ggplot_tutorial.R

haldrerogers

Mon Dec 3 09:50:04 2018

```
############
# Introduction to ggplot2
# LunchinatoRs
# Author: Haldre Rogers
# last updated 5 November 2018

#### References ######
# Super useful cheatsheet: https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf
# Inspired by several websites, including: http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html#o
# http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3/
#
# Other resources
# http://www.r-graph-gallery.com/
# https://github.com/Gibbsdavidl/CatterPlots - make scatter plots with cat shaped points!
# http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html check out this website for

############
library(tidyverse)
```

```
## -- Attaching packages ------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.7
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ---------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(ggthemes)
library(lme4)
```

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

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

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```
##############################

#Read in seed data and location of seed traps.
pstraps<-read.csv("pstraps.csv", header=TRUE)

#Dataset is to examine the proportion of seeds that are handled relative to distance from the nearest c

#check out the dataframe
str(pstraps)
```

```
## 'data.frame':    124 obs. of  15 variables:
##  $ trap     : int  1 1 1 1 1 1 1 1 1 10 ...
##  $ site     : Factor w/ 9 levels "CHIG","FORB",..: 1 2 3 4 5 6 7 8 9 1 ...
##  $ species  : Factor w/ 1 level "PS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ x        : num  354938 369776 353419 308548 309140 ...
##  $ y        : num  1665353 1676002 1663137 1565005 1565232 ...
##  $ source   : Factor w/ 1 level "HandDrawnMap": 1 1 1 1 1 1 1 1 1 1 ...
##  $ island   : Factor w/ 3 levels "rota","saipan",..: 3 2 3 1 1 2 3 2 1 3 ...
##  $ dist     : int  0 0 0 0 0 0 0 0 0 10 ...
##  $ total    : int  472 418 655 253 2 173 86 57 61 288 ...
##  $ handled  : int  68 152 39 23 2 115 74 33 3 14 ...
##  $ unhandled: num  202 133 308 115 0 29 6 12 29 137 ...
##  $ numsamp  : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ mindist  : num  0.772 1.299 1.328 1.306 1.037 ...
##  $ bird     : Factor w/ 1 level "yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sumtotal : num  270 285 347 138 2 144 80 45 32 151 ...
```

```
summary(pstraps)
```

```
##       trap             site    species        x                y
##  Min.   : 1.000   JCP    :17   PS:124   Min.   :308166   Min.   :1563661
##  1st Qu.: 4.000   MTR1   :17            1st Qu.:308996   1st Qu.:1565175
##  Median : 9.000   CHIG   :16            Median :353436   Median :1663391
##  Mean   : 8.702   JAPC   :16            Mean   :347828   Mean   :1644824
##  3rd Qu.:13.000   LADT   :16            3rd Qu.:370784   3rd Qu.:1686518
##  Max.   :17.000   MTLA   :15            Max.   :372159   Max.   :1688506
##                   (Other):27
##           source       island        dist           total
##  HandDrawnMap:124   rota  :33   Min.   : 0.00   Min.   :  1.00
##                     saipan:44   1st Qu.: 1.00   1st Qu.:  3.00
##                     tinian:47   Median :10.00   Median :  7.00
##                                 Mean   :10.61   Mean   : 73.55
##                                 3rd Qu.:20.00   3rd Qu.: 53.00
##                                 Max.   :20.00   Max.   :900.00
```

```
##
##     handled         unhandled         numsamp     mindist         bird
##  Min.   :  0.00   Min.   :  0.00   Min.   :3   Min.   : 0.4692   yes:124
##  1st Qu.:  2.00   1st Qu.:  0.00   1st Qu.:3   1st Qu.: 1.9789
##  Median :  5.00   Median :  0.00   Median :3   Median : 3.5606
##  Mean   : 25.52   Mean   : 24.01   Mean   :3   Mean   : 4.4036
##  3rd Qu.: 22.25   3rd Qu.: 13.00   3rd Qu.:3   3rd Qu.: 6.3100
##  Max.   :335.00   Max.   :342.50   Max.   :3   Max.   :15.8465
##
##     sumtotal
##  Min.   :  1.000
##  1st Qu.:  2.375
##  Median :  6.500
##  Mean   : 49.536
##  3rd Qu.: 38.000
##  Max.   :557.500
##
```

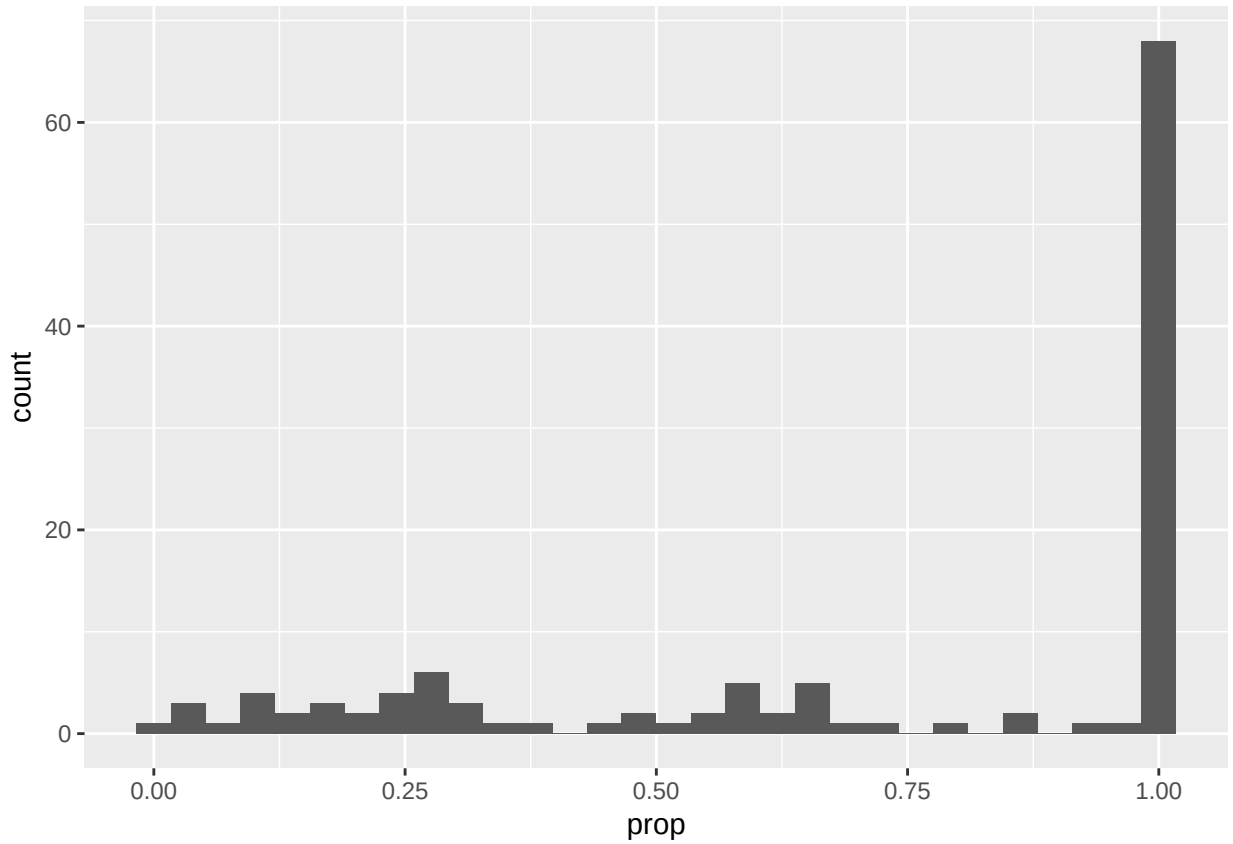
handled' seeds and 'total' seeds (ignore unhandled)

```
#create proportion column using these two variables
pstraps$prop<-as.numeric(pstraps$handled/pstraps$total)
summary(pstraps$prop)
```

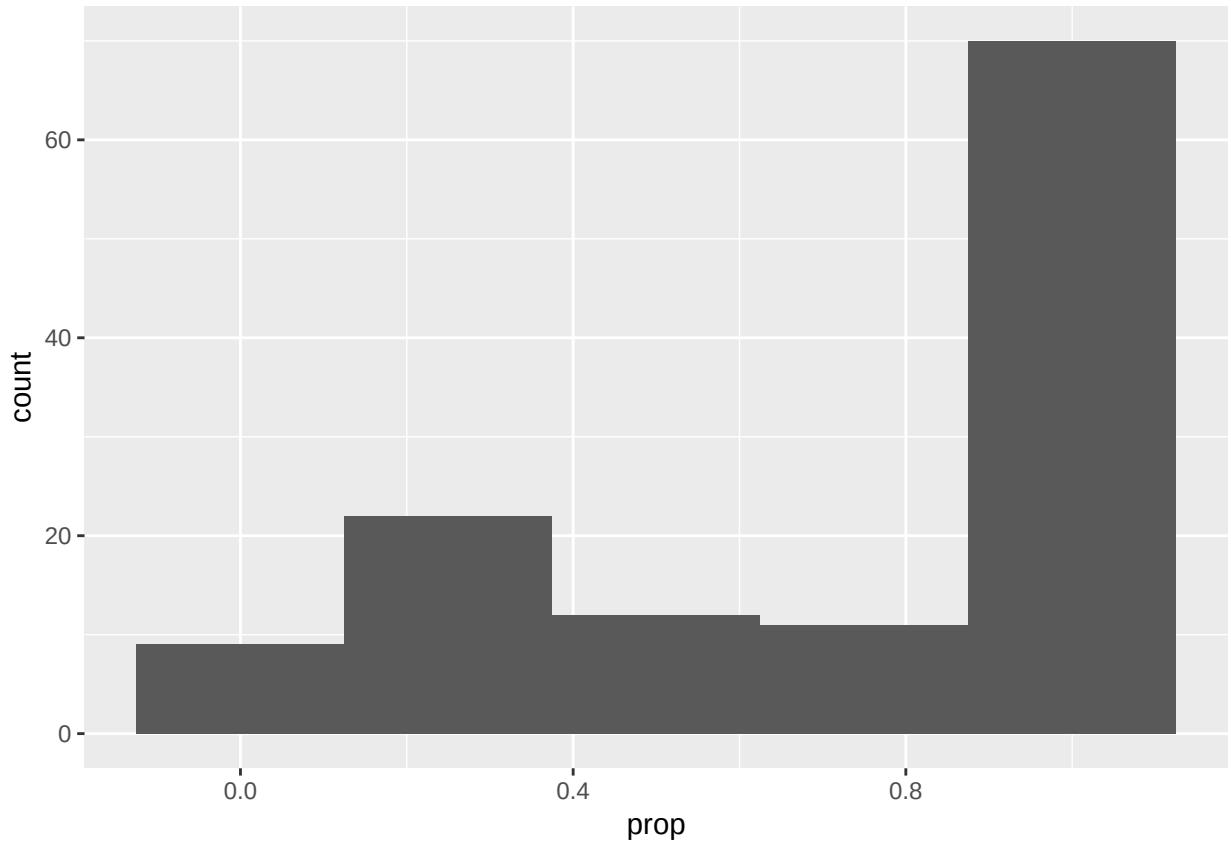
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.4143  1.0000  0.7294  1.0000  1.0000
```

```
#predictors
#island (factor), mindist (distance from nearest conspecific, numeric)

#other things that might be important
#site (factor)

##########################################
#1 Geometric Objects

#plot just a single variable
#continuous x
ggplot(pstraps, aes(prop))+
  geom_histogram() #default stat for geom_histogram is "bin"
```

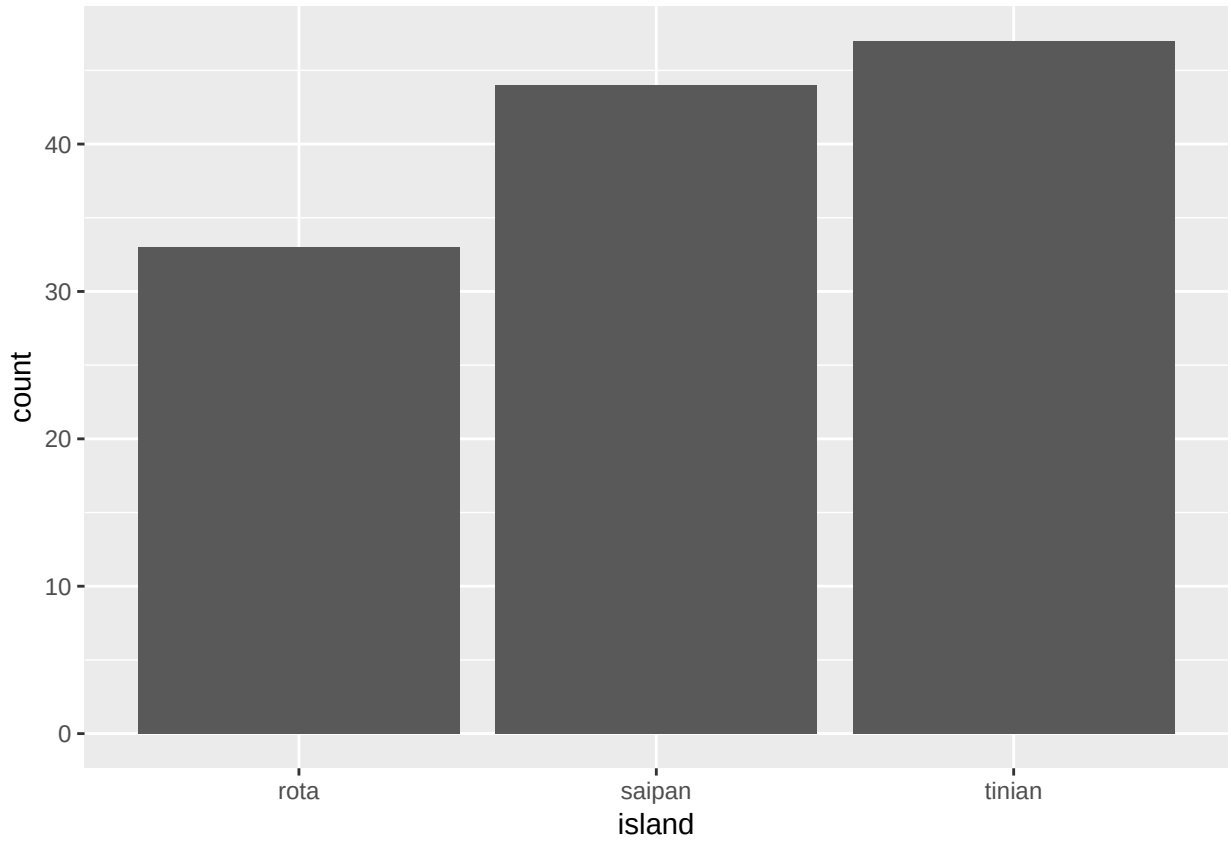
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



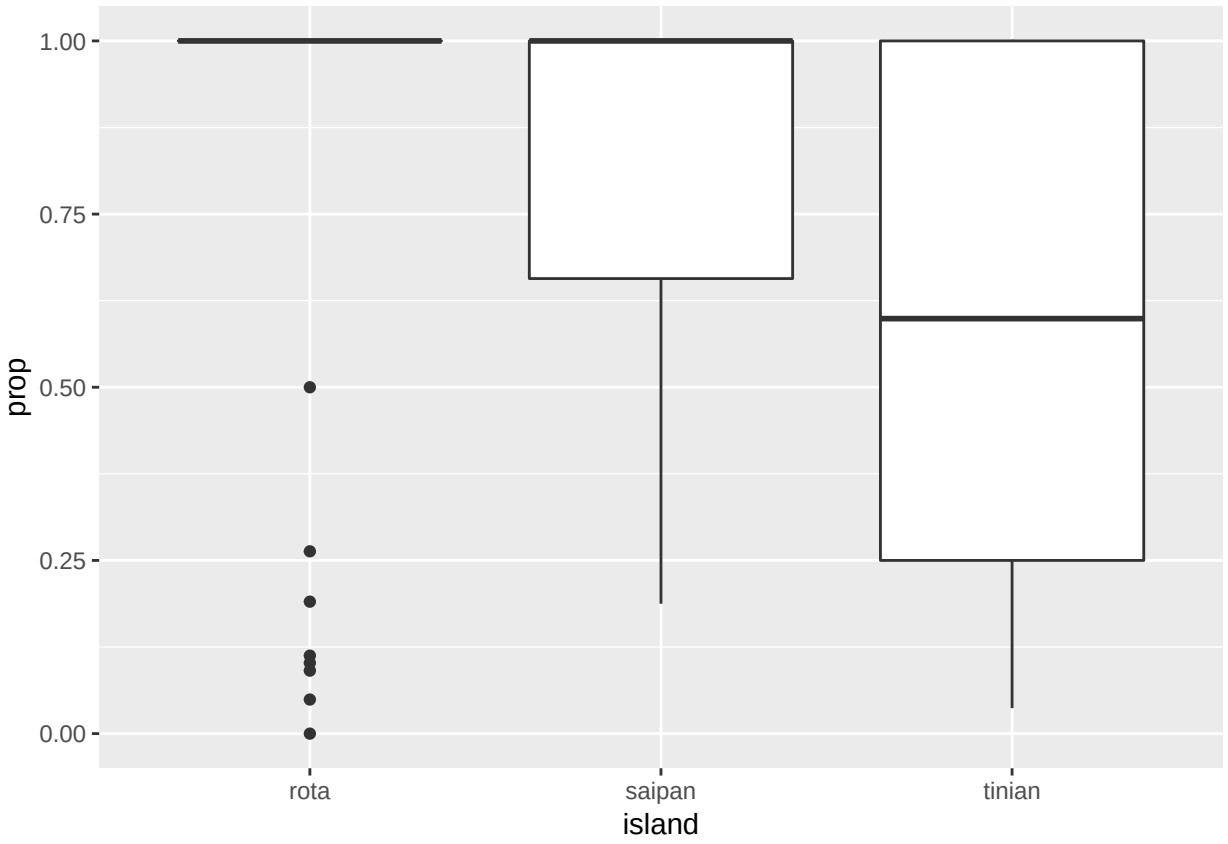

```
ggplot(pstraps, aes(prop))+
  geom_histogram(stat="bin", bins=5)
```



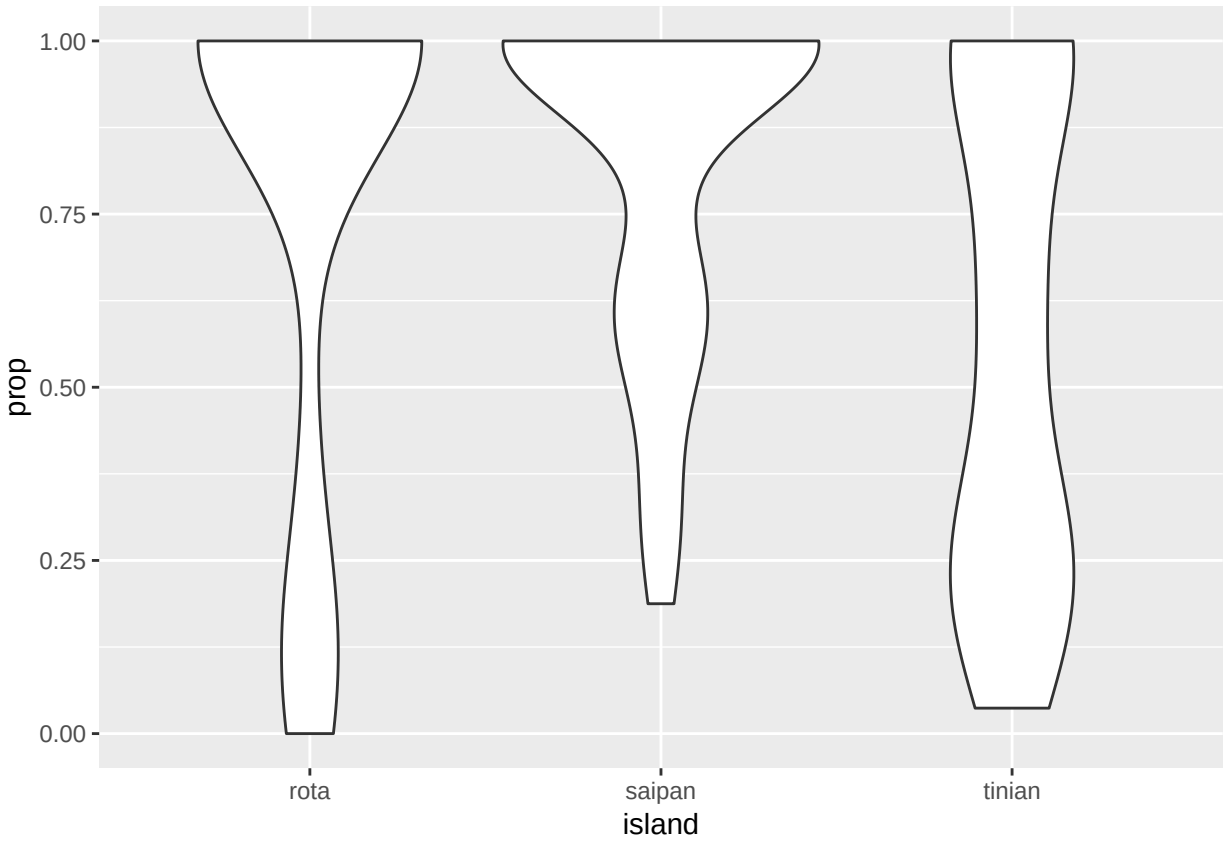

```
#categorical x
ggplot(pstraps, aes(island))+
  geom_bar(stat="count") #default stat for geom_bar is count. Count takes a count of the number of case
```




```
#plot x and y variables
ggplot(data=pstraps, aes(x=island, y=prop))+
  geom_boxplot()
```

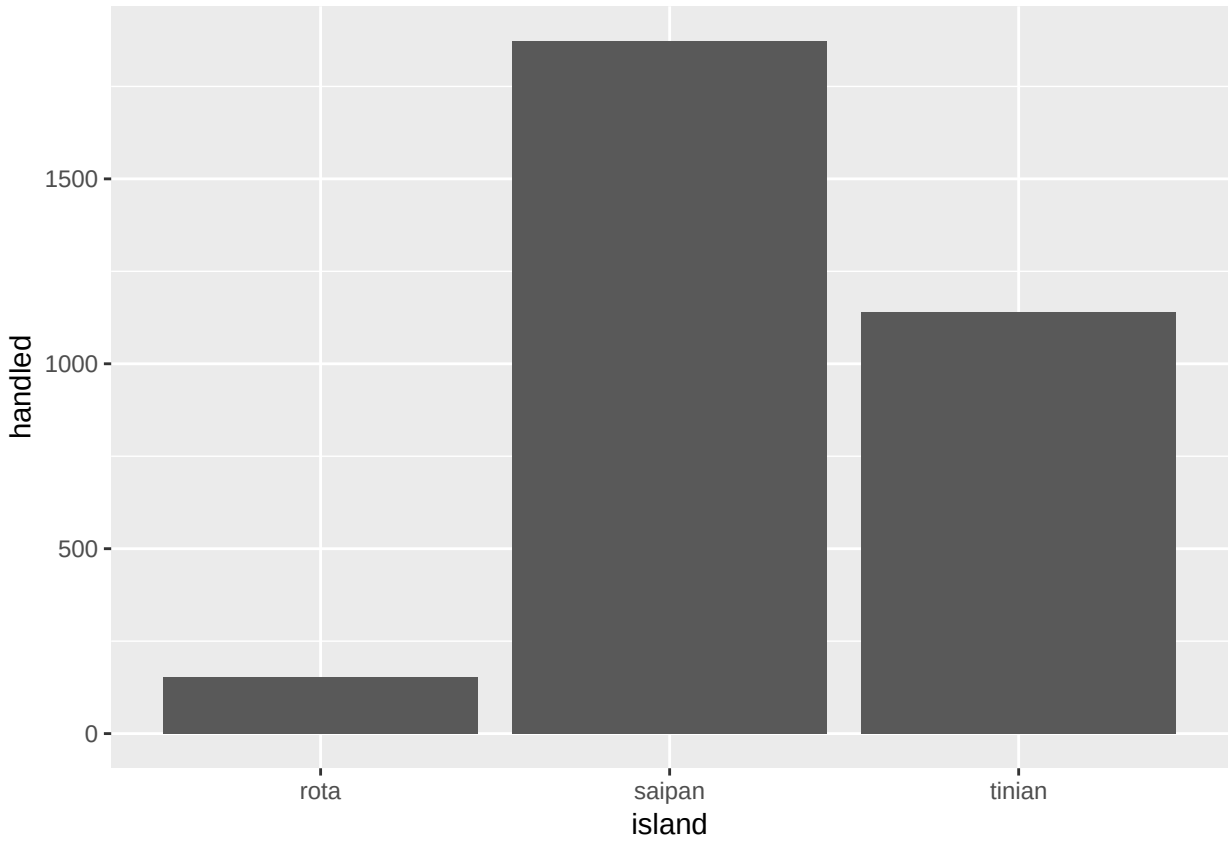



```
ggplot(pstraps, aes(island, prop))+
  geom_violin()
```

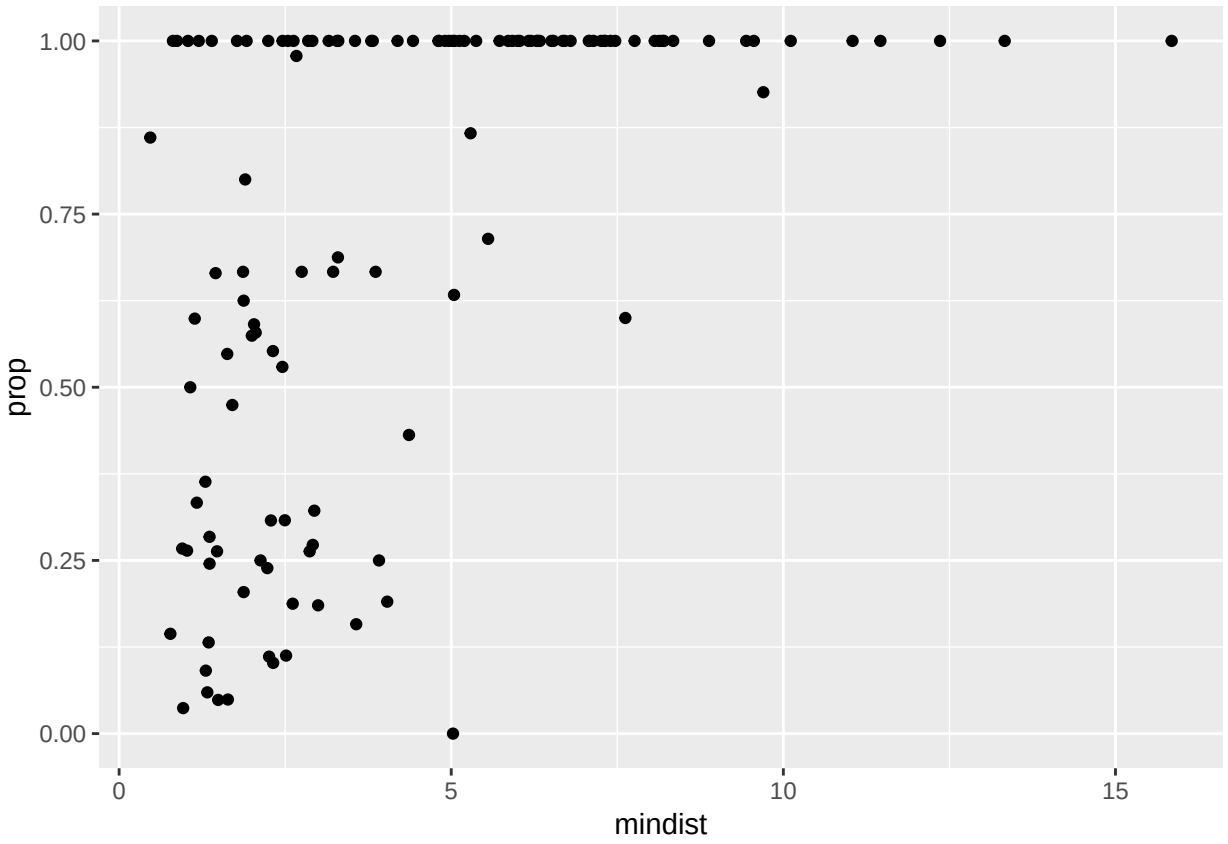



```
p1 <- ggplot(data=pstraps, aes(x=island, y=prop))+
  geom_boxplot()

p1+geom_violin() #add a second layer to p1 ggplot
```

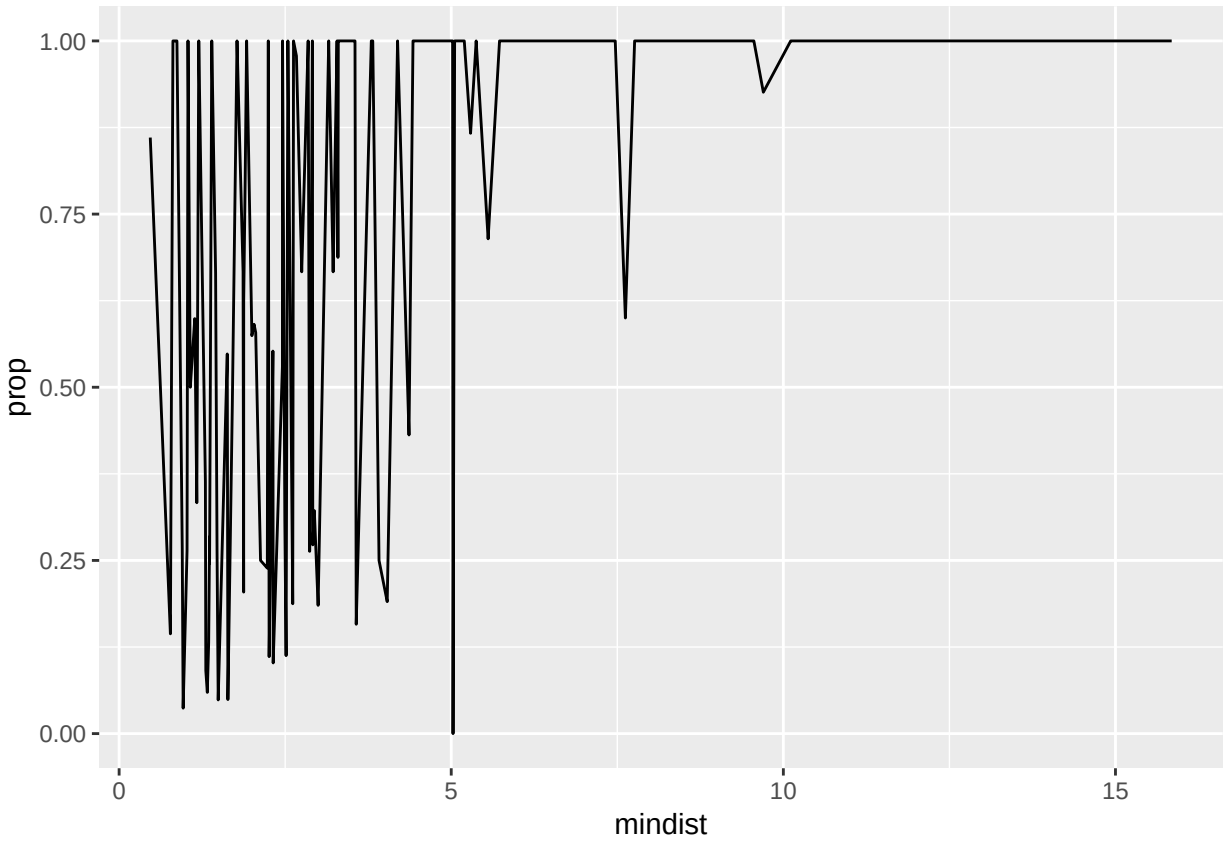

```
ggplot(data=pstraps, aes(x=island, y=handled))+
  geom_bar(stat="identity") #stat="identity" produces a bar graph of values not counts. Need x and y va
```



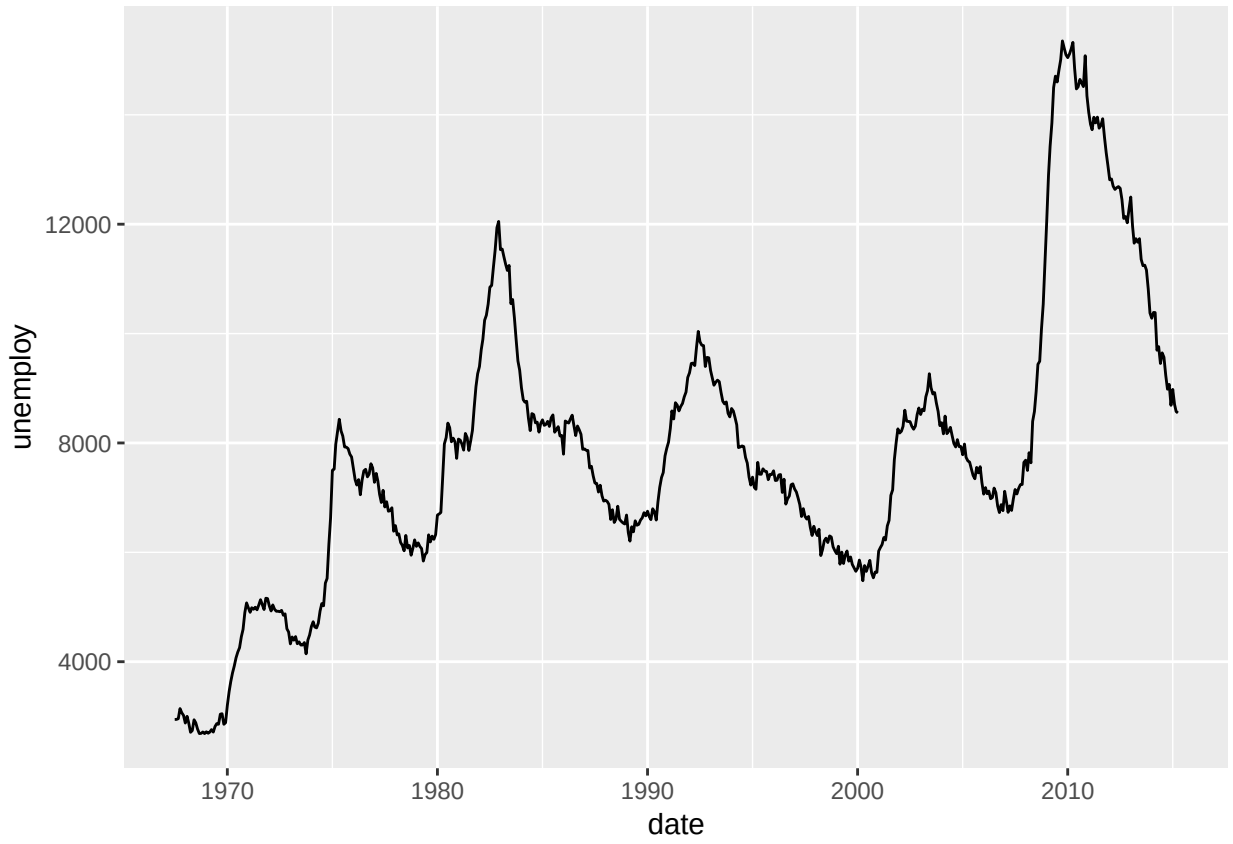

```
#this website is useful http://www.cookbook-r.com/Graphs/Bar_and_line_graphs_(ggplot2)/

ggplot(data=pstraps, aes(x=mindist, y=prop))+
  geom_point()
```



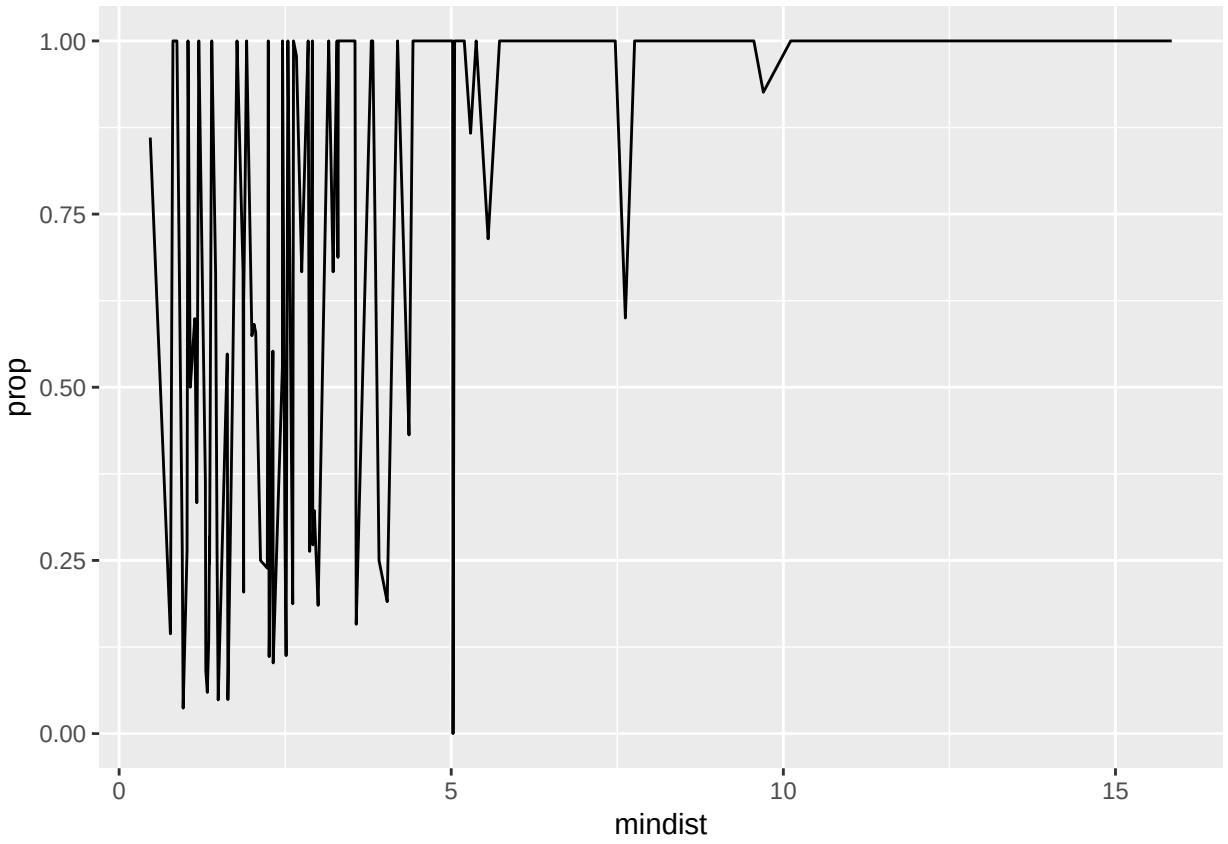

```
ggplot(pstraps, aes(mindist, prop))+
  geom_line() #not very useful. geom_line essentially connects the dots.
```



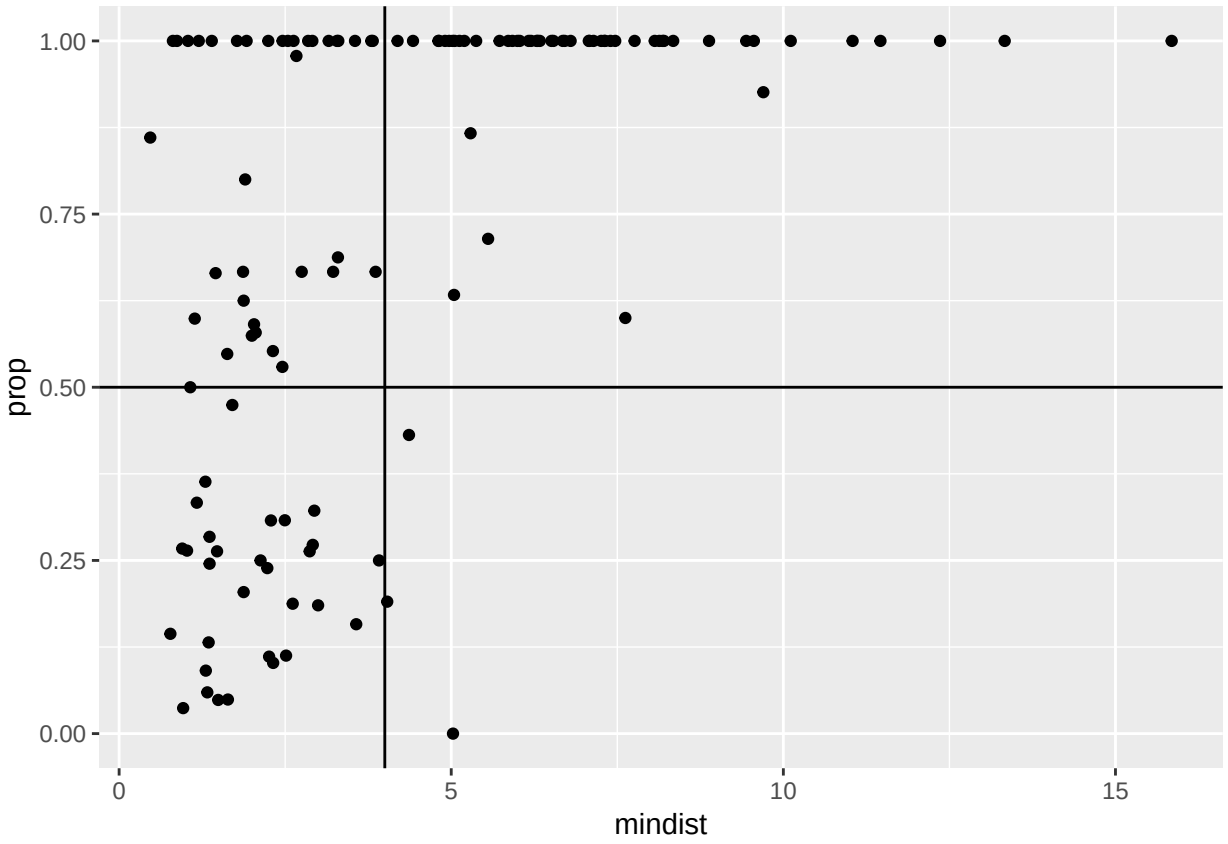

```
#here's an example that shows where geom_line is useful
ggplot(economics, aes(date, unemploy)) +
  geom_line()
```



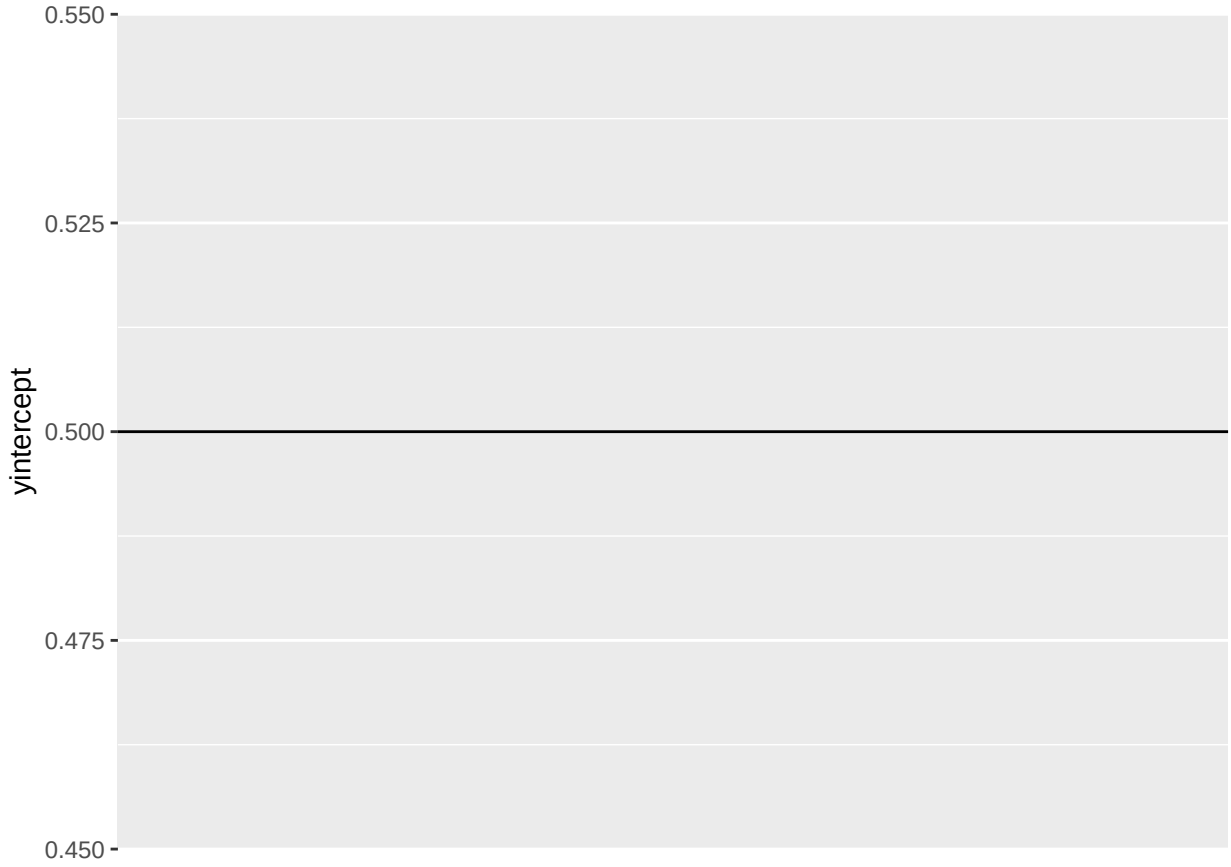

```
ggplot(pstraps, aes(mindist, prop))+
  geom_line(stat="summary", fun.y="mean") #default stat used. alternative statistical transformations a
```



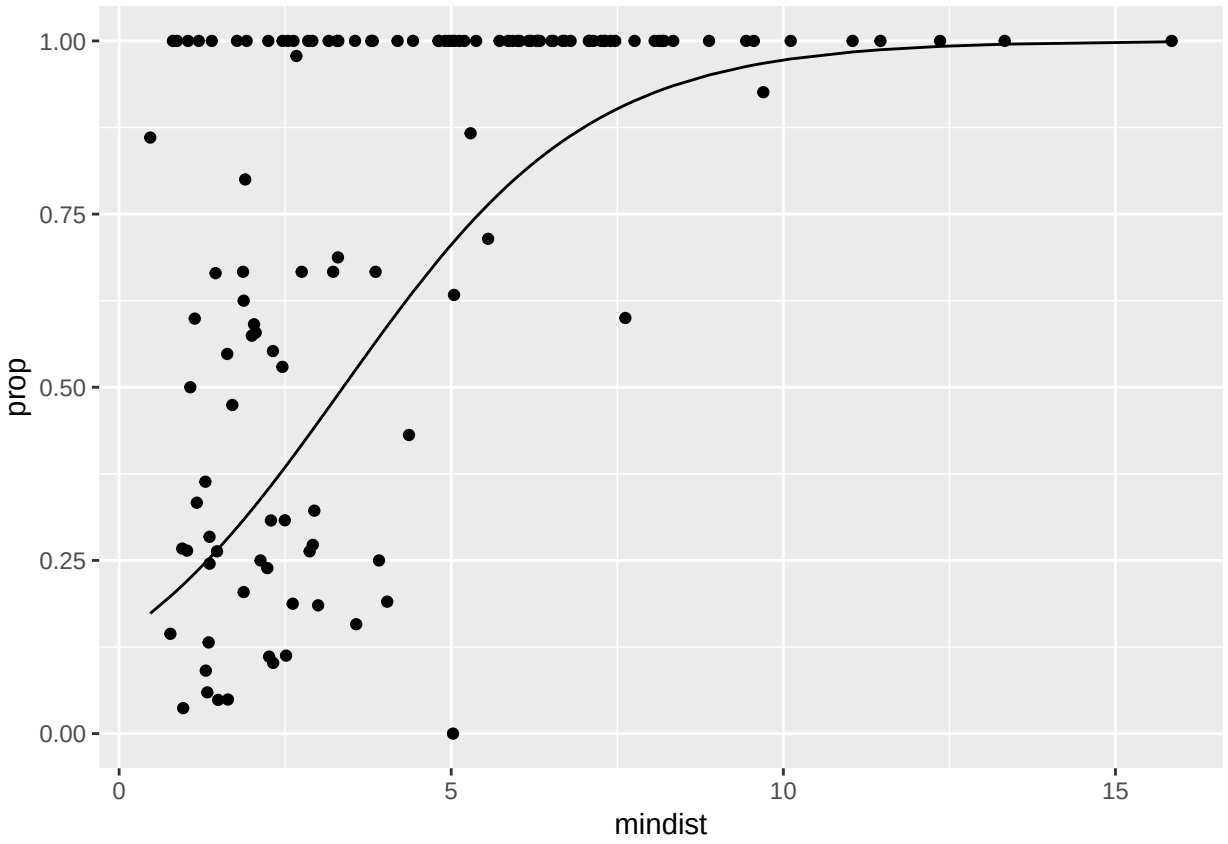

```
#add vertical and horizontal lines
ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  geom_hline(yintercept=0.5)+
  geom_vline(xintercept=4)
```




```
ggplot()+
  geom_hline(yintercept=0.5) #shortest line of code for ggplot graph?
```

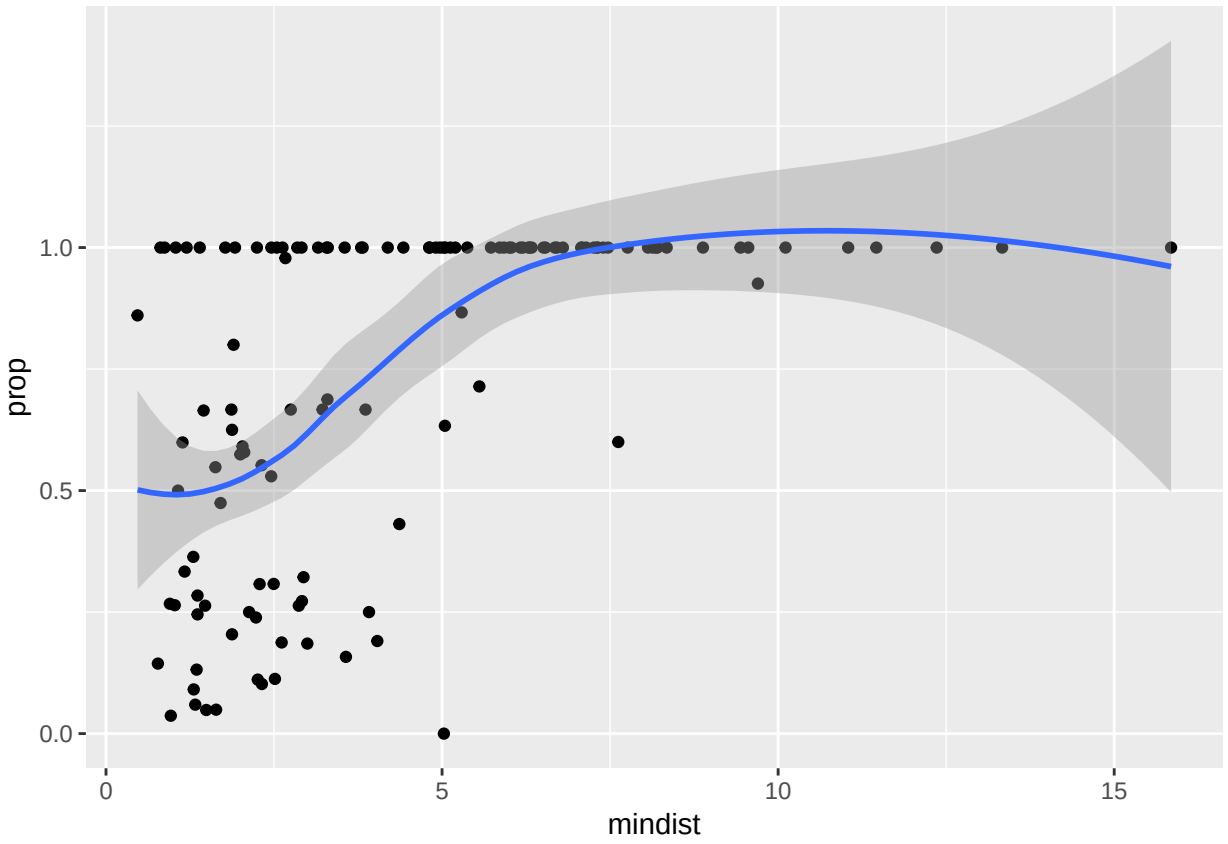



```
########################################
#2 Add prediction lines to graph
#use model results
m1 <- glm(cbind(handled, total-handled) ~ mindist, data = pstraps, family=binomial)
pstraps$pred <- predict(m1, type="response")

ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  geom_line(aes(y=pred))
```



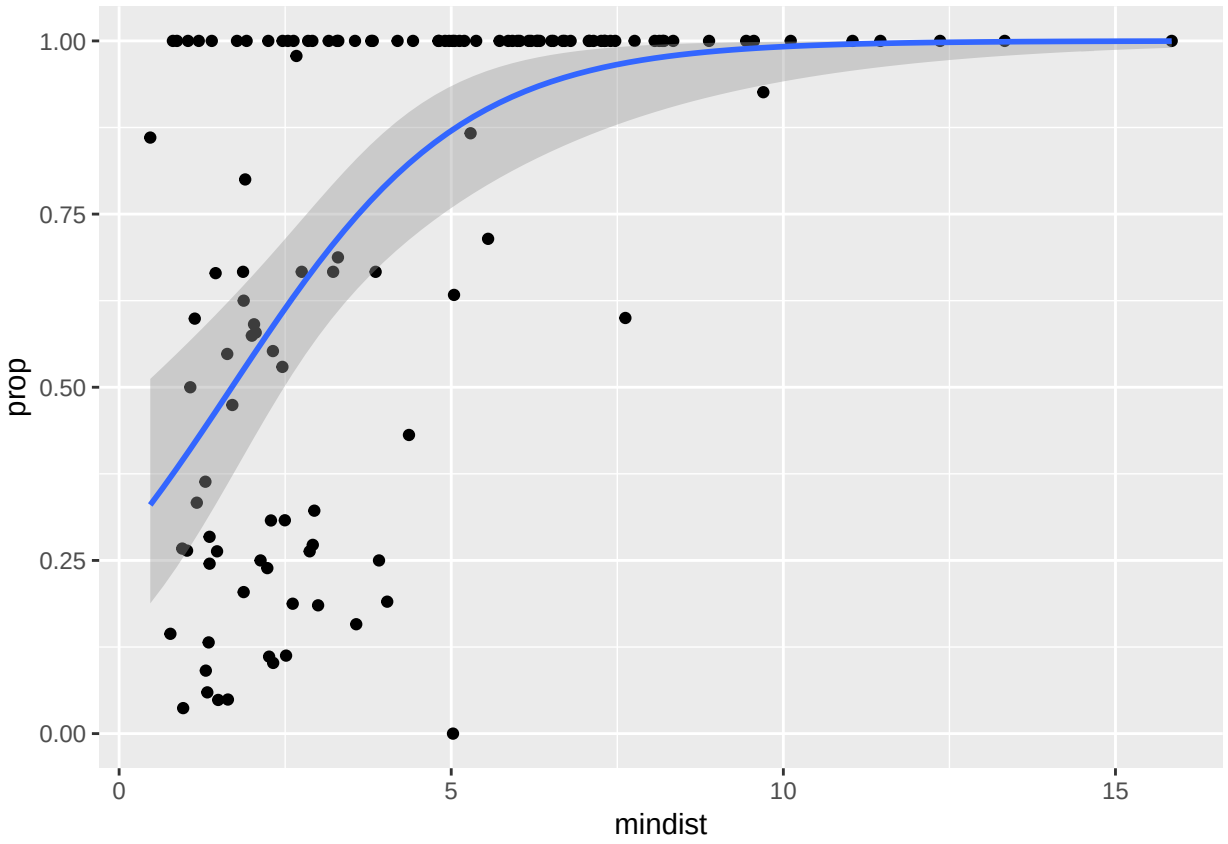

```
#use stat_smooth
ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  geom_smooth() #default method is loess, dark shaded band is +/- confidence intervals
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

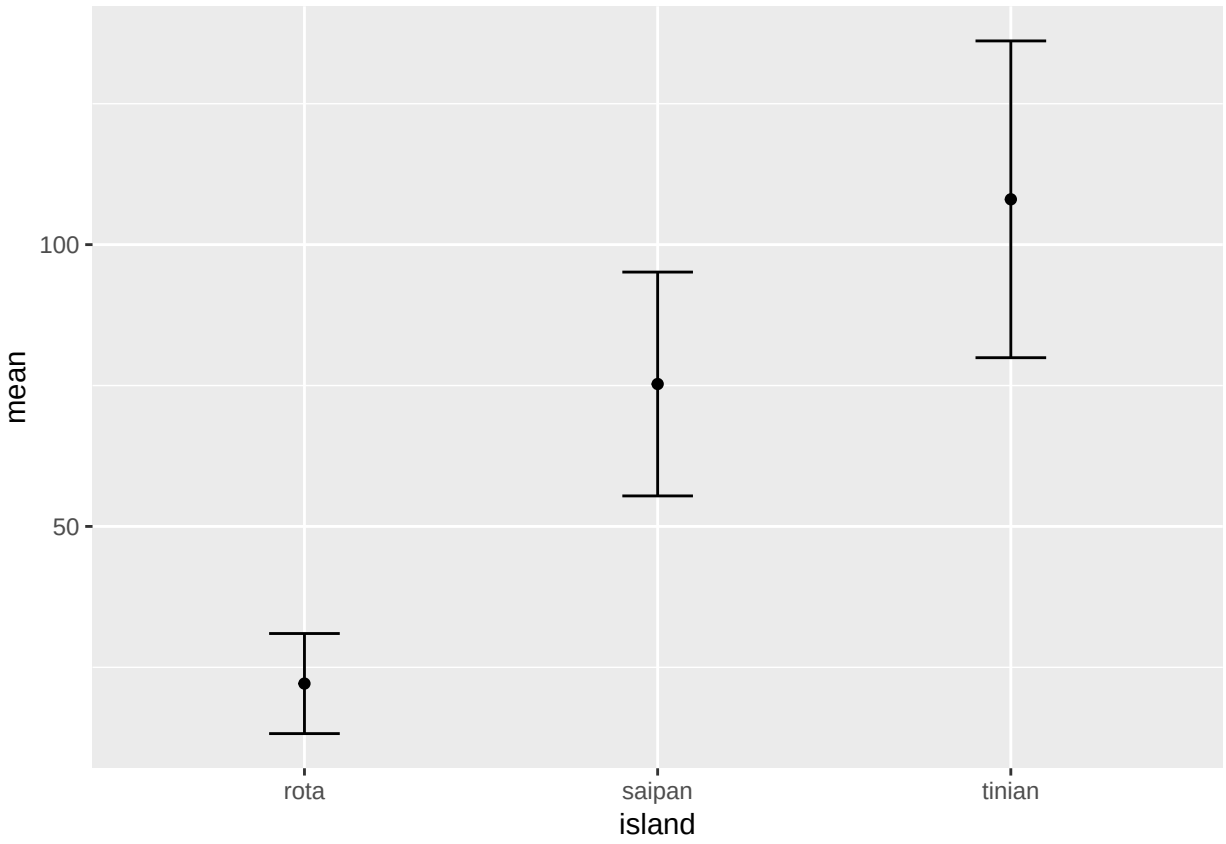



```
ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  geom_smooth(method="glm", method.args=list(family="binomial"))
```

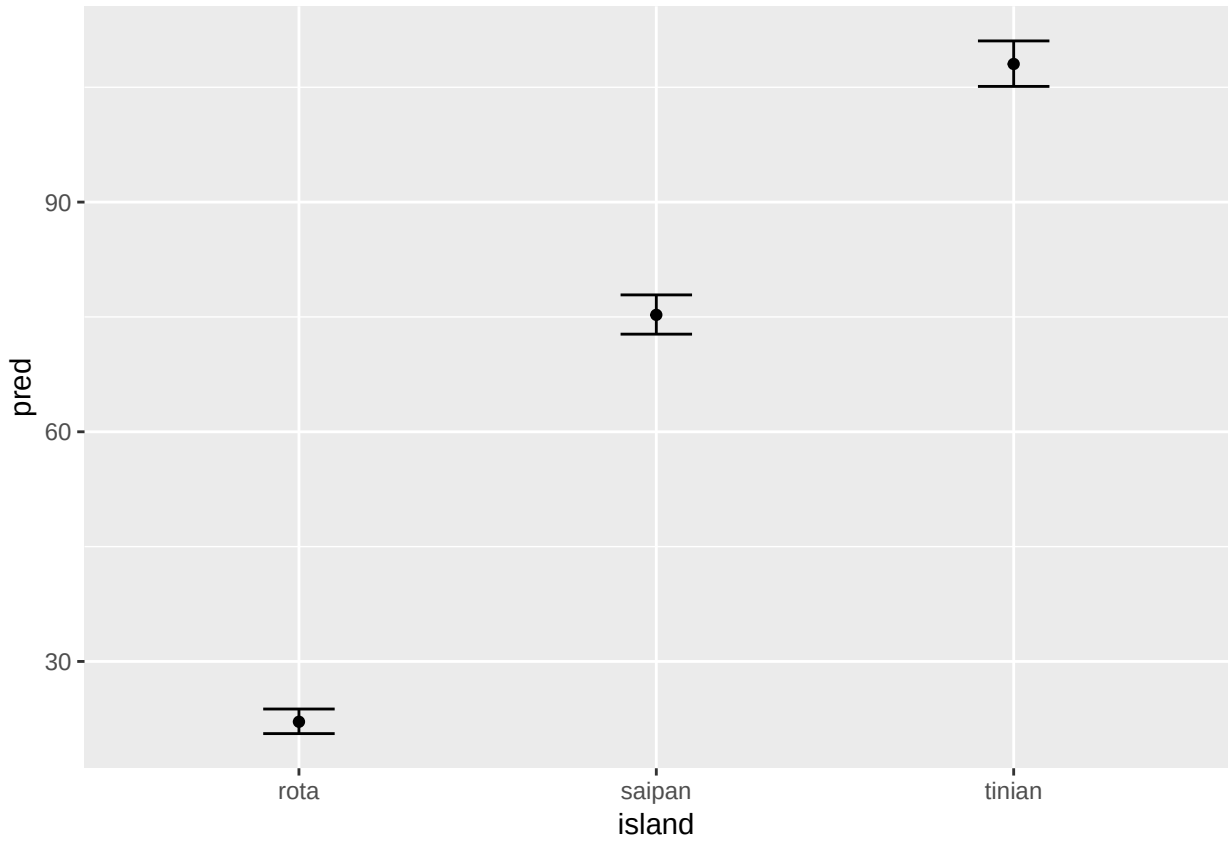
```
## Warning in eval(family$initialize): non-integer #successes in a binomial
## glm!
```



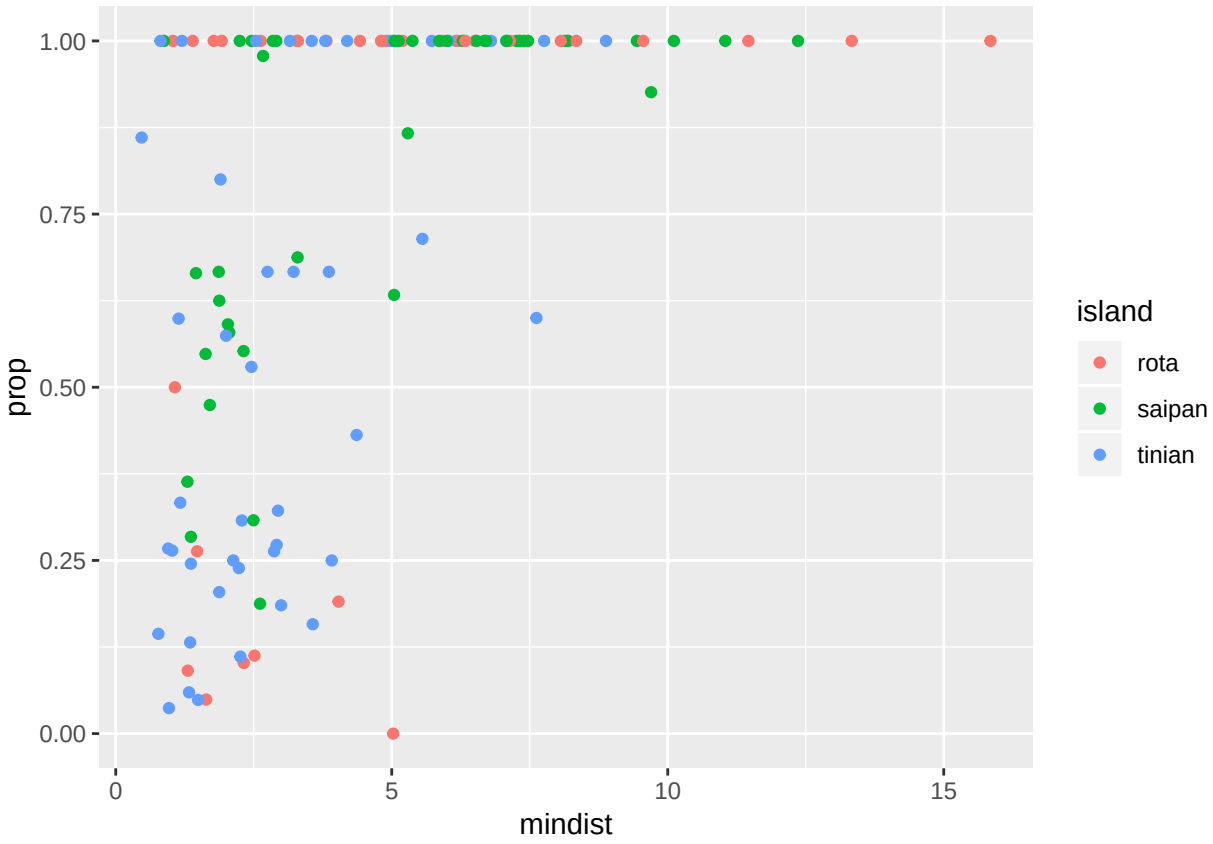

```
#I'd use predict approach above- this is to show you the geom_smooth option.

###### Add error bars to graph ###############
#Option 1- based on raw data
#First need to calculate error and create new dataframe
sumpstraps <- ddply(pstraps, c("island"), summarise,
               N    = length(total),
               mean = mean(total),
               sd   = sd(total),
               se   = sd / sqrt(N))

ggplot(sumpstraps, aes(island, mean))+
  geom_point(stat="identity")+
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=0.2)
```



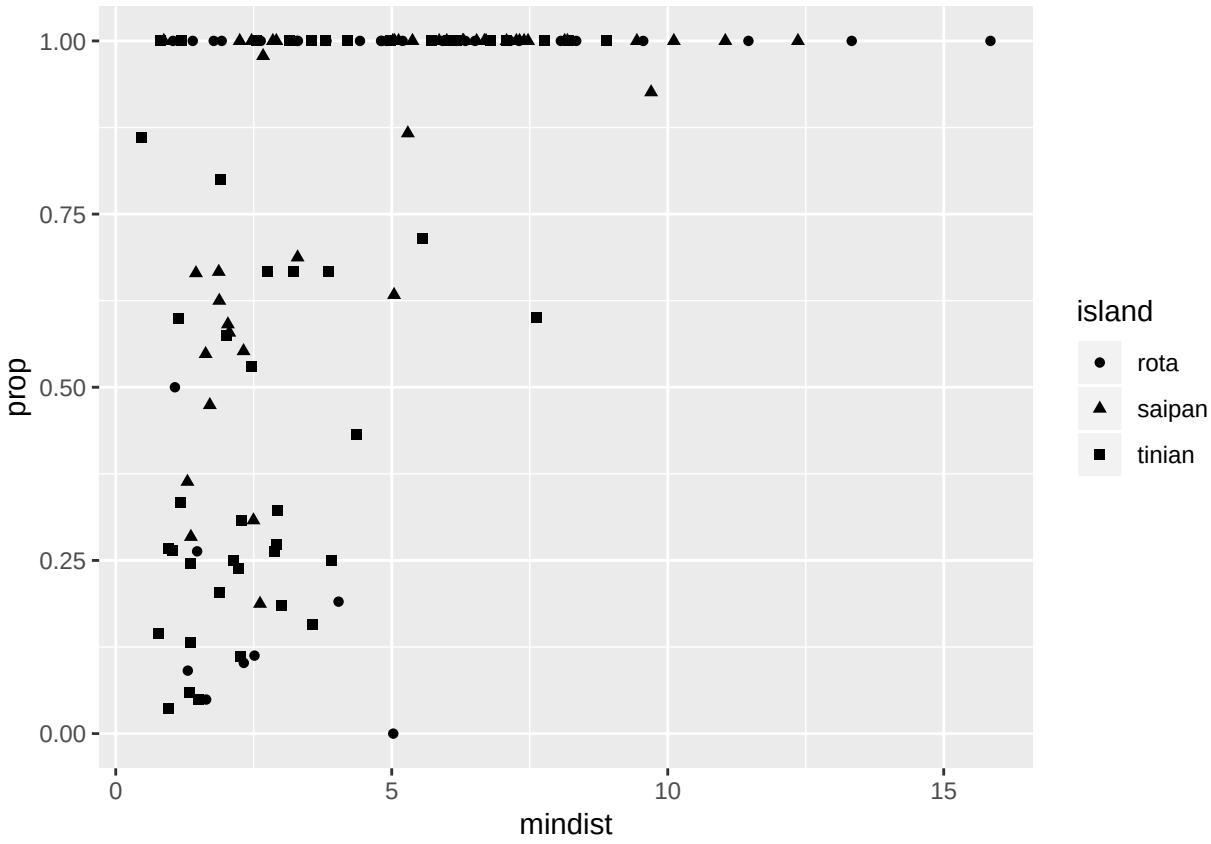

```
#####Option 2- based on model output#####
m1<-glm(total ~ island, data = pstraps, family=poisson)

#create dataframe over which to predict model results
preddata <- with(pstraps, expand.grid(island = levels(island)))

#predict model results
preddata2 <- as.data.frame(predict(m1, newdata=preddata, type="link", se.fit=TRUE))
preddata2<-cbind(preddata, preddata2)

#calculate upper and lower CI's
preddata2 <- within(preddata2, {
  pred <- exp(fit)
  lwr <- exp(fit - (1.96 * se.fit))
  upr <- exp(fit + (1.96 * se.fit))
})

ggplot(preddata2, aes(island, pred))+
  geom_point()+
  geom_errorbar(aes(ymin=lwr, ymax=upr), width=0.2)
```



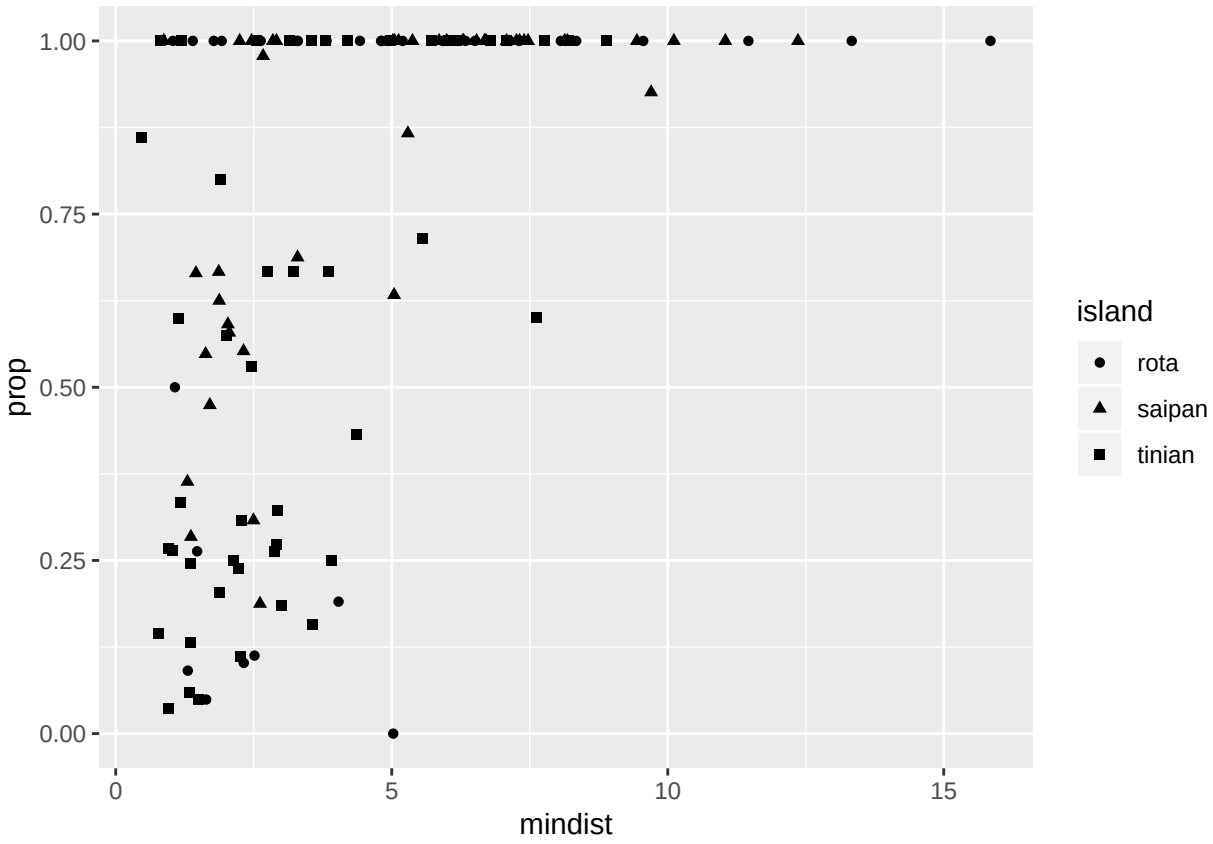

```
#Look here for similar code for binomial model
#http://stats.idre.ucla.edu/r/dae/logit-regression/

##########################################
#3 - change qualities of the points & lines
ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point()
```



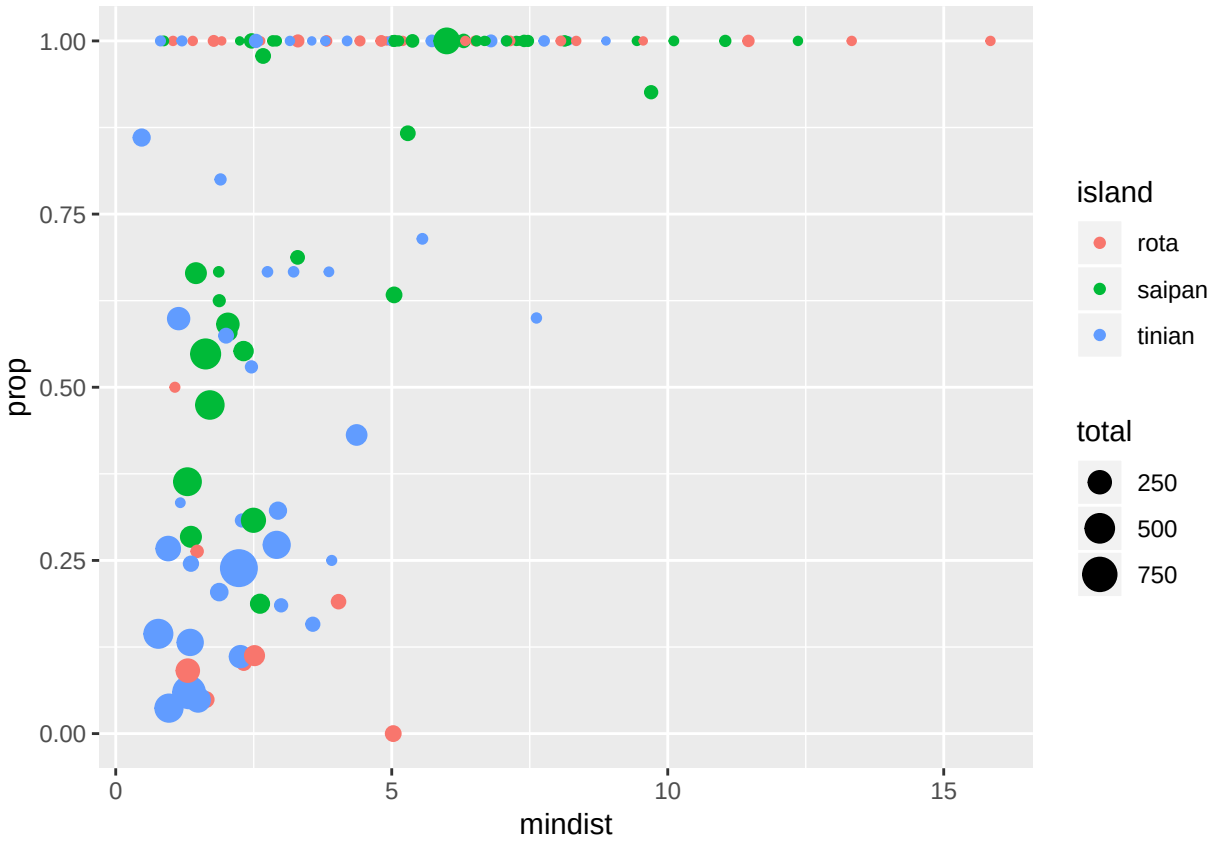

```
ggplot(pstraps, aes(mindist, prop, shape=island))+
  geom_point()
```



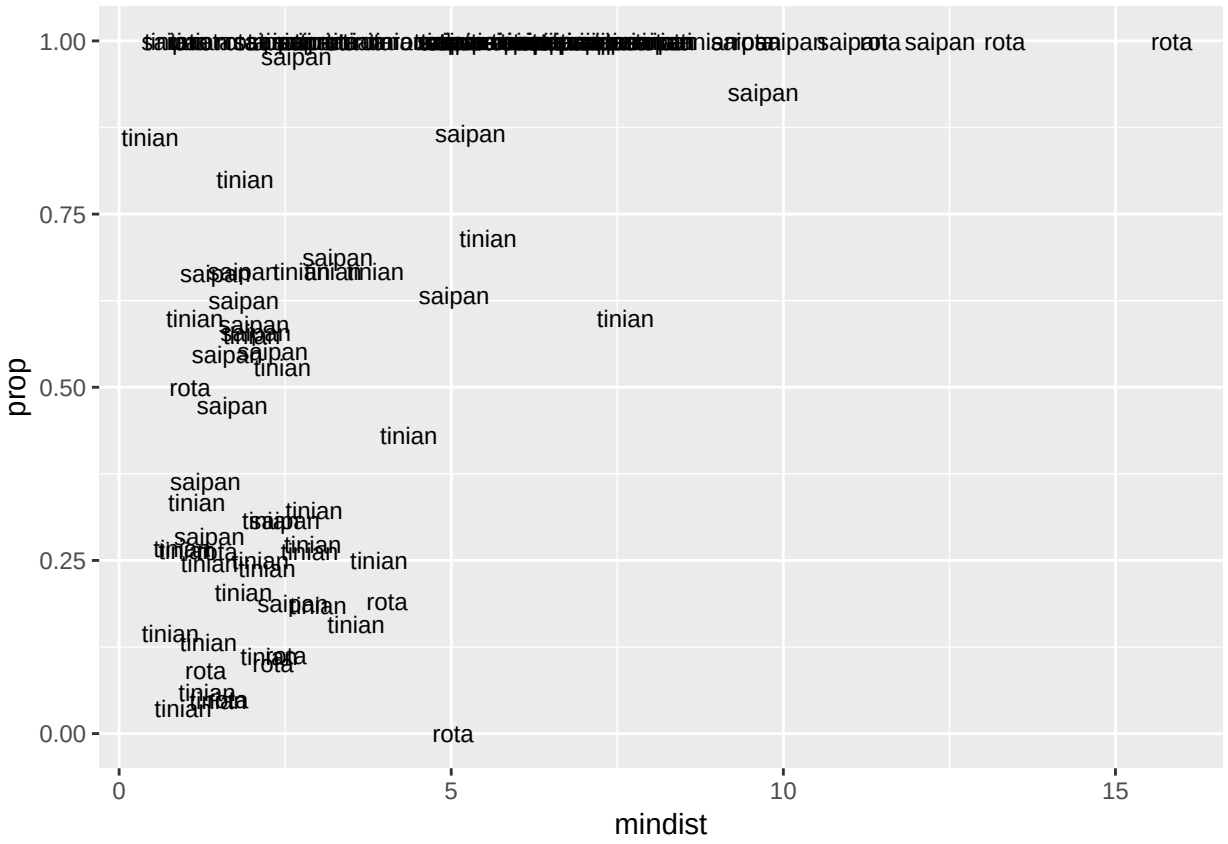

```
ggplot(pstraps, aes(mindist, prop))+
  geom_point(aes(shape=island)) #same as above- this is useful if you have several geoms on a graph, an
```



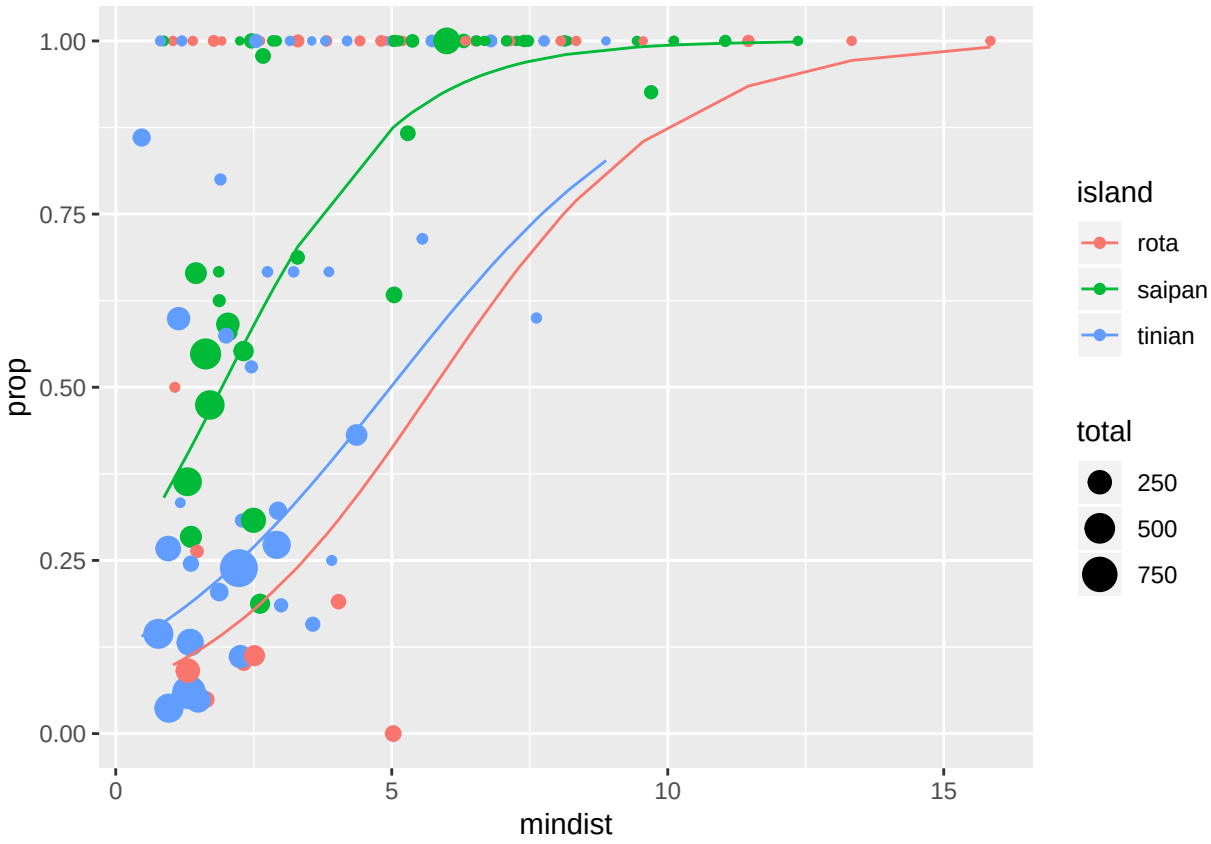

```
ggplot(pstraps, aes(mindist, prop, color=island, size=total))+
  geom_point()
```



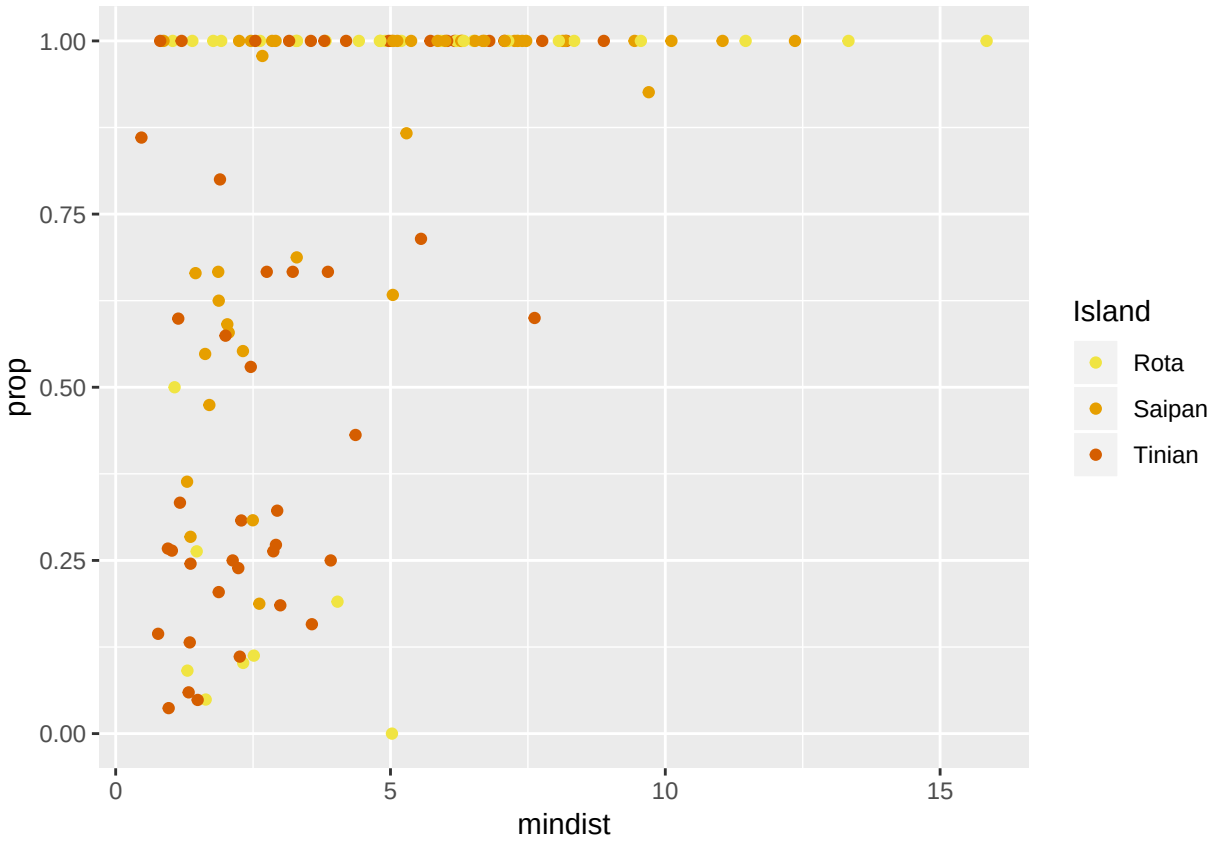

```
ggplot(pstraps, aes(mindist, prop))+
  geom_text(aes(label=island), size = 3) #not super useful here, but you can see how this might be help
```



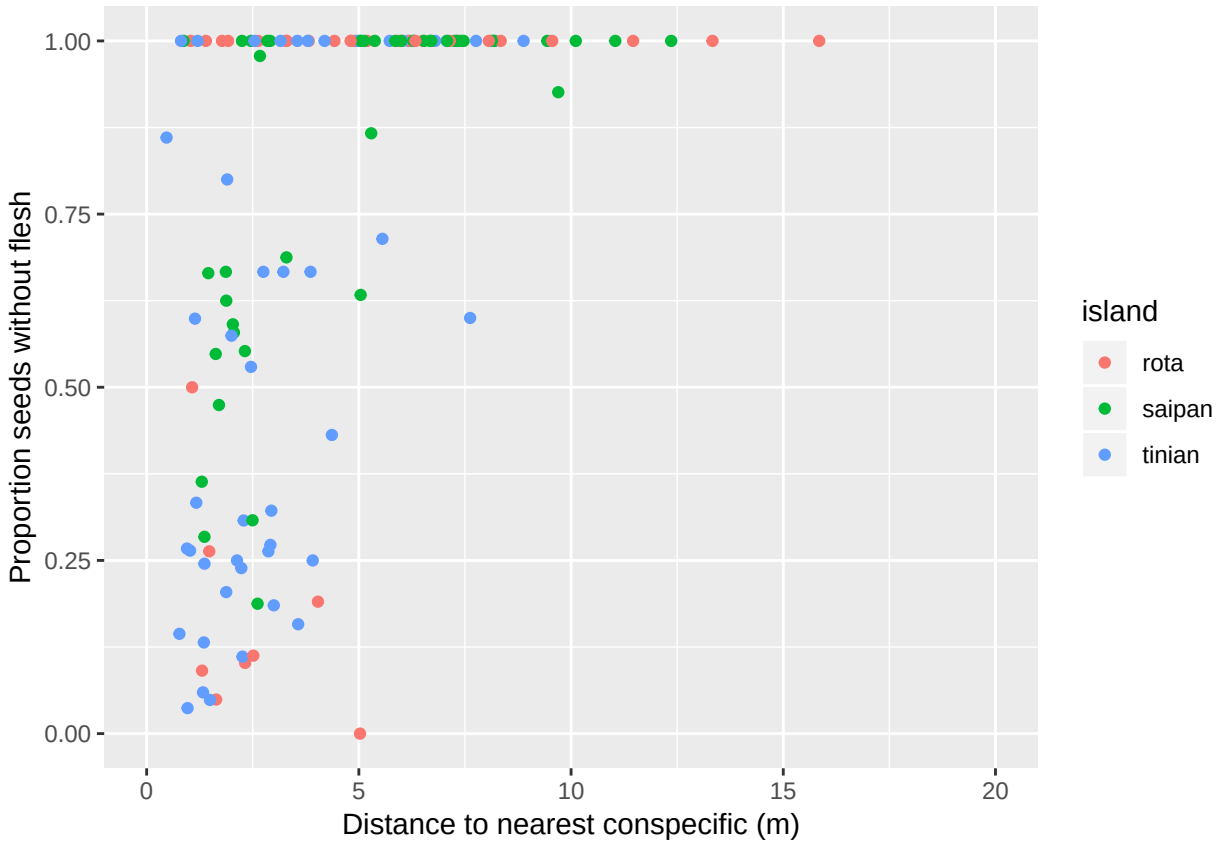

```
#change formula to include island
pstraps$pred <- predict(glm(cbind(handled, total-handled) ~ mindist*island, data = pstraps, family=bino

ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point(aes(size=total))+
  geom_line(aes(y=pred))
```



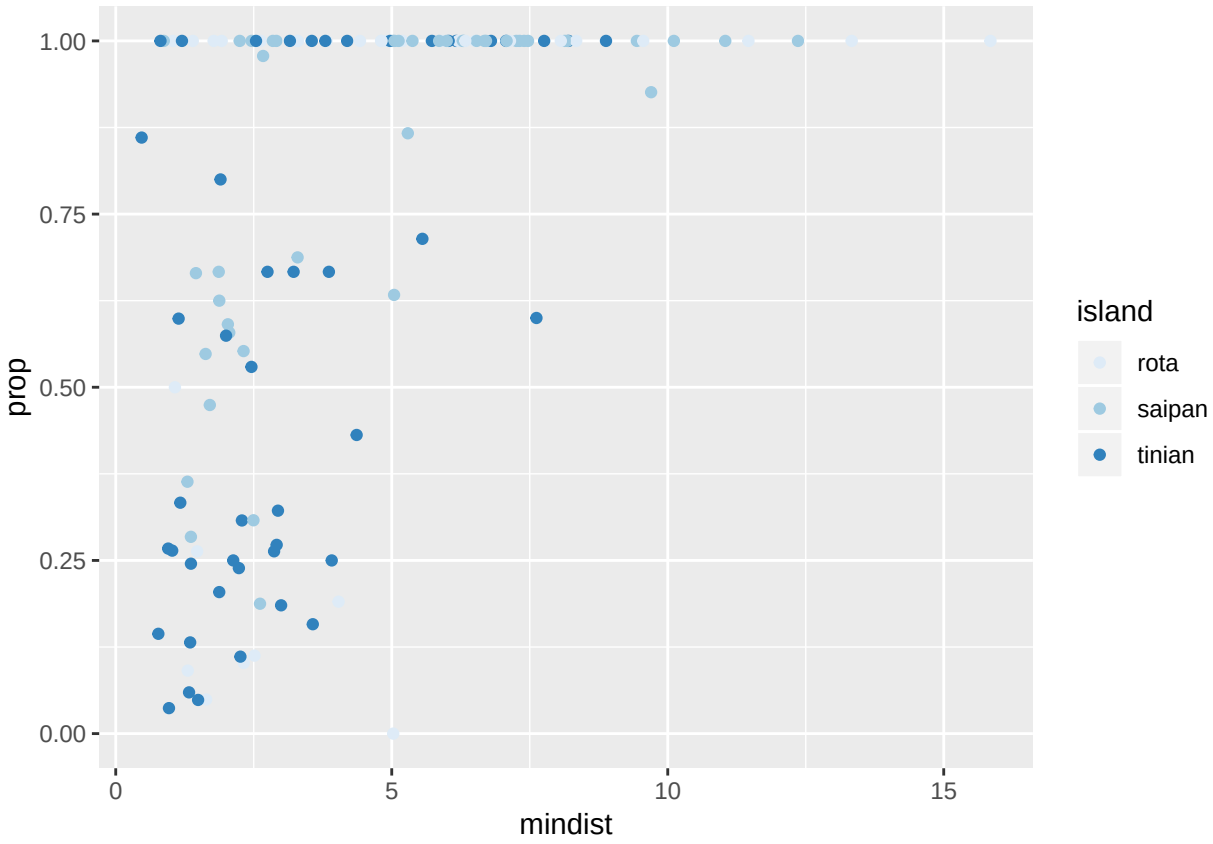

```
##########################################
#4) Scales - Controlling Aesthetic Mapping
#Scales adjust the aesthetics we specified above. Can adjust position (e.g. jitter), color, fill, shape

#general function formula is scale_<aesthetic>_<type>, where <aesthetic> is replaced by color, shape, s
#Options include: scale_color_<type>, scale_fill_<type>, scale_size_<type>, scale_shape_<type>, scale_l

#this website has a useful table for the options here: http://tutorials.iq.harvard.edu/R/Rgraphics/Rgra
#I'm going to use these colors.
#Check out http://colorbrewer2.org/ for color scheme ideas
group.colors <- c("saipan"="#E69F00", "tinian"="#D55E00FF", "rota"="#F0E442")

ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point()+
  scale_color_manual(name="Island", breaks=c("rota", "saipan", "tinian"), labels=c("Rota", "Saipan", "T
```



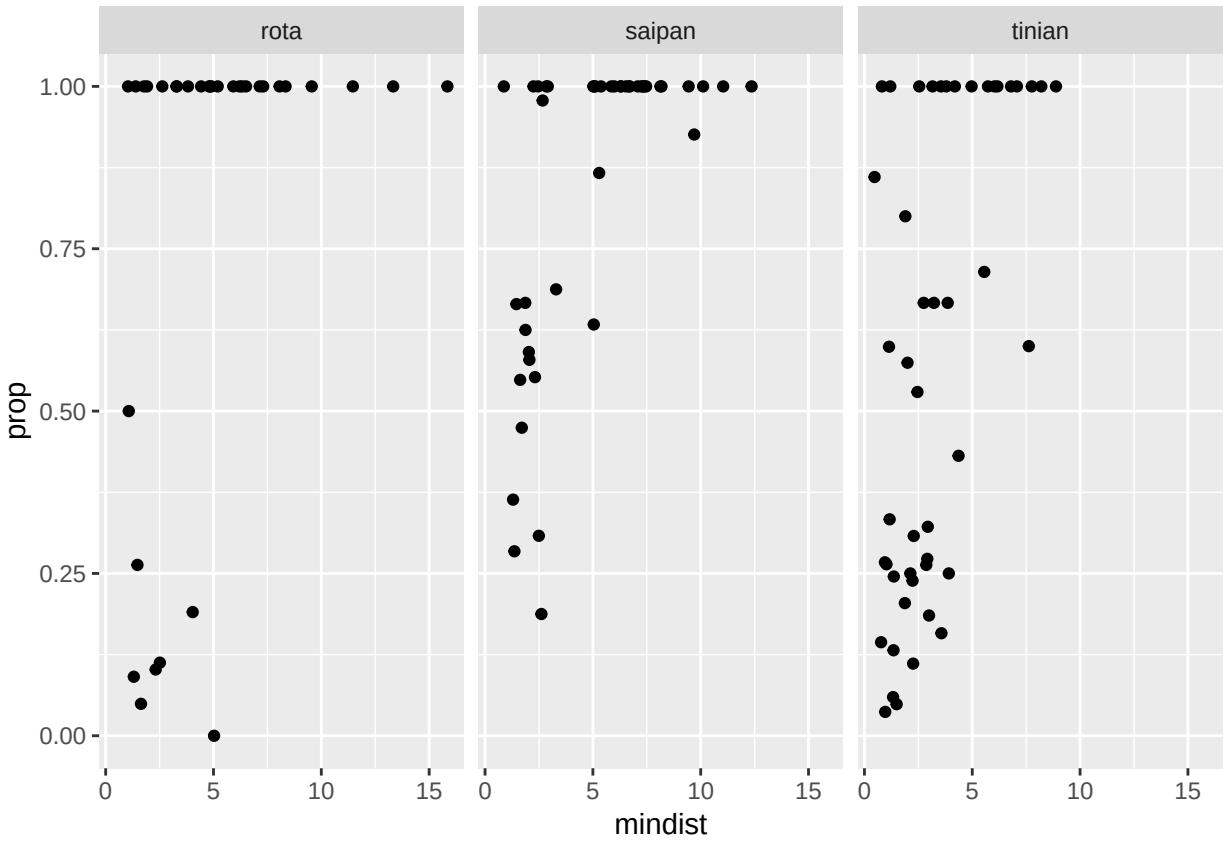

```
ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point()+
  scale_y_continuous(limits=c(0,1), "Proportion seeds without flesh") +
  scale_x_continuous(limits=c(0,20), "Distance to nearest conspecific (m)")
```



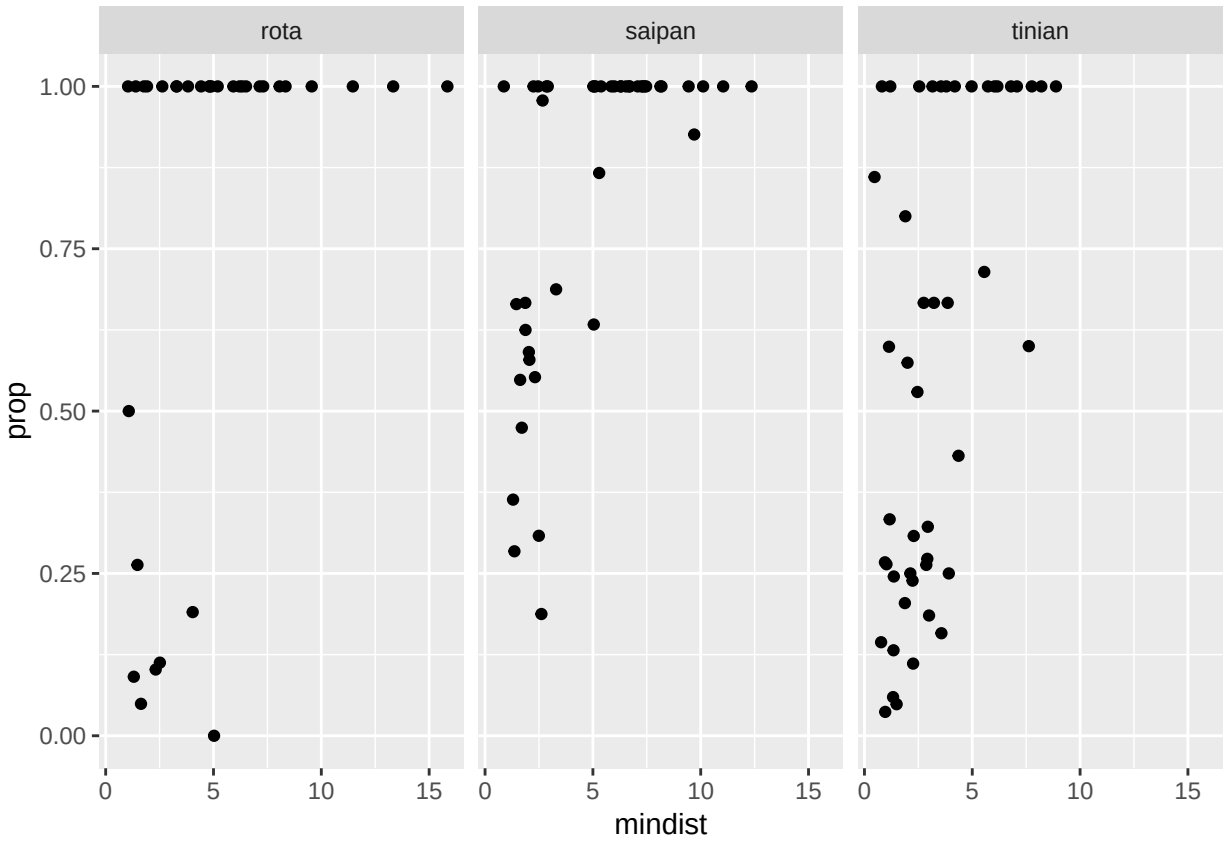

```
ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point()+
  scale_color_brewer(palette = "Blues") #use color brewer to choose colors
```



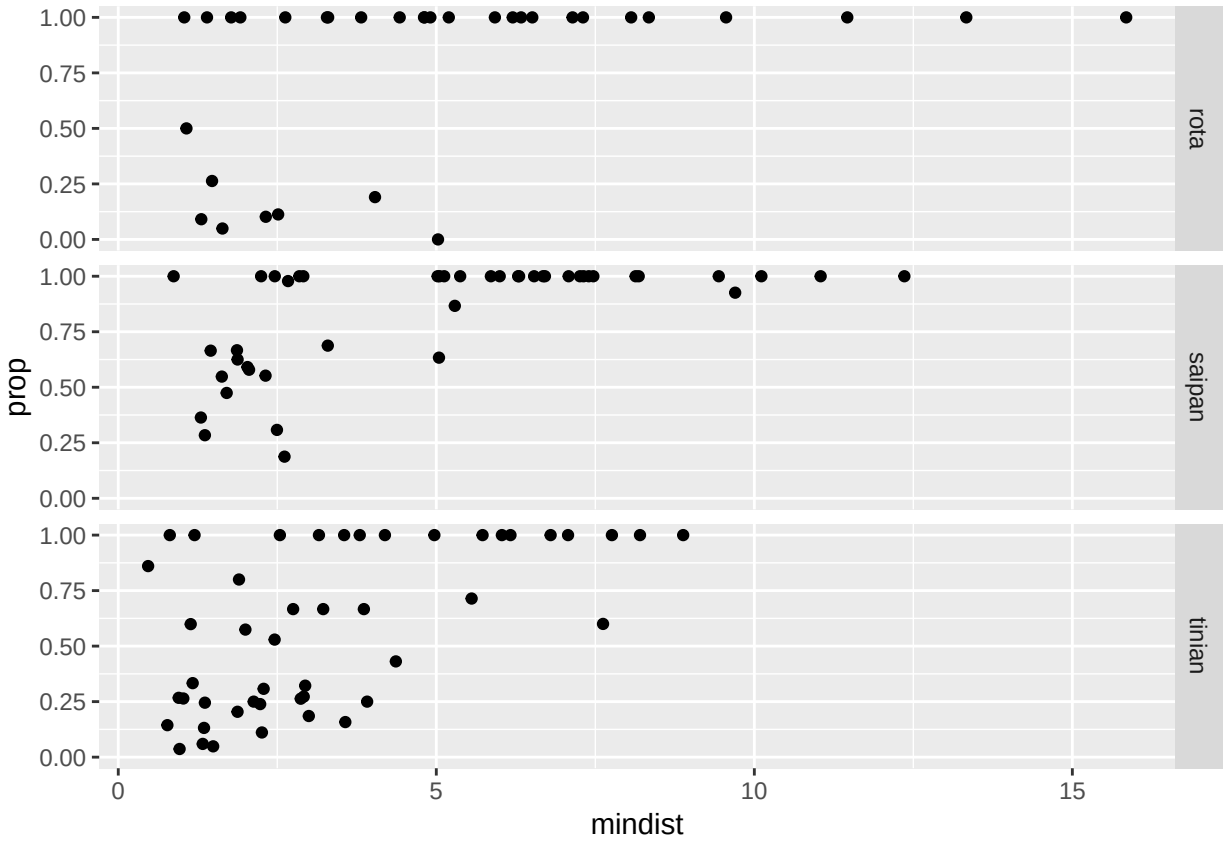

```
##########################################
#5) multiple plots from same dataset- faceting

ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  facet_wrap(~island)
```



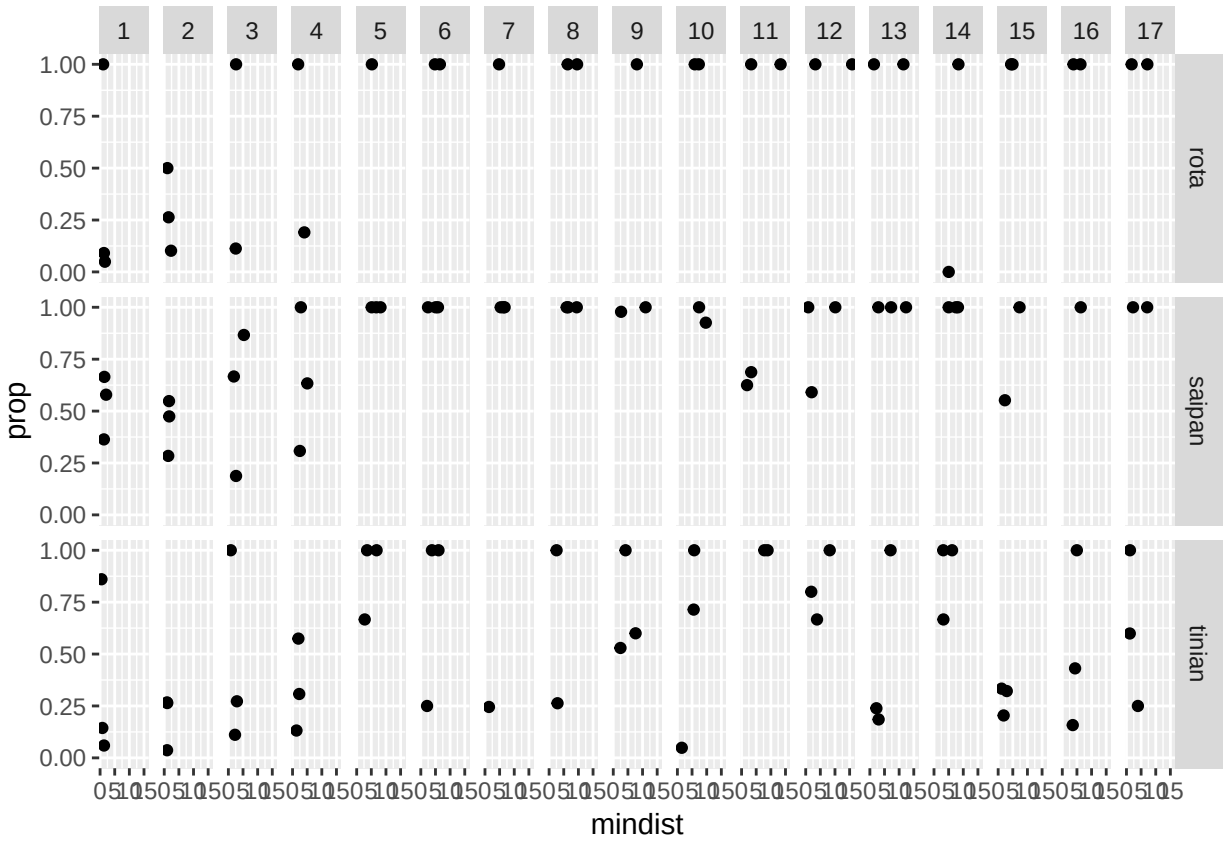

```
ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  facet_grid(.~island)
```



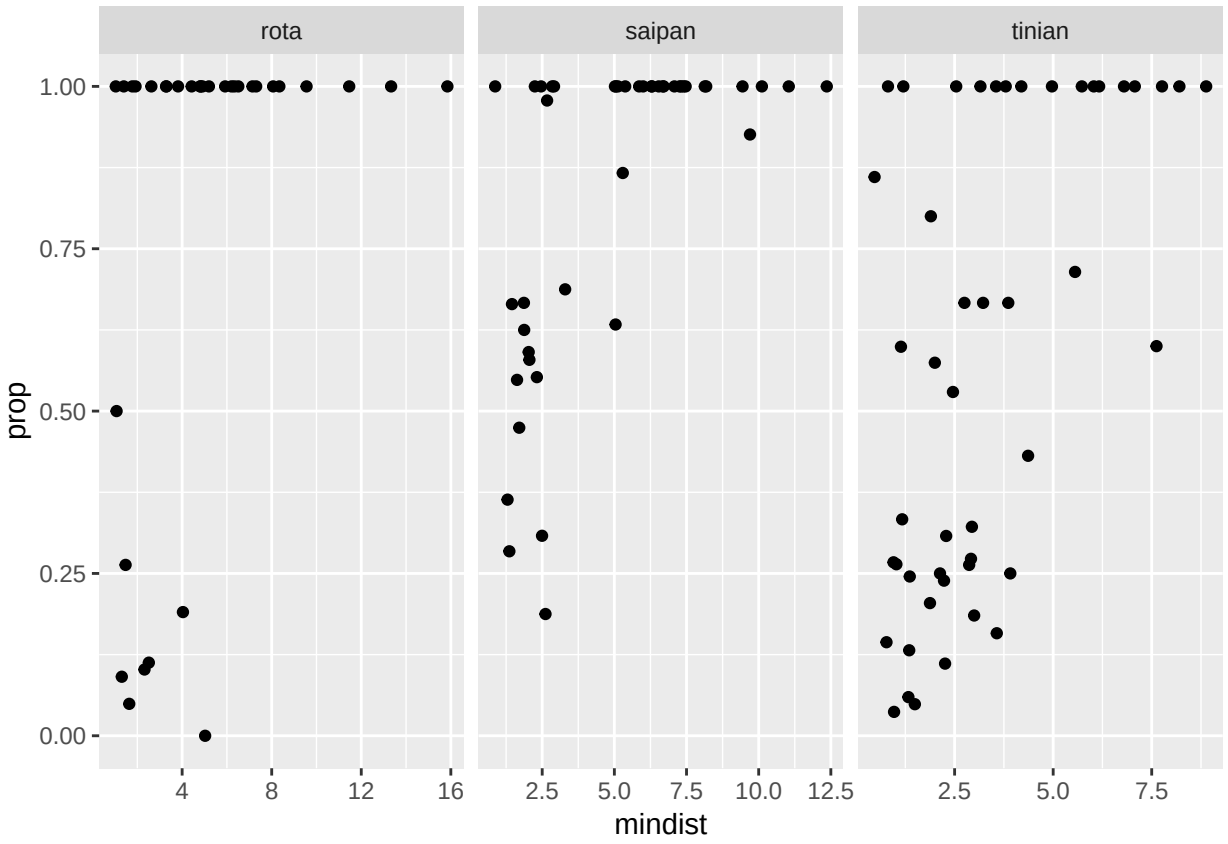

```
ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  facet_grid(island~.)
```



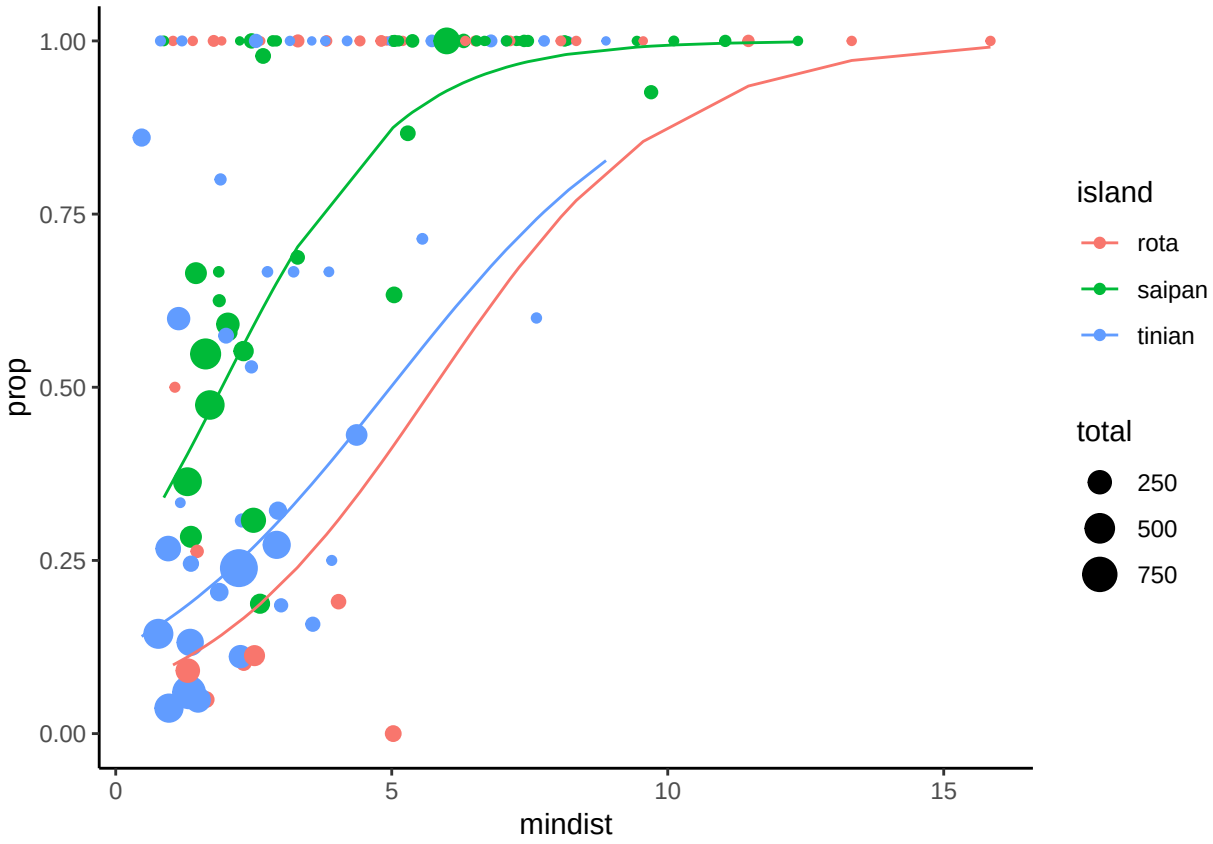

```
ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  facet_grid(island~trap) #not super useful, but you can see how a different factor might be.
```



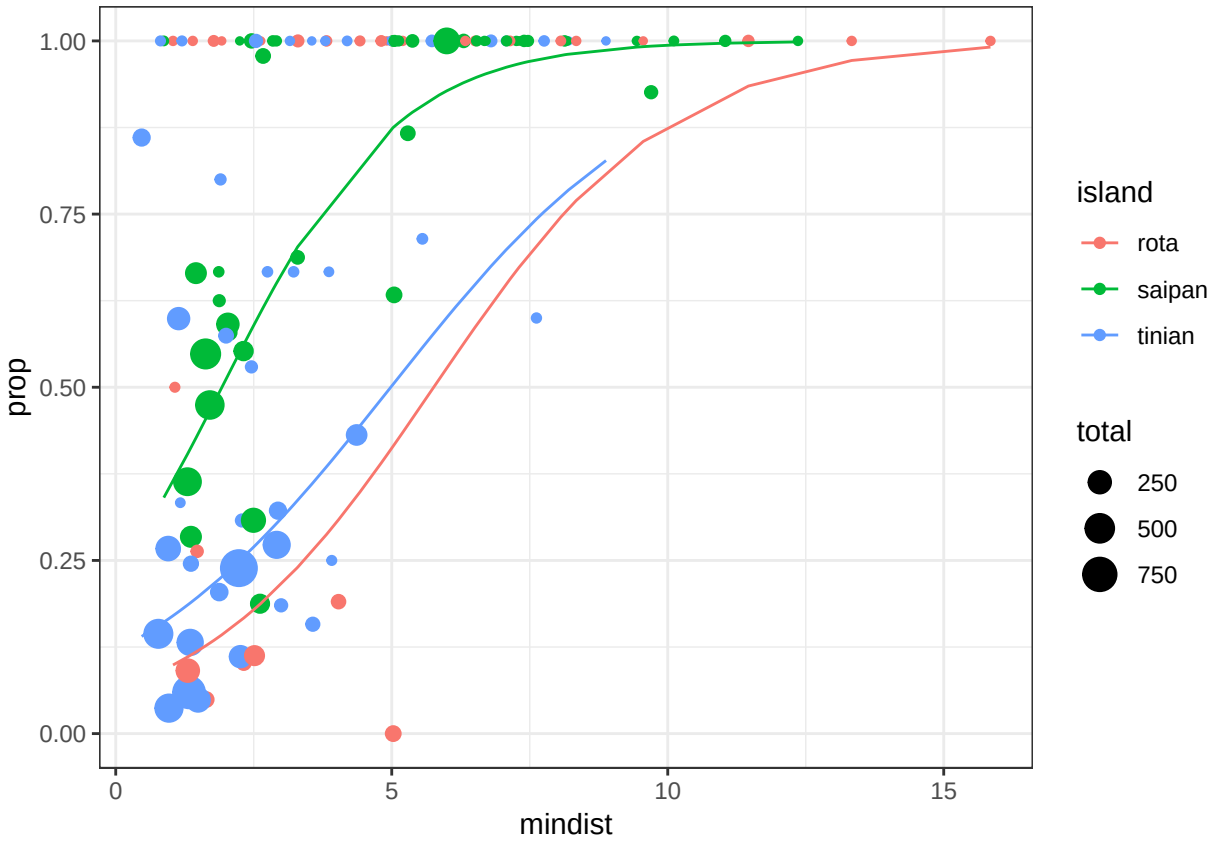

```
ggplot(pstraps, aes(mindist, prop))+
  geom_point()+
  facet_grid(.~island, scales="free_x") #can let each panel have it's own x or y axes
```



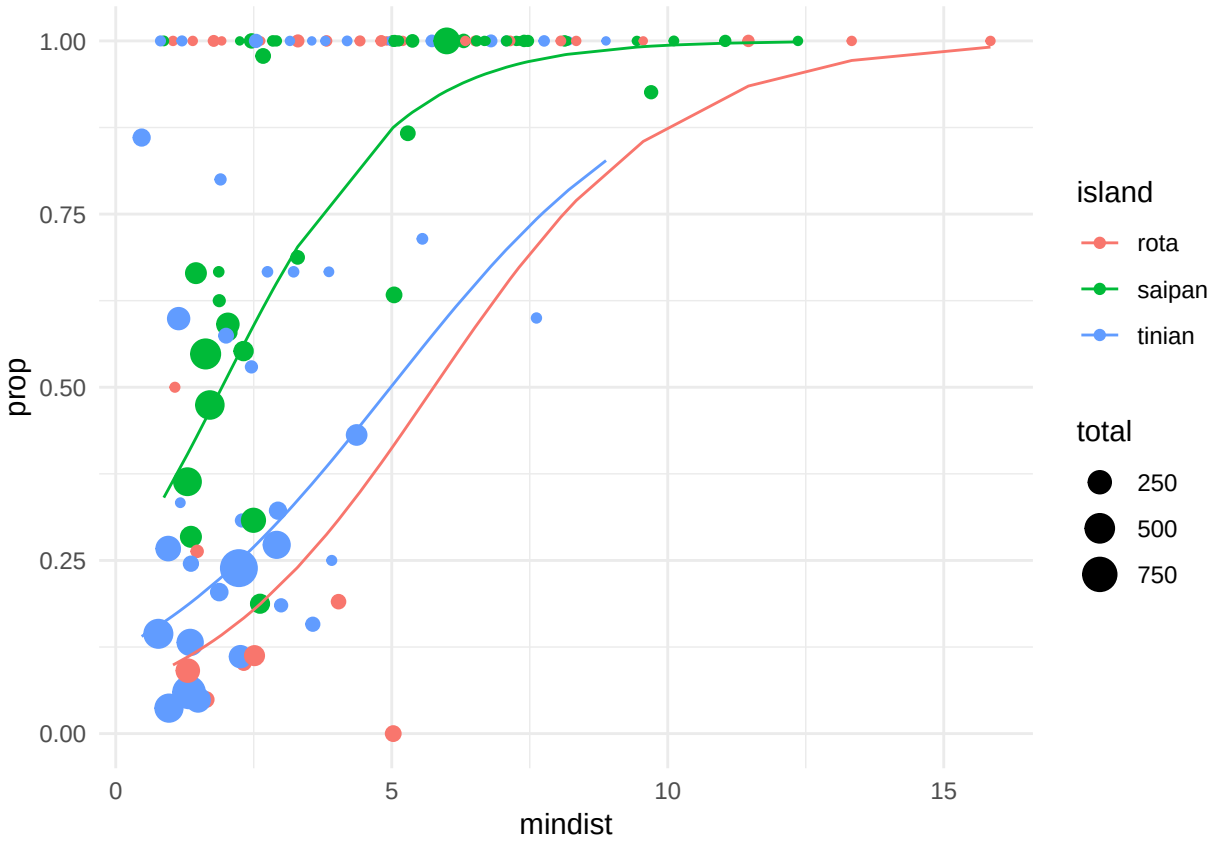

```
##########################################
#6) themes. This is where you adjust non-data plot elements such as axis labels, plot background, facet

#built-in themes

ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point(aes(size=total))+
  geom_line(aes(y=pred))+
  theme_classic()
```



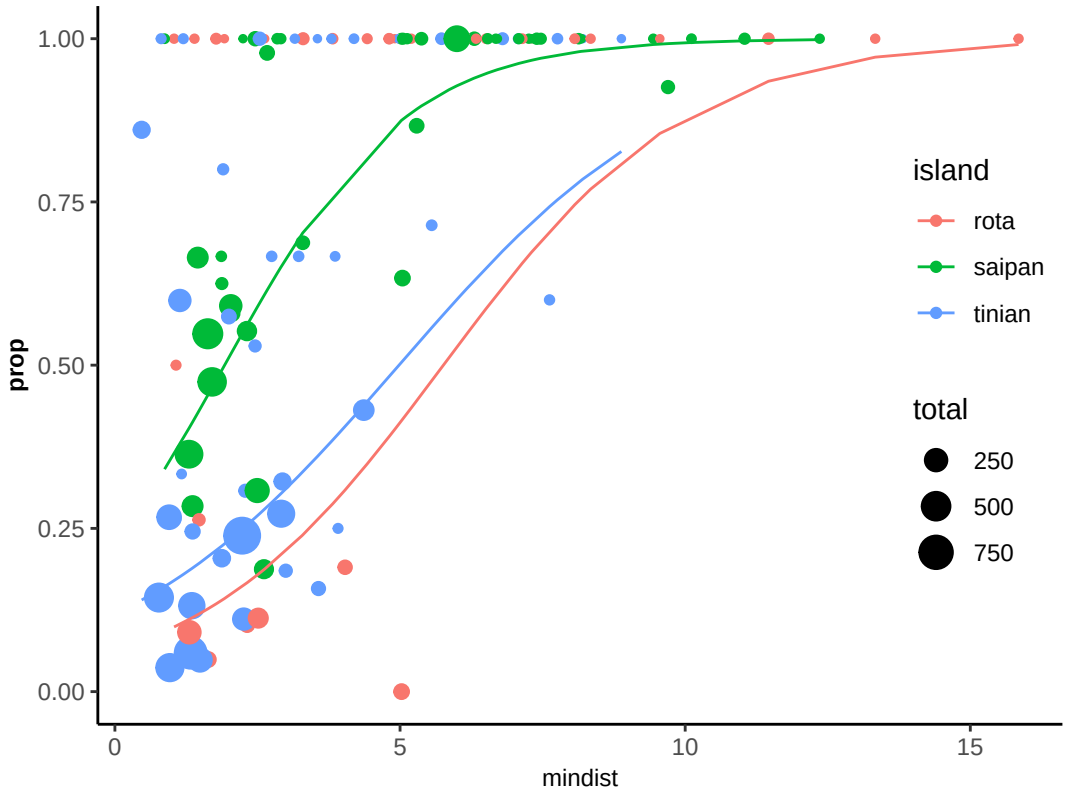

```
ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point(aes(size=total))+
  geom_line(aes(y=pred))+
  theme_bw()
```




```
ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point(aes(size=total))+
  geom_line(aes(y=pred))+
  theme_minimal()
```




```
#try theme_grey, theme_linedraw, theme_light, theme_fivethirtyeight, theme_economist, theme_few, theme_

#you can adjust aspects of the theme manually
ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point(aes(size=total))+
  geom_line(aes(y=pred))+
  theme_bw()+
  theme(axis.text.x=element_text(size=9),
        axis.text.y=element_text(size=9),
        axis.title.y=element_text(size=9, face="bold"),
        axis.title.x=element_text(size=9),
        axis.line=element_line(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        legend.position=c(0.83,0.5),
        legend.box.just = "left",
        legend.justification = "left",
        legend.key = element_blank(),
        plot.margin = unit(c(1,4,1,1), units="lines"))
```

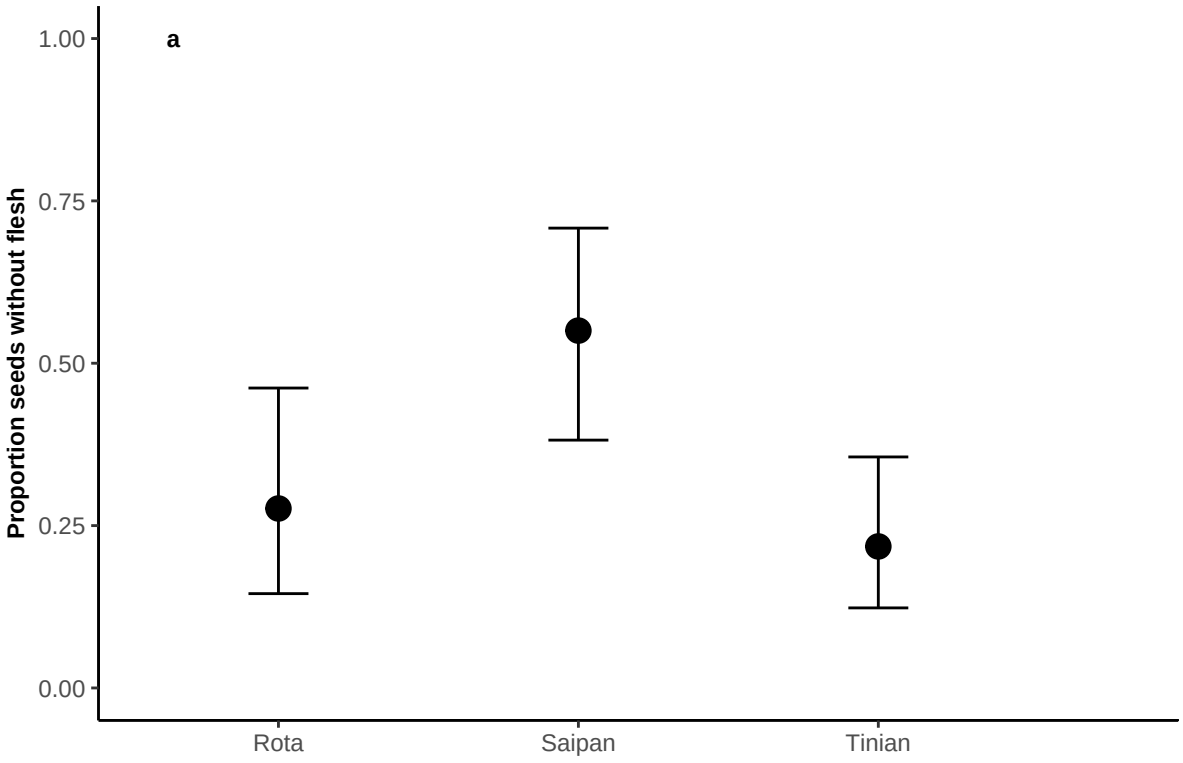



```
#you can also save your own theme.
#See here: http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html#org01640c8

##########################################
#6) Saving graphs.
mindist_graph<-ggplot(pstraps, aes(mindist, prop, color=island))+
  geom_point(aes(size=total))+
  geom_line(aes(y=pred))

ggsave("mindist.pdf", width=4, height=4, units="in")

ggsave("mindist.png", width=4, height=4, units="in")

######## Other stuff ####################

#Plot model results from mixed effects model
model1ps<-glmer(cbind(handled, total-handled)~island+(1|site), family=binomial, data=pstraps)

#need inverse logit function
invlogit<-function(x){exp(x)/(1+exp(x))}

###Graph it
#### Confidence intervals are from http://glmm.wikidot.com/faq and from this code: http://glmm.wdfiles.

#Get predicted value and upper and lower confidence intervals
#conset up prediction frame
preddata <- with(pstraps, expand.grid(island = levels(island)))

## construct model matrix
```

```
mm <- model.matrix(~island,data=preddata)

## predictions from each model; first construct linear
##  predictor, then transform to raw scale
pframe2 <- data.frame(preddata,eta=mm%*%fixef(model1ps))
pframe2 <- with(pframe2,data.frame(pframe2,prop=invlogit(eta)))
pvar1 <- diag(mm %*% tcrossprod(vcov(model1ps),mm))
tvar1 <- pvar1+VarCorr(model1ps)$site  ## must be adapted for more complex models
```

```
## Warning in pvar1 + VarCorr(model1ps)$site: Recycling array of length 1 in vector-array arithmetic is
##   Use c() or as.vector() instead.
```

```
pframe2 <- data.frame(
  pframe2
  , plo = invlogit(pframe2$eta-2*sqrt(pvar1))
  , phi = invlogit(pframe2$eta+2*sqrt(pvar1))
  , tlo = invlogit(pframe2$eta-2*sqrt(tvar1))
  , thi = invlogit(pframe2$eta+2*sqrt(tvar1))
)

#plot confidence intervals, based on fixed effects uncertainty only (plo and phi)

ggplot(pframe2, aes(x=island, y=prop))+
  geom_point(size=4)+
  geom_rangeframe(data=data.frame(x=c(1,4), y=c(0, 1)), aes(x, y))+
  scale_y_continuous(limits=c(0,1), "Proportion seeds without flesh") +
  scale_x_discrete("", labels=c("Rota", "Saipan", "Tinian"))+
  geom_errorbar(aes(ymin = plo, ymax = phi), width=0.2, size=0.5)+
  annotate("text", label = "a", x = 0.65, y = 1.00, size=3, fontface="bold")+
  theme_bw()+
  theme(axis.line = element_line(colour = "black"),
        axis.text.x=element_text(),
        axis.title.x=element_text(),
        axis.title.y=element_text(size=9, face="bold"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        plot.margin = unit(c(1,1,1,1), units="lines"))
```


a
1.00
0.75
0.50
0.25
0.00
Proportion seeds without flesh
Rota
Saipan
Tinian