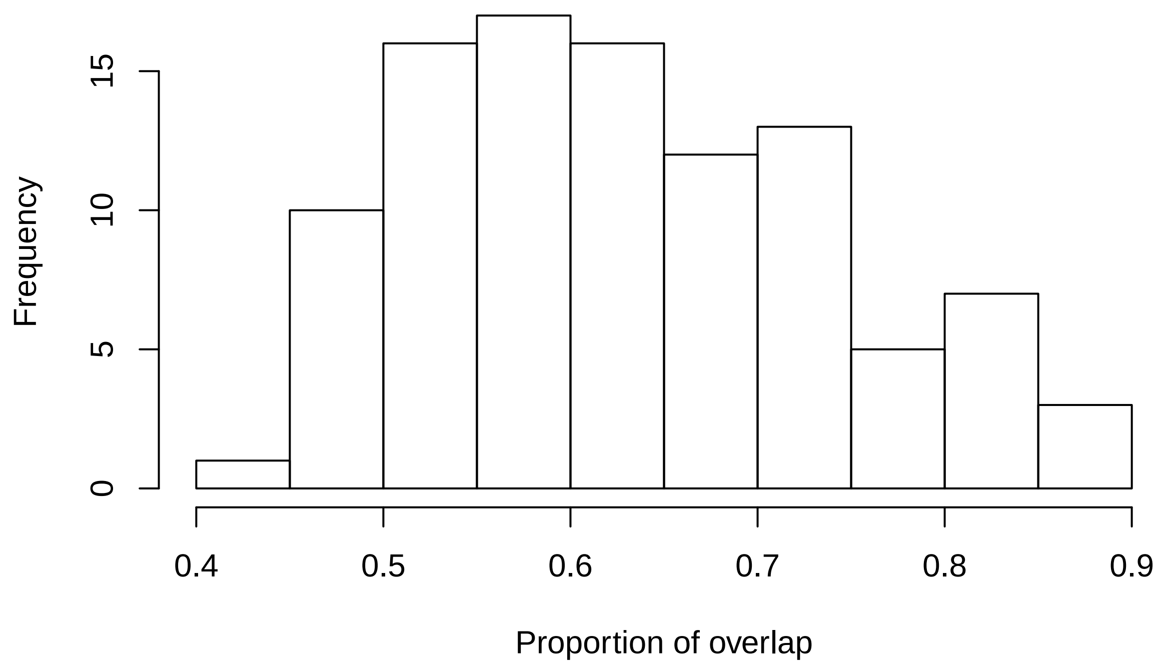
```
ggsave("E_CE.png", dpi=300, width=4, height=3)

#hdr(E_CE_95.overlap_prop)





#####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 = "")
```



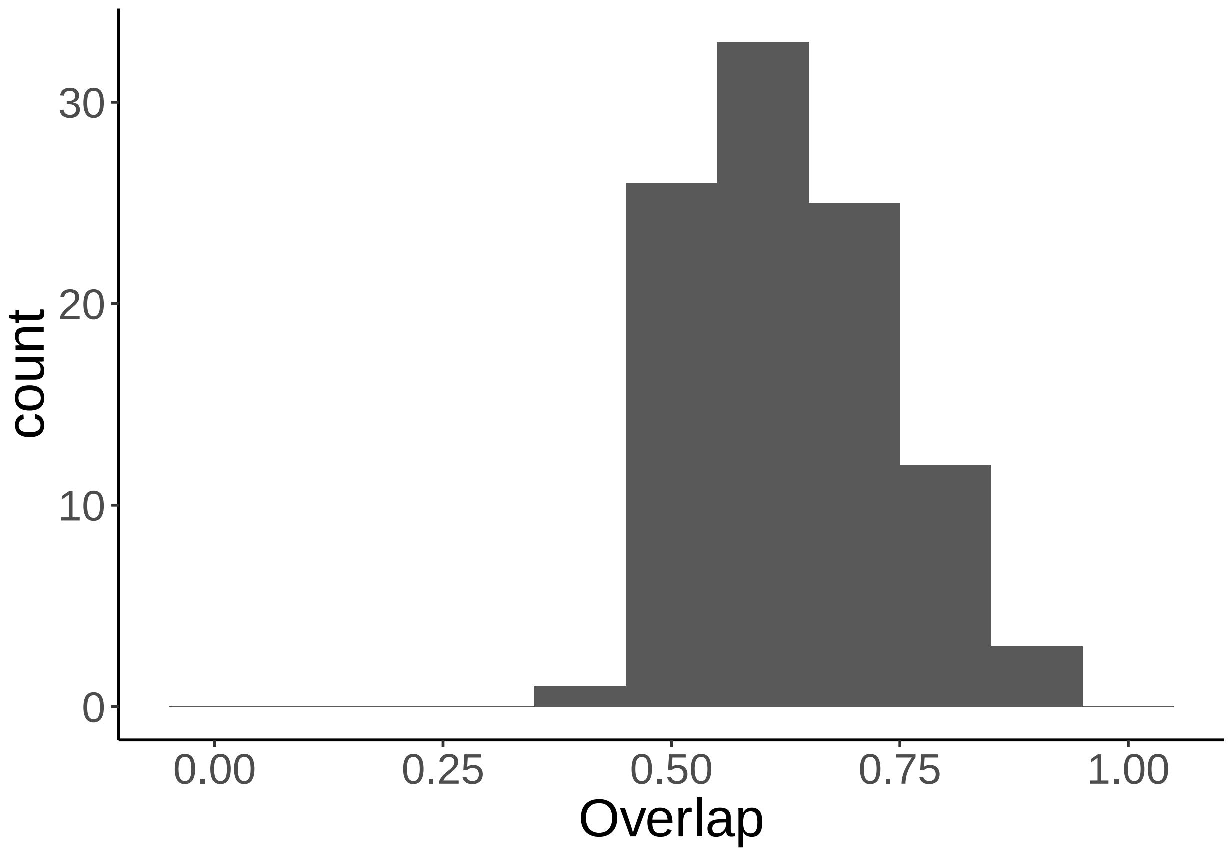
```r
myplot_E_V = ggplot(data.frame(Overlap =  E_V_95.overlap_prop),
                    aes(Overlap)) +
  geom_histogram(binwidth=0.1) +
  scale_x_continuous(limits = c(-0.05, 1.05))

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))
```



```r
ggsave("E_V.png", dpi=300, width=4, height=3)

#hdr(E_V_95.overlap_prop)






#####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)) +
  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))
```



```r
ggsave("CE_V.png", dpi=300, width=4, height=3)

#hdr(CE_V_95.overlap_prop)
```