Supplementary Information 1: Using the *multifunc* package for analysis of Biodiversity Ecosystem Multifunctionality Relationships in R

This is a step-by-step walk-through of the analysis of the German BIODEPTH site in the paper to illustrate the different analyses we conducted to evaluate the relationship between biodiversity and ecosystem multifunctionality.

We begin by loading the *multifunc* package. If it is not currently installed, it can be found on github, and installed using the *devtools* library.

```
library(devtools)
install_github("multifunc", username="jebyrnes")
```

```
library(multifunc)

#for plotting
library(ggplot2)
library(gridExtra)
```

Next, we load in the BIODEPTH data. We use some of the helper functions in the package to identify the column numbers of the relevant functions for later use.

```
#Read in data  to run sample analyses on the biodepth data
data(all_biodepth)

allVars<-qw(biomassY3, root3, N.g.m2,  light3, N.Soil, wood3, cotton3)
varIdx<-which(names(all_biodepth) %in% allVars)
```

As we want to work with just the German data, we next subset down the data to just Germany. We then look at the whether species were seeded or not, and determine which species were indeed seeded in Germany. If a column of species is full of zeroes, then it was not seeded in these plots. We will need the column indices of relevant species for later use in the overlap analyses.

```
#######
#Now, specify what data we're working with - Germany
#and the relevant variables we'll be working with
#######
germany<-subset(all_biodepth, all_biodepth$location=="Germany")


vars<-whichVars(germany, allVars)
species<-relevantSp(germany,26:ncol(germany))
spIDX <- which(names(germany) %in% species) #in case we need these
```

Single Function Approach

First, we will demonstrate the qualitative single function approach. As we're using *ggplot2* for plotting, we'll use *reshape* to melt the data into something suitable to make a nice faceted plot of each relationship.

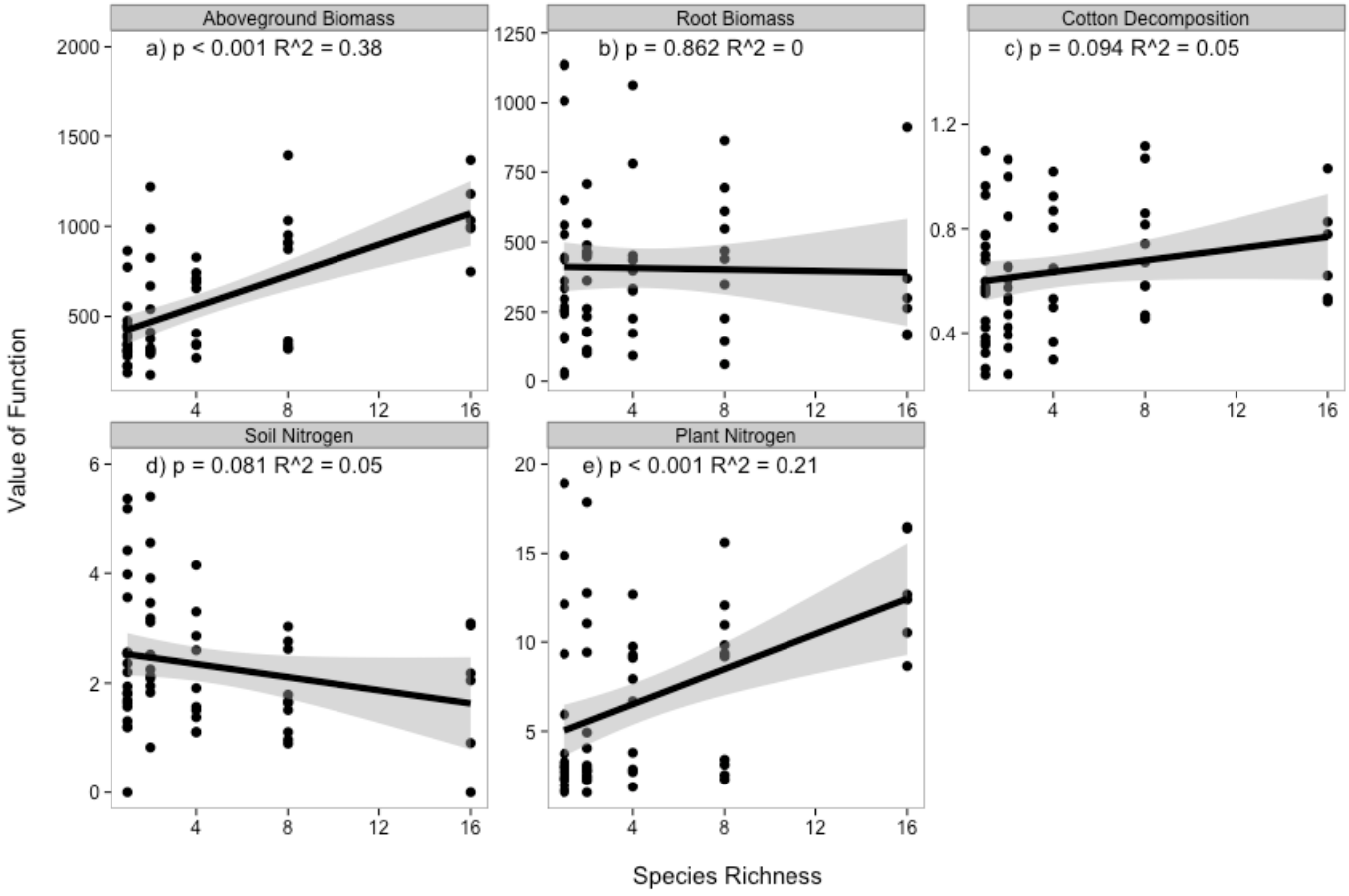
```
germanyForPlotting<-melt(germany[,c(8,which(names(germany) %in% vars))],
 id.vars="Diversity")
germanyForPlotting$variable <- factor(germanyForPlotting$variable)

#make the levels of the functions into something nice for plots
levels(germanyForPlotting$variable) <- c('Aboveground Biomass', 'Root
Biomass', 'Cotton Decomposition', 'Soil Nitrogen', 'Plant Nitrogen')
```

Nest, as we want to display additional information about the fit of diversity to each function, we'll need to iterate over each function, derive a fit, and then make some labels to be used in the eventual plot.

```
germanyFits <- dlply(germanyForPlotting, .(variable), function(x) lm(value ~
 Diversity, data=x))
germanyLabels <- data.frame(variable = levels(germanyForPlotting$variable),
                            Diversity=7, value=c(2000, 1200, 1.5, 6, 20),
                            lab=paste(letters[1:5], ")", sep=""),
                            r2 = sapply(germanyFits, function(x)
 summary(x)$r.squared),
                            p = sapply(germanyFits, function(x) anova(x)
[1,5])
                            )

germanyLabels$labels <- with(germanyLabels, paste(lab, "p =",
 round(p,3),expression(R^2), "=", round(r2,2),   sep=" "))
germanyLabels$labels <- gsub("p = 0 ", "p < 0.001 ", germanyLabels$labels)
```

Finally, we will put it all together into a single plot.

```
ggplot(aes(x=Diversity, y=value),data=germanyForPlotting) +
  geom_point(size=3)+
  facet_wrap(~variable, scales="free") +
  theme_bw(base_size=15)+
  stat_smooth(method="lm", colour="black", size=2) +
  xlab("\nSpecies Richness") +
  ylab("Value of Function\n") +
  geom_text(data=germanyLabels,
            aes(label=labels)) +
 theme(panel.grid = element_blank())
```



Next, we ask whether the best performing species differs between functions. To do that, we have to pull out all monoculture plots, then reshape the data and run a quick qualitative analysis on it to determine the best performer.

```
#pull out only those plots planted in monoculture
monoGermany<- subset(germany, rowSums(germany[,spIDX])==1)

#figure out which monoculture is which
monoGermany$mono<-apply(monoGermany[,spIDX], 1, function(x)
 species[which(x==1)])

#melt by function for easier analysis
monoGermanyMelt<-
melt(monoGermany[,c(ncol(monoGermany),which(names(monoGermany) %in% vars))],
 id.vars="mono")

#get the mean for each monoculture that has >1 replicate
monoGermanyMelt <- ddply(monoGermanyMelt, .(variable, mono), summarize,
 value=mean(value))

#now, who is the best performer?
ddply(monoGermanyMelt, .(variable), summarize,
 maxMono=mono[which(value==max(value, na.rm=T))])
```

```
#    variable maxMono
# 1 biomassY3 TRIPRA1
# 2     root3 FESRUB1
# 3   cotton3 TRIREP1
# 4    N.Soil TRIREP1
# 5    N.g.m2 TRIPRA1
```

Overlap Approach

One of the functions we are looking at, soil N, first needs to be reflected before we can apply the overlap approach.

```
germany$N.Soil<- -1*germany$N.Soil +max(germany$N.Soil, na.rm=T)
```

We can examine any single species and generate coefficient values and 'effects' - i.e., whether a species has a positive (1), negative (-1), or neutral (0) effect on a function. Let's examine this for biomass production in year 3.

```
spList<-sAICfun("biomassY3", species, germany)
spList
```

```
# $pos.sp
# [1] "DACGLO1" "FESPRA1" "LATPRA1" "LOTCOR1" "TRIPRA1" "TRIREP1"
#
# $neg.sp
# [1] "CAMPAT1" "PHLPRA1" "RANACR1"
#
# $neu.sp
#  [1] "ACHMIL1" "ALOPRA1" "ANTODO1" "ARRELA1" "BROHOR1" "CENJAC1" "CHRLEU1"
#  [8] "CREBIE1" "CYNCRI1" "FESRUB1" "GERPRA1" "HOLLAN1" "KNAARV1" "LEOAUT1"
# [15] "LOLPER1" "LYCFLO1" "PIMMAJ1" "PLALAN1" "RUMACE1" "TAROFF1" "VICCRA1"
# [22] "VICSEP1"
#
# $functions
# [1] "biomassY3"
#
# $coefs
#     ACHMIL1     ALOPRA1     ANTODO1     ARRELA1     BROHOR1     CAMPAT1
#         0.0         0.0         0.0         0.0         0.0      -391.5
#     CENJAC1     CHRLEU1     CREBIE1     CYNCRI1     DACGLO1     FESPRA1
#         0.0         0.0         0.0         0.0       104.6       135.7
#     FESRUB1     GERPRA1     HOLLAN1     KNAARV1     LATPRA1     LEOAUT1
#         0.0         0.0         0.0         0.0       248.6         0.0
#     LOLPER1     LOTCOR1     LYCFLO1     PHLPRA1     PIMMAJ1     PLALAN1
#         0.0       375.7         0.0      -297.7         0.0         0.0
#     RANACR1     RUMACE1     TAROFF1     TRIPRA1     TRIREP1     VICCRA1
#      -140.3         0.0         0.0       492.6       325.0         0.0
#     VICSEP1 (Intercept)
#         0.0       335.8
#
# $effects
# ACHMIL1 ALOPRA1 ANTODO1 ARRELA1 BROHOR1 CAMPAT1 CENJAC1 CHRLEU1 CREBIE1
#       0       0       0       0       0      -1       0       0       0
# CYNCRI1 DACGLO1 FESPRA1 FESRUB1 GERPRA1 HOLLAN1 KNAARV1 LATPRA1 LEOAUT1
#       0       1       1       0       0       0       0       1       0
# LOLPER1 LOTCOR1 LYCFLO1 PHLPRA1 PIMMAJ1 PLALAN1 RANACR1 RUMACE1 TAROFF1
#       0       1       0      -1       0       0      -1       0       0
# TRIPRA1 TRIREP1 VICCRA1 VICSEP1
#       1       1       0       0
```

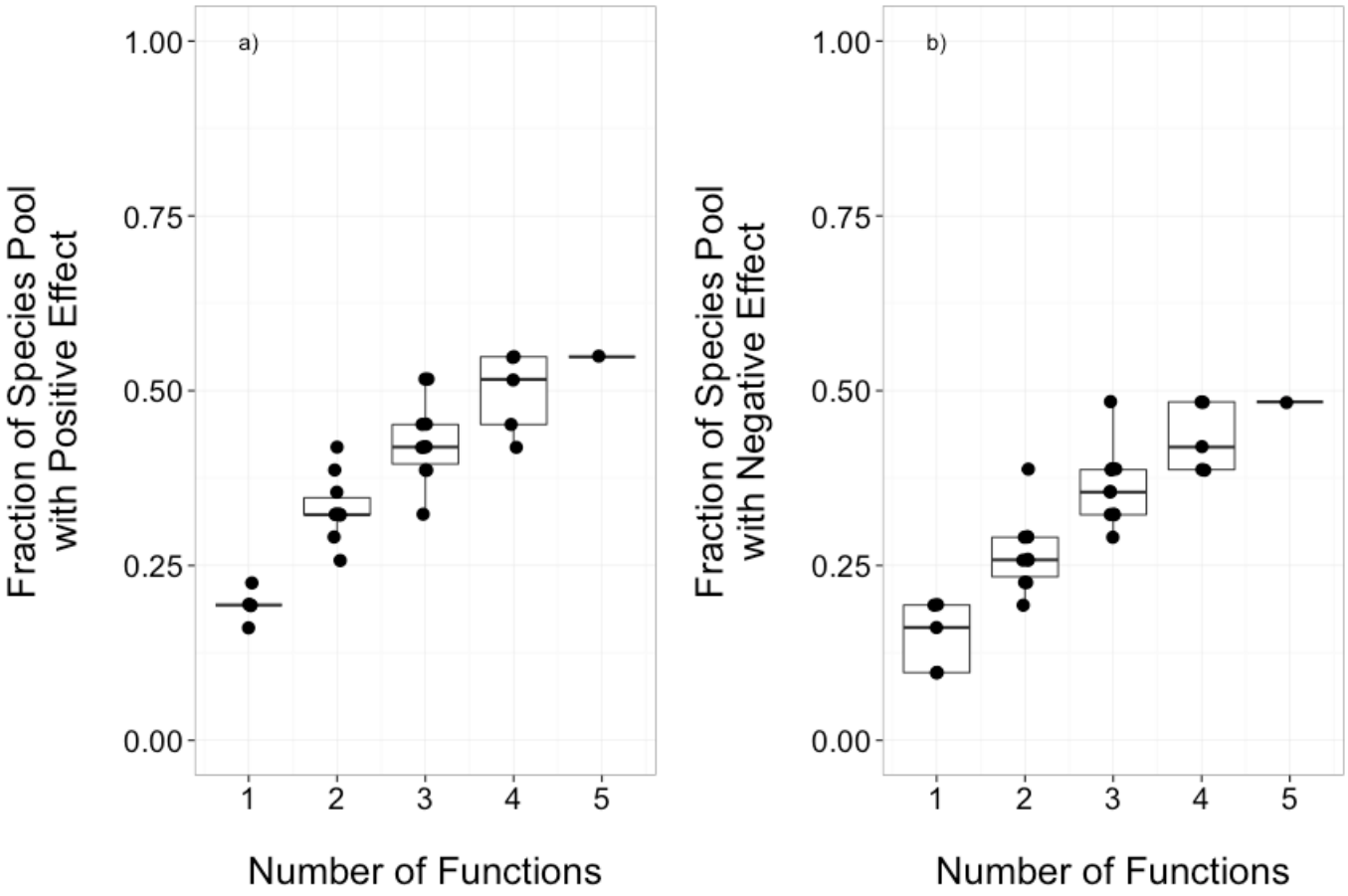
Using this as our starting point, we can use the getRedundancy function to generate a) an effect matrix for all functions, b) a coefficient matrix for all functions, and c) a standardized coefficient matrix for all functions.

```
redund<-getRedundancy(vars, species, germany)
coefs<-getRedundancy(vars, species, germany, output="coef")
stdCoefs<-stdEffects(coefs, germany, vars, species)

#for example
redund
```

```
#           ACHMIL1 ALOPRA1 ANTODO1 ARRELA1 BROHOR1 CAMPAT1 CENJAC1 CHRLEU1
# biomassY3       0       0       0       0       0      -1       0       0
# root3           1       0      -1       0       1       0       0       0
# N.g.m2          1       0       0      -1       0       0       0       0
# N.Soil         -1       0      -1       0       1      -1       0       0
# cotton3         0       1       1       0      -1       1       0       0
#           CREBIE1 CYNCRI1 DACGLO1 FESPRA1 FESRUB1 GERPRA1 HOLLAN1 KNAARV1
# biomassY3       0       0       1       1       0       0       0       0
# root3           0       0       0       0       1       0      -1       0
# N.g.m2         -1       0       0       1       0       0      -1       0
# N.Soil          1       0       1      -1       0       0       0       0
# cotton3         0       0       0       0       0       0       0       0
#           LATPRA1 LEOAUT1 LOLPER1 LOTCOR1 LYCFLO1 PHLPRA1 PIMMAJ1 PLALAN1
# biomassY3       1       0       0       1       0      -1       0       0
# root3           1       0       1       0       0       0       0      -1
# N.g.m2          0       0       1       1       0      -1       0      -1
# N.Soil          1       0       0       1       0       1       0       0
# cotton3        -1       0      -1       0       0       0       0       1
#           RANACR1 RUMACE1 TAROFF1 TRIPRA1 TRIREP1 VICCRA1 VICSEP1
# biomassY3      -1       0       0       1       1       0       0
# root3          -1       0       0      -1       0       0       0
# N.g.m2         -1       0       0       1       1       0       0
# N.Soil          1       0       0      -1      -1       0       0
# cotton3         0       0       0       1       1       0       0
```

Using the effect matrix, we then estimate the average number of species affecting all possible combinations of functions - both positive and negative - and then plot the results.

```
#plot the num. functions by fraction of the species pool needed
posCurve<-divNeeded(redund, type="positive")
posCurve$div<-posCurve$div/ncol(redund)
pc<-qplot(nfunc, div, data=posCurve, group=nfunc, geom=c("boxplot"))+
  geom_jitter(size=4, position = position_jitter(height=0.001, width = .04))+
  ylab("Fraction of Species Pool\nwith Positive Effect\n")+
  xlab("\nNumber of Functions")+theme_bw(base_size=24)+ylim(c(0,1))


negCurve<-divNeeded(redund, type="negative")
negCurve$div<-negCurve$div/ncol(redund)
nc<-qplot(nfunc, div, data=negCurve, group=nfunc, geom=c("boxplot"))+
  geom_jitter(size=4, position = position_jitter(height=0.001, width = .04))+
  ylab("Fraction of Species Pool\nwith Negative Effect\n")+
  xlab("\nNumber of Functions")+theme_bw(base_size=24)+ylim(c(0,1))

#combine these into one plot
grid.arrange(pc+annotate("text", x=1, y=1, label="a)"), nc+annotate("text",
 x=1, y=1, label="b)"), ncol=2)
```

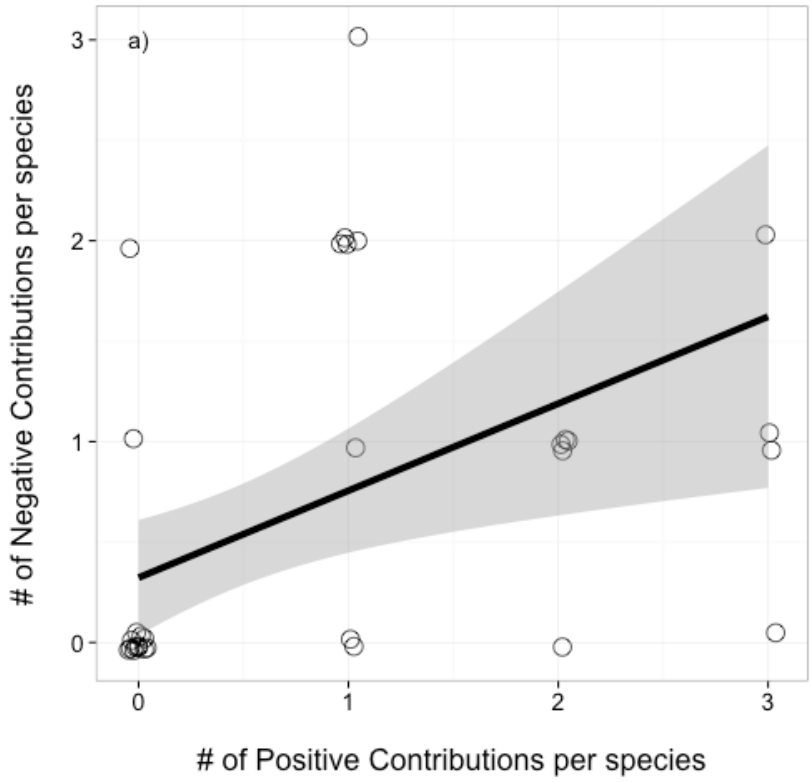


We could have also looked at overlap indices, such as the Sorenson index.

```
allOverlapPos<-ddply(data.frame(m=2:nrow(redund)), .(m), function(adf)
 getOverlapSummary(redund, m=adf$m[1], denom="set"))
```

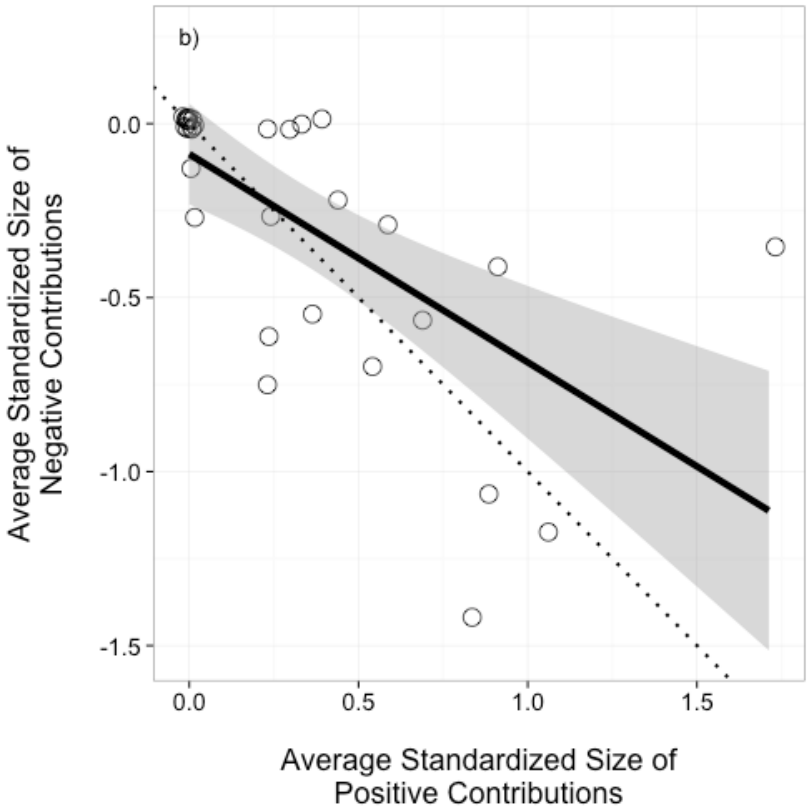
One we've examined the consequences of functional overlap, we can then move on to compare the number and size of these effects. First, what is the relative balance of positive to negative contributions for each species?

```
posNeg<-data.frame(Species = colnames(redund),
                   Pos = colSums(filterOverData(redund)),
                   Neg = colSums(filterOverData(redund, type="negative")))

#plot it
ggplot(aes(x=Pos,y= Neg), data=posNeg) +
  geom_jitter(position = position_jitter(width = 0.05, height = 0.05),
 size=5, shape=1) +
  theme_bw(base_size=18) +
  xlab("\n# of Positive Contributions per species\n") +
  ylab("# of Negative Contributions per species\n") +
  annotate("text", x=0, y=3, label="a)") +
  stat_smooth(method="glm", colour="black", size=2,
 family=poisson(link="identity"))
```



So, a generally positive relationship. We can see from the figure that the slope is slightly less than 1:1, so positive effects accumulate faster. Still, more positive effects = more negative effects. What about the size of those positive versus negative effects?

```
#now get the standardized effect sizes
posNeg<-within(posNeg, {

  stdPosMean <- colSums(filterCoefData(stdCoefs))/Pos
  stdPosMean[which(is.nan(stdPosMean))] <-0

  stdNegMean <- colSums(filterCoefData(stdCoefs, type="negative"))/Neg
  stdNegMean[which(is.nan(stdNegMean))] <-0
})


ggplot(aes(x=stdPosMean,y= stdNegMean), data=posNeg) +
  #  geom_point(size=3) +
  geom_jitter(position = position_jitter(width = .02, height = .02), size=5,
 shape=1) +
  theme_bw(base_size=18) +
  xlab("\nAverage Standardized Size of\nPositive Contributions") +
  ylab("Average Standardized Size of\nNegative Contributions\n") +
  stat_smooth(method="lm", colour="black", size=2)+
  annotate("text", x=0, y=0.25, label="b)") +
  geom_abline(slope=-1, intercept=0, size=1, colour="black", lty=3)
```



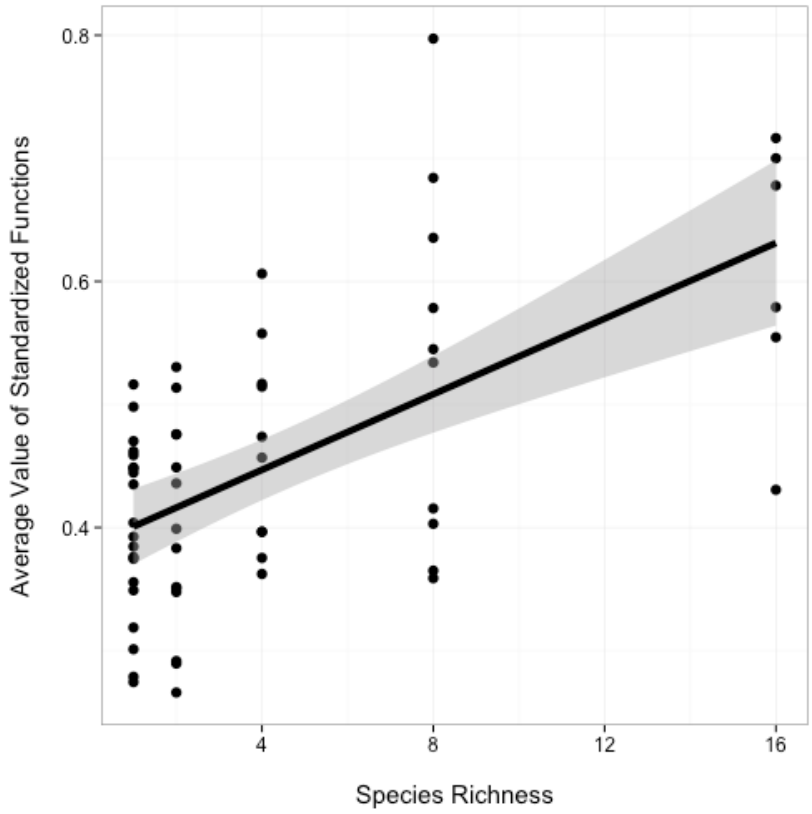
Averaging Approach

Now we'll try the averaging approach. The first step is to create a set of columns where we have standardized all of the functions of interest, and then create an average of those standardized functions.

```
#add on the new functions along with the averaged multifunctional index
germany<-cbind(germany, getStdAndMeanFunctions(germany, vars))
#germany<-cbind(germany, getStdAndMeanFunctions(germany, vars,
standardizeZScore))
```

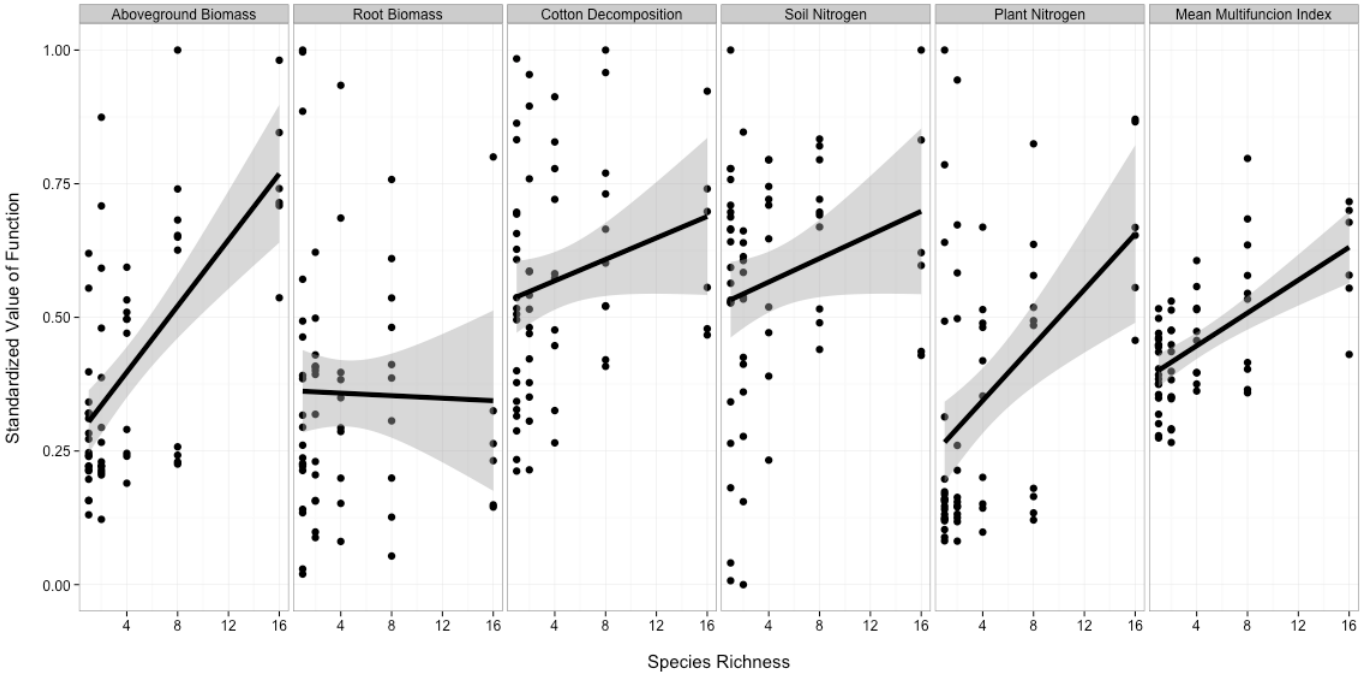
We can then plot the averaged multifunctionality quite simply

```
#plot it
ggplot(aes(x=Diversity, y=meanFunction),data=germany)+geom_point(size=3)+
  theme_bw(base_size=15)+
  stat_smooth(method="lm", colour="black", size=2) +
  xlab("\nSpecies Richness") +
  ylab("Average Value of Standardized Functions\n")
```



We may want to look at all of the functions on a standardized scale, and add in the averaged line for comparison as well. We can do this by reshaping the data, and plotting it.

```
#reshape for plotting everything with ggplot2
germanyMeanForPlotting<-melt(germany[,c(8,129:134)], id.vars="Diversity")

#nice names for plotting
levels(germanyMeanForPlotting$variable) <- c('Aboveground Biomass', 'Root
Biomass', 'Cotton Decomposition', 'Soil Nitrogen', 'Plant Nitrogen', 'Mean
Multifuncion Index')

#plot it
ggplot(aes(x=Diversity,
 y=value),data=germanyMeanForPlotting)+geom_point(size=3)+
  facet_grid(~variable) +
  theme_bw(base_size=15)+
  stat_smooth(method="lm", colour="black", size=2) +
  xlab("\nSpecies Richness") +
  ylab("Standardized Value of Function\n")
```



Last, let's test the statistical fit of the effect of diversity on the averaged multifunctionality index.

```
#statistical fit
aveFit<-lm(meanFunction ~ Diversity, data=germany)
Anova(aveFit)
```

```
# Anova Table (Type II tests)
#
# Response: meanFunction
#           Sum Sq Df F value Pr(>F)
# Diversity  0.295  1    32.6  4e-07 ***
# Residuals  0.524 58
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aveFit)
```

```
#
# Call:
# lm(formula = meanFunction ~ Diversity, data = germany)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -0.20053 -0.06932  0.00651  0.06272  0.28894
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  0.38542    0.01705   22.61   <2e-16 ***
# Diversity    0.01536    0.00269    5.71    4e-07 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 0.0951 on 58 degrees of freedom
# Multiple R-squared: 0.36, Adjusted R-squared: 0.349
# F-statistic: 32.6 on 1 and 58 DF,  p-value: 4.04e-07
```

Threshold Approach

First, we need to create a new data set that looks at the number of functions greater than or equal to a threshold across a wide range of thresholds.

```
germanyThresh<-getFuncsMaxed(germany, vars, threshmin=0.05, threshmax=0.99,
 prepend=c("plot","Diversity"), maxN=7)
```

Next, let's perform an analysis on just the 0.8 threshold data.

```
mfuncGermanyLinear08<-glm(funcMaxed ~ Diversity, data=subset(germanyThresh,
 germanyThresh$thresholds=="0.8"), family=quasipoisson(link="identity"))

Anova(mfuncGermanyLinear08, test.statistic="F")
```

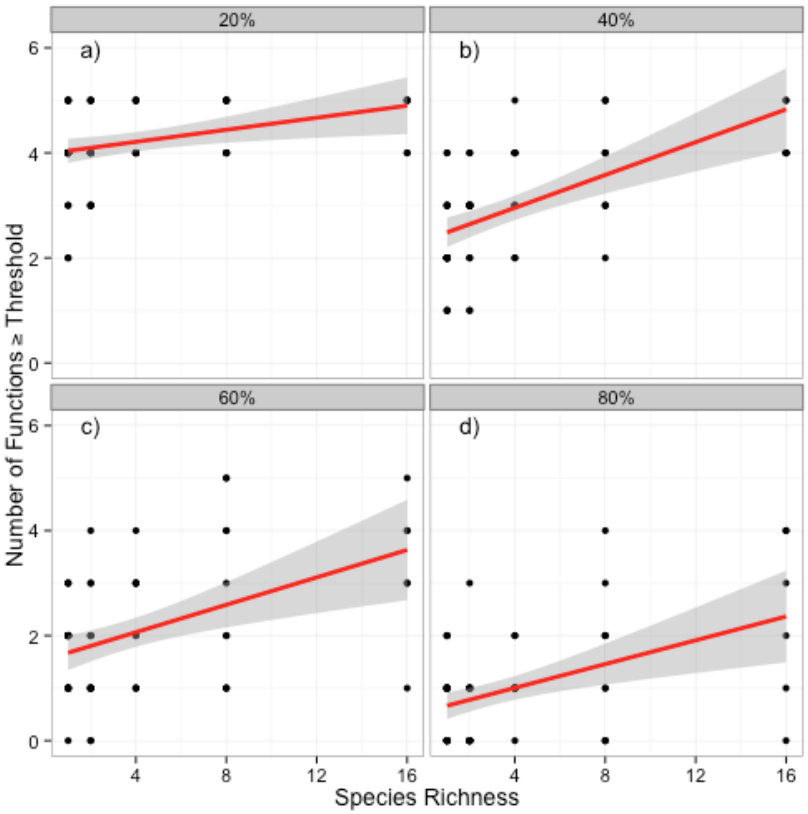
```
# Analysis of Deviance Table (Type II tests)
#
# Response: funcMaxed
#             SS Df    F  Pr(>F)
# Diversity 13.1  1 15.9 0.00019 ***
# Residuals 47.7 58
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mfuncGermanyLinear08)
```

```
#
# Call:
# glm(formula = funcMaxed ~ Diversity, family = quasipoisson(link =
"identity"),
#     data = subset(germanyThresh, germanyThresh$thresholds ==
#         "0.8"))
#
# Deviance Residuals:
#     Min       1Q   Median       3Q      Max
# -2.1743  -1.1532  -0.0047   0.3821   1.9112
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)   0.5517     0.1483    3.72  0.00045 ***
# Diversity     0.1133     0.0331    3.42  0.00116 **
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# (Dispersion parameter for quasipoisson family taken to be 0.8227)
#
#     Null deviance: 71.852  on 59  degrees of freedom
# Residual deviance: 58.758  on 58  degrees of freedom
# AIC: NA
#
# Number of Fisher Scoring iterations: 4
```

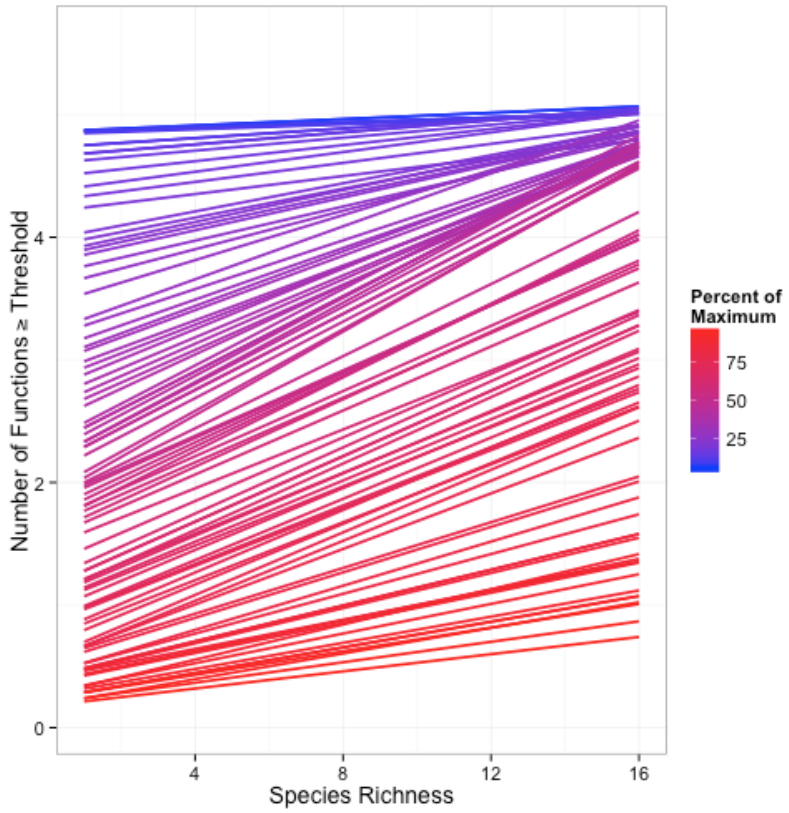
To get a better sense of how robust these results are to threshold choice, let's plot the diversity versus number of functions greater than or equal to a threshold for four different thresholds.

```
gcPlot<-subset(germanyThresh, germanyThresh$thresholds %in% qw(0.2, 0.4, 0.6,
0.8)) #note, using qw as %in% is a string comparison operator

gcPlot$percent<-paste(100*gcPlot$thresholds, "%", sep="")

qplot(Diversity, funcMaxed, data=gcPlot, facets=~percent) +
  stat_smooth(method="glm", family=quasipoisson(link="identity"),
 colour="red", lwd=1.2) +
  ylab(expression("Number of Functions" >= Threshold)) +
  xlab("Species Richness") +
  theme_bw(base_size=14) +
  geom_text(data=data.frame(percent = unique(gcPlot$percent),
                            lab = paste(letters[1:4], ")", sep=""),
                           Diversity=2,
                           funcMaxed=6
                           ), mapping=aes(x=Diversity, y=funcMaxed, label=lab))
```



Given that variation, let's look at the entire spread of thresholds.

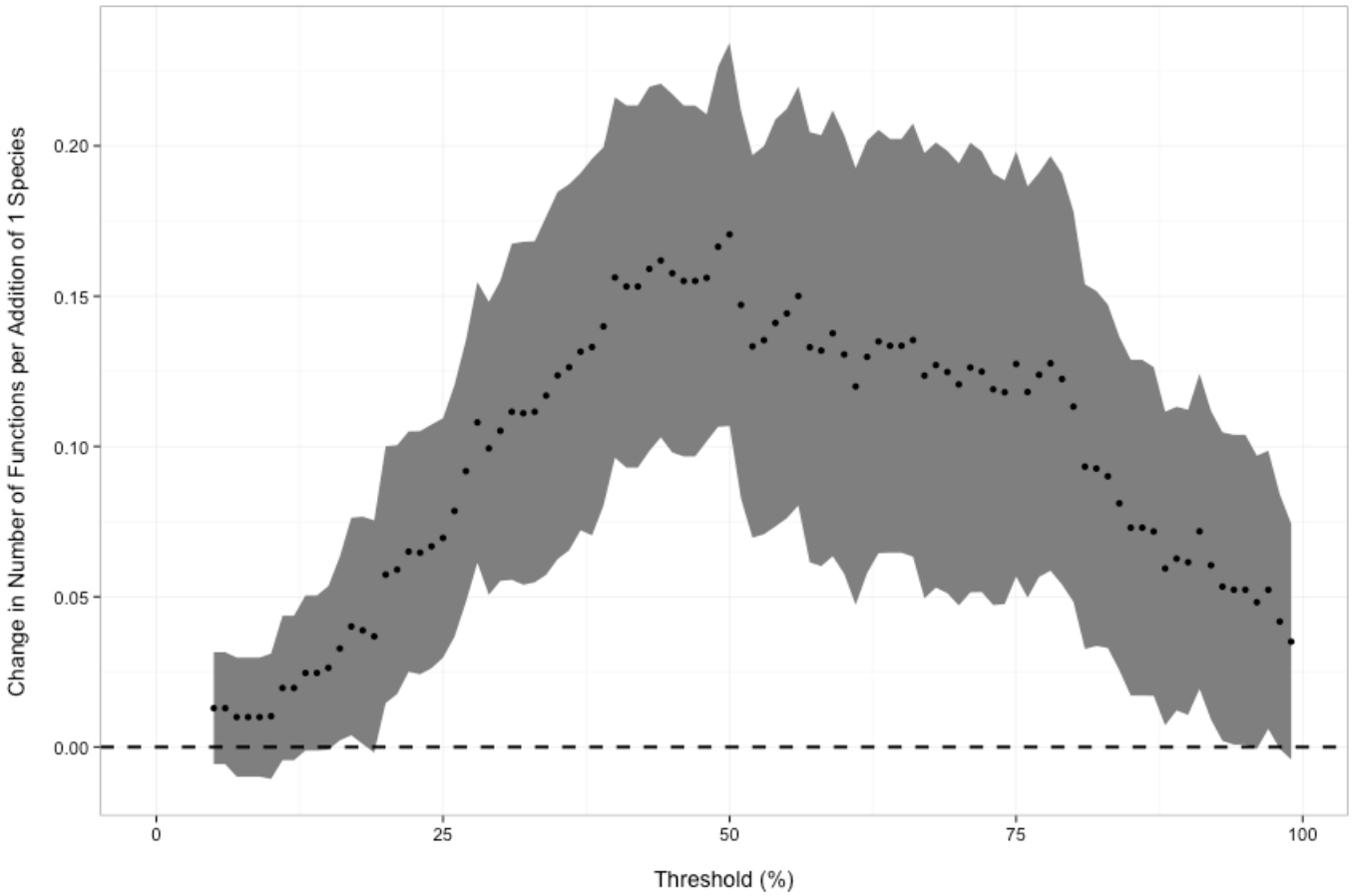
```
germanyThresh$percent <- 100*germanyThresh$thresholds
ggplot(data=germanyThresh, aes(x=Diversity, y=funcMaxed, group=percent)) +
    ylab(expression("Number of Functions" >= Threshold)) +
    xlab("Species Richness") +
    stat_smooth(method="glm", family=quasipoisson(link="identity"), lwd=0.8,
 fill=NA, aes(color=percent)) +
    theme_bw(base_size=14) +
    scale_color_gradient(name="Percent of \nMaximum", low="blue", high="red")
```



Multiple Threshold Approach

To systematically explore the variation in the relationship based on threshold choice, let's look at the slope of the relationship and its confidence interval over all thresholds. We will then plot this relationship.

```
germanyLinearSlopes<-getCoefTab(funcMaxed ~ Diversity, data=germanyThresh,
 coefVar="Diversity", family=quasipoisson(link="identity"))


######
# Plot the values of the diversity slope at
# different levels of the threshold
######
germanSlopes <- ggplot(germanyLinearSlopes, aes(x=thresholds)) +
  geom_ribbon(fill="grey50", aes(x=thresholds*100, ymin=Estimate-
1.96*germanyLinearSlopes[["Std. Error"]],

 ymax=Estimate+1.96*germanyLinearSlopes[["Std. Error"]])) +
  geom_point(aes(x=thresholds*100, y=Estimate)) +
  ylab("Change in Number of Functions per Addition of 1 Species\n") +
  xlab("\nThreshold (%)") +
  stat_abline(intercept=0, slope=0, lwd=1, linetype=2) +
  theme_bw(base_size=14)

germanSlopes
```



We can easily see a number of indices mentioned in the text - Tmin, Tmax, Tmde, etc. Let's pull them out with our indices function.

```
germanIDX <- getIndices(germanyLinearSlopes, germanyThresh, funcMaxed ~
 Diversity)
germanIDX
```

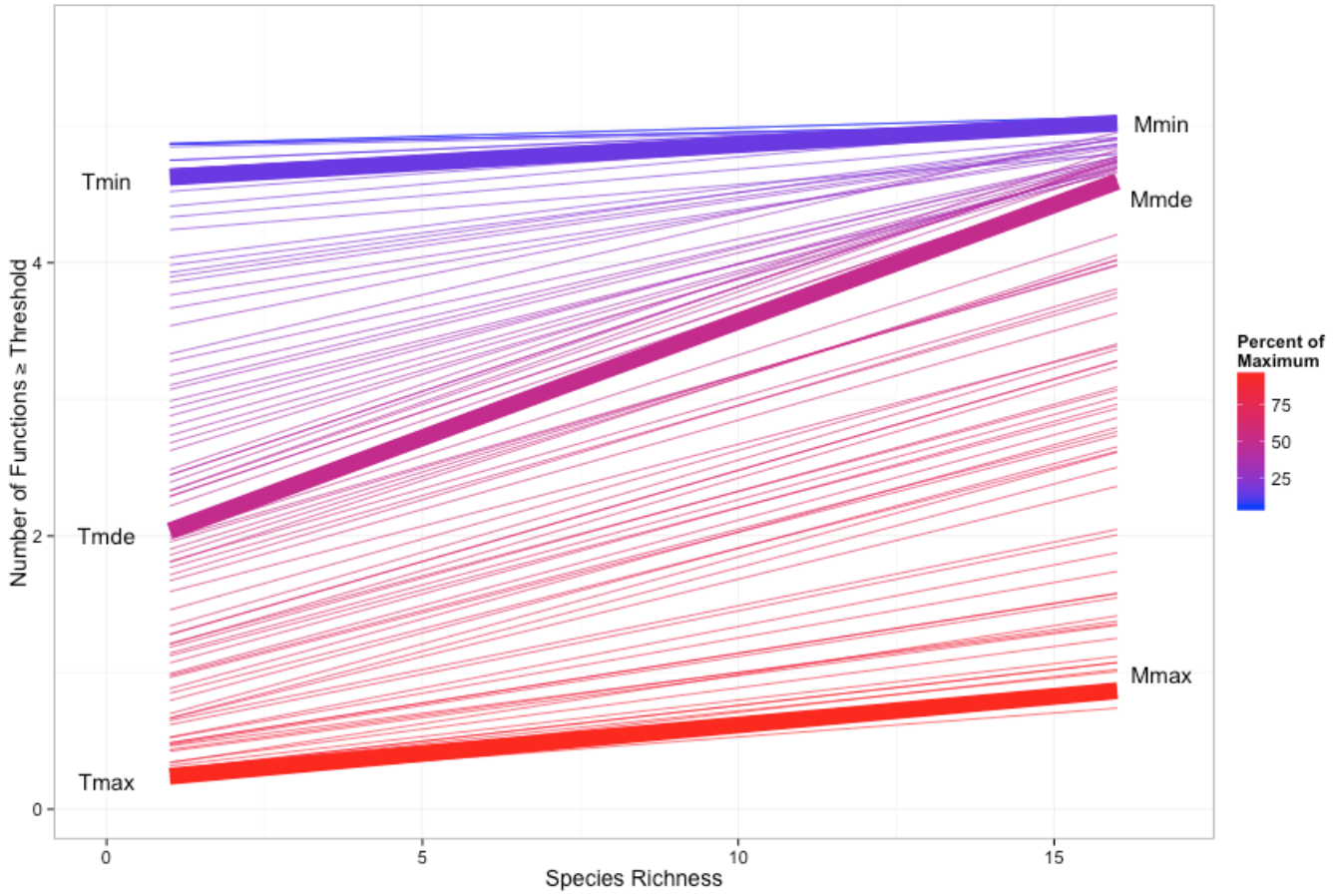
```
#     Tmin Tmax Tmde Rmde.linear Pmde.linear  Mmin   Mmax  Mmde
# 602 0.15 0.98  0.5      0.1705      0.5457 5.021 0.9845 4.474
```

We can now add annotations to our earlier plot to make it more rich, and show us those crucial threshold values.

```
germanyLinearSlopes$Estimate[which(germanyLinearSlopes$thresholds==germanIDX$Tmde)]
```

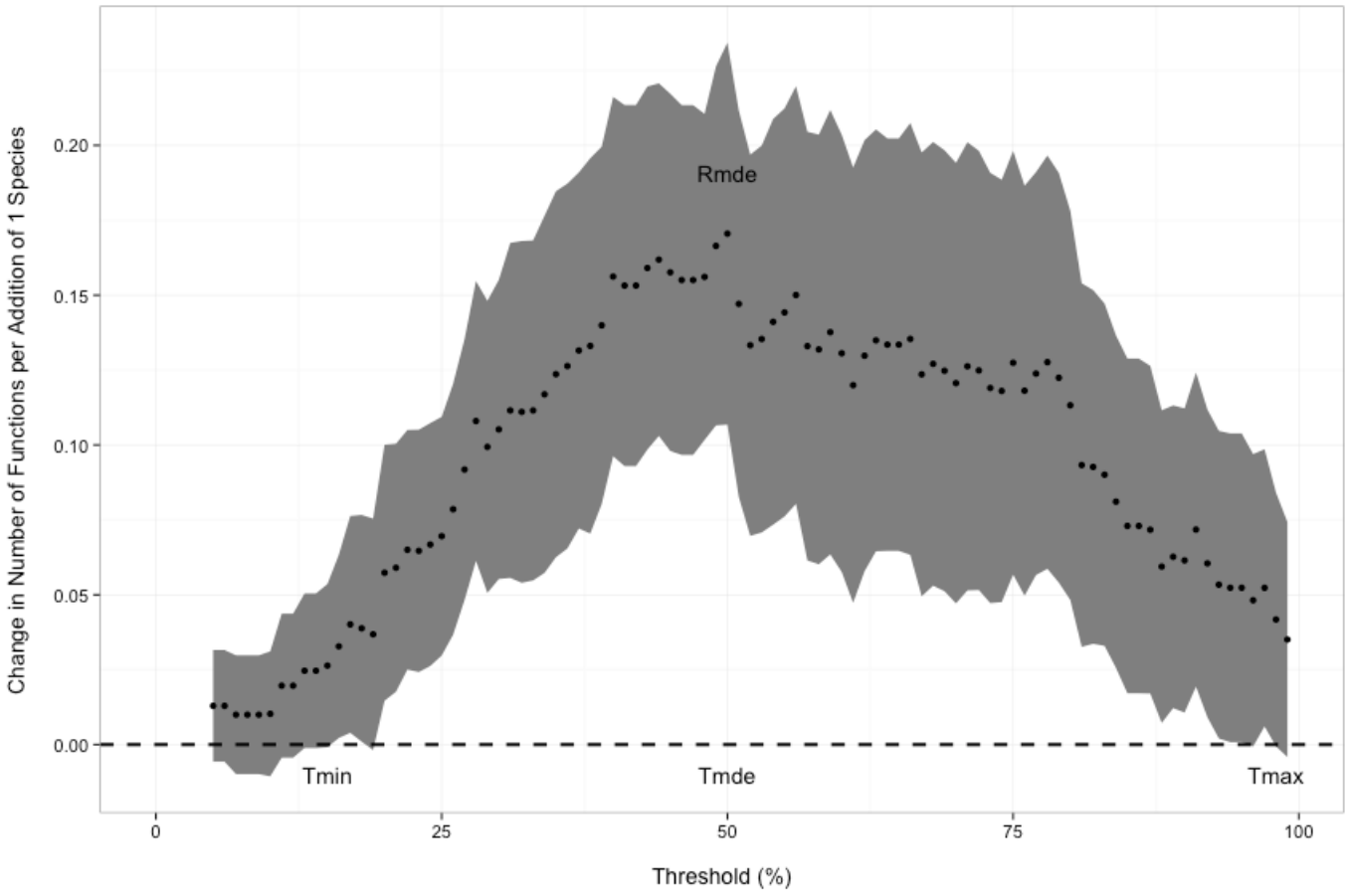
```
# [1] 0.1705
```

```
germanyThresh$IDX <- 0
germanyThresh$IDX [which(germanyThresh$thresholds %in%
                           c(germanIDX$Tmin, germanIDX$Tmax,
 germanIDX$Tmde))] <- 1



ggplot(data=germanyThresh, aes(x=Diversity, y=funcMaxed, group=percent)) +
  ylab(expression("Number of Functions" >= Threshold)) +
  xlab("Species Richness") +
  geom_smooth(method="glm", family=quasipoisson(link="identity"),
              fill=NA, aes(color=percent, lwd=IDX)) +
  theme_bw(base_size=14) +
  scale_color_gradient(name="Percent of \nMaximum", low="blue", high="red") +
  scale_size(range=c(0.3,5), guide="none") +
  annotate(geom="text", x=0, y=c(0.2,2,4.6), label=c("Tmax", "Tmde", "Tmin"))
+
  annotate(geom="text", x=16.7, y=c(germanIDX$Mmin, germanIDX$Mmax,
 germanIDX$Mmde), label=c("Mmin", "Mmax", "Mmde"))
```



Or we can see where they are on the curve of slopes at different thresholds.

```
germanSlopes + annotate(geom="text", y=c(-0.01, -0.01, -0.01,
 germanIDX$Rmde.linear+0.02), x=c(germanIDX$Tmin*100, germanIDX$Tmde*100,
 germanIDX$Tmax*100, germanIDX$Tmde*100),  label=c("Tmin", "Tmde", "Tmax",
 "Rmde"), color="black")
```



Comparison of Sites

Furthermore, we can extend this to the more of the BIODEPTH dataset to compare the strength of multifunctionality across sites. We begin by loading in the data and calculating slopes at different thresholds. We'll subset down to Sheffield, Portugal, and Sweden for simplicity, but this could be done for all sites quite easily.

```
####
# Now we will look at the entire BIODEPTH dataset
#
# Note the use of ddply from plyr to generate the new data frame using all
locations from biodepth.
# If you're not using plyr for your data aggregation, you should be.
http://plyr.had.co.nz/
# It will save you a lot of time and headaches in the future.
####

#Read in data  to run sample analyses on the biodepth data
all_biodepth<-read.table("./FunctContrastData.txt",header=T)
sub_biodepth<-subset(all_biodepth, all_biodepth$location %in% c("Sheffield",
 "Portugal", "Sweden"))
sub_biodepth$location<-factor(sub_biodepth$location, levels=c("Portugal",
 "Sweden", "Sheffield"))

allVars<-qw(biomassY3, root3, N.g.m2,  light3, N.Soil, wood3, cotton3)
varIdx<-which(names(sub_biodepth) %in% allVars)



#re-normalize so that everything is on the same sign-scale (e.g. the maximum
level of a function is the "best" function)
sub_biodepth<-ddply(sub_biodepth, .(location), function(adf){
  adf$light3<- -1*adf$light3+max(adf$light3, na.rm=T)
  adf$N.Soil<- -1*adf$N.Soil +max(adf$N.Soil, na.rm=T)

  adf
})


#get thresholds
bdThreshes<-ddply(sub_biodepth, .(location), function(x) getFuncsMaxed(x,
 vars=allVars, prepend=c("plot","Diversity"), maxN=8))


####look at slopes

#note, maxIT argument is due to some fits needing more iterations to converge
bdLinearSlopes<-getCoefTab(funcMaxed ~ Diversity, data=bdThreshes,
 groupVar=c("location", "thresholds"),
                           coefVar="Diversity",
 family=quasipoisson(link="identity"), control=list(maxit=800))
```

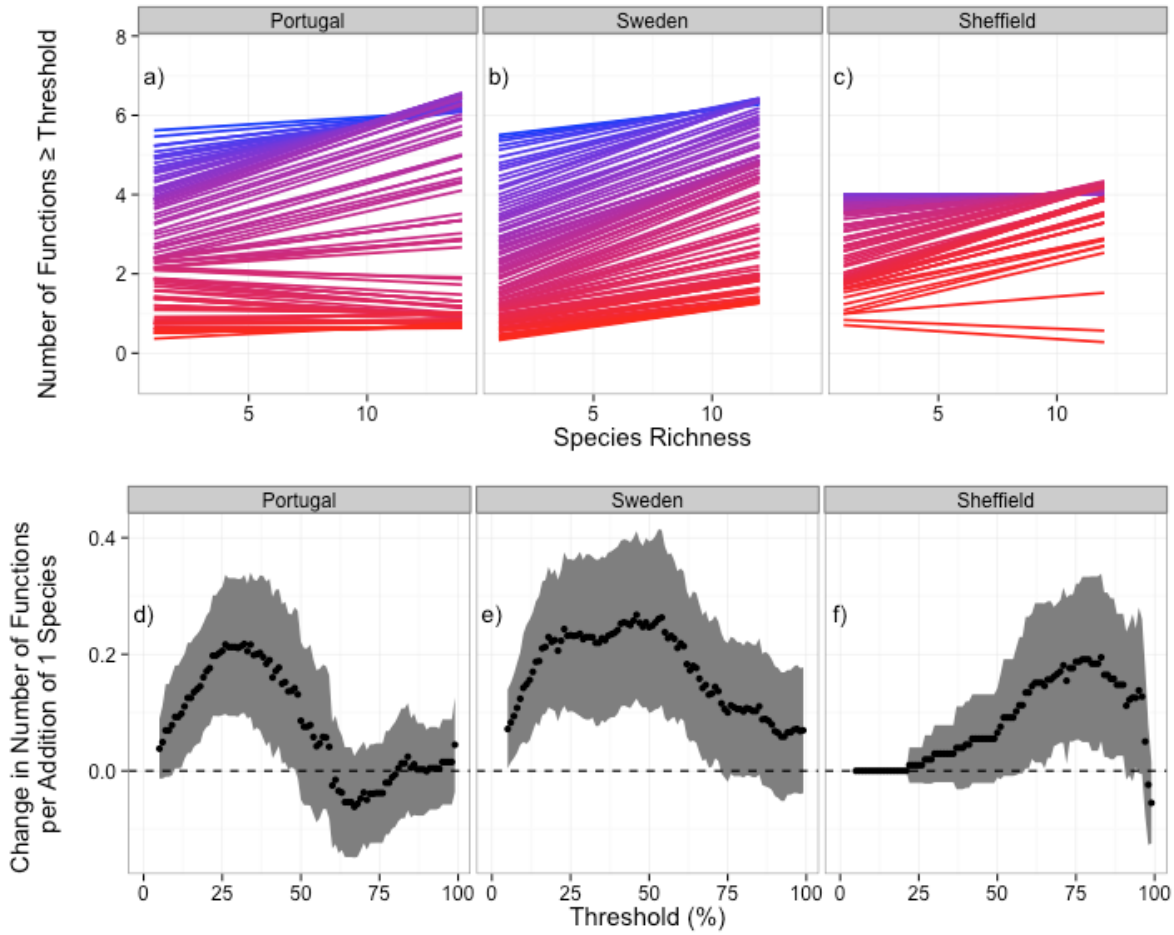
Great, now we can look at the indices of these three new sites. We'll add Germany in so we can compare them against the German values.

```
indexTable <- lapply(levels(bdLinearSlopes$location), function(x){
  slopedata <- subset(bdLinearSlopes, bdLinearSlopes$location==x)
  threshdata <- subset(bdThreshes, bdThreshes$location==x)
  ret <- getIndices(slopedata, threshdata, funcMaxed ~ Diversity)
  ret<-cbind(location=x, ret)
  ret
})
indexTable <- ldply(indexTable)

indexTable <- rbind(data.frame(location="Germany", germanIDX), indexTable)


indexTable
```

```
#       location Tmin Tmax Tmde Rmde.linear Pmde.linear  Mmin   Mmax  Mmde
# 602    Germany 0.15 0.98 0.50      0.1705      0.5457 5.021 0.9845 4.474
# 2     Portugal 0.09 0.49 0.32      0.2177      0.5080 6.097 4.1058 6.441
# 3       Sweden   NA 0.73 0.46      0.2677      0.5353    NA 2.2013 4.776
# 4    Sheffield 0.58 0.95 0.83      0.1946      0.5839 4.260 2.7135 3.890
```

Finally, let's compare a visualization of the relevant curves at different thresholds to the relationship between threshold and slope of curve to give the table of numbers some visual meaning.

```
library(gridExtra)

bdCurvesAll<-qplot(Diversity, funcMaxed, data=bdThreshes, group=thresholds,
 alpha=I(0)) +
  facet_wrap(~location)+#, scales="free") +
  scale_color_gradient(low="blue", high="red", name="Proportion of
\nMaximum", guide=FALSE) +
  stat_smooth(method="glm", lwd=0.8, fill=NA,
 family=gaussian(link="identity"), control=list(maxit=200),
 aes(color=thresholds)) +
  ylab("\nNumber of Functions ≥ Threshold\n\n") +
  xlab("Species Richness") +
  theme_bw(base_size=15)  +
  geom_text(data=data.frame(location = levels(bdThreshes$location), lab =
 paste(letters[1:3], ")", sep=""), thresholds=c(NA, NA, NA)), x=1, y=7,
 mapping=aes(label=lab))



#Plot it!
slopePlot<-ggplot(bdLinearSlopes, aes(x=thresholds, y=Estimate)) +
  geom_ribbon(fill="grey50", aes( x=thresholds*100,ymin=Estimate-
2*bdLinearSlopes[["Std. Error"]],
                                  ymax=Estimate+2*bdLinearSlopes[["Std.
Error"]])) +
  geom_point(aes(x=thresholds*100, y=Estimate)) +
  ylab("Change in Number of Functions \nper Addition of 1 Species\n") +
  xlab("Threshold (%)") +
  facet_wrap(~location)+#, scale="free") +
  stat_abline(intercept=0, slope=0, lwd=0.6, linetype=2) +
  theme_bw(base_size=15)+
  geom_text(data=data.frame(location = levels(bdThreshes$location),
                            lab = paste(letters[4:6], ")", sep=""),
                            thresholds=c(NA, NA, NA)), x=0.05, y=0.27,
 mapping=aes(label=lab))


###Plot them in a single figure
grid.arrange(bdCurvesAll,
             slopePlot)
```


Portugal
Sweden
Sheffield
a)
b)
c)
Number of Functions ≥ Threshold
Species Richness
d)
e)
f)
Change in Number of Functions per Addition of 1 Species
Threshold (%)

Supplementary Information 2: Choice of underlying model for threshold approach

We chose a quasipoisson error distribution as 1) we are dealing with count data (O'Hara & Kotze 2010) and 2) the data are likely to be overdispersed (and if not, the dispersion coefficient should converge to 0). We eschewed a negative binomial error, as we had no expectation that the variance around our fitted relationship is skewed (Ver Hoef & Boveng 2007). We chose a linear model for the ease of interpretation of the slope coefficient – the change in the number of functions greater than the threshold per change in species richness. Taking its inverse, we have an estimate of the number of species needed per function. However, we could have fit our model with a log link function, in which case the coefficient for species richness can be interpreted by exponentiating it and subtracting one to assess the change in percent of functions greater than the threshold per change in species richness. The log link also has the advantage of never predicting values of number of functions less than 0. The curves generated by the two fit quite similarly (the difference in deviance of the two models is ~1), so for the purposes of this example, we will continue discussing the linear model. Finally, we note that one could have divided the number of functions greater than or equal to the threshold by the total number of functions measured to estimate the proportion of functions greater than the threshold. In this case, we would then fit a logit curve with a binomial error (Warton 2011). However, this method may be misinterpreted if all relevant functions in an ecosystem have not been measured (most likely in any real research situation). Conversely, if the set of functions measured are chosen for management purposes and the goal of the analysis is to

957 examine the relationship between biodiversity and just that one set of functions are
958 interest, then a logit approach may indeed be appropriate.
959