Multivariate Analysis

Scott Coffin

1/19/2021

Table of Contents

# Setup

## Data import

# Load finalized dataset.  
aoc <- read\_csv("AquaticOrganisms\_Clean\_final.csv", guess\_max = 10000) %>%   
 mutate(ID = paste0("ID",row\_number()))  
  
#### Introduction Setup ####  
  
# All text inputs below.  
  
#### Overview AO Setup ####  
  
#Final\_effect\_dataset <- read\_csv("Final\_effect\_dataset.csv")%>%  
 #mutate(plot\_f = case\_when(  
 #plot\_f == "Polymer" ~ "Polymer",  
 #plot\_f == "Size" ~ "Size",  
 #plot\_f == "Shape" ~ "Shape",  
 #plot\_f == "Organism" ~ "Organism",  
 #plot\_f == "Lvl1" ~ "Endpoint Category",  
 #plot\_f == "Life.stage" ~ "Life Stage",  
 #plot\_f == "Invivo.invivo" ~ "In Vivo or In Vitro",  
 #plot\_f == "Exposure.route" ~ "Exposure Route"))%>%  
 #mutate(plot\_f = factor(plot\_f))%>%  
 #mutate(logEndpoints = log(Endpoints))%>%  
 #rename(Percent = Freq)  
  
polydf<-rowPerc(xtabs( ~polymer +effect, aoc)) #pulls polymers by effect   
polyf<-as.data.frame(polydf)%>% #Makes data frame   
 filter(effect %in% c("Y","N"))%>% #Sorts into Yes and No  
 mutate(polymer = case\_when(  
 polymer == "BIO" ~ "Biopolymer",  
 polymer == "EVA" ~ "Polyethylene Vinyl Acetate",  
 polymer == "LTX" ~ "Latex",  
 polymer == "PA" ~ "Polyamide",  
 polymer == "PE" ~ "Polyethylene",  
 polymer == "PC" ~ "Polycarbonate",  
 polymer == "PET" ~ "Polyethylene Terephthalate",  
 polymer == "PI" ~ "Polyisoprene",  
 polymer == "PMMA" ~ "Polymethylmethacrylate",  
 polymer == "PP" ~ "Polypropylene",  
 polymer == "PS" ~ "Polystyrene",  
 polymer == "PUR" ~ "Polyurathane",  
 polymer == "PVC" ~ "Polyvinylchloride",  
 polymer == "PLA" ~ "Polylactic Acid"))%>%  
 mutate\_if(is.numeric, round,0) #rounds percents   
Endpoints<-xtabs(~polymer +effect ,aoc) #Pulls all study obs. for polymer from dataset  
polyfinal<- data.frame(cbind(polyf, Endpoints))%>% #adds it as a column  
 rename(Endpoints='Freq.1')%>% #renames column  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
  
sizedf<-rowPerc(xtabs(~size.category +effect, aoc))  
sizef<-as.data.frame(sizedf)%>%  
 filter(effect %in% c("Y","N"))%>%  
 mutate(size.category = case\_when(  
 size.category == 1 ~ "<1µm",  
 size.category == 2 ~ "1µm < 10µm",  
 size.category == 3 ~ "10µm < 100µm",  
 size.category == 4 ~ "100µm < 1mm",  
 size.category == 5 ~ "1mm < 5mm",  
 size.category == 0 ~ "unavailable"))%>%  
 rename(Type = "size.category")%>%  
 mutate\_if(is.numeric, round,0)%>%  
 mutate(plot="Size")  
study\_s<-xtabs(~size.category +effect ,aoc)  
sizefinal<- data.frame(cbind(sizef, study\_s))%>%   
 rename(Endpoints='Freq.1')%>%  
 rename(category='size.category')%>%  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
  
   
shapedf<-rowPerc(xtabs(~shape + effect, aoc))  
shapef<-as.data.frame(shapedf)%>%  
 filter(effect %in% c("Y","N"))%>%  
 rename(Type="shape")%>%  
 mutate\_if(is.numeric, round,0)%>%  
 mutate(plot="Shape")%>%  
 mutate(Type = case\_when(  
 Type == "cube" ~ "Cube",  
 Type == "sphere" ~ "Sphere",  
 Type == "fragment" ~ "Fragment",  
 Type == "fiber" ~ "Fiber"))  
study\_sh<-xtabs(~shape + effect,aoc)  
shapefinal<- data.frame(cbind(shapef, study\_sh))%>%   
 rename(Endpoints='Freq.1')%>%  
 rename(category='shape')%>%  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
  
taxdf<-rowPerc(xtabs(~organism.group +effect, aoc))  
taxf<-as.data.frame(taxdf)%>%  
 filter(effect %in% c("Y","N"))%>%  
 rename(Type= "organism.group")%>%  
 mutate\_if(is.numeric, round,0)%>%  
 mutate(plot="Organism")  
study\_t<-xtabs(~organism.group +effect,aoc)  
taxfinal<- data.frame(cbind(taxf, study\_t))%>%   
 rename(Endpoints='Freq.1')%>%  
 rename(category='organism.group')%>%  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
  
lvl1df<-rowPerc(xtabs(~lvl1 +effect, aoc))  
lvl1f<-as.data.frame(lvl1df)%>%  
 filter(effect %in% c("Y","N"))%>%  
 rename(Type= "lvl1")%>%  
 mutate\_if(is.numeric, round,0)%>%  
 mutate(plot="Lvl1")%>%  
 mutate(Type = case\_when(  
 Type == "alimentary.excretory" ~ "Alimentary, Excretory",  
 Type == "behavioral.sense.neuro" ~ "Behavioral, Sensory, Neurological",  
 Type == "circulatory.respiratory" ~ "Circulatory, Respiratory",  
 Type == "community" ~ "Community",  
 Type == "fitness" ~ "Fitness",  
 Type == "immune" ~ "Immune",  
 Type == "metabolism" ~ "Metabolism",  
 Type == "microbiome" ~ "Microbiome",  
 Type == "stress" ~ "Stress"))   
study\_l<-xtabs(~lvl1 +effect,aoc)  
lvl1final<- data.frame(cbind(lvl1f, study\_l))%>%   
 rename(Endpoints='Freq.1')%>%  
 rename(category='lvl1')%>%  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
   
lifedf<-rowPerc(xtabs(~life.stage +effect, aoc))  
lifef<-as.data.frame(lifedf)%>%  
 filter(effect %in% c("Y","N"))%>%  
 rename(Type= "life.stage")%>%  
 mutate\_if(is.numeric, round,0)%>%  
 mutate(plot="Life.stage")  
studyli<-xtabs(~life.stage +effect ,aoc)  
lifefinal<- data.frame(cbind(lifef, studyli))%>%   
 rename(Endpoints='Freq.1')%>%  
 rename(category='life.stage')%>%  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
  
vivodf<-rowPerc(xtabs(~invitro.invivo +effect, aoc))  
vivof<-as.data.frame(vivodf)%>%  
 filter(effect %in% c("Y","N"))%>%  
 rename(Type= "invitro.invivo")%>%  
 mutate\_if(is.numeric, round,0)%>%  
 mutate(plot="Invivo.invivo")%>%  
 mutate(Type = case\_when(  
 Type=="invivo"~"In Vivo",  
 Type=="invitro"~"In Vitro"))  
study\_v<-xtabs(~invitro.invivo +effect,aoc)  
vivofinal<- data.frame(cbind(vivof, study\_v))%>%   
 rename(Endpoints='Freq.1')%>%  
 rename(category='invitro.invivo')%>%  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
  
routedf<-rowPerc(xtabs(~exposure.route +effect, aoc))  
routef<-as.data.frame(routedf)%>%  
 filter(effect %in% c("Y","N"))%>%  
 rename(Type= "exposure.route")%>%  
 mutate\_if(is.numeric, round,0)%>%  
 mutate(plot="Exposure.route")%>%  
 mutate(Type = case\_when(  
 Type == "coparental.exposure" ~"Co-Parental Exposure",  
 Type == "paternal.exposure" ~ "Paternal Exposure",  
 Type == "maternal.exposure" ~ "Maternal Exposure",  
 Type == "food" ~ "Food",  
 Type == "water" ~ "Water",  
 Type == "sediment" ~ "Sediment",  
 Type == "media" ~ "Media"))  
study\_r<-xtabs(~exposure.route +effect,aoc)  
routefinal<- data.frame(cbind(routef, study\_r))%>%   
 rename(Endpoints='Freq.1')%>%  
 rename(category='exposure.route')%>%  
 mutate(logEndpoints = log(Endpoints))%>%  
 rename(Percent = Freq)#renames column  
   
   
#### Exploration AO Setup ####  
  
# Master dataset for scatterplots - for Heili's tab.  
aoc\_v1 <- aoc %>% # start with original dataset  
 # full dataset filters.  
 mutate(effect\_f = factor(case\_when(effect == "Y" ~ "Yes",  
 effect == "N" ~ "No"),  
 levels = c("No", "Yes"))) %>%  
 # removing NAs to make data set nicer  
 replace\_na(list(size.category = 0, shape = "Not Reported", polymer = "Not Reported", life.stage = "Not Reported"))  
  
aoc\_setup <- aoc\_v1 %>% # start with original dataset  
 mutate(size\_f = factor(case\_when(  
 size.category == 1 ~ "1nm < 100nm",  
 size.category == 2 ~ "100nm < 1µm",  
 size.category == 3 ~ "1µm < 100µm",  
 size.category == 4 ~ "100µm < 1mm",  
 size.category == 5 ~ "1mm < 5mm",  
 size.category == 0 ~ "Not Reported"),  
 levels = c("1nm < 100nm", "100nm < 1µm", "1µm < 100µm", "100µm < 1mm", "1mm < 5mm", "Not Reported"))) %>% # creates new column with nicer names and order by size levels.  
 # shape category data tidying.  
 mutate(shape\_f = factor(case\_when(  
 shape == "fiber" ~ "Fiber",  
 shape == "fragment" ~ "Fragment",  
 shape == "sphere" ~ "Sphere",  
 shape == "Not Reported" ~ "Not Reported"),  
 levels = c("Fiber", "Fragment", "Sphere", "Not Reported"))) %>% # order our different shapes.  
 # polymer category data tidying.  
 mutate(poly\_f = factor(case\_when(  
 polymer == "BIO" ~ "Biopolymer",  
 polymer == "EVA" ~ "Polyethylene Vinyl Acetate",  
 polymer == "LTX" ~ "Latex",  
 polymer == "PA" ~ "Polyamide",  
 polymer == "PE" ~ "Polyethylene",  
 polymer == "PC" ~ "Polycarbonate",  
 polymer == "PET" ~ "Polyethylene Terephthalate",  
 polymer == "PI" ~ "Polyisoprene",  
 polymer == "PMMA" ~ "Polymethylmethacrylate",  
 polymer == "PP" ~ "Polypropylene",  
 polymer == "PS" ~ "Polystyrene",  
 polymer == "PUR" ~ "Polyurathane",  
 polymer == "PVC" ~ "Polyvinylchloride",  
 polymer == "PLA" ~ "Polylactic Acid",  
 polymer == "Not Reported" ~ "Not Reported"))) %>%  
 # taxonomic category data tidying.  
 mutate(org\_f = factor(organism.group, levels = c("Algae", "Annelida", "Bacterium", "Cnidaria", "Crustacea",  
 "Echinoderm", "Fish", "Insect", "Mollusca", "Nematoda", "Plant", "Rotifera", "Mixed"))) %>% # order our different organisms.  
 mutate(lvl1\_f = factor(case\_when(lvl1 == "alimentary.excretory" ~ "Alimentary, Excretory",  
 lvl1 == "behavioral.sense.neuro" ~ "Behavioral, Sensory, Neurological",  
 lvl1 == "circulatory.respiratory" ~ "Circulatory, Respiratory",  
 lvl1 == "community" ~ "Community",  
 lvl1 == "fitness" ~ "Fitness",  
 lvl1 == "immune" ~ "Immune",  
 lvl1 == "metabolism" ~ "Metabolism",  
 lvl1 == "microbiome" ~ "Microbiome",  
 lvl1 == "stress" ~ "Stress"))) %>% # creates new column with nicer names.  
 # Level 2 Data tidying  
 mutate(lvl2\_f = factor(case\_when(lvl2 == "abundance"~"Abundance",  
 lvl2 == "actinobacteria" ~ "Actinobacteria",  
 lvl2 == "aggressivity"~"Agressivity",  
 lvl2 == "ammonia.excretion" ~ "Ammonia Excretion",  
 lvl2 == "bacteroidetes"~ "Bacteriodetes",  
 lvl2 == "blood"~"Blood",  
 lvl2 == "body.condition"~"Body Condition",  
 lvl2 == "boldness"~"Boldness",  
 lvl2 == "brain.histo"~"Brain Histological Abnormalities",  
 lvl2 == "burrowing"~"Burrowing",  
 lvl2 == "carb.metabolism"~"Carb Metabolism",  
 lvl2 == "chemokines.cytokines"~"Chemokines",  
 lvl2 == "circulatory"~"Circulatory",  
 lvl2 == "detoxification"~"Detoxification",  
 lvl2 == "development"~"Development",  
 lvl2 == "digestion"~"Digestion",  
 lvl2 == "digestive.enzymes"~"Digestive Enzymes",  
 lvl2 == "digestive.tract.histo"~"Digestive Tract Histological Abnormalities",  
 lvl2 == "diversity"~ "Diversity",  
 lvl2 == "feeding"~ "Feeding",  
 lvl2 == "firmicutes"~ "Firmicutes",  
 lvl2 == "gall.bladder.histo" ~ "Gall Bladder Histological Abnormalities",  
 lvl2 == "gen.metabolism"~ "General Metabolism",  
 lvl2 == "gill.histo"~ "Gill Histological Abnormalities",  
 lvl2 == "gonad.histo"~"Gonad Histological Abnormalities",  
 lvl2 == "growth"~ "Growth",  
 lvl2 == "immune.cells"~"Immune Cells",  
 lvl2 == "immune.other"~"Immune Other ",  
 lvl2 == "intestinal.permeability"~"Intestinal Permeability",  
 lvl2 == "kidney.histo"~"Kidney Histological abnormalities",  
 lvl2 == "lipid.metabolism"~"Lipid Metabolism",  
 lvl2 == "liver.histo"~"Liver Histological Abnormalities",  
 lvl2 == "liver.kidney.products" ~ "Liver and Kidney Products",  
 lvl2 == "locomotion"~"Locomotion",  
 lvl2 == "mortality"~"Mortality",  
 lvl2 == "nervous.system"~"Nervous System",  
 lvl2 == "oxidative.stress"~"Oxidative Stress",  
 lvl2 == "photosynthesis"~ "Photosynthesis",  
 lvl2 == "proteobacteria"~"Protebacteria",  
 lvl2 == "reproduction"~"Reproduction",  
 lvl2 == "respiration"~"Respiration",  
 lvl2 == "sexhormones"~"Sex Hormones",  
 lvl2 == "shoaling"~"Shoaling",  
 lvl2 == "stress"~"Stress",  
 lvl2 == "vision.system"~"Vision System"))) %>% #Renames for widget  
 mutate(bio\_f = factor(case\_when(bio.org == "cell"~"Cell", #Bio Org Data Tidying  
 bio.org == "organism"~"Organism",  
 bio.org == "population"~ "Population",  
 bio.org == "subcell"~"Subcell",  
 bio.org == "tissue" ~ "Tissue")))%>%  
 mutate(vivo\_f = factor(case\_when(invitro.invivo == "invivo"~"In Vivo",  
 invitro.invivo == "invitro"~"In Vitro")))%>% ##Renames for widget (Not using a widget right now, but saving for human health database)  
 mutate(life\_f = factor(case\_when(life.stage == "Early"~"Early",  
 life.stage == "Juvenile"~"Juvenile",  
 life.stage == "Adult"~"Adult",  
 life.stage == "Not Reported"~"Not Reported")))%>% #Renames for widget  
 mutate(env\_f = factor(case\_when(environment == "Freshwater"~"Freshwater",  
 environment == "Marine" ~ "Marine",  
 environment == "Terrestrial" ~ "Terrestrial"))) %>%  
 mutate(dose.mg.L.master.converted.reported = factor(dose.mg.L.master.converted.reported)) %>%  
 mutate(dose.particles.mL.master.converted.reported = factor(dose.particles.mL.master.converted.reported)) %>%   
 mutate(dose.um3.mL.master = particle.volume.um3 \* dose.particles.mL.master) %>% #calculate volume/mL  
 mutate(af.time\_noNA = replace\_na(af.time, "Unavailable")) %>%   
 mutate(acute.chronic\_f = factor(case\_when(af.time\_noNA == 10 ~ "Acute",  
 af.time\_noNA == 1 ~ "Chronic",  
 af.time\_noNA == "Unavailable" ~ "Unavailable"))) %>% #factorize assesment factor time into chronic/acute  
 mutate(dose.mg.L.master.AF.noec = dose.mg.L.master \* af.noec) %>%   
 mutate(dose.particles.mL.master.AF.noec = dose.particles.mL.master \* af.noec) %>%   
 mutate(effect\_f = factor(effect)) %>%   
 mutate\_if(is.character, as.factor) %>%   
 mutate(effect\_10 = case\_when(  
 effect\_f == "Y" ~ 1,  
 effect\_f == "N" ~ 0))  
   
  
#### SSD AO Setup ####  
  
# Master dataset for SSDs  
aoc\_z <- aoc\_setup %>% # start with Heili's altered dataset (no filtration for terrestrial data)  
 # environment category data tidying.  
 mutate(environment.noNA = replace\_na(environment, "Not Reported")) %>% # replaces NA to better relabel.  
 mutate(env\_f = factor(environment.noNA, levels = c("Marine", "Freshwater", "Terrestrial", "Not Reported")))   
   
# final cleanup and factoring   
aoc\_z$Species <- as.factor(paste(aoc\_z$genus,aoc\_z$species)) #must make value 'Species" (uppercase)  
aoc\_z$Group <- as.factor(aoc\_z$organism.group) #must make value "Group"  
aoc\_z$Group <- fct\_explicit\_na(aoc\_z$Group) #makes sure that species get counted even if they're missing a group

# subset data to selected variables  
  
multiVar <- aoc\_z %>% dplyr::select(#doi, size.category,   
 size\_f,  
 size.length.um.used.for.conversions,   
 shape,   
 polymer,   
 particle.volume.um3,   
 density.mg.um.3,   
 organism.group,  
 environment,   
 bio.org, #biological level of organization  
 #af.time, #assessment factor based on exposure time  
 treatments, #number of doses (no including control)  
 effect, #yes no  
 effect\_f,  
 effect\_10,  
 size\_f,  
 exposure.duration.d,   
 exposure.route, #Factor  
 organism.group, #factor  
 media.temp, #numeric  
 lvl1\_f, #endpoints  
 lvl2\_f, #endpoints  
 # lvl3,   
 dose.mg.L.master,   
 sex, #factor  
 media.ph, #numeric  
 media.sal.ppt, #numeric  
 dose.particles.mL.master,  
 effect.metric, #NOEC LOEC  
 functional.group, #factor  
 charge, #positive or negatibe  
 zetapotential.mV, # numeric   
 max.size.ingest.mm,#max ingestible size  
 acute.chronic\_f,  
 dose.mg.L.master.AF.noec,  
 dose.particles.mL.master.AF.noec,  
 max.size.ingest.mm, #maximum ingestible size range  
 effect.score) %>% #1 = minor, 2 = photosynthesis, feeding, 3 = growth, chlorophyll content, 4 = reproduction, 5 = population growth, 6 = survival  
 filter(!size\_f == "Not Reported") %>% #take out not reported   
 # max.size.ingest.mm) %>% #max ingestible size  
 filter(!size\_f == "Not Reported") #take out not reported   
  
#recode variables  
# multiVar <- multiVar %>% mutate(effect\_10 = case\_when(  
# effect == "Y" ~ 1,  
# effect == "N" ~ 0  
# ))# %>%   
# #mutate\_all(is.character, ~as.factor())

# Analysis

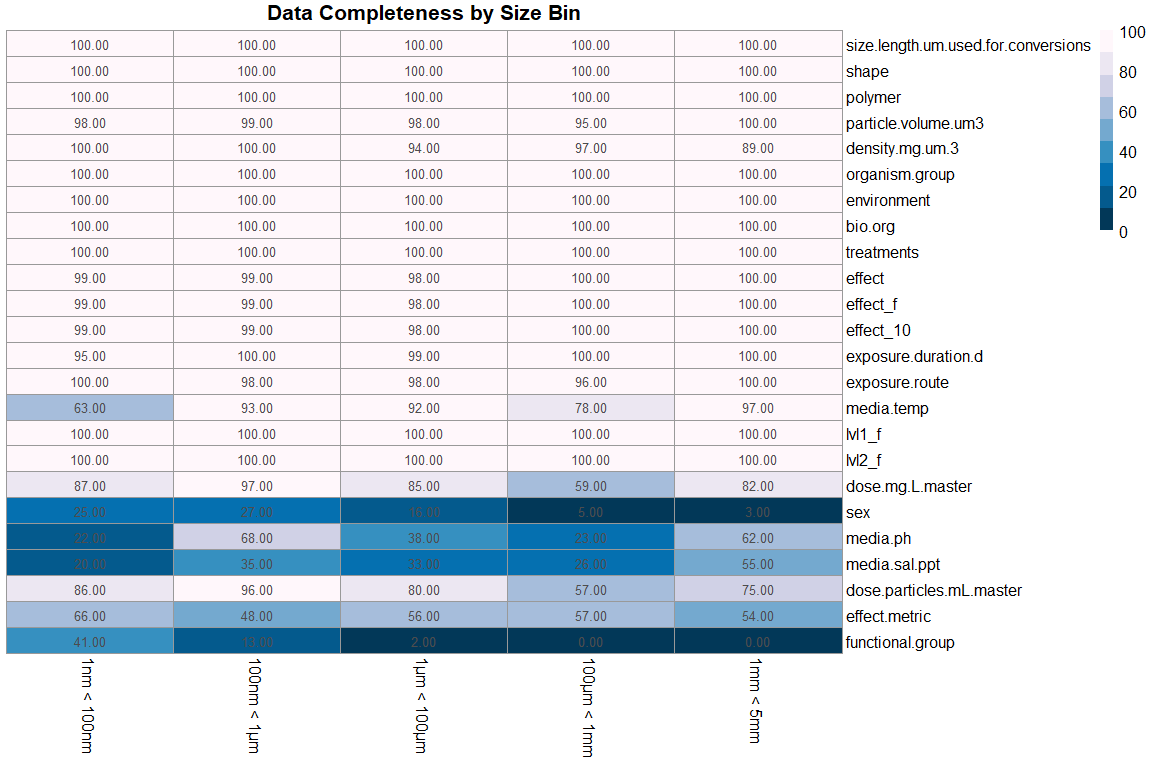
## Data Exploration

### Completeness vby Size

CompletenessSummary\_size <- multiVar %>%  
 group\_by(size\_f) %>%   
 summarise\_all((name = ~sum(is.na(.))/length(.))) %>%   
 mutate(across(is.numeric,~round(., 2))) %>%   
 mutate(across(is.numeric, ~100 \*(1 -.)))  
CompletenessSummary\_size

## # A tibble: 5 x 32  
## size\_f size.length.um.~ shape polymer particle.volume~ density.mg.um.3  
## <fct> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1nm <~ 100 100 100 98 100  
## 2 100nm~ 100 100 100 99 100  
## 3 1µm <~ 100 100 100 98 94  
## 4 100µm~ 100 100 100 95 97  
## 5 1mm <~ 100 100 100 100 89  
## # ... with 26 more variables: organism.group <dbl>, environment <dbl>,  
## # bio.org <dbl>, treatments <dbl>, effect <dbl>, effect\_f <dbl>,  
## # effect\_10 <dbl>, exposure.duration.d <dbl>, exposure.route <dbl>,  
## # media.temp <dbl>, lvl1\_f <dbl>, lvl2\_f <dbl>, dose.mg.L.master <dbl>,  
## # sex <dbl>, media.ph <dbl>, media.sal.ppt <dbl>,  
## # dose.particles.mL.master <dbl>, effect.metric <dbl>,  
## # functional.group <dbl>, charge <dbl>, zetapotential.mV <dbl>,  
## # max.size.ingest.mm <dbl>, acute.chronic\_f <dbl>,  
## # dose.mg.L.master.AF.noec <dbl>, dose.particles.mL.master.AF.noec <dbl>,  
## # effect.score <dbl>

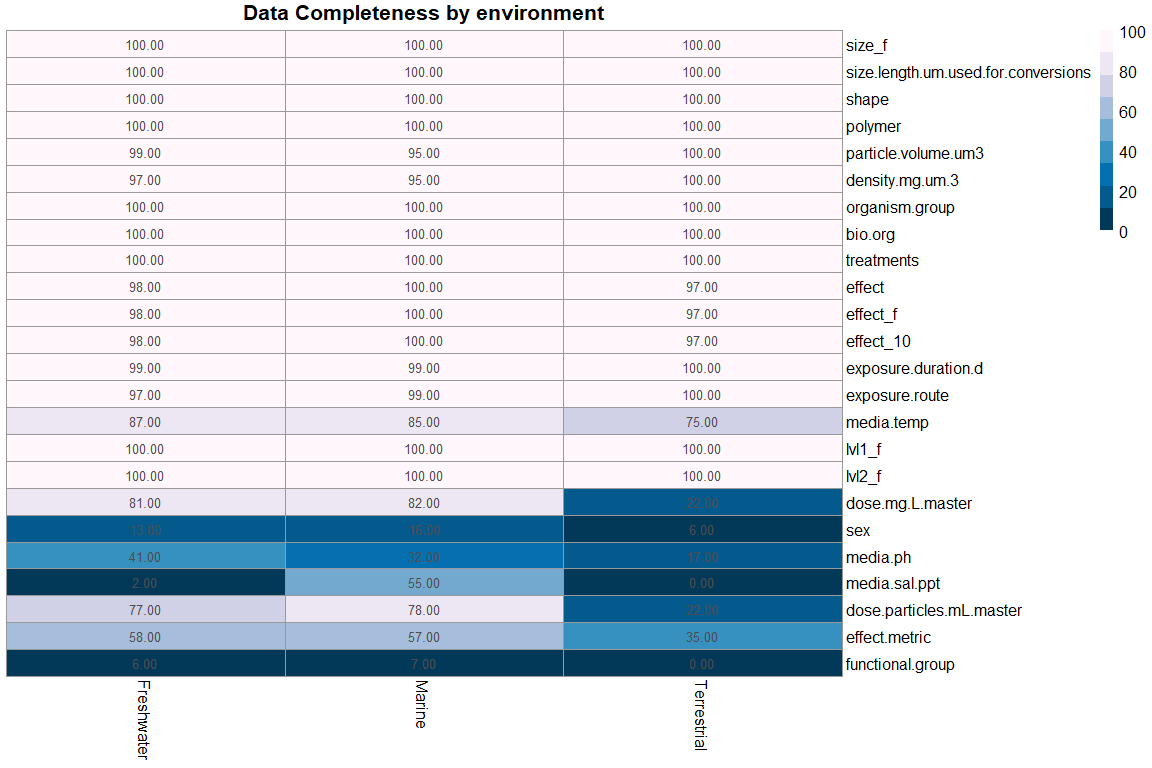
require(pheatmap)  
#convert to matrix and transpose  
transposed\_size <- as.data.frame(t(as.matrix(CompletenessSummary\_size[2:25]))) %>% #1 is category, 2-6 are variables  
 arrange('1nm < 100nm')  
#reassign column names  
colnames(transposed\_size) <- c("1nm < 100nm", "100nm < 1µm", "1µm < 100µm", "100µm < 1mm", "1mm < 5mm")  
#transposedTag  
#format as matrix  
MissingMatrix\_size <- data.matrix(transposed\_size)  
#build heatmap  
pheatmap(MissingMatrix\_size,  
 main = "Data Completeness by Size Bin", #title  
 fontsize = 12,  
 cluster\_rows = FALSE, cluster\_cols = FALSE,#disable dendrograms  
 display\_numbers = TRUE,  
 treeheight\_row = 0, treeheight\_col = 0, #keeps clustering after dropping dendrograms  
 col = rev(brewer.pal(n = 9, name = "PuBu"))) #blue color scheme with 9 colors)

 ### Completeness by Environment

CompletenessSummary\_environment <- multiVar %>%  
 filter(!environment == "NA") %>%   
 group\_by(environment) %>%   
 summarise\_all((name = ~sum(is.na(.))/length(.))) %>%   
 mutate(across(is.numeric,~round(., 2))) %>%   
 mutate(across(is.numeric, ~100 \*(1 -.)))  
CompletenessSummary\_environment

## # A tibble: 3 x 32  
## environment size\_f size.length.um.~ shape polymer particle.volume~  
## <fct> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Freshwater 100 100 100 100 99  
## 2 Marine 100 100 100 100 95  
## 3 Terrestrial 100 100 100 100 100  
## # ... with 26 more variables: density.mg.um.3 <dbl>, organism.group <dbl>,  
## # bio.org <dbl>, treatments <dbl>, effect <dbl>, effect\_f <dbl>,  
## # effect\_10 <dbl>, exposure.duration.d <dbl>, exposure.route <dbl>,  
## # media.temp <dbl>, lvl1\_f <dbl>, lvl2\_f <dbl>, dose.mg.L.master <dbl>,  
## # sex <dbl>, media.ph <dbl>, media.sal.ppt <dbl>,  
## # dose.particles.mL.master <dbl>, effect.metric <dbl>,  
## # functional.group <dbl>, charge <dbl>, zetapotential.mV <dbl>,  
## # max.size.ingest.mm <dbl>, acute.chronic\_f <dbl>,  
## # dose.mg.L.master.AF.noec <dbl>, dose.particles.mL.master.AF.noec <dbl>,  
## # effect.score <dbl>

require(pheatmap)  
#convert to matrix and transpose  
transposed\_environment <- as.data.frame(t(as.matrix(CompletenessSummary\_environment[2:25])))  
#reassign column names  
colnames(transposed\_environment) <- c("Freshwater", "Marine", "Terrestrial")  
#transposedTag  
#format as matrix  
MissingMatrix\_environment <- data.matrix(transposed\_environment)  
#build heatmap  
pheatmap(MissingMatrix\_environment,  
 main = "Data Completeness by environment", #title  
 fontsize = 12,  
 cluster\_rows = FALSE, cluster\_cols = FALSE,#disable dendrograms  
 display\_numbers = TRUE,  
 treeheight\_row = 0, treeheight\_col = 0, #keeps clustering after dropping dendrograms  
 col = rev(brewer.pal(n = 9, name = "PuBu"))) #blue color scheme with 9 colors)

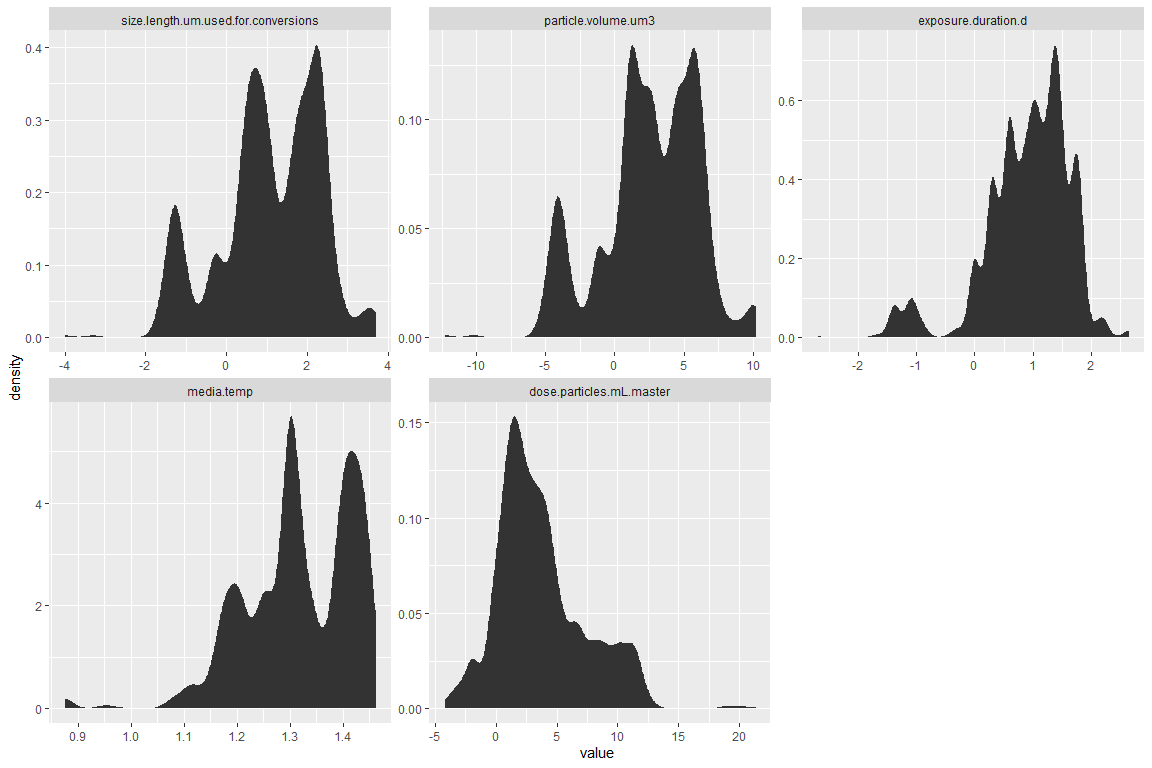


## Stepwise Regression

## Estimate an OLS Regression  
fitols <- lm(effect\_10 ~ size.length.um.used.for.conversions + particle.volume.um3 + exposure.duration.d + media.temp + dose.particles.mL.master,   
 na.action = na.omit,   
 data = multiVar)  
summary(fitols)

##   
## Call:  
## lm(formula = effect\_10 ~ size.length.um.used.for.conversions +   
## particle.volume.um3 + exposure.duration.d + media.temp +   
## dose.particles.mL.master, data = multiVar, na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.5545 -0.3385 -0.3091 0.6448 0.8356   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.104e-01 3.675e-02 5.727 1.10e-08 \*\*\*  
## size.length.um.used.for.conversions -1.251e-04 2.923e-05 -4.280 1.91e-05 \*\*\*  
## particle.volume.um3 4.504e-11 9.431e-12 4.776 1.85e-06 \*\*\*  
## exposure.duration.d 9.134e-04 3.120e-04 2.927 0.00344 \*\*   
## media.temp 5.048e-03 1.691e-03 2.986 0.00284 \*\*   
## dose.particles.mL.master 4.778e-20 1.957e-20 2.441 0.01469 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4682 on 3951 degrees of freedom  
## (1968 observations deleted due to missingness)  
## Multiple R-squared: 0.01223, Adjusted R-squared: 0.01098   
## F-statistic: 9.787 on 5 and 3951 DF, p-value: 2.612e-09

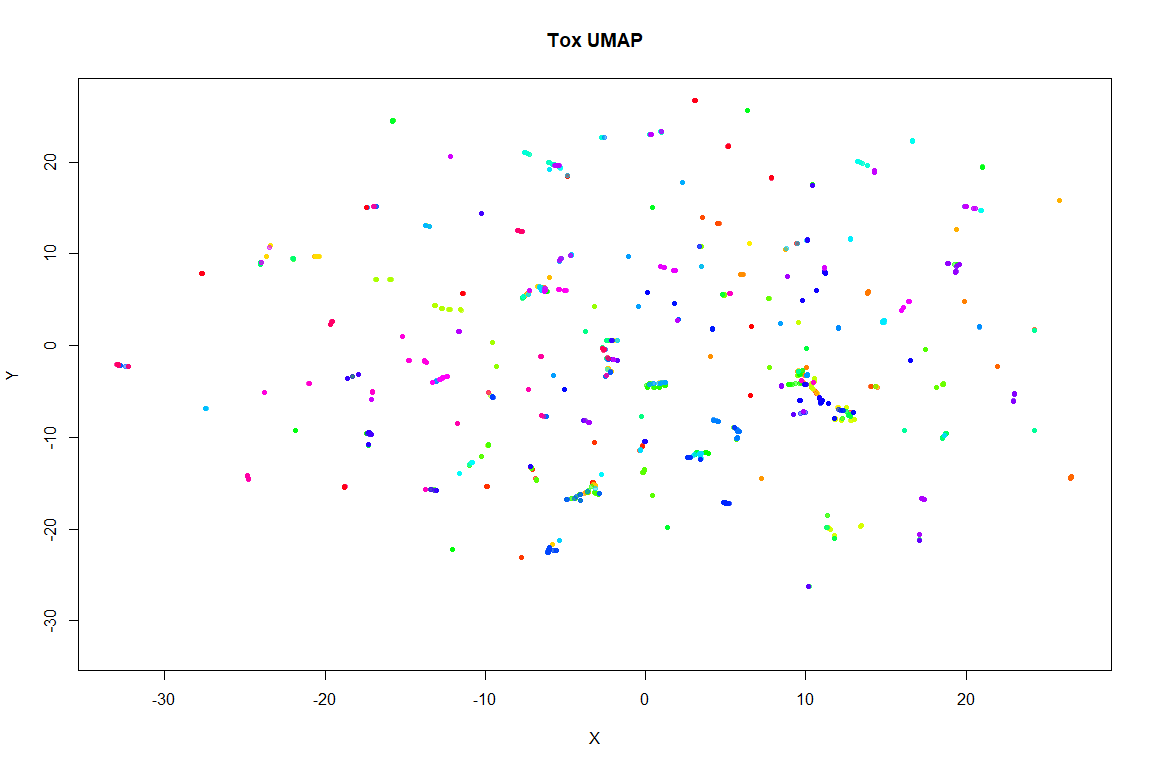
require(reshape2)  
multiVar %>%   
 dplyr::select(size.length.um.used.for.conversions, particle.volume.um3, exposure.duration.d ,media.temp, dose.particles.mL.master) %>%   
 melt() %>% #convert wide to long  
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 ggplot(aes(x = value)) +   
 stat\_density() +   
 facet\_wrap(~variable, scales = "free")



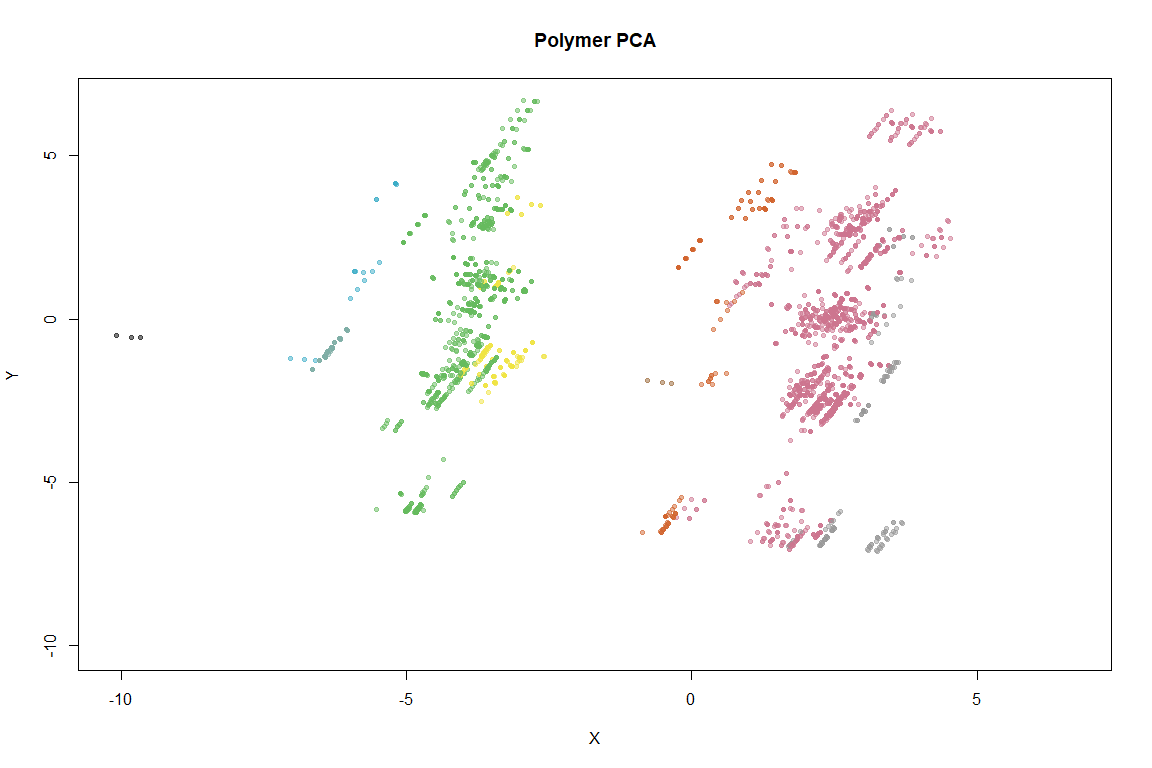
# PCA

# UMAP

require(uwot)  
require(Rtsne)  
require(vizier) #devtools::install\_github("jlmelville/vizier")  
  
  
  
# For some functions we need to strip out non-numeric columns and convert data to matrix  
x2m <- function(X) {  
 if (!methods::is(X, "matrix")) {  
 m <- as.matrix(X[, which(vapply(X, is.numeric, logical(1)))])  
 }  
 else {m <- X}   
 m}  
  
  
#choose values with most completeness  
multiVar2 <- multiVar %>%   
 filter(!environment == "Terrestrial") %>%   
 dplyr::select(size\_f, size.length.um.used.for.conversions, shape, polymer, particle.volume.um3, density.mg.um.3, organism.group, bio.org, treatments, effect, exposure.duration.d, exposure.route, lvl1\_f, dose.mg.L.master) %>%   
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 drop\_na() #drop missing  
  
#convert discrete variables to numeric  
multiVar2[] <- data.matrix(multiVar2)  
  
# build umap for small dataset (<10,000 points)  
multiVar\_map <- umap(multiVar2, pca = 10)  
  
# Remove duplicates for t-SNE  
#multiVar2\_noNa\_dup <- multiVar2\_noNa[-which(duplicated(x2m(multiVar2\_noNa))), ]  
   
#build t-SNE  
#multiVar\_tsne <- Rtsne::Rtsne(multiVar2\_noNa\_dup, perplexity = 15, initial\_dims = 100, partical\_pca = TRUE, exaggeration\_factor = 4)  
  
# Non-numeric columns are ignored, so in a lot of cases you can pass a data  
# frame directly to umap  
#iris\_umap <- umap(iris, n\_neighbors = 50, learning\_rate = 0.5, init = "random")  
  
#visualize umap  
embed\_img <- function(X, Y, k = 15, ...) {  
 args <- list(...)  
 args$coords <- Y  
 args$x <- X  
  
 do.call(vizier::embed\_plot, args)  
}  
  
#plot  
embed\_img(multiVar2, multiVar\_map, pc\_axes = TRUE, equal\_axes = TRUE, alpha\_scale = 0.5, title = "Tox UMAP", cex = 1)



#PCA  
pca <- stats::prcomp(multiVar2[,-5], retx = TRUE, rank. = 2)  
#build color pallete  
my\_colors = colorRampPalette(c("red", "yellow", "green"))(nrow(multiVar2))  
#plot  
embed\_plot(pca$x, multiVar2$polymer, color\_scheme = palette.colors(palette = "Okabe-Ito"), #turbo, #rainbow, #my\_colors,   
 title = "Polymer PCA", alpha\_scale = 0.5, equal\_axes = TRUE)



require(plotly)  
#PCA for discrete variables  
multiVar\_discrete <- multiVar %>%   
 select(size.length.um.used.for.conversions, polymer, dose.mg.L.master, exposure.duration.d) %>%   
 drop\_na %>%   
 mutate\_if(~is.numeric(.) && (.) > 0, log10)  
  
#buildPCA  
pca\_discrete <- stats::prcomp(multiVar\_discrete[,-2], retx = TRUE, rank. = 2)  
  
embed\_plotly(pca\_discrete$x, multiVar\_discrete$polymer, color\_scheme = palette.colors(palette = "Okabe-Ito"),   
 title = "Polymer PCA", alpha\_scale = 0.5, equal\_axes = TRUE,  
 tooltip = paste("Polymer:", multiVar\_discrete$polymer))



## Random Forest

### Recursive Partitioning And Regression Trees

The rpart algorithm works by splitting the dataset recursively, which means that the subsets that arise from a split are further split until a predetermined termination criterion is reached. At each step, the split is made based on the independent variable that results in the largest possible reduction in heterogeneity of the dependent (predicted) variable.

It is important to note that the algorithm works by making the best possible choice at each particular stage, without any consideration of whether those choices remain optimal in future stages. That is, the algorithm makes a locally optimal decision at each stage. It is thus quite possible that such a choice at one stage turns out to be sub-optimal in the overall scheme of things. In other words, the algorithm does not find a globally optimal tree.

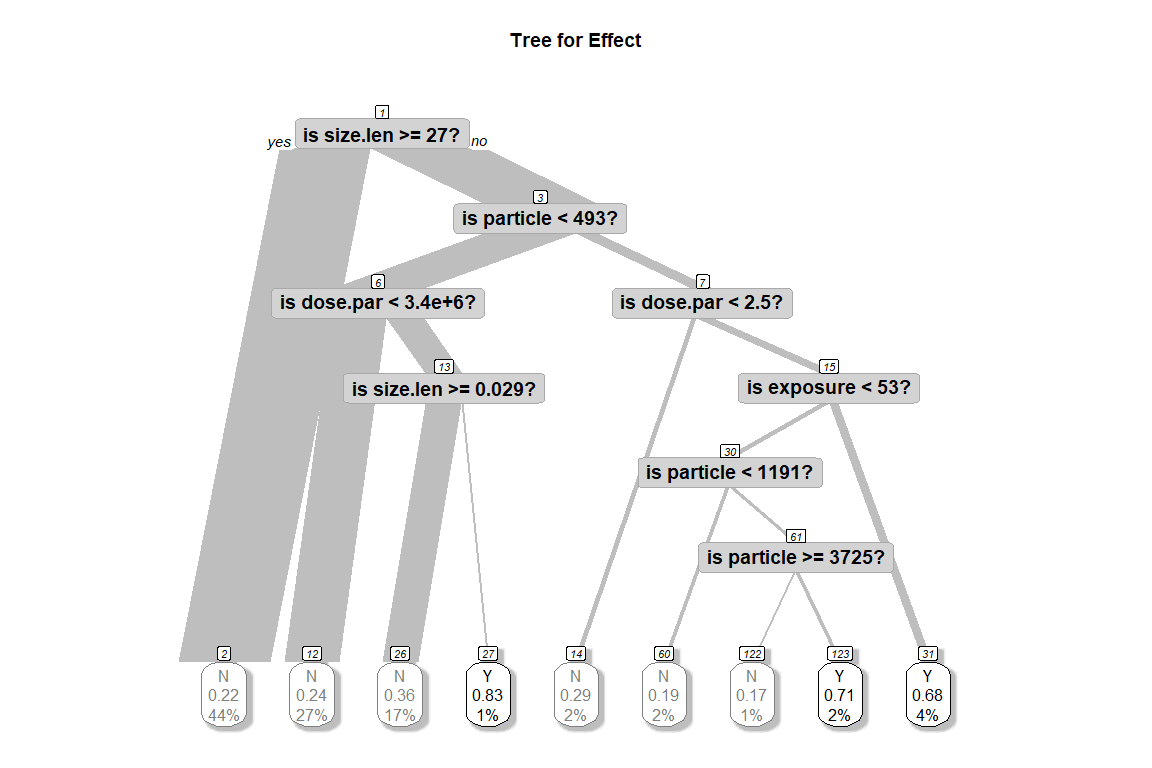
#trim data so effect is always known  
multiVar\_sub <- multiVar %>%   
 drop\_na(effect\_10)  
  
# Split data into training and test sets  
set.seed(42)  
multiVar\_sub[,"train"] <- ifelse(runif(nrow(multiVar\_sub)) < 0.8, 1, 0)  
# Separate trainig and test sets  
trainSet <- multiVar\_sub[multiVar\_sub$train==1,]  
testSet <- multiVar\_sub[multiVar\_sub$train==0,]  
#get column index of train flag  
trainColNum <- grep("train", names(trainSet))  
# Remove train flag column from train and test sets  
trainSet <- trainSet[, -trainColNum]  
testSet <- testSet[, -trainColNum]

Make a classification tree.

## n= 4649   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 4649 1330 N (0.7139 0.2861)   
## 2) size.length.um.used.for.conversions>=26.5 2065 464 N (0.7753 0.2247) \*  
## 3) size.length.um.used.for.conversions< 26.5 2584 866 N (0.6649 0.3351)   
## 6) particle.volume.um3< 493.4 2078 616 N (0.7036 0.2964)   
## 12) dose.particles.mL.master< 3.429e+06 1232 290 N (0.7646 0.2354) \*  
## 13) dose.particles.mL.master>=3.429e+06 846 326 N (0.6147 0.3853)   
## 26) size.length.um.used.for.conversions>=0.02914 805 292 N (0.6373 0.3627) \*  
## 27) size.length.um.used.for.conversions< 0.02914 41 7 Y (0.1707 0.8293) \*  
## 7) particle.volume.um3>=493.4 506 250 N (0.5059 0.4941)   
## 14) dose.particles.mL.master< 2.5 103 30 N (0.7087 0.2913) \*  
## 15) dose.particles.mL.master>=2.5 403 183 Y (0.4541 0.5459)   
## 30) exposure.duration.d< 52.5 203 83 N (0.5911 0.4089)   
## 60) particle.volume.um3< 1191 86 16 N (0.8140 0.1860) \*  
## 61) particle.volume.um3>=1191 117 50 Y (0.4274 0.5726)   
## 122) particle.volume.um3>=3725 30 5 N (0.8333 0.1667) \*  
## 123) particle.volume.um3< 3725 87 25 Y (0.2874 0.7126) \*  
## 31) exposure.duration.d>=52.5 200 63 Y (0.3150 0.6850) \*

Plot an interpretable tree.

## cex 1 xlim c(-0.2, 1.2) ylim c(0, 1)

 Next we see how good the model is by seeing how it fares against the test data.

t1\_predict <- predict(t1, newdata = testSet[,-typeColNum],  
 type="class")  
mean(t1\_predict==testSet$effect)

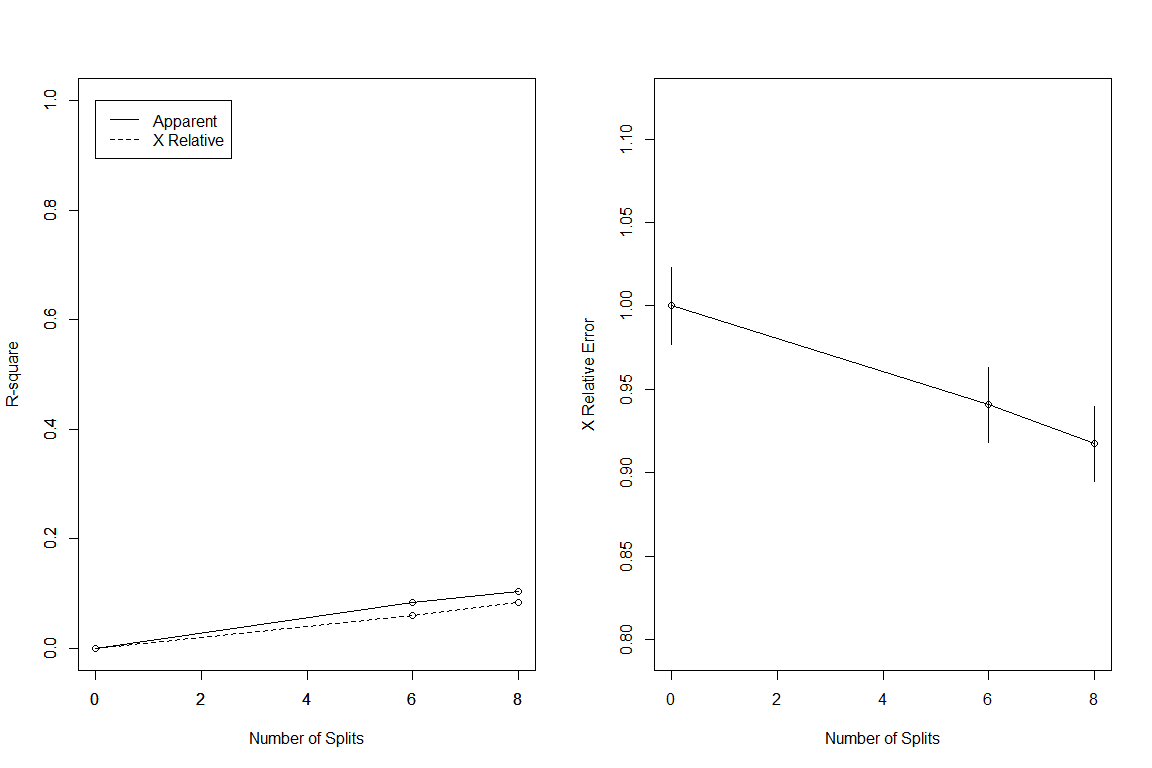
## [1] 0.7094088

# [1] 0.7094088  
#confusion matrix  
table(pred=t1\_predict,true=testSet$effect)

## true  
## pred N Y  
## N 792 323  
## Y 26 60

par(mfrow=c(1,2)) # two plots on one page  
#plot approximate R-squared and relative error for different splits (2 plots). labels are only appropriate for the "anova" method.  
rsq.rpart(t1)

##   
## Classification tree:  
## rpart(formula = effect ~ size.length.um.used.for.conversions +   
## particle.volume.um3 + exposure.duration.d + media.temp +   
## dose.particles.mL.master, data = trainSet, method = "class",   
## control = rpart.control(minbucket = 20, cp = 0.008))  
##   
## Variables actually used in tree construction:  
## [1] dose.particles.mL.master exposure.duration.d   
## [3] particle.volume.um3 size.length.um.used.for.conversions  
##   
## Root node error: 1330/4649 = 0.28608  
##   
## n= 4649   
##   
## CP nsplit rel error xerror xstd  
## 1 0.01391 0 1.00000 1.00000 0.023169  
## 2 0.01015 6 0.91654 0.94060 0.022736  
## 3 0.00800 8 0.89624 0.91729 0.022554



Next, we prune the tree using the cost complexity criterion. Basically, the intent is to see if a shallower subtree can give us comparable results. If so, we’d be better of choosing the shallower tree because it reduces the likelihood of overfitting.

As described earlier, we choose the appropriate pruning parameter (aka cost-complexity parameter) by picking the value that results in the lowest prediction error. Note that all relevant computations have already been carried out by R when we built the original tree (the call to rpart in the code above). All that remains now is to pick the value of :

#### Pruning

#cost-complexity pruning  
printcp(t1)

##   
## Classification tree:  
## rpart(formula = effect ~ size.length.um.used.for.conversions +   
## particle.volume.um3 + exposure.duration.d + media.temp +   
## dose.particles.mL.master, data = trainSet, method = "class",   
## control = rpart.control(minbucket = 20, cp = 0.008))  
##   
## Variables actually used in tree construction:  
## [1] dose.particles.mL.master exposure.duration.d   
## [3] particle.volume.um3 size.length.um.used.for.conversions  
##   
## Root node error: 1330/4649 = 0.28608  
##   
## n= 4649   
##   
## CP nsplit rel error xerror xstd  
## 1 0.01391 0 1.00000 1.00000 0.023169  
## 2 0.01015 6 0.91654 0.94060 0.022736  
## 3 0.00800 8 0.89624 0.91729 0.022554

It is clear from the above, that the lowest cross-validation error (xerror in the table) occurs for =0.008 (this is CP in the table above). One can find CP programatically like so:

# get index of CP with lowest xerror  
opt <- which.min(t1$cptable[,"xerror"])  
#get its value  
cp <- t1$cptable[opt, "CP"]

Next, we prune the tree based on this value of CP:

# # prune the tree  
# pt1 <- prune(t1,cp)  
# pt1<- prune(t1, cp= t1$cptable[which.min(t1$cptable[,"xerror"]),"CP"])  
#   
# # plot the pruned tree  
# plot(pt1, uniform=TRUE,  
# main="Pruned Classification Tree");text(pt1, use.n=TRUE, all=TRUE, cex=.8)  
#   
# #post(pfit, file = "c:/ptree.ps",  
# # title = "Pruned Classification Tree for Kyphosis")

# #find proportion of correct predictions using test set  
# t1\_pruned\_predict <- predict(pt1, testSet, type="class")  
# mean(t1\_pruned\_predict == testSet$effect)  
# #

This is not an improvement over an unprunend tree.. We need to check that this holds up for different training and test sets. This is easily done by creating multiple random partitions of the dataset and checking the efficacy of pruning for each. To do this efficiently, I’ll create a function that takes the training fraction, number of runs (partitions) and the name of the dataset as inputs and outputs the proportion of correct predictions for each run. It also optionally prunes the tree.

# {r} <!-- # #function to do multiple runs --> <!-- # multiple\_runs\_classification <- function(train\_fraction,n,dataset,prune\_tree=FALSE){ --> <!-- # fraction\_correct <- rep(NA,n) --> <!-- # set.seed(42) --> <!-- # for (i in 1:n){ --> <!-- # dataset[,"train"] <- ifelse(runif(nrow(dataset))<0.8,1,0) --> <!-- # trainColNum <- grep("train",names(dataset)) --> <!-- # typeColNum <- grep("effect",names(dataset)) --> <!-- # trainSet <- dataset[dataset$train==1,-trainColNum] --> <!-- # testSet <- dataset[dataset$train==0,-trainColNum] --> <!-- # rpart\_model <- rpart(effect~ size.length.um.used.for.conversions + particle.volume.um3 + exposure.duration.d + media.temp + dose.particles.mL.master,data = trainSet, method="class") --> <!-- # if(prune\_tree==FALSE) { --> <!-- # rpart\_test\_predict <- predict(rpart\_model,testSet[,-typeColNum],type="class") --> <!-- # fraction\_correct[i] <- mean(rpart\_test\_predict==testSet$effect) --> <!-- # }else{ --> <!-- # opt <- which.min(rpart\_model$cptable[,"xerror"]) --> <!-- # cp <- rpart\_model$cptable[opt, "CP"] --> <!-- # pruned\_model <- prune(rpart\_model,cp) --> <!-- # rpart\_pruned\_predict <- predict(pruned\_model,testSet[,-typeColNum],type="class") --> <!-- # fraction\_correct[i] <- mean(rpart\_pruned\_predict == testSet$effect) --> <!-- # } --> <!-- # } --> <!-- # return(fraction\_correct) --> <!-- # } --> <!-- # –>

### CForest

require(party)  
crf <- cforest(as.factor(effect\_10) ~ size.length.um.used.for.conversions + particle.volume.um3 + exposure.duration.d +  
 media.temp + dose.particles.mL.master,  
 controls = cforest\_control(ntree = 500,  
 mincriterion = qnorm(0.8),   
 trace = TRUE), # adds project bar because it's very slow  
 data = multiVar\_sub)

##   
## [> ] 0% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[=> ] 2% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[==> ] 4% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[===> ] 6% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[====> ] 8% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[=====> ] 10% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[======> ] 12% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[=======> ] 14% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[========> ] 16% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[=========> ] 18% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[==========> ] 20% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[===========> ] 22% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[============> ] 24% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[=============> ] 26% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[==============> ] 28% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[===============> ] 30% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[================> ] 32% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[=================> ] 34% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[==================> ] 36% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[===================> ] 38% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[====================> ] 40% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[=====================> ] 42% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[======================> ] 44% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[=======================> ] 46% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[========================> ] 48% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[=========================> ] 50% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[==========================> ] 52% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[===========================> ] 54% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[============================> ] 56% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[=============================> ] 58% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[==============================> ] 60% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[===============================> ] 62% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[================================> ] 64% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[=================================> ] 66% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[==================================> ] 68% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[===================================> ] 70% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[====================================> ] 72% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[=====================================> ] 74% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[======================================> ] 76% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[=======================================> ] 78% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[========================================> ] 80% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[=========================================> ] 82% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[==========================================> ] 84% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[===========================================> ] 86% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[============================================> ] 88% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[=============================================> ] 90% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[==============================================> ] 92% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[===============================================> ] 94% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[================================================> ] 96% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[=================================================> ] 98% completed[==================================================>] 100% completed[==================================================>] 100% completed[==================================================>] 100% completed[==================================================>] 100% completed[==================================================>] 100% completed[==================================================>] 100% completed[==================================================>] 100% completed[==================================================>] 100% completed[==================================================>] 100% completed

crf

##   
## Random Forest using Conditional Inference Trees  
##   
## Number of trees: 500   
##   
## Response: as.factor(effect\_10)   
## Inputs: size.length.um.used.for.conversions, particle.volume.um3, exposure.duration.d, media.temp, dose.particles.mL.master   
## Number of observations: 5850

train2 <- multiVar\_sub %>% dplyr::select(size.length.um.used.for.conversions,  
 particle.volume.um3,exposure.duration.d,media.temp,dose.particles.mL.master)  
train3 <-multiVar\_sub %>% dplyr::select(size.length.um.used.for.conversions,  
 particle.volume.um3,exposure.duration.d,media.temp,dose.particles.mL.master,  
 effect\_10)  
  
fitted <- predict(crf, train2, OOB = TRUE, type ="response")  
#rpart.prob <- predict(t1, newdata=imputedSmalls.requested.voluntary,type="prob")  
  
misClasificError <- mean(fitted != train3$effect\_10)  
print(paste('Training Accuracy', 1 - misClasificError))

## [1] "Training Accuracy 0.792649572649573"

print(crf)

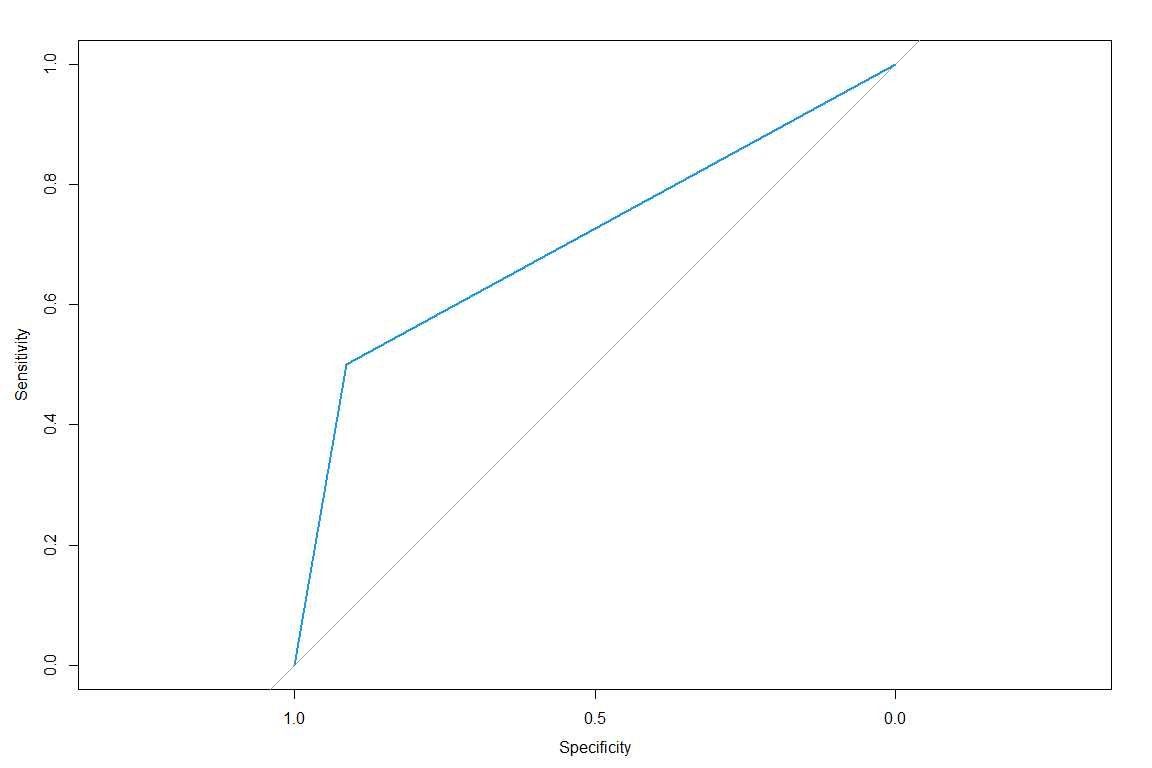
##   
## Random Forest using Conditional Inference Trees  
##   
## Number of trees: 500   
##   
## Response: as.factor(effect\_10)   
## Inputs: size.length.um.used.for.conversions, particle.volume.um3, exposure.duration.d, media.temp, dose.particles.mL.master   
## Number of observations: 5850

Alternative ROC Curve

require(pROC)  
predicted <- predict(crf, train2, OOB=TRUE, type= "response")  
auc(as.numeric(train3$effect\_10), as.numeric(predicted))

## [1] 0.7071698

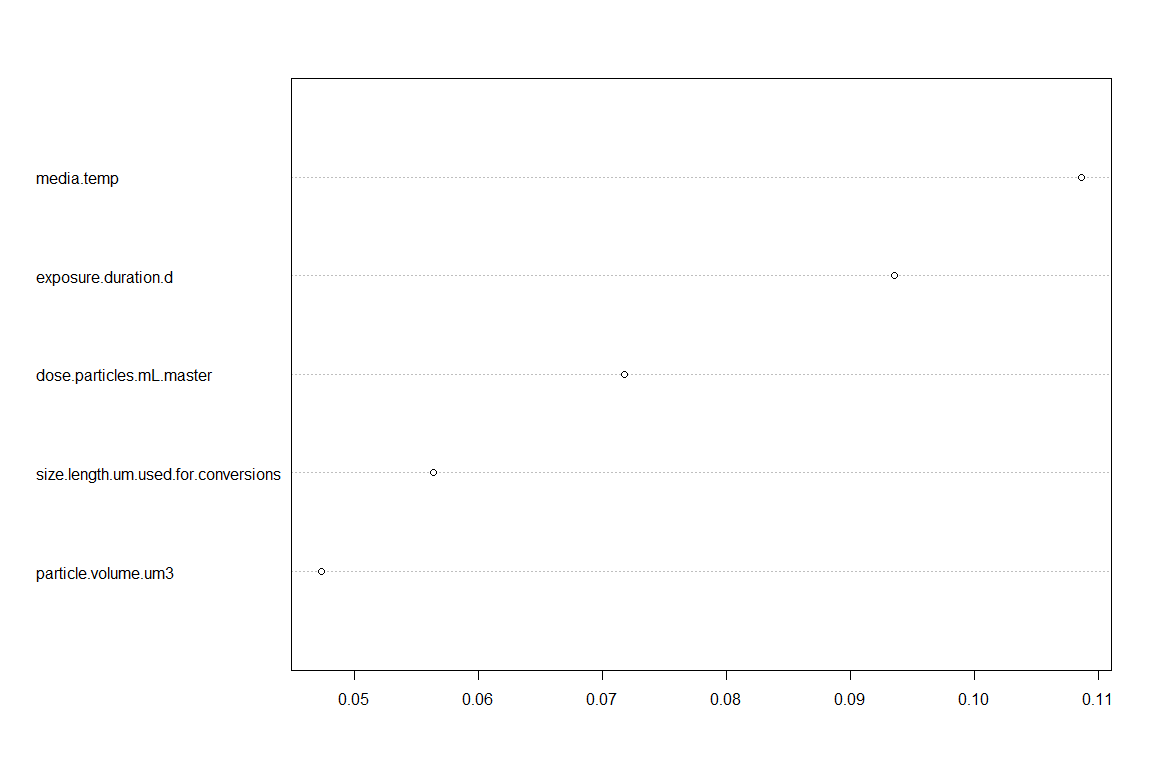
#Calculate ROC curve  
rocCurve.tree <- roc(train3$effect\_10,as.numeric(predicted))  
#plot the ROC curve  
plot(rocCurve.tree,col=c(4))



# compute in-sample results  
caret::confusionMatrix(fitted,as.factor(train3$effect\_10))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3779 855  
## 1 358 858  
##   
## Accuracy : 0.7926   
## 95% CI : (0.782, 0.803)  
## No Information Rate : 0.7072   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4528   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9135   
## Specificity : 0.5009   
## Pos Pred Value : 0.8155   
## Neg Pred Value : 0.7056   
## Prevalence : 0.7072   
## Detection Rate : 0.6460   
## Detection Prevalence : 0.7921   
## Balanced Accuracy : 0.7072   
##   
## 'Positive' Class : 0   
##

#plot feature importance  
cforestImpPlot <- function(x) {  
 cforest\_importance <<- v <- varimp(x)  
 dotchart(v[order(v)])  
}  
  
importancePlot <- cforestImpPlot(crf)



importancePlot

## NULL

#GLMM

#tutorial https://aosmith.rbind.io/2020/08/20/simulate-binomial-glmm/  
require(lme4)  
  
mod = glm(effect\_f ~ size.length.um.used.for.conversions + polymer + particle.volume.um3 + density.mg.um.3 + organism.group + bio.org + treatments + effect\_f + exposure.duration.d + exposure.route + lvl1\_f + dose.mg.L.master,   
 data = aoc\_z,  
 family = binomial(link = "logit") )  
mod

##   
## Call: glm(formula = effect\_f ~ size.length.um.used.for.conversions +   
## polymer + particle.volume.um3 + density.mg.um.3 + organism.group +   
## bio.org + treatments + effect\_f + exposure.duration.d + exposure.route +   
## lvl1\_f + dose.mg.L.master, family = binomial(link = "logit"),   
## data = aoc\_z)  
##   
## Coefficients:  
## (Intercept)   
## 2.003e+00   
## size.length.um.used.for.conversions   
## 1.168e-04   
## polymerNot Reported   
## 1.946e-01   
## polymerPA   
## -9.836e-01   
## polymerPE   
## -5.899e-01   
## polymerPET   
## -9.957e-01   
## polymerPMMA   
## 1.376e+00   
## polymerPP   
## -4.278e-01   
## polymerPS   
## 1.663e-01   
## polymerPVC   
## 4.586e-01   
## particle.volume.um3   
## 9.749e-11   
## density.mg.um.3   
## -9.536e+08   
## organism.groupAnnelida   
## 6.760e-01   
## organism.groupBacterium   
## 6.240e-02   
## organism.groupCnidaria   
## 2.218e-01   
## organism.groupCrustacea   
## 8.432e-01   
## organism.groupEchinoderm   
## 1.932e-01   
## organism.groupFish   
## 1.706e-01   
## organism.groupMixed   
## -1.203e-01   
## organism.groupMollusca   
## -4.522e-01   
## organism.groupNematoda   
## 9.088e-01   
## organism.groupPlant   
## 1.037e+00   
## organism.groupRotifera   
## 1.372e+00   
## bio.orgorganism   
## -1.448e+00   
## bio.orgpopulation   
## -1.837e+00   
## bio.orgsubcell   
## -6.195e-01   
## bio.orgtissue   
## -1.003e+00   
## treatments   
## 5.067e-02   
## exposure.duration.d   
## 7.919e-03   
## exposure.routemedia   
## NA   
## exposure.routesediment   
## NA   
## exposure.routewater   
## -8.275e-01   
## lvl1\_fBehavioral, Sensory, Neurological   
## 2.349e-02   
## lvl1\_fCirculatory, Respiratory   
## -7.212e-01   
## lvl1\_fCommunity   
## NA   
## lvl1\_fFitness   
## -8.014e-01   
## lvl1\_fImmune   
## -4.756e-01   
## lvl1\_fMetabolism   
## -3.565e-01   
## lvl1\_fMicrobiome   
## -1.775e-01   
## lvl1\_fStress   
## -4.839e-01   
## dose.mg.L.master   
## -2.967e-08   
##   
## Degrees of Freedom: 4264 Total (i.e. Null); 4227 Residual  
## (1785 observations deleted due to missingness)  
## Null Deviance: 5341   
## Residual Deviance: 4902 AIC: 4978

bin\_glmm\_fun = function(n\_sites = 10,  
 b0 = 0,  
 b1 = 1.735,  
 num\_samp = 50,  
 site\_var = 0.5) {  
 site = rep(LETTERS[1:n\_sites], each = 2)  
 plot = paste(site, rep(1:2, times = n\_sites), sep = "." )  
 treatment = rep( c("treatment", "control"), times = n\_sites)  
 dat = data.frame(site, plot, treatment)   
   
 site\_eff = rep( rnorm(n = n\_sites, mean = 0, sd = sqrt(site\_var) ), each = 2)  
   
 log\_odds = with(dat, b0 + b1\*(treatment == "treatment") + site\_eff)  
 prop = plogis(log\_odds)  
 dat$num\_samp = num\_samp  
 dat$y = rbinom(n = n\_sites\*2, size = num\_samp, prob = prop)  
   
 glmer(cbind(y, num\_samp - y) ~ treatment + (1|site),  
 data = dat,  
 family = binomial(link = "logit") )  
}  
  
  
set.seed(16)  
bin\_glmm\_fun()

## Generalized linear mixed model fit by maximum likelihood (Laplace  
## Approximation) [glmerMod]  
## Family: binomial ( logit )  
## Formula: cbind(y, num\_samp - y) ~ treatment + (1 | site)  
## Data: dat  
## AIC BIC logLik deviance df.resid   
## 122.6154 125.6025 -58.3077 116.6154 17   
## Random effects:  
## Groups Name Std.Dev.  
## site (Intercept) 0.4719   
## Number of obs: 20, groups: site, 10  
## Fixed Effects:  
## (Intercept) treatmenttreatment   
## 0.1576 1.4859

# Random Forest Package

## Preparation

library(quantregForest)  
library(caret)  
library(tidyverse)  
library(tidymodels)  
library(skimr)  
library(sf)  
library(ggspatial)  
library(nhdplusTools)  
library(patchwork)  
library(Metrics)  
library(gt)

aoc\_z\_2 <- aoc\_z %>%   
 mutate\_if(is.character, as.factor)# %>%  
 # mutate(effect\_10 = case\_when( #convert ordinal to numeric  
 # effect\_f == "Yes" ~ 1,  
 # effect\_f == "No" ~ 0))  
  
#choose relevant predictors and log-transform  
multiVar\_small\_log <- aoc\_z\_2 %>%   
 dplyr::select(size\_f,  
 ID, #row number ID for split/joining by study  
 doi, #need to split studies  
 size.length.um.used.for.conversions, shape, polymer, particle.volume.um3, density.mg.um.3, organism.group, bio.org, treatments, effect\_f, exposure.duration.d, exposure.route, lvl1\_f, dose.mg.L.master) %>%   
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 drop\_na() %>% #drop missing  
 mutate(effect\_10 = case\_when( #convert ordinal to numeric  
 effect\_f == "Y" ~ 1,  
 effect\_f == "N" ~ 0  
 ))# %>%   
 # select(-effect\_f)  
skim(multiVar\_small\_log)

Data summary

|  |  |
| --- | --- |
| Name | multiVar\_small\_log |
| Number of rows | 4265 |
| Number of columns | 17 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 10 |
| numeric | 7 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| size\_f | 0 | 1 | FALSE | 5 | 1µm: 2419, 100: 794, 1nm: 542, 100: 380 |
| ID | 0 | 1 | FALSE | 4265 | ID1: 1, ID1: 1, ID1: 1, ID1: 1 |
| doi | 0 | 1 | FALSE | 121 | 10.: 372, 10.: 144, 10.: 144, 10.: 139 |
| shape | 0 | 1 | FALSE | 3 | sph: 2763, fra: 1391, fib: 111, Not: 0 |
| polymer | 0 | 1 | FALSE | 9 | PS: 2311, PE: 1318, PP: 211, PET: 143 |
| organism.group | 0 | 1 | FALSE | 12 | Fis: 1449, Cru: 1092, Mol: 1032, Alg: 333 |
| bio.org | 0 | 1 | FALSE | 5 | sub: 1886, org: 1689, cel: 395, tis: 238 |
| effect\_f | 0 | 1 | FALSE | 2 | N: 2904, Y: 1361 |
| exposure.route | 0 | 1 | FALSE | 4 | wat: 4151, foo: 63, med: 34, sed: 17 |
| lvl1\_f | 0 | 1 | FALSE | 9 | Fit: 2033, Met: 1331, Beh: 293, Imm: 182 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| size.length.um.used.for.conversions | 0 | 1 | 0.89 | 1.26 | -4.00 | 0.30 | 1.00 | 1.84 | 3.70 | ▁▂▅▇▃ |
| particle.volume.um3 | 0 | 1 | 2.12 | 3.58 | -12.28 | 0.43 | 2.43 | 4.68 | 10.21 | ▁▂▅▇▃ |
| density.mg.um.3 | 0 | 1 | -8.98 | 0.04 | -9.06 | -9.01 | -8.97 | -8.97 | -8.85 | ▃▂▇▁▁ |
| treatments | 0 | 1 | 0.38 | 0.28 | 0.00 | 0.00 | 0.48 | 0.60 | 1.00 | ▇▃▆▇▁ |
| exposure.duration.d | 0 | 1 | 0.83 | 0.73 | -2.70 | 0.48 | 0.90 | 1.32 | 2.23 | ▁▁▁▇▅ |
| dose.mg.L.master | 0 | 1 | -0.31 | 2.06 | -11.64 | -1.49 | -0.30 | 1.00 | 8.17 | ▁▁▇▅▁ |
| effect\_10 | 0 | 1 | 0.32 | 0.47 | 0.00 | 0.00 | 0.00 | 1.00 | 1.00 | ▇▁▁▁▃ |

#no log transform for comparison  
multiVar\_small <- aoc\_z %>%   
 dplyr::select(size\_f, size.length.um.used.for.conversions, shape, polymer, particle.volume.um3, density.mg.um.3, organism.group, bio.org, treatments, effect\_f, life.stage,  
 exposure.duration.d, lvl1\_f, dose.mg.L.master, dose.particles.mL.master, dose.um3.mL.master)# %>%   
 #drop\_na() #%>% #drop missing  
# mutate(effect\_10 = case\_when( #convert ordinal to numeric  
# effect == "Y" ~ 1,  
# effect == "N" ~ 0  
# )) %>%   
 # select(-effect)  
  
#ensure completeness  
skim(multiVar\_small)

Data summary

|  |  |
| --- | --- |
| Name | multiVar\_small |
| Number of rows | 6050 |
| Number of columns | 16 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 8 |
| numeric | 8 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| size\_f | 0 | 1.00 | FALSE | 6 | 1µm: 3241, 100: 1461, 1nm: 640, 100: 410 |
| shape | 0 | 1.00 | FALSE | 4 | sph: 3213, fra: 2410, Not: 267, fib: 160 |
| polymer | 0 | 1.00 | FALSE | 15 | PS: 2859, PE: 1815, PVC: 377, Not: 288 |
| organism.group | 0 | 1.00 | FALSE | 13 | Fis: 2125, Cru: 1415, Mol: 1279, Alg: 425 |
| bio.org | 0 | 1.00 | FALSE | 5 | org: 2574, sub: 2505, cel: 552, tis: 298 |
| effect\_f | 75 | 0.99 | FALSE | 2 | N: 4198, Y: 1777 |
| life.stage | 0 | 1.00 | FALSE | 4 | Adu: 2559, Ear: 1804, Juv: 1108, Not: 579 |
| lvl1\_f | 0 | 1.00 | FALSE | 9 | Fit: 2804, Met: 1807, Beh: 491, Imm: 303 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| size.length.um.used.for.conversions | 125 | 0.98 | 1.708200e+02 | 6.172300e+02 | 0 | 3.00 | 10.5 | 126.0 | 5.000000e+03 | ▇▁▁▁▁ |
| particle.volume.um3 | 289 | 0.95 | 2.540824e+08 | 1.818038e+09 | 0 | 10.47 | 523.6 | 261848.6 | 1.636246e+10 | ▇▁▁▁▁ |
| density.mg.um.3 | 293 | 0.95 | 0.000000e+00 | 0.000000e+00 | 0 | 0.00 | 0.0 | 0.0 | 0.000000e+00 | ▅▇▁▂▁ |
| treatments | 0 | 1.00 | 2.910000e+00 | 1.810000e+00 | 1 | 1.00 | 3.0 | 4.0 | 1.000000e+01 | ▇▆▂▁▁ |
| exposure.duration.d | 72 | 0.99 | 2.095000e+01 | 3.656000e+01 | 0 | 4.00 | 10.0 | 28.0 | 4.500000e+02 | ▇▁▁▁▁ |
| dose.mg.L.master | 1265 | 0.79 | 7.615795e+04 | 2.443881e+06 | 0 | 0.03 | 0.5 | 12.5 | 1.484403e+08 | ▇▁▁▁▁ |
| dose.particles.mL.master | 1561 | 0.74 | 1.113595e+18 | 3.952454e+19 | 0 | 14.00 | 1000.0 | 350777.9 | 2.280000e+21 | ▇▁▁▁▁ |
| dose.um3.mL.master | 1561 | 0.74 | 1.777625e+11 | 4.606684e+12 | 0 | 26231.62 | 934579.4 | 20746888.0 | 2.112000e+14 | ▇▁▁▁▁ |

## Training Data

### NON-LOG TRANSFORMED

#rough fix  
multiVar\_small\_roughfix <- na.roughfix(multiVar\_small)  
  
# Create calibration and validation splits with tidymodels initial\_split() function.  
set.seed(4)  
multiVar\_small\_split <- multiVar\_small\_roughfix %>%  
 initial\_split(prop = 0.75, strata = polymer) # splits data into training and testing set.  
# default is 3/4ths split (but 75% training, 25% testing).  
# Stratification (strata) = grouping training/testing sets by region, state, etc.  
# Using the "strata" call ensures the number of data points in the training data is equivalent to the proportions in the original data set. (Strata below 10% of the total are pooled together.)  
  
# Create a training data set with the training() function  
# Pulls from training and testing sets created by initial\_split()  
multiVar\_small\_train <- training(multiVar\_small\_split)  
multiVar\_small\_test <- testing(multiVar\_small\_split)  
# Examine the environment to be sure # of observations looks like the 75/25 split. 3199:1066.  
  
# Create a separate dataset of available IDs that were not used in the training dataset.  
notTrain <- aoc %>% # all COMIDS from StreamCat data, sampled or not  
 filter(!ID %in% aoc$ID) # Removing sites used to train the model. n = 140,097  
  
count\_optimized <- paste0('n = ',nrow(multiVar\_small\_roughfix))  
  
skim(multiVar\_small\_roughfix)

Data summary

|  |  |
| --- | --- |
| Name | multiVar\_small\_roughfix |
| Number of rows | 6050 |
| Number of columns | 16 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 8 |
| numeric | 8 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| size\_f | 0 | 1 | FALSE | 6 | 1µm: 3241, 100: 1461, 1nm: 640, 100: 410 |
| shape | 0 | 1 | FALSE | 4 | sph: 3213, fra: 2410, Not: 267, fib: 160 |
| polymer | 0 | 1 | FALSE | 15 | PS: 2859, PE: 1815, PVC: 377, Not: 288 |
| organism.group | 0 | 1 | FALSE | 13 | Fis: 2125, Cru: 1415, Mol: 1279, Alg: 425 |
| bio.org | 0 | 1 | FALSE | 5 | org: 2574, sub: 2505, cel: 552, tis: 298 |
| effect\_f | 0 | 1 | FALSE | 2 | N: 4273, Y: 1777 |
| life.stage | 0 | 1 | FALSE | 4 | Adu: 2559, Ear: 1804, Juv: 1108, Not: 579 |
| lvl1\_f | 0 | 1 | FALSE | 9 | Fit: 2804, Met: 1807, Beh: 491, Imm: 303 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| size.length.um.used.for.conversions | 0 | 1 | 1.675100e+02 | 6.112400e+02 | 0 | 3.00 | 10.5 | 111.70 | 5.000000e+03 | ▇▁▁▁▁ |
| particle.volume.um3 | 0 | 1 | 2.419453e+08 | 1.774904e+09 | 0 | 11.78 | 523.6 | 143793.31 | 1.636246e+10 | ▇▁▁▁▁ |
| density.mg.um.3 | 0 | 1 | 0.000000e+00 | 0.000000e+00 | 0 | 0.00 | 0.0 | 0.00 | 0.000000e+00 | ▃▇▁▁▁ |
| treatments | 0 | 1 | 2.910000e+00 | 1.810000e+00 | 1 | 1.00 | 3.0 | 4.00 | 1.000000e+01 | ▇▆▂▁▁ |
| exposure.duration.d | 0 | 1 | 2.082000e+01 | 3.636000e+01 | 0 | 4.00 | 10.0 | 28.00 | 4.500000e+02 | ▇▁▁▁▁ |
| dose.mg.L.master | 0 | 1 | 6.023412e+04 | 2.173592e+06 | 0 | 0.10 | 0.5 | 4.90 | 1.484403e+08 | ▇▁▁▁▁ |
| dose.particles.mL.master | 0 | 1 | 8.262695e+17 | 3.404834e+19 | 0 | 50.10 | 1000.0 | 15620.76 | 2.280000e+21 | ▇▁▁▁▁ |
| dose.um3.mL.master | 0 | 1 | 1.318971e+11 | 3.968775e+12 | 0 | 93457.94 | 934579.4 | 7462686.57 | 2.112000e+14 | ▇▁▁▁▁ |

## Kitchen Sink Model

### LINEAR DATA

# Step Three - Kitchen Sink model -----------------------------------------  
# Random forest --   
# a decision tree model, using predictors to answer dichotomous questions to create nested splits.  
# no pruning happens - rather, multiple trees are built (the forest) and then you are looking for consensus across trees  
# training data goes down the tree and ends up in a terminal node.  
# if testing data goes down the same route, then this upholds our conclusions. Or, if it goes awry, this allows us to look for patterns in how it goes awry.  
  
set.seed(2) # assures the data pulled is random, but sets it for the run below (makes outcome stable)  
myrf <- randomForest(y = multiVar\_small\_roughfix$effect\_f, # dependent variable  
 x = multiVar\_small\_roughfix %>%  
 dplyr::select(-effect\_f), # selecting all predictor variables  
 importance = T, # how useful is a predictor in predicting values (nothing causal)  
 proximity = T,   
 na.action = na.roughfix,  
 ntrees = 100) # 500 trees default.   
  
myrf # examine the results.

##   
## Call:  
## randomForest(x = multiVar\_small\_roughfix %>% dplyr::select(-effect\_f), y = multiVar\_small\_roughfix$effect\_f, importance = T, proximity = T, na.action = na.roughfix, ntrees = 100)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 19.97%  
## Confusion matrix:  
## N Y class.error  
## N 3848 425 0.09946174  
## Y 783 994 0.44063028

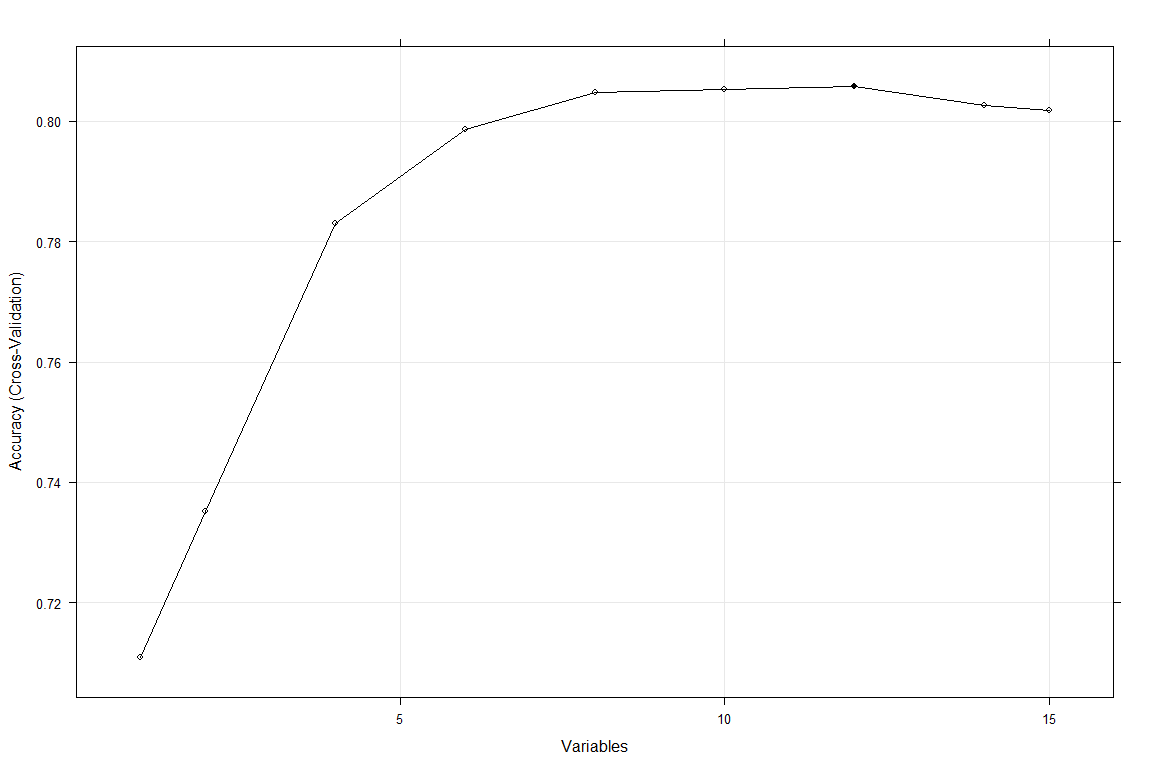
~20% error rate. Let’s compare this with the same model with log-transformed values.

###### Predictor Selector

# Using caret to select the best predictors  
# What are the parameters you want to use to run recursive feature elimination (rfe)?  
my\_ctrl <- rfeControl(functions = rfFuncs,  
 method = "cv",  
 verbose = FALSE,  
 returnResamp = "all")  
  
# rfe = recursive feature elimination  
# THIS STEP TAKES FOR-EV-ER!!!  
set.seed(22)  
my\_rfe <- rfe(y = multiVar\_small\_roughfix$effect\_f, # set dependent variable  
 x = multiVar\_small\_roughfix %>%   
 dplyr::select(-effect\_f),  
 rfeControl = my\_ctrl,  
 size = c(1:2, 4, 6, 8, 10, 12, 14, 15))#,  
 # na.action = na.roughfix())  
  
# sets how many variables are in the overall model  
# # I have 13 total possible variables, so I've chosen increments of 3 to look at.  
# rfeControl = my\_ctrl,  
# na.action = na.roughfix,  
# testX = multiVar\_small\_test %>% dplyr::select(-effect\_f),  
# testY = multiVar\_small\_test$effect\_f)  
  
# can you make your model even simpler?  
# the following will pick a model with the smallest number of predictor variables based on the tolerance ("tol") that you specify (how much less than the best are you willing to tolerate?)  
  
#inspect  
my\_rfe

##   
## Recursive feature selection  
##   
## Outer resampling method: Cross-Validated (10 fold)   
##   
## Resampling performance over subset size:  
##   
## Variables Accuracy Kappa AccuracySD KappaSD Selected  
## 1 0.7109 0.04366 0.006508 0.02098   
## 2 0.7352 0.24393 0.012324 0.03795   
## 4 0.7830 0.43883 0.010690 0.03355   
## 6 0.7987 0.48547 0.010395 0.03278   
## 8 0.8048 0.50390 0.009335 0.02513   
## 10 0.8053 0.50767 0.011775 0.03293   
## 12 0.8058 0.50979 0.011016 0.03207 \*  
## 14 0.8026 0.50041 0.012484 0.03555   
## 15 0.8018 0.49671 0.011058 0.03184   
##   
## The top 5 variables (out of 12):  
## organism.group, exposure.duration.d, lvl1\_f, dose.mg.L.master, dose.um3.mL.master

trellis.par.set(caretTheme())  
#visualize  
plot(my\_rfe,type = c("g", "o"))



my\_size <- pickSizeTolerance(my\_rfe$results, metric = "Accuracy", tol = 5, maximize = TRUE)  
# higher tol (~10) gives you less variables  
# lower tol (~1) gives you more variables - "I'd like the simplest model within 1% of the best model."  
pickVars(my\_rfe$variables, size = my\_size)

## [1] "organism.group" "exposure.duration.d" "lvl1\_f"   
## [4] "dose.mg.L.master"

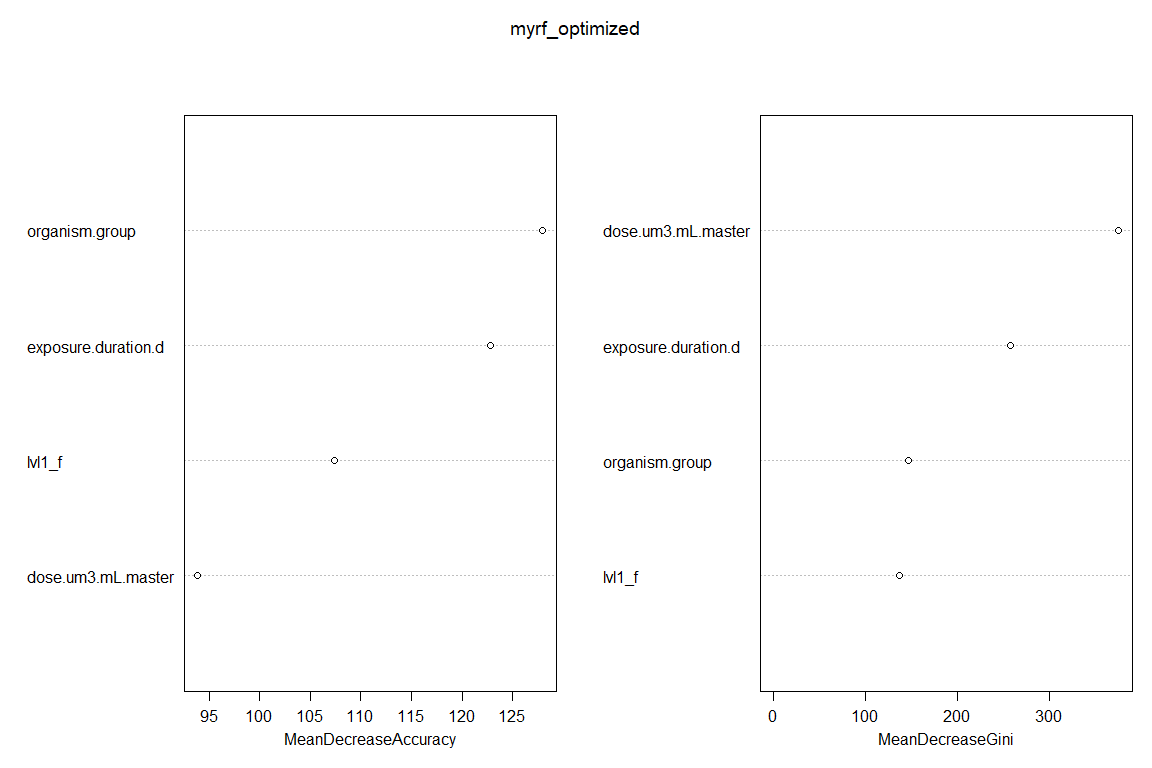
The pickSizeTolerance determines the absolute best value then the percent difference of the other points to this value. This approach can produce good results for many of the tree based models, such as random forest, where there is a plateau of good performance for larger subset sizes. For trees, this is usually because unimportant variables are infrequently used in splits and do not significantly affect performance.

Just use best predictors.

set.seed(2) # assures the data pulled is random, but sets it for the run below (makes outcome stable)  
myrf\_optimized <- randomForest(y = multiVar\_small\_train$effect\_f, # dependent variable  
 x = multiVar\_small\_train %>%  
 dplyr::select(c(organism.group, exposure.duration.d, lvl1\_f, dose.um3.mL.master)), # selecting all predictor variables  
 importance = T, # how useful is a predictor in predicting values (nothing causal)  
 proximity = T,   
 na.action = na.roughfix,  
 ntrees = 100) # 500 trees default.   
  
myrf\_optimized # examine the results.

##   
## Call:  
## randomForest(x = multiVar\_small\_train %>% dplyr::select(c(organism.group, exposure.duration.d, lvl1\_f, dose.um3.mL.master)), y = multiVar\_small\_train$effect\_f, importance = T, proximity = T, na.action = na.roughfix, ntrees = 100)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 23.23%  
## Confusion matrix:  
## N Y class.error  
## N 2849 359 0.1119077  
## Y 695 635 0.5225564

varImpPlot(myrf\_optimized)



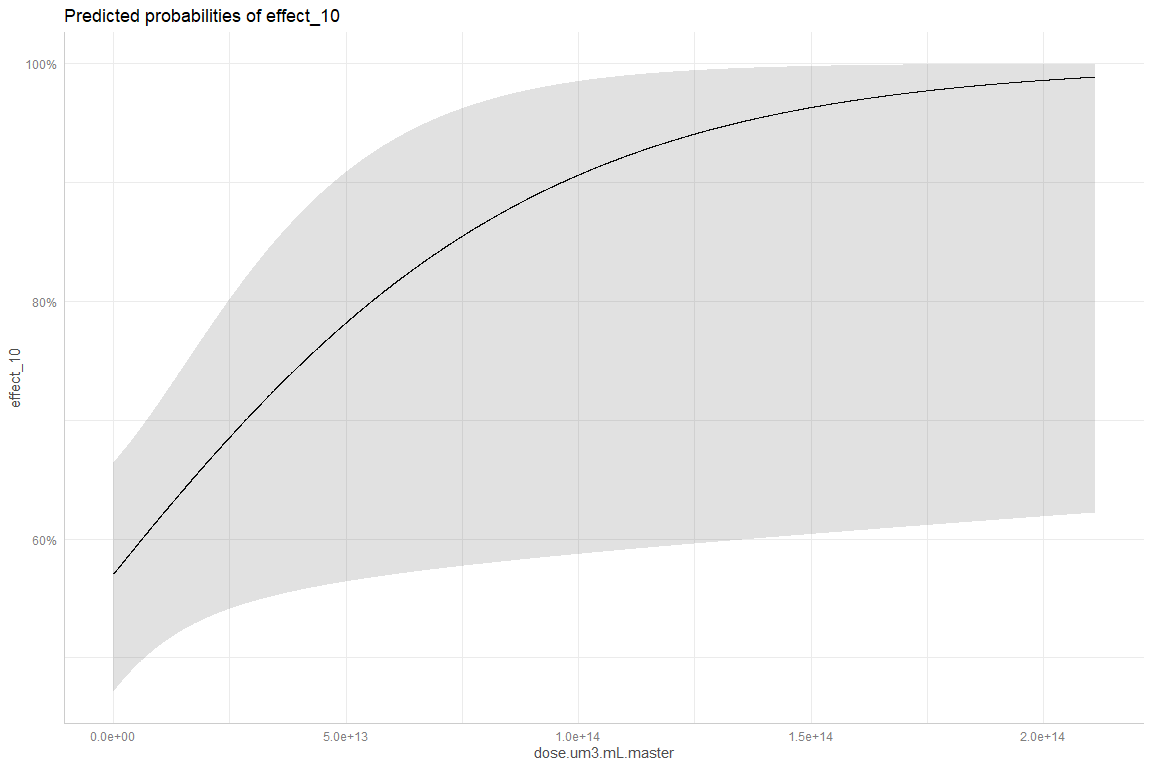
optimized\_fitted <- predict(myrf\_optimized, multiVar\_small\_test %>% dplyr::select(-effect\_f), OOB = TRUE, type ="response")  
misClasificError\_optimized <- mean(optimized\_fitted != multiVar\_small\_test$effect\_f)  
accuracy\_optimized <- paste0('Accuracy: ', round(100\*(1 - misClasificError\_optimized), 2), '%')  
accuracy\_optimized

## [1] "Accuracy: 80.03%"

df1 <- aoc\_z %>%   
 dplyr::select(c(organism.group, exposure.duration.d, lvl1\_f, dose.um3.mL.master, effect\_10)) %>%   
 drop\_na()  
  
m1 <- glm(effect\_10 ~ organism.group + exposure.duration.d + lvl1\_f + dose.um3.mL.master,   
 data = df1, na.action = na.omit, family = "binomial")  
summary(m1)

##   
## Call:  
## glm(formula = effect\_10 ~ organism.group + exposure.duration.d +   
## lvl1\_f + dose.um3.mL.master, family = "binomial", data = df1,   
## na.action = na.omit)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6477 -0.8702 -0.7715 1.2140 2.3209   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) 9.313e-02 2.015e-01 0.462 0.64390  
## organism.groupAnnelida 3.993e-01 5.046e-01 0.791 0.42882  
## organism.groupBacterium 1.005e+00 4.189e-01 2.400 0.01639  
## organism.groupCnidaria -1.914e+00 3.736e-01 -5.123 3.01e-07  
## organism.groupCrustacea -3.697e-01 1.283e-01 -2.882 0.00396  
## organism.groupEchinoderm -9.854e-01 3.324e-01 -2.964 0.00303  
## organism.groupFish -7.221e-01 1.271e-01 -5.683 1.32e-08  
## organism.groupMixed -1.598e+00 7.263e-01 -2.201 0.02775  
## organism.groupMollusca -1.133e+00 1.280e-01 -8.848 < 2e-16  
## organism.groupNematoda 8.204e-01 3.662e-01 2.241 0.02506  
## organism.groupPlant -3.141e-01 3.598e-01 -0.873 0.38281  
## organism.groupRotifera 2.186e-01 2.524e-01 0.866 0.38653  
## exposure.duration.d 1.097e-02 1.789e-03 6.131 8.72e-10  
## lvl1\_fBehavioral, Sensory, Neurological -7.990e-02 2.060e-01 -0.388 0.69809  
## lvl1\_fCirculatory, Respiratory -5.358e-01 2.908e-01 -1.842 0.06545  
## lvl1\_fCommunity NA NA NA NA  
## lvl1\_fFitness -8.049e-01 1.799e-01 -4.473 7.71e-06  
## lvl1\_fImmune 1.672e-01 2.205e-01 0.759 0.44810  
## lvl1\_fMetabolism -3.949e-02 1.780e-01 -0.222 0.82440  
## lvl1\_fMicrobiome -1.226e-02 2.945e-01 -0.042 0.96681  
## lvl1\_fStress 5.007e-02 2.772e-01 0.181 0.85667  
## dose.um3.mL.master 1.986e-14 9.563e-15 2.076 0.03785  
##   
## (Intercept)   
## organism.groupAnnelida   
## organism.groupBacterium \*   
## organism.groupCnidaria \*\*\*  
## organism.groupCrustacea \*\*   
## organism.groupEchinoderm \*\*   
## organism.groupFish \*\*\*  
## organism.groupMixed \*   
## organism.groupMollusca \*\*\*  
## organism.groupNematoda \*   
## organism.groupPlant   
## organism.groupRotifera   
## exposure.duration.d \*\*\*  
## lvl1\_fBehavioral, Sensory, Neurological   
## lvl1\_fCirculatory, Respiratory .   
## lvl1\_fCommunity   
## lvl1\_fFitness \*\*\*  
## lvl1\_fImmune   
## lvl1\_fMetabolism   
## lvl1\_fMicrobiome   
## lvl1\_fStress   
## dose.um3.mL.master \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 5543.1 on 4375 degrees of freedom  
## Residual deviance: 5289.9 on 4355 degrees of freedom  
## AIC: 5331.9  
##   
## Number of Fisher Scoring iterations: 4

require(ggeffects)  
m1\_g <- ggeffects::ggpredict(m1, terms = "dose.um3.mL.master")  
plot(m1\_g)



###LOG DATA

#log subset  
## First split data by DOI, then re-join other data  
set.seed(4)  
doi\_split <- multiVar\_small\_log %>%   
 dplyr::select(doi) %>%   
 unique() %>%   
 initial\_split(prop = 0.65)  
#split just by doi   
doi\_train <- training(doi\_split)  
doi\_test <- testing(doi\_split)  
  
train\_full <- left\_join(doi\_train, multiVar\_small\_log, by = "doi") %>%   
 dplyr::select(-c(doi, ID, effect\_10)) %>%   
 droplevels()  
  
test\_full <- left\_join(doi\_test, multiVar\_small\_log, by = "doi") %>%   
 dplyr::select(-c(doi, ID, effect\_10)) %>%   
 droplevels()  
  
#inspect proportion in test and train  
nrow(test\_full)

## [1] 921

nrow(train\_full)

## [1] 3344

Now that we’ve split the data by study and ensured a good proportion in test and train, let’s run the model.

set.seed(2) # assures the data pulled is random, but sets it for the run below (makes outcome stable)  
myrf\_log <- randomForest(y = train\_full$effect\_f, # dependent variable  
 x = train\_full %>%  
 dplyr::select(-effect\_f), # selecting all predictor variables  
 importance = T, # how useful is a predictor in predicting values (nothing causal)  
 proximity = T,   
 ntrees = 100) # 500 trees default.   
  
myrf\_log # examine the results.

##   
## Call:  
## randomForest(x = train\_full %>% dplyr::select(-effect\_f), y = train\_full$effect\_f, importance = T, proximity = T, ntrees = 100)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 20.66%  
## Confusion matrix:  
## N Y class.error  
## N 1980 254 0.1136974  
## Y 437 673 0.3936937

No performance enhancement with log-transformed values.

## Continuous Predictors

Repeat with continous variables whenever possible, and max of 8 predictors.

skim(multiVar)

Data summary

|  |  |
| --- | --- |
| Name | multiVar |
| Number of rows | 5925 |
| Number of columns | 32 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 16 |
| numeric | 16 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| size\_f | 0 | 1.00 | FALSE | 5 | 1µm: 3241, 100: 1461, 1nm: 640, 100: 410 |
| shape | 0 | 1.00 | FALSE | 4 | sph: 3213, fra: 2388, Not: 164, fib: 160 |
| polymer | 0 | 1.00 | FALSE | 14 | PS: 2858, PE: 1728, PVC: 374, PP: 267 |
| organism.group | 0 | 1.00 | FALSE | 13 | Fis: 2041, Cru: 1393, Mol: 1277, Alg: 425 |
| environment | 0 | 1.00 | FALSE | 3 | Mar: 3198, Fre: 2547, Ter: 180 |
| bio.org | 0 | 1.00 | FALSE | 5 | org: 2527, sub: 2427, cel: 552, tis: 298 |
| effect | 75 | 0.99 | FALSE | 2 | N: 4137, Y: 1713 |
| effect\_f | 75 | 0.99 | FALSE | 2 | N: 4137, Y: 1713 |
| exposure.route | 129 | 0.98 | FALSE | 7 | wat: 4498, foo: 697, sed: 531, med: 40 |
| lvl1\_f | 0 | 1.00 | FALSE | 9 | Fit: 2767, Met: 1766, Beh: 486, Imm: 297 |
| lvl2\_f | 0 | 1.00 | FALSE | 45 | Oxi: 881, Mor: 804, Rep: 581, Gro: 481 |
| sex | 5060 | 0.15 | FALSE | 3 | F: 439, M: 414, M,F: 12 |
| effect.metric | 2568 | 0.57 | FALSE | 8 | HON: 1877, LOE: 1020, NOE: 385, LC5: 42 |
| functional.group | 5547 | 0.06 | FALSE | 3 | COO: 201, NH2: 154, SO3: 23 |
| charge | 5223 | 0.12 | FALSE | 2 | neg: 537, pos: 165 |
| acute.chronic\_f | 0 | 1.00 | FALSE | 3 | Acu: 2893, Chr: 2243, Una: 789 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| size.length.um.used.for.conversions | 0 | 1.00 | 1.708200e+02 | 6.172300e+02 | 0.00 | 3.00 | 10.50 | 126.00 | 5.000000e+03 | ▇▁▁▁▁ |
| particle.volume.um3 | 164 | 0.97 | 2.540824e+08 | 1.818038e+09 | 0.00 | 10.47 | 523.60 | 261848.61 | 1.636246e+10 | ▇▁▁▁▁ |
| density.mg.um.3 | 262 | 0.96 | 0.000000e+00 | 0.000000e+00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.000000e+00 | ▅▇▁▂▁ |
| treatments | 0 | 1.00 | 2.930000e+00 | 1.820000e+00 | 1.00 | 1.00 | 3.00 | 4.00 | 1.000000e+01 | ▇▆▂▁▁ |
| effect\_10 | 75 | 0.99 | 2.900000e-01 | 4.600000e-01 | 0.00 | 0.00 | 0.00 | 1.00 | 1.000000e+00 | ▇▁▁▁▃ |
| exposure.duration.d | 72 | 0.99 | 2.107000e+01 | 3.692000e+01 | 0.00 | 4.00 | 10.00 | 28.00 | 4.500000e+02 | ▇▁▁▁▁ |
| media.temp | 868 | 0.85 | 2.131000e+01 | 4.590000e+00 | 7.50 | 18.00 | 20.00 | 25.00 | 2.900000e+01 | ▁▃▇▃▇ |
| dose.mg.L.master | 1224 | 0.79 | 7.751876e+04 | 2.465602e+06 | 0.00 | 0.03 | 0.50 | 18.36 | 1.484403e+08 | ▇▁▁▁▁ |
| media.ph | 3826 | 0.35 | 7.540000e+00 | 5.800000e-01 | 6.00 | 7.20 | 7.60 | 8.00 | 8.500000e+00 | ▂▂▅▇▅ |
| media.sal.ppt | 4113 | 0.31 | 2.882000e+01 | 8.170000e+00 | 0.06 | 28.00 | 32.00 | 34.00 | 3.840000e+01 | ▁▁▁▃▇ |
| dose.particles.mL.master | 1436 | 0.76 | 1.113595e+18 | 3.952454e+19 | 0.00 | 14.00 | 1000.00 | 350777.93 | 2.280000e+21 | ▇▁▁▁▁ |
| zetapotential.mV | 5244 | 0.11 | -1.837000e+01 | 2.767000e+01 | -87.06 | -30.90 | -24.70 | -5.10 | 1.060000e+02 | ▂▇▅▁▁ |
| max.size.ingest.mm | 4800 | 0.19 | 1.900000e-01 | 1.400000e-01 | 0.04 | 0.11 | 0.11 | 0.40 | 4.000000e-01 | ▅▇▁▁▆ |
| dose.mg.L.master.AF.noec | 3253 | 0.45 | 1.062252e+05 | 2.977044e+06 | 0.00 | 0.08 | 0.70 | 20.00 | 1.484403e+08 | ▇▁▁▁▁ |
| dose.particles.mL.master.AF.noec | 3387 | 0.43 | 4.015210e+17 | 1.285796e+19 | 0.00 | 39.77 | 2000.00 | 285586.44 | 5.720000e+20 | ▇▁▁▁▁ |
| effect.score | 0 | 1.00 | 2.230000e+00 | 1.770000e+00 | 1.00 | 1.00 | 1.00 | 3.00 | 6.000000e+00 | ▇▂▁▁▂ |

acute <- aoc\_z %>%   
 filter(acute.chronic\_f == "Acute") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(bio.org == "organism") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 filter(!exposure.route == "food") %>%   
 filter(!polymer == "Not Reported") %>%   
 filter(!shape == "Not Reported") %>%   
 dplyr::select(c(doi, dose.mg.L.master, organism.group, size.length.um.used.for.conversions, polymer, shape, effect.score, dose.particles.mL.master, effect\_f)) %>%   
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 drop\_na() %>%   
 droplevels()  
  
count\_acute <- paste0('n = ',nrow(acute))  
  
skim(acute)

Data summary

|  |  |
| --- | --- |
| Name | acute |
| Number of rows | 840 |
| Number of columns | 9 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 5 |
| numeric | 4 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| doi | 0 | 1 | FALSE | 49 | 10.: 103, 10.: 96, doi: 64, 10.: 60 |
| organism.group | 0 | 1 | FALSE | 3 | Cru: 449, Mol: 211, Fis: 180 |
| polymer | 0 | 1 | FALSE | 6 | PS: 382, PE: 339, PET: 56, PP: 53 |
| shape | 0 | 1 | FALSE | 3 | sph: 477, fra: 281, fib: 82 |
| effect\_f | 0 | 1 | FALSE | 2 | N: 665, Y: 175 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| dose.mg.L.master | 0 | 1 | 0.37 | 2.07 | -11.64 | -0.74 | 0.28 | 1.40 | 8.17 | ▁▁▇▇▁ |
| size.length.um.used.for.conversions | 0 | 1 | 0.80 | 1.42 | -1.46 | -0.30 | 0.74 | 1.74 | 3.70 | ▅▆▇▅▃ |
| effect.score | 0 | 1 | 0.63 | 0.14 | 0.48 | 0.48 | 0.60 | 0.78 | 0.78 | ▇▁▂▁▇ |
| dose.particles.mL.master | 0 | 1 | 4.52 | 3.75 | -4.20 | 1.70 | 4.04 | 7.25 | 12.65 | ▂▇▇▅▃ |

With a complete dataset for just acute studies in aquatic organisms with aqueous route of exposure, we are left with 453 complete cases with 6 predictor variables, 1 response variable (effect y/n), and an ID. Let’s determine if we have enough data for ML.

exp(1)^6

## [1] 403.4288

is less than n (453), so we may proceed.

## First split data by DOI, then re-join other data  
set.seed(4)  
acute\_doi\_split <- acute %>%   
 dplyr::select(doi) %>%   
 unique() %>%   
 initial\_split(prop = 0.65)  
#split just by doi   
acute\_doi\_train <- training(acute\_doi\_split)  
acute\_doi\_test <- testing(acute\_doi\_split)  
  
acute\_train <- left\_join(acute\_doi\_train, acute, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
acute\_test <- left\_join(acute\_doi\_test, acute, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
#inspect proportion in test and train  
nrow(acute\_test)

## [1] 192

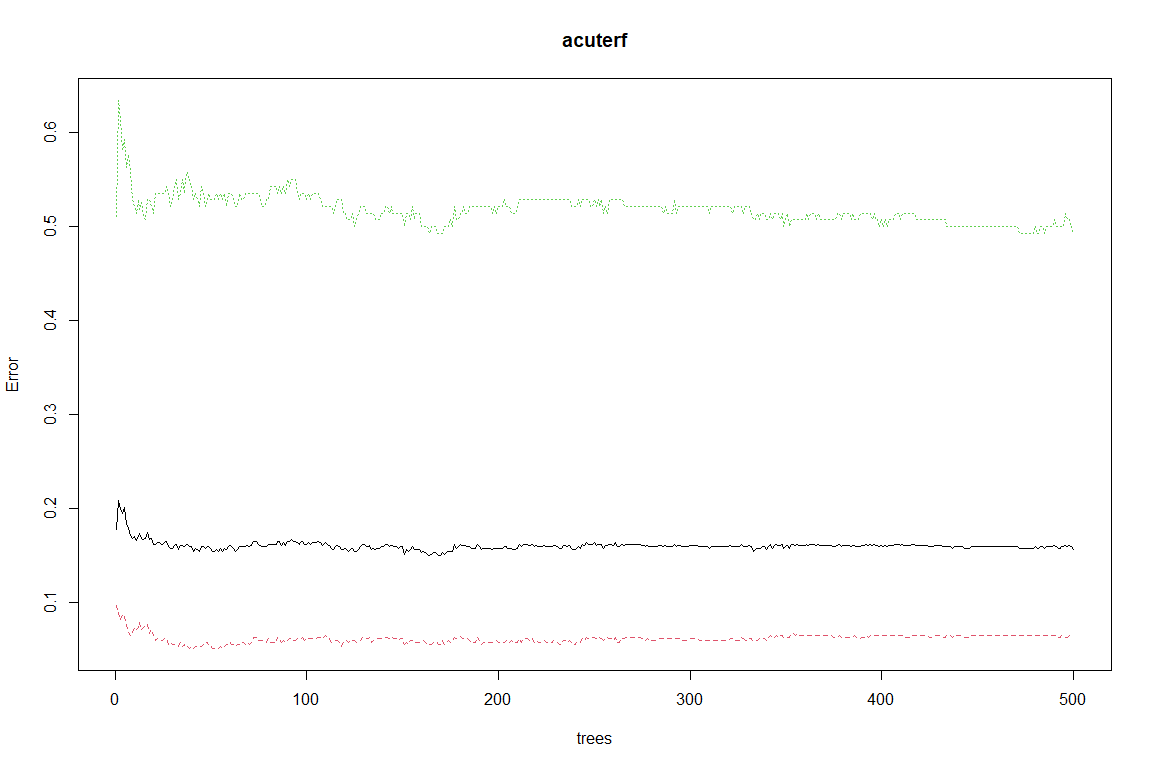
nrow(acute\_train)

## [1] 648

# Create calibration and validation splits with tidymodels initial\_split() function.  
# set.seed(4)  
# acute\_split <- acute %>%  
# initial\_split(prop = 0.75, strata = polymer) # splits data into training and testing set.  
# # default is 3/4ths split (but 75% training, 25% testing).  
# # Stratification (strata) = grouping training/testing sets by region, state, etc.  
# # Using the "strata" call ensures the number of data points in the training data is equivalent to the proportions in the original data set. (Strata below 10% of the total are pooled together.)  
#   
# # Create a training data set with the training() function  
# # Pulls from training and testing sets created by initial\_split()  
# acute\_train <- training(acute\_split)  
# acute\_test <- testing(acute\_split)  
# # Examine the environment to be sure # of observations looks like the 75/25 split. 3199:1066.  
  
# Random forest --   
set.seed(2) # assures the data pulled is random, but sets it for the run below (makes outcome stable)  
acuterf <- randomForest(y = acute\_train$effect\_f, # dependent variable  
 x = acute\_train %>%  
 dplyr::select(-effect\_f), # selecting all predictor variables  
 importance = T, # how useful is a predictor in predicting values (nothing causal)  
 proximity = T,   
 ntrees = 100) # 500 trees default.   
  
acuterf # examine the results.

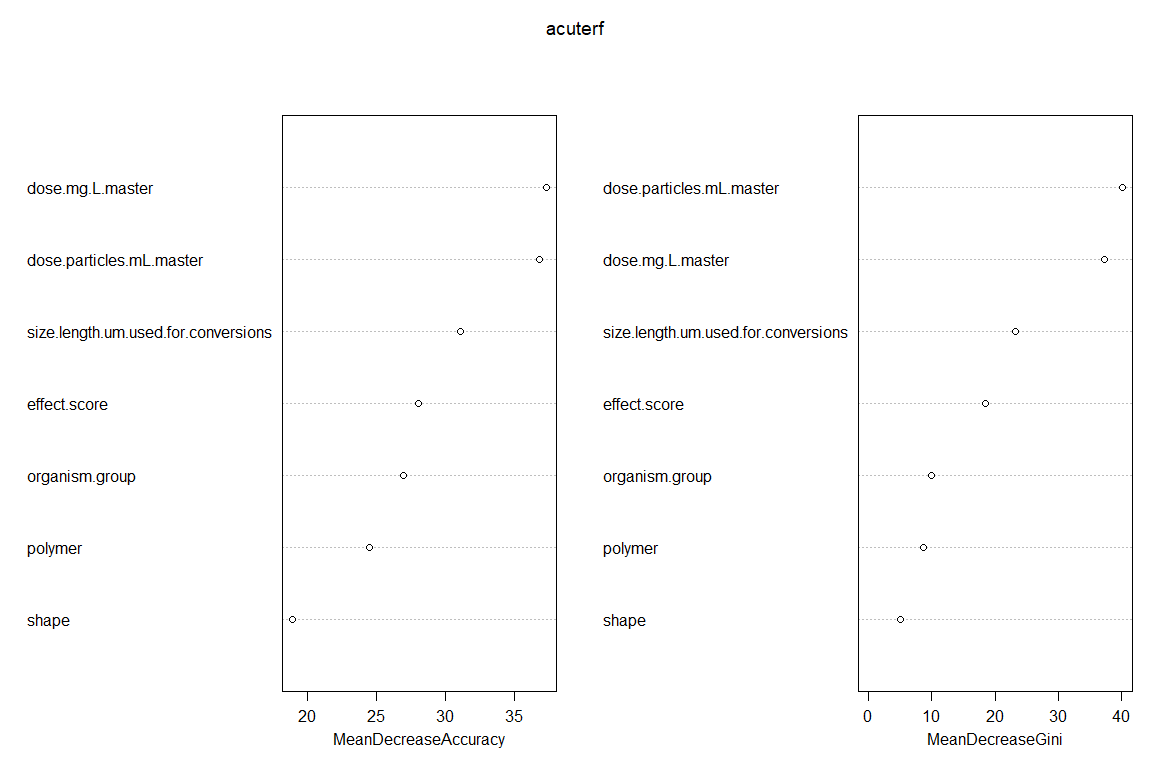
##   
## Call:  
## randomForest(x = acute\_train %>% dplyr::select(-effect\_f), y = acute\_train$effect\_f, importance = T, proximity = T, ntrees = 100)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 15.59%  
## Confusion matrix:  
## N Y class.error  
## N 476 32 0.06299213  
## Y 69 71 0.49285714

plot(acuterf)



# model performance appears to improve most at ~75 trees

varImpPlot(acuterf)



# displays which variables are most important  
# helps to winnow down list of predictors  
# recommended to weigh left pane more  
# right pane also shows how evenly things split based on the list of predictors  
# values close to 0 can be dropped, but don't have to be

Dose, organism group, exposure duration are important.

importance <- acuterf$importance  
#View(importance)  
# displays the data plotted in the plot above

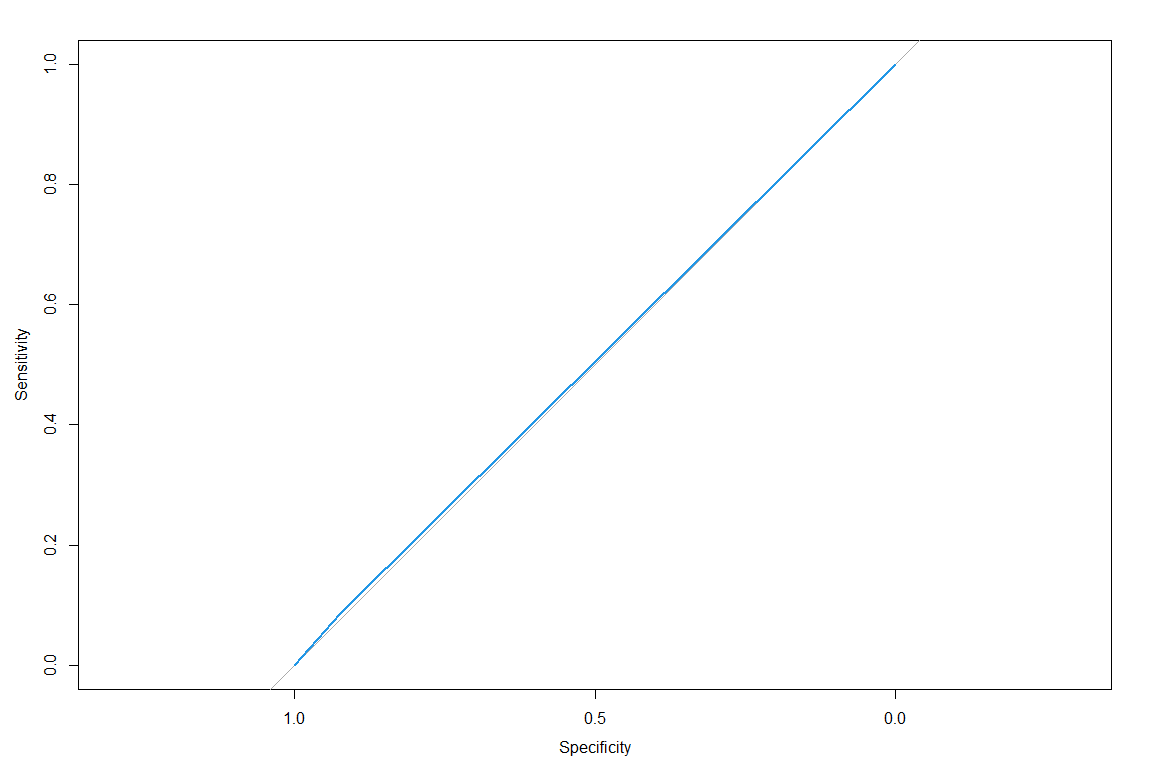
###Validation

# fix levels  
levels(acute\_test$polymer) <- levels(acute\_train$polymer)  
levels(acute\_test$shape) <- levels(acute\_train$shape)  
levels(acute\_test$organism.group) <- levels(acute\_train$organism.group)  
  
fitted <- predict(acuterf,   
 newdata = acute\_test %>% dplyr::select(-effect\_f),   
 OOB = TRUE, type ="response")  
  
misClasificError\_acute <- mean(fitted != acute\_test$effect\_f)  
  
accuracy\_acute <- paste0('Accuracy: ', round(100\*(1 - misClasificError\_acute), 2), '%')  
accuracy\_acute

## [1] "Accuracy: 77.08%"

Alternative ROC Curve

require(pROC)  
predicted <- predict(acuterf, acute\_test %>% dplyr::select(-effect\_f),  
 OOB=TRUE, type= "response")  
#Calculate ROC curve  
rocCurve.tree <- roc(as.numeric(acute\_test$effect\_f),as.numeric(predicted))  
  
##gplot  
# rocks <- roc()  
  
#plot the ROC curve  
plot(rocCurve.tree,col=c(4))

 ##### Chronic

chronic <- aoc\_z %>%   
 filter(acute.chronic\_f == "Chronic") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(bio.org == "organism") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 filter(!exposure.route == "food") %>%   
 dplyr::select(c(doi, dose.mg.L.master, organism.group, size.length.um.used.for.conversions, polymer, shape, effect.score, dose.particles.mL.master, effect\_f)) %>%  
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 drop\_na() %>%   
 droplevels()  
  
count\_chronic <- paste0('n = ',nrow(chronic))  
  
skim(chronic)

Data summary

|  |  |
| --- | --- |
| Name | chronic |
| Number of rows | 428 |
| Number of columns | 9 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 5 |
| numeric | 4 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| doi | 0 | 1 | FALSE | 24 | 10.: 96, doi: 48, 10.: 44, 10.: 36 |
| organism.group | 0 | 1 | FALSE | 3 | Cru: 368, Mol: 35, Fis: 25 |
| polymer | 0 | 1 | FALSE | 10 | PS: 191, PE: 80, Not: 73, PET: 30 |
| shape | 0 | 1 | FALSE | 3 | sph: 271, fra: 156, fib: 1 |
| effect\_f | 0 | 1 | FALSE | 2 | N: 357, Y: 71 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| dose.mg.L.master | 0 | 1 | -0.45 | 1.56 | -3.62 | -1.64 | -0.54 | 0.60 | 3.33 | ▃▆▇▅▂ |
| size.length.um.used.for.conversions | 0 | 1 | 0.61 | 1.01 | -1.15 | 0.37 | 0.74 | 1.00 | 3.18 | ▅▆▇▅▁ |
| effect.score | 0 | 1 | 0.60 | 0.11 | 0.48 | 0.48 | 0.60 | 0.60 | 0.78 | ▆▁▇▁▃ |
| dose.particles.mL.master | 0 | 1 | 4.34 | 2.98 | -4.20 | 2.58 | 4.00 | 5.14 | 11.89 | ▁▂▇▁▂ |

With a complete dataset for just acute studies in aquatic organisms with aqueous route of exposure, we are left with 453 complete cases with 6 predictor variables, 1 response variable (effect y/n), and an ID. Let’s determine if we have enough data for ML.

exp(1)^6

## [1] 403.4288

is less than n (453), so we may proceed.

## First split data by DOI, then re-join other data  
set.seed(4)  
chronic\_doi\_split <- chronic %>%   
 dplyr::select(doi) %>%   
 unique() %>%   
 initial\_split(prop = 0.9)  
#split just by doi   
chronic\_doi\_train <- training(chronic\_doi\_split)  
chronic\_doi\_test <- testing(chronic\_doi\_split)  
  
chronic\_train <- left\_join(chronic\_doi\_train, chronic, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
chronic\_test <- left\_join(chronic\_doi\_test, chronic, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
#inspect proportion in test and train  
nrow(chronic\_test)

## [1] 80

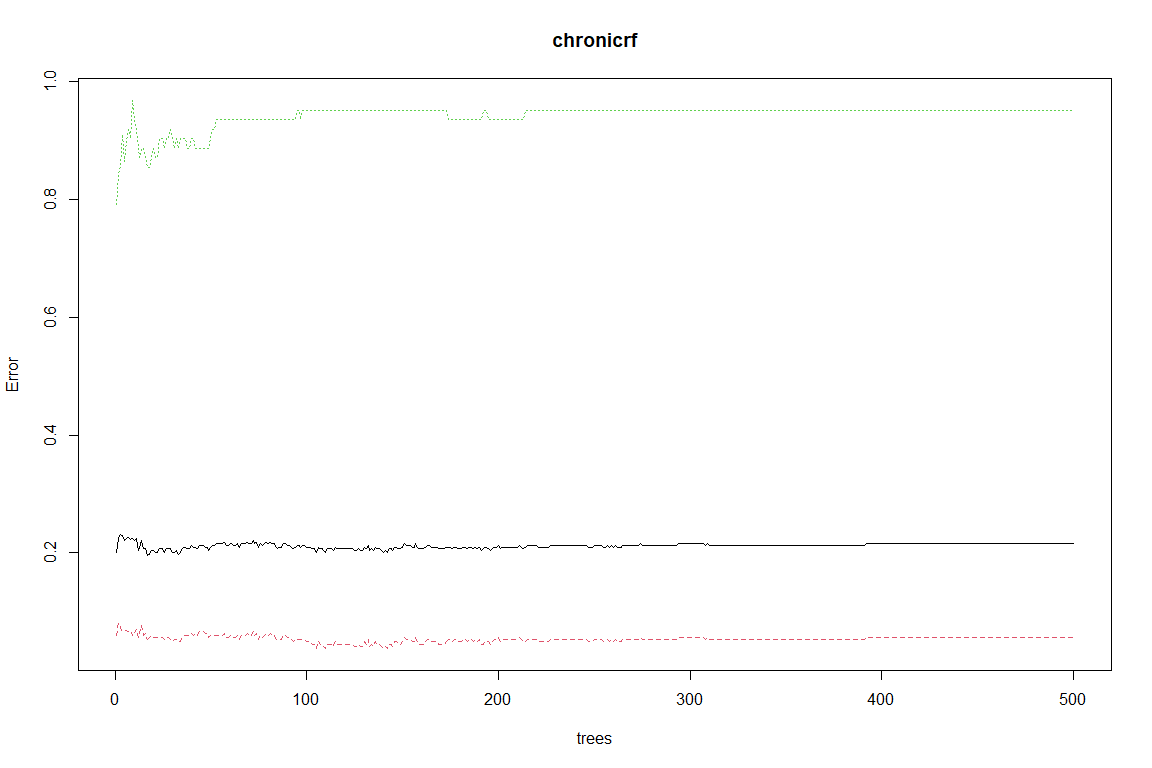
nrow(chronic\_train)

## [1] 348

# # Create calibration and validation splits with tidymodels initial\_split() function.  
# set.seed(4)  
# chronic\_split <- chronic %>%  
# initial\_split(prop = 0.75, strata = polymer) # splits data into training and testing set.  
# # default is 3/4ths split (but 75% training, 25% testing).  
# # Stratification (strata) = grouping training/testing sets by region, state, etc.  
# # Using the "strata" call ensures the number of data points in the training data is equivalent to the proportions in the original data set. (Strata below 10% of the total are pooled together.)  
#   
# # Create a training data set with the training() function  
# # Pulls from training and testing sets created by initial\_split()  
# chronic\_train <- training(chronic\_split)  
# chronic\_test <- testing(chronic\_split)  
# # Examine the environment to be sure # of observations looks like the 75/25 split. 3199:1066.  
  
# Random forest --   
set.seed(2) # assures the data pulled is random, but sets it for the run below (makes outcome stable)  
chronicrf <- randomForest(y = chronic\_train$effect\_f, # dependent variable  
 x = chronic\_train %>%  
 dplyr::select(-effect\_f), # selecting all predictor variables  
 importance = T, # how useful is a predictor in predicting values (nothing causal)  
 proximity = T,   
 ntrees = 100) # 500 trees default.   
  
chronicrf # examine the results.

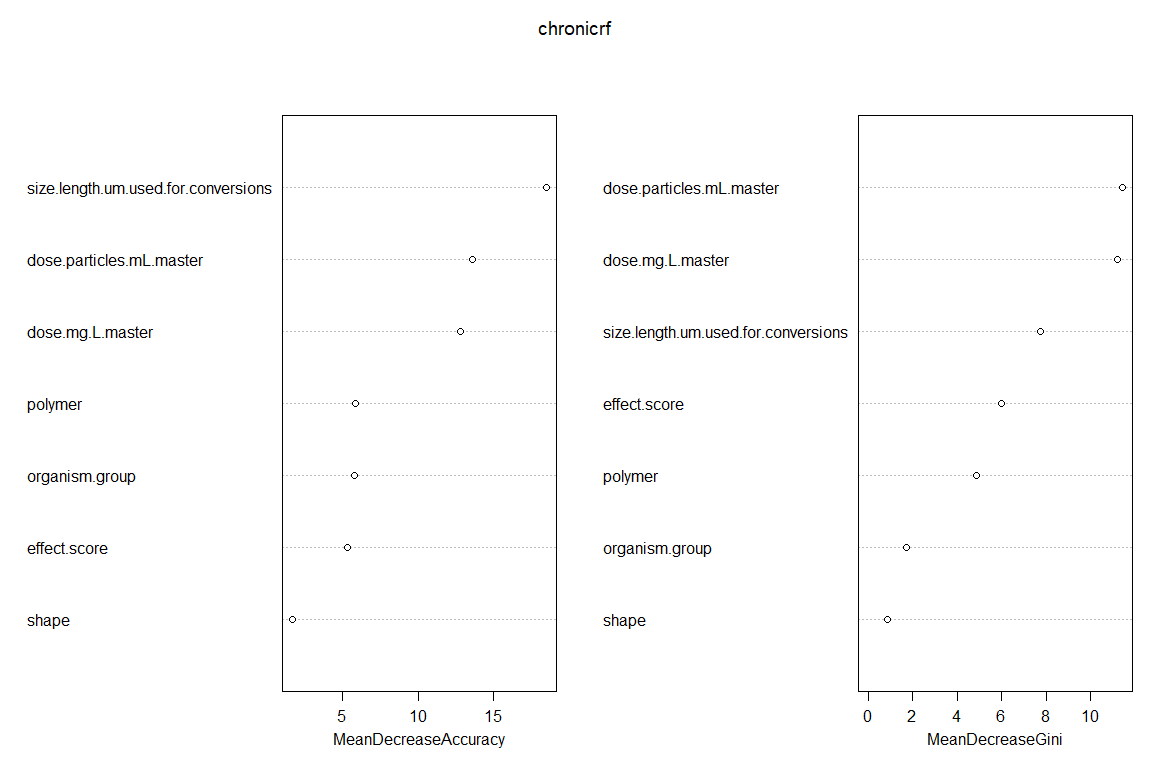
##   
## Call:  
## randomForest(x = chronic\_train %>% dplyr::select(-effect\_f), y = chronic\_train$effect\_f, importance = T, proximity = T, ntrees = 100)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 21.55%  
## Confusion matrix:  
## N Y class.error  
## N 270 16 0.05594406  
## Y 59 3 0.95161290

plot(chronicrf)



# model performance appears to improve most at ~75 trees

varImpPlot(chronicrf)



# displays which variables are most important  
# helps to winnow down list of predictors  
# recommended to weigh left pane more  
# right pane also shows how evenly things split based on the list of predictors  
# values close to 0 can be dropped, but don't have to be

Dose, organism group, exposure duration are important.

importance <- chronicrf$importance  
#View(importance)  
# displays the data plotted in the plot above

# fix levels  
levels(chronic\_test$polymer) <- levels(chronic\_train$polymer)  
levels(chronic\_test$shape) <- levels(chronic\_train$shape)  
levels(chronic\_test$organism.group) <- levels(chronic\_train$organism.group)  
  
fitted <- predict(chronicrf, chronic\_test %>%   
 dplyr::select(-effect\_f),   
 OOB = TRUE, type ="response")  
  
misClasificError\_chronic <- mean(fitted != chronic\_test$effect\_f)  
  
accuracy\_chronic <- paste0('Accuracy: ', round(100\*(1 - misClasificError\_chronic), 2), '%')  
accuracy\_chronic

## [1] "Accuracy: 81.25%"

#### All Data

all <- aoc\_z %>%   
 #filter(acute.chronic\_f == "Chronic") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(bio.org == "organism") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 filter(!exposure.route == "food") %>%   
 dplyr::select(c(size.length.um.used.for.conversions, shape, polymer, particle.volume.um3, organism.group, dose.mg.L.master, effect.score, lvl2\_f, effect\_f, doi)) %>%  
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 drop\_na()  
  
count\_all <- paste0('n = ',nrow(all))  
  
skim(all)

Data summary

|  |  |
| --- | --- |
| Name | all |
| Number of rows | 1510 |
| Number of columns | 10 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 6 |
| numeric | 4 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| shape | 0 | 1 | FALSE | 3 | sph: 848, fra: 579, fib: 83, Not: 0 |
| polymer | 0 | 1 | FALSE | 11 | PS: 623, PE: 524, Not: 116, PET: 86 |
| organism.group | 0 | 1 | FALSE | 8 | Cru: 878, Mol: 246, Fis: 211, Ech: 63 |
| lvl2\_f | 0 | 1 | FALSE | 5 | Mor: 574, Rep: 338, Dev: 295, Gro: 222 |
| effect\_f | 0 | 1 | FALSE | 2 | N: 1199, Y: 311 |
| doi | 0 | 1 | FALSE | 83 | 10.: 144, 10.: 136, 10.: 103, doi: 72 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| size.length.um.used.for.conversions | 0 | 1 | 0.79 | 1.25 | -1.46 | 0.30 | 0.74 | 1.77 | 3.70 | ▅▆▇▅▂ |
| particle.volume.um3 | 0 | 1 | 1.76 | 3.54 | -4.67 | 0.17 | 1.15 | 4.24 | 10.21 | ▃▇▆▃▂ |
| dose.mg.L.master | 0 | 1 | 0.14 | 1.94 | -11.64 | -0.91 | 0.10 | 1.40 | 8.17 | ▁▁▇▇▁ |
| effect.score | 0 | 1 | 0.62 | 0.13 | 0.48 | 0.48 | 0.60 | 0.78 | 0.78 | ▇▁▅▁▇ |

With a complete dataset for just organismal fitness studies in aquatic organisms with aqueous route of exposure, we are left with 1510 complete cases with 8 predictor variables, and 1 response variable (effect y/n). Let’s determine if we have enough data for ML.

exp(1)^7

## [1] 1096.633

is less than n (453), so we may proceed.

## First split data by DOI, then re-join other data  
set.seed(4)  
all\_doi\_split <- all %>%   
 dplyr::select(doi) %>%   
 unique() %>%   
 initial\_split(prop = 0.7)  
#split just by doi   
all\_doi\_train <- training(all\_doi\_split)  
all\_doi\_test <- testing(all\_doi\_split)  
  
all\_train <- left\_join(all\_doi\_train, all, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
all\_test <- left\_join(all\_doi\_test, all, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
#inspect proportion in test and train  
nrow(all\_test)

## [1] 296

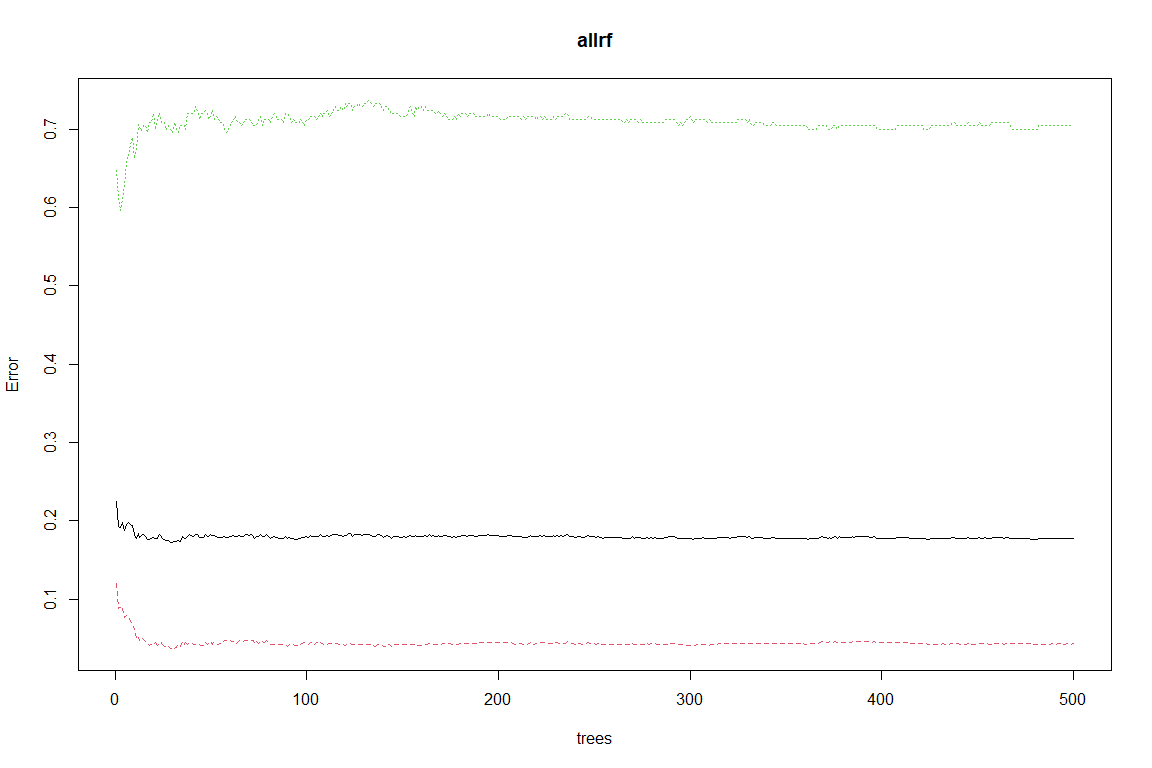
nrow(all\_train)

## [1] 1214

# Create calibration and validation splits with tidymodels initial\_split() function.  
set.seed(4)  
# all\_split <- all %>%  
# initial\_split(prop = 0.75, strata = polymer) # splits data into training and testing set.  
# # default is 3/4ths split (but 75% training, 25% testing).  
# # Stratification (strata) = grouping training/testing sets by region, state, etc.  
# # Using the "strata" call ensures the number of data points in the training data is equivalent to the proportions in the original data set. (Strata below 10% of the total are pooled together.)  
#   
# # Create a training data set with the training() function  
# # Pulls from training and testing sets created by initial\_split()  
# all\_train <- training(all\_split)  
# all\_test <- testing(all\_split)  
# # Examine the environment to be sure # of observations looks like the 75/25 split. 3199:1066.  
  
# Random forest --   
set.seed(2) # assures the data pulled is random, but sets it for the run below (makes outcome stable)  
allrf <- randomForest(y = all\_train$effect\_f, # dependent variable  
 x = all\_train %>%  
 dplyr::select(-effect\_f), # selecting all predictor variables  
 importance = T, # how useful is a predictor in predicting values (nothing causal)  
 proximity = T,   
 ntrees = 100) # 500 trees default.   
  
allrf # examine the results.

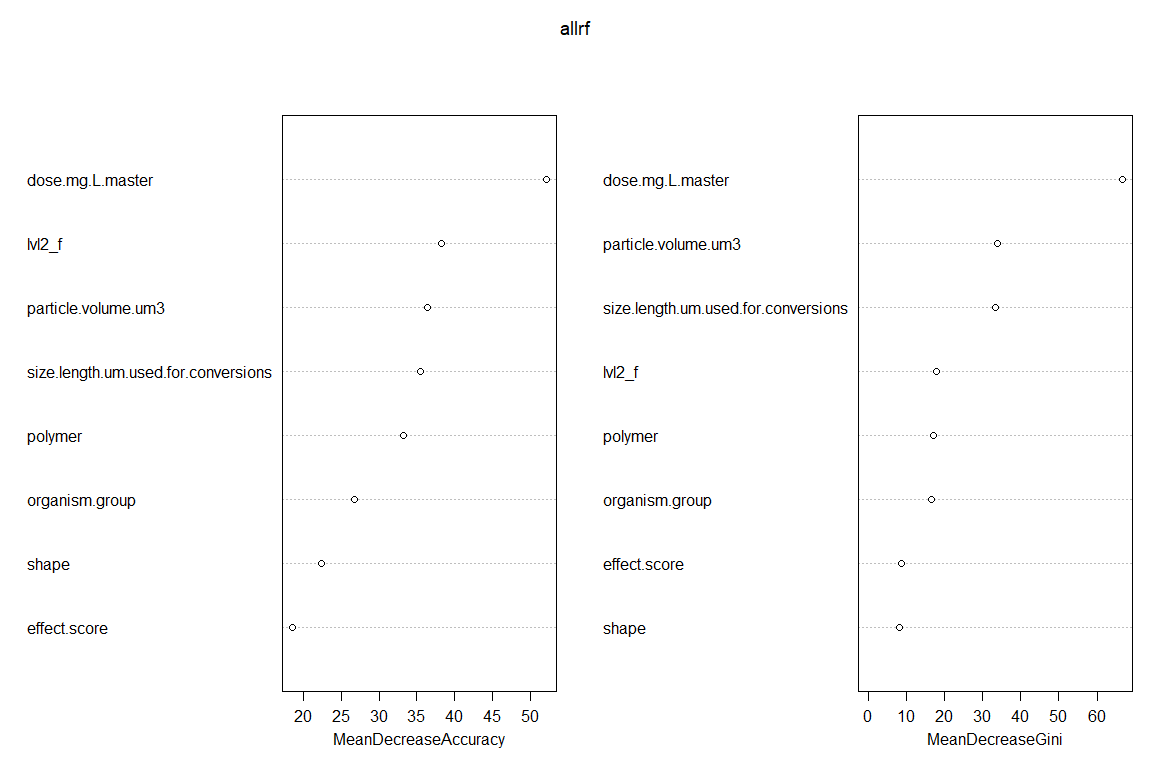
##   
## Call:  
## randomForest(x = all\_train %>% dplyr::select(-effect\_f), y = all\_train$effect\_f, importance = T, proximity = T, ntrees = 100)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 17.79%  
## Confusion matrix:  
## N Y class.error  
## N 925 42 0.0434333  
## Y 174 73 0.7044534

plot(allrf)



# model performance appears to improve most at ~75 trees

varImpPlot(allrf)



# displays which variables are most important  
# helps to winnow down list of predictors  
# recommended to weigh left pane more  
# right pane also shows how evenly things split based on the list of predictors  
# values close to 0 can be dropped, but don't have to be

Dose, organism group, exposure duration are important.

importance <- allrf$importance  
#View(importance)  
# displays the data plotted in the plot above

levels(all\_test$polymer) <- levels(all\_train$polymer)  
levels(all\_test$shape) <- levels(all\_train$shape)  
levels(all\_test$organism.group) <- levels(all\_train$organism.group)  
  
fitted <- predict(allrf, all\_test %>%   
 dplyr::select(-effect\_f),   
 OOB = TRUE, type ="response")  
  
misClasificError\_all <- mean(fitted != all\_test$effect\_f)  
accuracy\_all <- paste0('Accuracy: ', round(100\*(1 - misClasificError\_all), 2), '%')  
accuracy\_all

## [1] "Accuracy: 75.34%"

#### NO FILTERS

nofilter <- aoc\_z %>%   
 #filter(acute.chronic\_f == "Chronic") %>%   
 ##filter(!environment == "Terrestrial") %>%   
 #filter(bio.org == "organism") %>%   
 #filter(lvl1\_f == "Fitness") %>%   
 #filter(!exposure.route == "food") %>%   
 filter(!shape == "Not Reported") %>%   
 filter(!polymer == "Not Reported") %>%   
 dplyr::select(c(doi, dose.mg.L.master, organism.group, size.length.um.used.for.conversions, polymer, shape,  
 #effect.score,   
 dose.particles.mL.master,   
 effect\_f,   
 particle.volume.um3,   
 exposure.duration.d,  
 lvl1\_f)) %>%  
 droplevels() %>%   
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 drop\_na()  
  
count\_nofilter <- paste0('n = ',nrow(nofilter))  
  
skim(nofilter)

Data summary

|  |  |
| --- | --- |
| Name | nofilter |
| Number of rows | 4266 |
| Number of columns | 11 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 6 |
| numeric | 5 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| doi | 0 | 1 | FALSE | 122 | 10.: 372, 10.: 144, 10.: 139, 10.: 126 |
| organism.group | 0 | 1 | FALSE | 12 | Fis: 1461, Mol: 1050, Cru: 1006, Alg: 393 |
| polymer | 0 | 1 | FALSE | 10 | PS: 2368, PE: 1288, PP: 241, PVC: 170 |
| shape | 0 | 1 | FALSE | 3 | sph: 2710, fra: 1445, fib: 111 |
| effect\_f | 0 | 1 | FALSE | 2 | N: 2858, Y: 1408 |
| lvl1\_f | 0 | 1 | FALSE | 9 | Fit: 1962, Met: 1379, Beh: 293, Imm: 203 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| dose.mg.L.master | 0 | 1 | -0.21 | 2.07 | -11.64 | -1.47 | -0.30 | 1.26 | 8.17 | ▁▁▇▅▁ |
| size.length.um.used.for.conversions | 0 | 1 | 0.90 | 1.27 | -4.00 | 0.24 | 1.00 | 1.85 | 3.70 | ▁▂▅▇▃ |
| dose.particles.mL.master | 0 | 1 | 3.74 | 3.70 | -4.20 | 1.15 | 3.00 | 5.76 | 21.36 | ▃▇▃▁▁ |
| particle.volume.um3 | 0 | 1 | 2.14 | 3.59 | -12.28 | 0.17 | 2.67 | 4.82 | 10.21 | ▁▂▅▇▂ |
| exposure.duration.d | 0 | 1 | 0.81 | 0.76 | -2.70 | 0.48 | 0.90 | 1.32 | 2.23 | ▁▁▁▇▅ |

With a complete dataset for just acute studies in aquatic organisms with aqueous route of exposure, we are left with 453 complete cases with 6 predictor variables, 1 response variable (effect y/n), and an ID. Let’s determine if we have enough data for ML.

exp(1)^8

## [1] 2980.958

is less than n (453), so we may proceed.

## First split data by DOI, then re-join other data  
set.seed(4)  
nofilter\_doi\_split <- nofilter %>%   
 dplyr::select(doi) %>%   
 unique() %>%   
 initial\_split(prop = 0.84)  
#split just by doi   
nofilter\_doi\_train <- training(nofilter\_doi\_split)  
nofilter\_doi\_test <- testing(nofilter\_doi\_split)  
  
nofilter\_train <- left\_join(nofilter\_doi\_train, nofilter, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
nofilter\_test <- left\_join(nofilter\_doi\_test, nofilter, by = "doi") %>%   
 dplyr::select(-doi) %>%   
 droplevels()  
  
#inspect proportion in test and train  
nrow(nofilter\_test)

## [1] 838

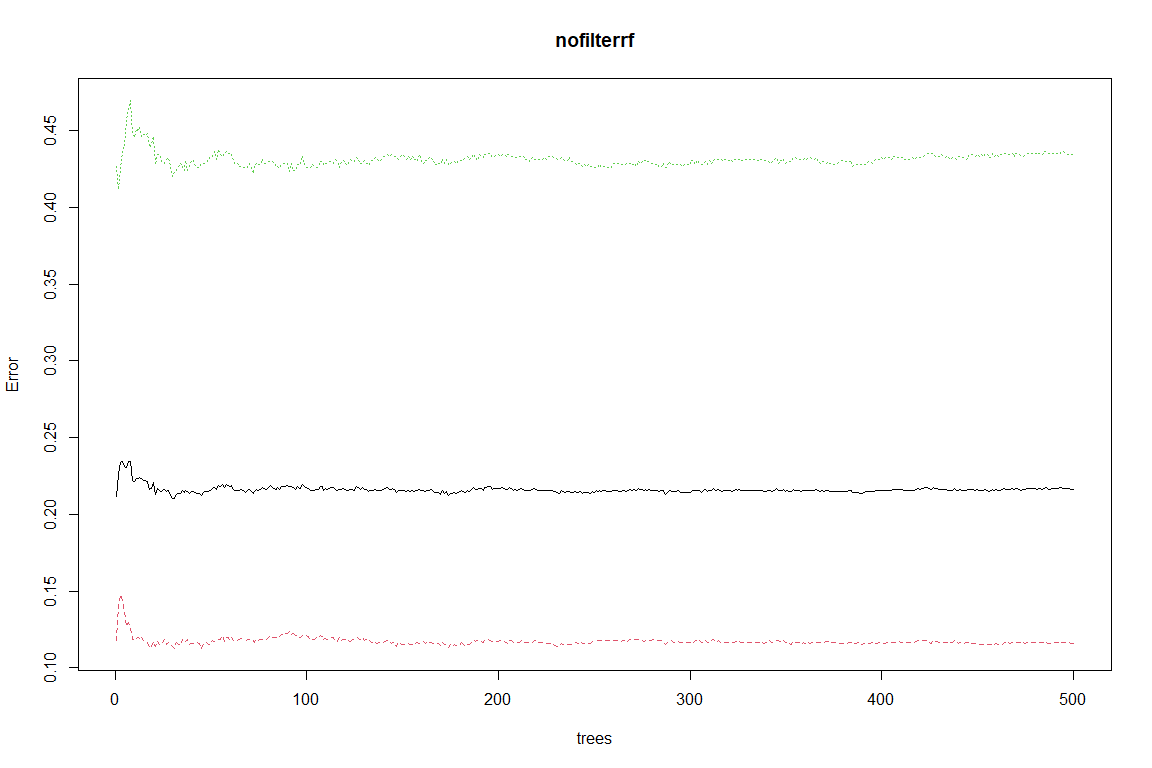
nrow(nofilter\_train)

## [1] 3428

# Create calibration and validation splits with tidymodels initial\_split() function.  
# set.seed(4)  
# nofilter\_split <- nofilter %>%  
# initial\_split(prop = 0.75, strata = polymer) # splits data into training and testing set.  
# # default is 3/4ths split (but 75% training, 25% testing).  
# # Stratification (strata) = grouping training/testing sets by region, state, etc.  
# # Using the "strata" cnofilter ensures the number of data points in the training data is equivalent to the proportions in the original data set. (Strata below 10% of the total are pooled together.)  
#   
# # Create a training data set with the training() function  
# # Pulls from training and testing sets created by initial\_split()  
# nofilter\_train <- training(nofilter\_split)  
# nofilter\_test <- testing(nofilter\_split)  
# # Examine the environment to be sure # of observations looks like the 75/25 split. 3199:1066.  
  
# Random forest --   
set.seed(2) # assures the data pulled is random, but sets it for the run below (makes outcome stable)  
nofilterrf <- randomForest(y = nofilter\_train$effect\_f, # dependent variable  
 x = nofilter\_train %>%  
 dplyr::select(-effect\_f), # selecting nofilter predictor variables  
 importance = T, # how useful is a predictor in predicting values (nothing causal)  
 proximity = T,   
 ntrees = 100) # 500 trees default.   
  
nofilterrf # examine the results.

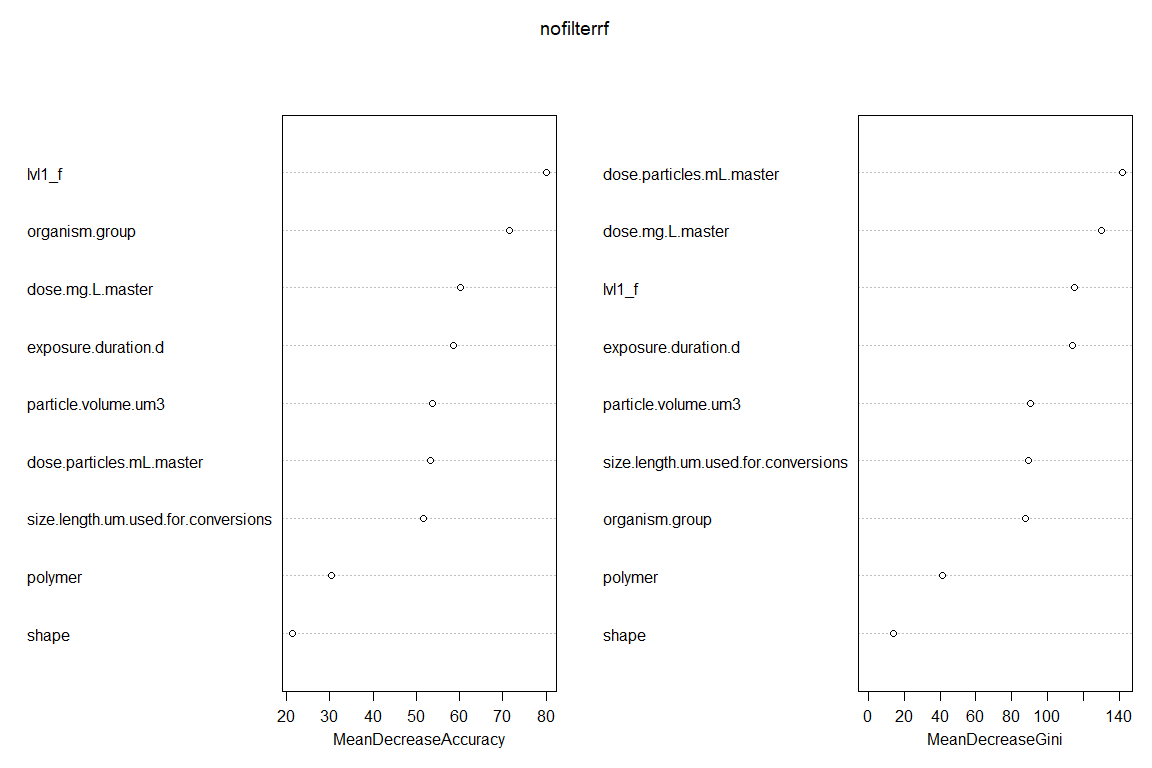
##   
## Call:  
## randomForest(x = nofilter\_train %>% dplyr::select(-effect\_f), y = nofilter\_train$effect\_f, importance = T, proximity = T, ntrees = 100)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 21.62%  
## Confusion matrix:  
## N Y class.error  
## N 2076 272 0.1158433  
## Y 469 611 0.4342593

plot(nofilterrf)



# model performance appears to improve most at ~75 trees

varImpPlot(nofilterrf)



# displays which variables are most important  
# helps to winnow down list of predictors  
# recommended to weigh left pane more  
# right pane also shows how evenly things split based on the list of predictors  
# values close to 0 can be dropped, but don't have to be

Dose, organism group, exposure duration are important.

importance <- nofilterrf$importance  
View(importance)  
# displays the data plotted in the plot above

#fix levels  
levels(nofilter\_test$polymer) <- levels(nofilter\_train$polymer)  
levels(nofilter\_test$shape) <- levels(nofilter\_train$shape)  
levels(nofilter\_test$lvl1\_f) <- levels(nofilter\_train$lvl1\_f)  
levels(nofilter\_test$organism.group) <- levels(nofilter\_train$organism.group)  
  
x1 <- nofilter\_test %>% dplyr::select(-effect\_f)  
fitted <- predict(nofilterrf,   
 x1,  
 OOB = TRUE, type ="response")  
  
misClasificError\_nofilter <- mean(fitted != nofilter\_test$effect\_f)  
accuracy\_nofilter <- paste0('Accuracy: ', round(100\*(1 - misClasificError\_nofilter), 2), '%')  
accuracy\_nofilter

## [1] "Accuracy: 50.48%"

#### ROC Curve

yet anothyer try

###CHRONIC  
chronicpredictions <- as.data.frame(predict(chronicrf, chronic\_test %>% dplyr::select(-effect\_f), type = "prob"))  
# predict class and then attach test class  
chronicpredictions$predict <- names(chronicpredictions)[1:2][apply(chronicpredictions[,1:2], 1, which.max)]  
chronicpredictions$observed <- chronic\_test$effect\_f  
  
####ACUTE  
predictions <- as.data.frame(predict(acuterf, acute\_test%>% dplyr::select(-effect\_f), type = "prob"))  
# predict class and then attach test class  
predictions$predict <- names(predictions)[1:2][apply(predictions[,1:2], 1, which.max)]  
predictions$observed <- acute\_test$effect\_f  
head(predictions)

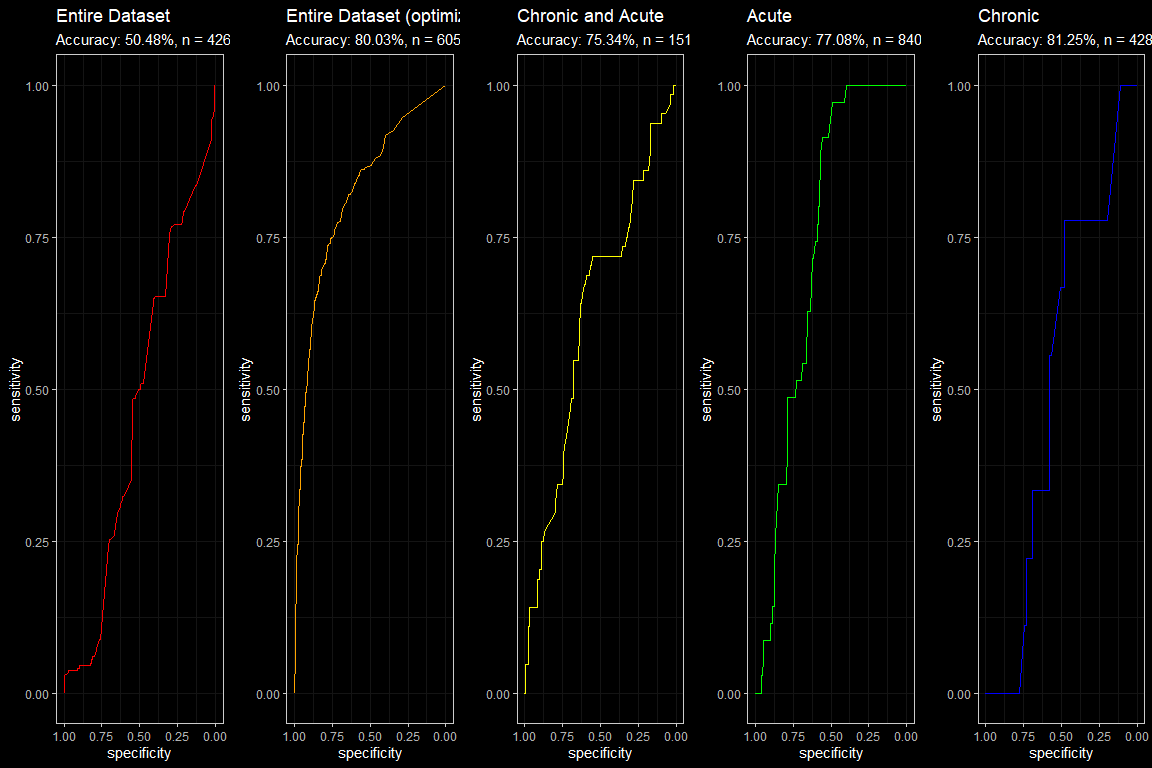
## N Y predict observed  
## 1 0.880 0.120 N N  
## 2 0.808 0.192 N N  
## 3 0.812 0.188 N N  
## 4 0.702 0.298 N Y  
## 5 0.622 0.378 N Y  
## 6 0.564 0.436 N N

#### all  
allpredictions <- as.data.frame(predict(allrf, all\_test%>% dplyr::select(-effect\_f), type = "prob"))  
# predict class and then attach test class  
allpredictions$predict <- names(allpredictions)[1:2][apply(allpredictions[,1:2], 1, which.max)]  
allpredictions$observed <- all\_test$effect\_f  
head(allpredictions)

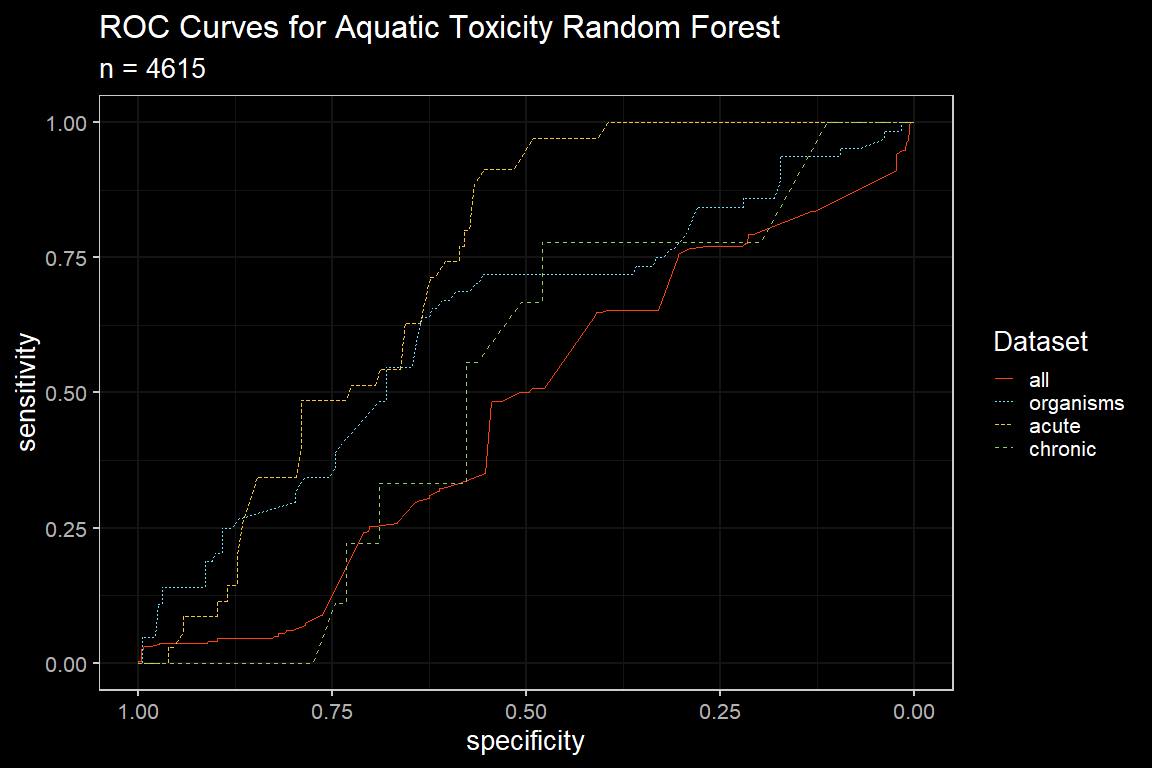
## N Y predict observed  
## 1 0.814 0.186 N Y  
## 2 0.812 0.188 N Y  
## 3 0.884 0.116 N N  
## 4 0.876 0.124 N N  
## 5 0.814 0.186 N Y  
## 6 0.812 0.188 N N

###nofilter  
nofilterpredictions <- as.data.frame(predict(nofilterrf, nofilter\_test%>% dplyr::select(-effect\_f), type = "prob"))  
# predict class and then attach test class  
nofilterpredictions$predict <- names(nofilterpredictions)[1:2][apply(nofilterpredictions[,1:2], 1, which.max)]  
nofilterpredictions$observed <- nofilter\_test$effect\_f  
  
###nofilterOptimized  
nofilter.optimiziedpredictions <- as.data.frame(predict(myrf\_optimized, multiVar\_small\_test %>% dplyr::select(-effect\_f), type = "prob"))  
# predict class and then attach test class  
nofilter.optimiziedpredictions$predict <- names(nofilter.optimiziedpredictions)[1:2][apply(nofilter.optimiziedpredictions[,1:2], 1, which.max)]  
nofilter.optimiziedpredictions$observed <- multiVar\_small\_test$effect\_f

require(ggdark)  
# 1 ROC curve, yes vs no for acute  
roc.acute <- roc(ifelse(predictions$observed=="Y", "Y", "N"), as.numeric(predictions$Y))  
  
#chronic  
roc.chronic <- roc(ifelse(chronicpredictions$observed=="Y", "Y", "N"), as.numeric(chronicpredictions$Y))  
#all  
roc.all <- roc(ifelse(allpredictions$observed=="Y", "Y", "N"), as.numeric(allpredictions$Y))  
  
#nofilter  
roc.nofilter <- roc(ifelse(nofilterpredictions$observed=="Y", "Y", "N"), as.numeric(nofilterpredictions$Y))  
  
#no filter (optimized)  
roc.nofilter.optimized <- roc(ifelse(nofilter.optimiziedpredictions$observed=="Y", "Y", "N"), as.numeric(nofilter.optimiziedpredictions$Y))  
  
##make ROC curves  
  
#all  
allROC <- ggroc(roc.all, col = "yellow") +   
 labs(title = "Chronic and Acute",  
 subtitle = paste0(accuracy\_all,', ',count\_all)) +   
 dark\_theme\_bw()  
  
#acute  
acuteROC <- ggroc(roc.acute, col = "green") +   
 labs(title = "Acute",  
 subtitle = paste0(accuracy\_acute,', ',count\_acute)) +   
 dark\_theme\_bw()  
  
#chronic  
chronicROC <- ggroc(roc.chronic, col = "blue") +   
 labs(title = "Chronic",  
 subtitle = paste0(accuracy\_chronic,', ',count\_chronic)) + #auto label  
 dark\_theme\_bw()  
  
#no filter  
nofilterROC <- ggroc(roc.nofilter, col = "red") +   
 labs(title = "Entire Dataset",  
 subtitle = paste0(accuracy\_nofilter,', ',count\_nofilter)) +   
 dark\_theme\_bw()  
  
#optimized  
nofilteroptimizedROC <- ggroc(roc.nofilter.optimized, col = "orange") +   
 labs(title = "Entire Dataset (optimized)",  
 subtitle = paste0(accuracy\_optimized,', ',count\_optimized)) +   
 dark\_theme\_bw()  
  
#arrange together and print  
require(gridExtra)  
grid.arrange(nofilterROC, nofilteroptimizedROC, allROC, acuteROC, chronicROC,  
 ncol = 5)



require(pROC)  
require(tidyverse)  
require(ggdark)  
require(ggsci)  
ggroc(list(all = roc.nofilter, organisms = roc.all, acute = roc.acute, chronic = roc.chronic), aes = c("linetype", "color")) +  
 labs(title = "ROC Curves for Aquatic Toxicity Random Forest",  
 subtitle = "n = 4615",  
 color = "Dataset",  
 linetype = "Dataset") +  
 scale\_color\_tron() +  
 # theme\_bw(base\_size = 20)  
 dark\_theme\_bw(base\_size = 20)# +



theme(plot.title.position = element\_text(hjust = 0.5),  
 plot.subtitle.position = element\_text(hjust = 0.5))

## List of 2  
## $ plot.title.position :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : num 0.5  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi FALSE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ plot.subtitle.position:List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : num 0.5  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi FALSE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## - attr(\*, "class")= chr [1:2] "theme" "gg"  
## - attr(\*, "complete")= logi FALSE  
## - attr(\*, "validate")= logi TRUE

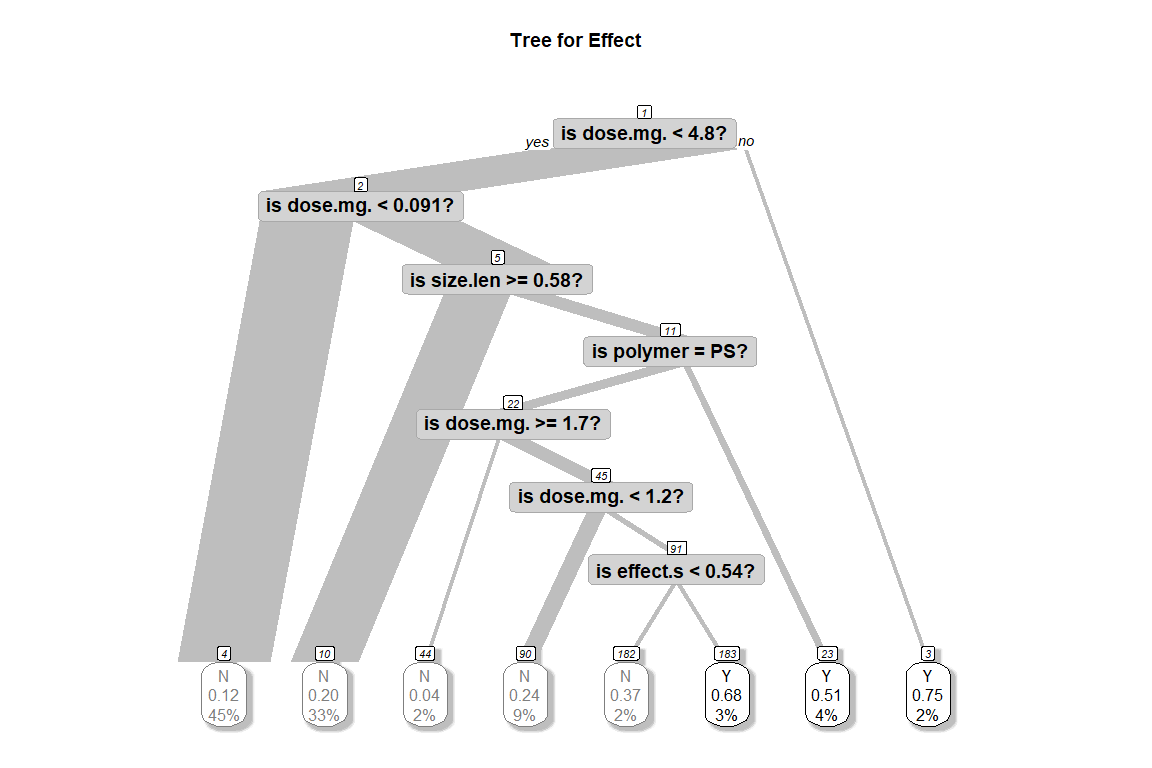
# Interpretable Tree

Make a classification tree.

## n= 1214   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 1214 247 N (0.79654 0.20346)   
## 2) dose.mg.L.master< 4.753 1190 229 N (0.80756 0.19244)   
## 4) dose.mg.L.master< 0.09109 549 66 N (0.87978 0.12022) \*  
## 5) dose.mg.L.master>=0.09109 641 163 N (0.74571 0.25429)   
## 10) size.length.um.used.for.conversions>=0.5784 399 79 N (0.80201 0.19799) \*  
## 11) size.length.um.used.for.conversions< 0.5784 242 84 N (0.65289 0.34711)   
## 22) polymer=PS 195 60 N (0.69231 0.30769)   
## 44) dose.mg.L.master>=1.651 23 1 N (0.95652 0.04348) \*  
## 45) dose.mg.L.master< 1.651 172 59 N (0.65698 0.34302)   
## 90) dose.mg.L.master< 1.224 111 27 N (0.75676 0.24324) \*  
## 91) dose.mg.L.master>=1.224 61 29 Y (0.47541 0.52459)   
## 182) effect.score< 0.5396 30 11 N (0.63333 0.36667) \*  
## 183) effect.score>=0.5396 31 10 Y (0.32258 0.67742) \*  
## 23) polymer=Not Reported,PE 47 23 Y (0.48936 0.51064) \*  
## 3) dose.mg.L.master>=4.753 24 6 Y (0.25000 0.75000) \*

Plot an interpretable tree.

## cex 1 xlim c(-0.2, 1.2) ylim c(0, 1)



GINI importance measures the average gain of purity by splits of a given variable. If the variable is useful, it tends to split mixed labeled nodes into pure single class nodes. Splitting by a permuted variables tend neither to increase nor decrease node purities. Permuting a useful variable, tend to give relatively large decrease in mean gini-gain. GINI importance is closely related to the local decision function, that random forest uses to select the best available split. Therefore, it does not take much extra time to compute. On the other hand, mean gini-gain in local splits, is not necessarily what is most useful to measure, in contrary to change of overall model performance. Gini importance is overall inferior to (permutation based) variable importance as it is relatively more biased, more unstable and tend to answer a more indirect question.

### Quantile Regression model

#predict(myqrf, what=c(0.2, 0.3, 0.999)) # to print specific quantiles

Again appears to improve after ~100 trees.

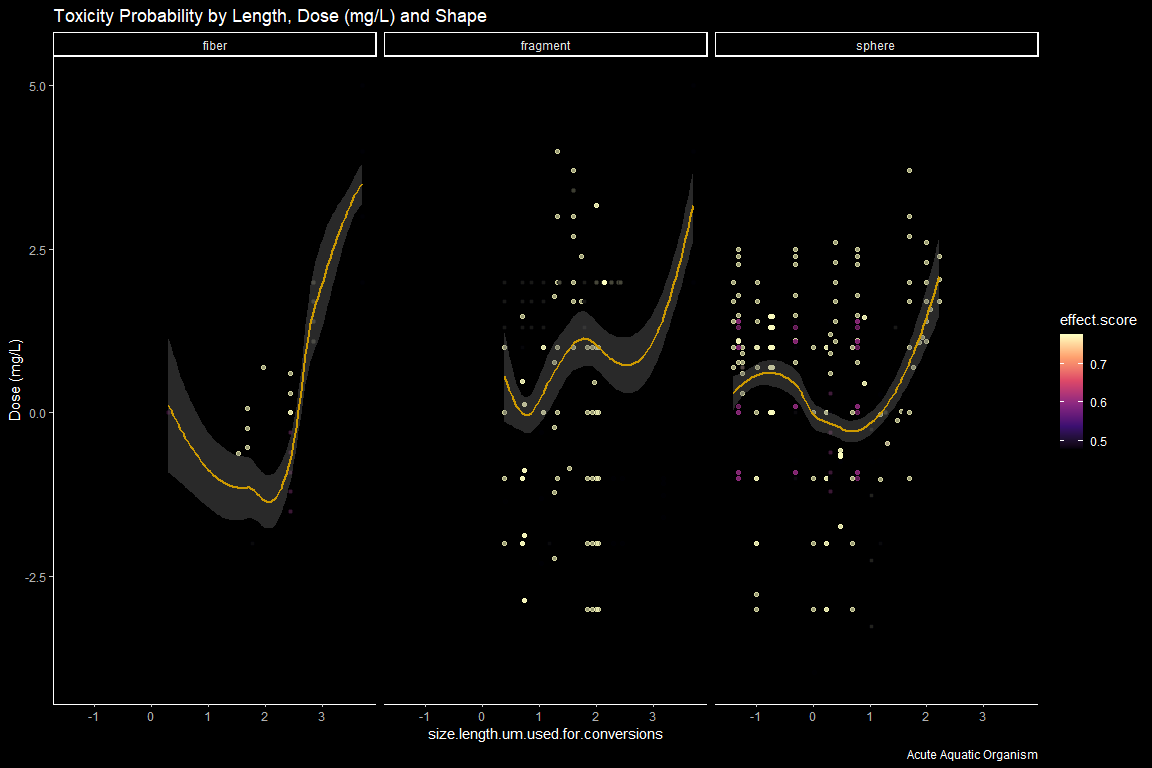
### Model Validation

#### Categorical Response

#### Continuous Response

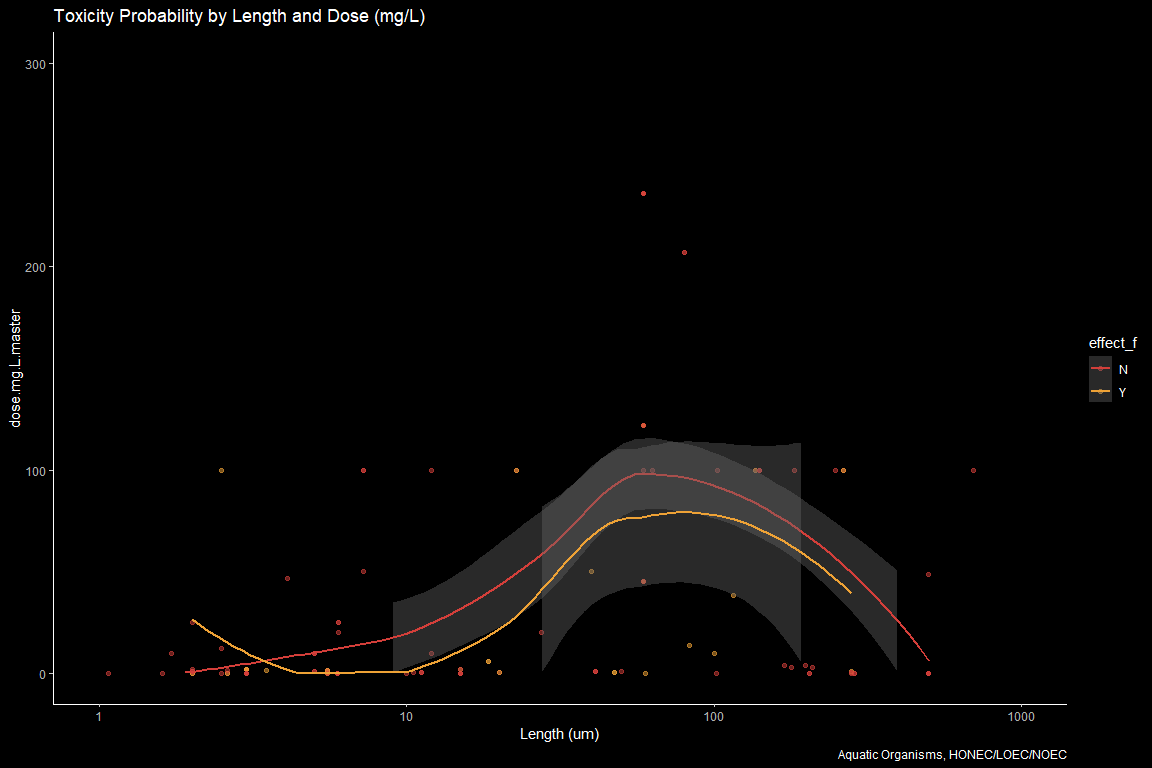
#3-D plots

#remotes::install\_github("tylermorganwall/rayshader")  
#require(rayshader)  
require(tidyverse)  
require(ggsci)  
require(ggdark)  
  
#scatterplot with size, dose and polymer  
acute3D <- acute %>%   
 #filter(effect\_f == "Yes") %>%   
 ggplot(aes(x = size.length.um.used.for.conversions, y = dose.mg.L.master, color = effect.score)) + #, shape = polymer)) +  
 geom\_point(alpha = 0.6) +  
 geom\_smooth() +  
 scale\_y\_continuous("Dose (mg/L)", limits=c(-4,5)) +  
 scale\_color\_viridis\_c(option = "A") +  
 ggtitle("Toxicity Probability by Length, Dose (mg/L) and Shape") +  
 labs(caption = "Acute Aquatic Organism") +  
 dark\_theme\_classic() +  
 facet\_wrap(shape~.)  
  
acute3D

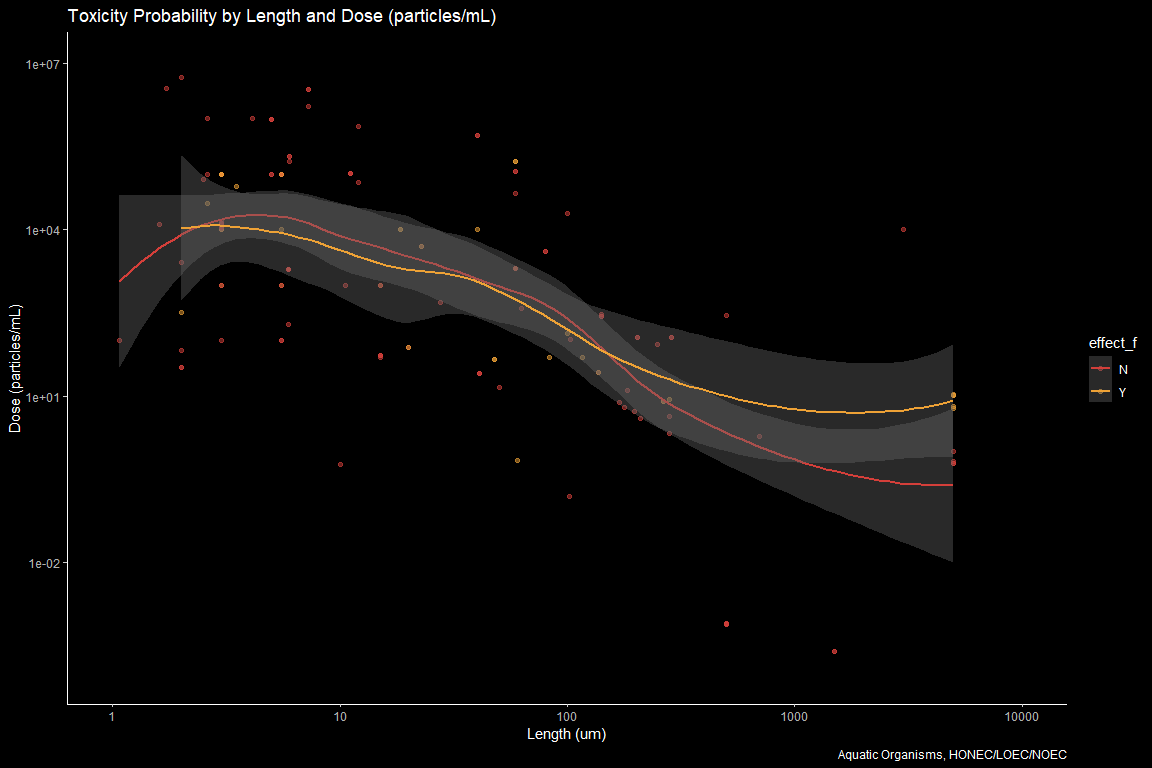


#3D plot  
#plot\_gg(acute3D, multicore = TRUE, width = 5, height = 5, scale =250) #3D plot

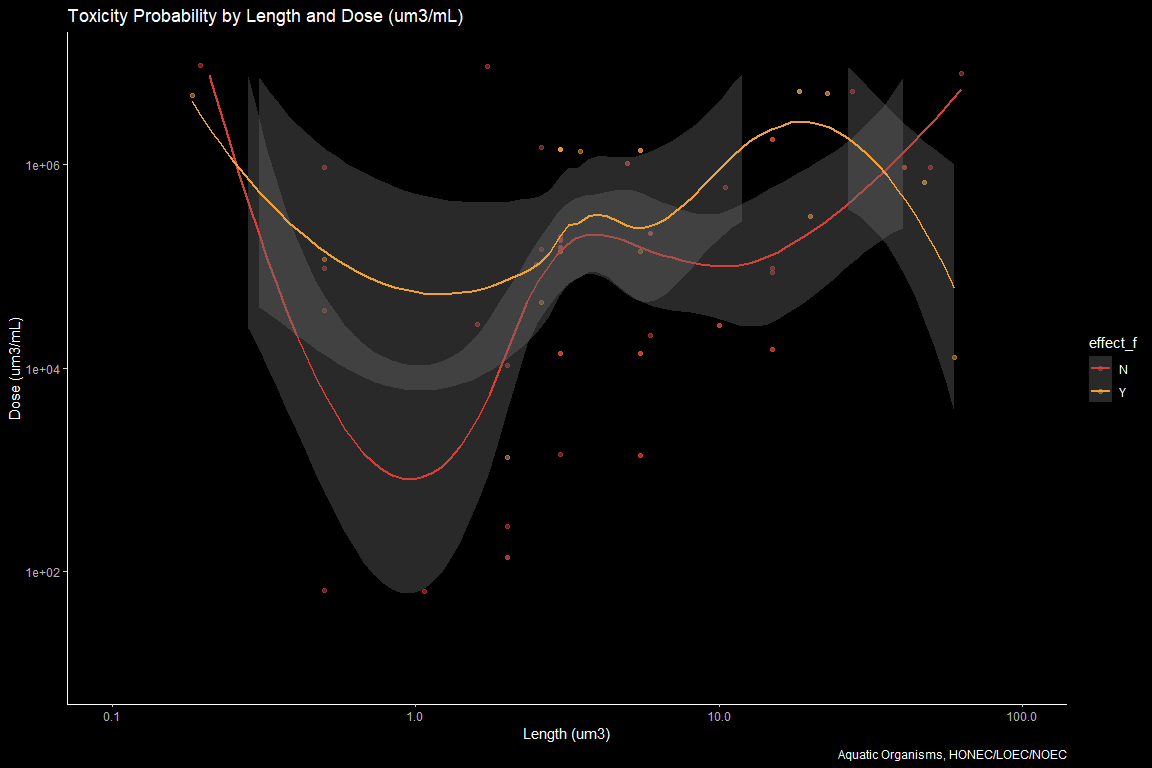
#scatterplot with size, dose and polymer  
mass <-multiVar %>%   
 filter(!effect\_f == "NA") %>%   
 filter(effect.metric == c("HONEC", "LOEC", "NOEC")) %>%   
 filter(!acute.chronic\_f == "Unavailable") %>%   
 filter(bio.org == "organism") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 ggplot(aes(x = size.length.um.used.for.conversions, y = dose.mg.L.master, color = effect\_f)) +   
 geom\_point(alpha = 0.5) +  
 geom\_smooth() +  
 scale\_y\_continuous(limits = c(0, 300))+  
 # scale\_y\_log10("Dose (mg/L)",limits = c(1e-4, 1e7))+  
 scale\_x\_log10("Length (um)", limits = c(1, 1e3)) +  
 #coord\_trans(x = "log10") +  
 #scale\_x\_continuous("Length (um)", breaks = scales::trans\_breaks("log10", function(x) 10^x, n = 5),  
 # labels = trans\_format("log10", scales::math\_format(10^.x))) +  
 scale\_colour\_locuszoom() +  
 ggtitle("Toxicity Probability by Length and Dose (mg/L)") +  
 labs(caption = "Aquatic Organisms, HONEC/LOEC/NOEC") +  
 dark\_theme\_classic()# +   
 #facet\_wrap(acute.chronic\_f ~.)  
mass



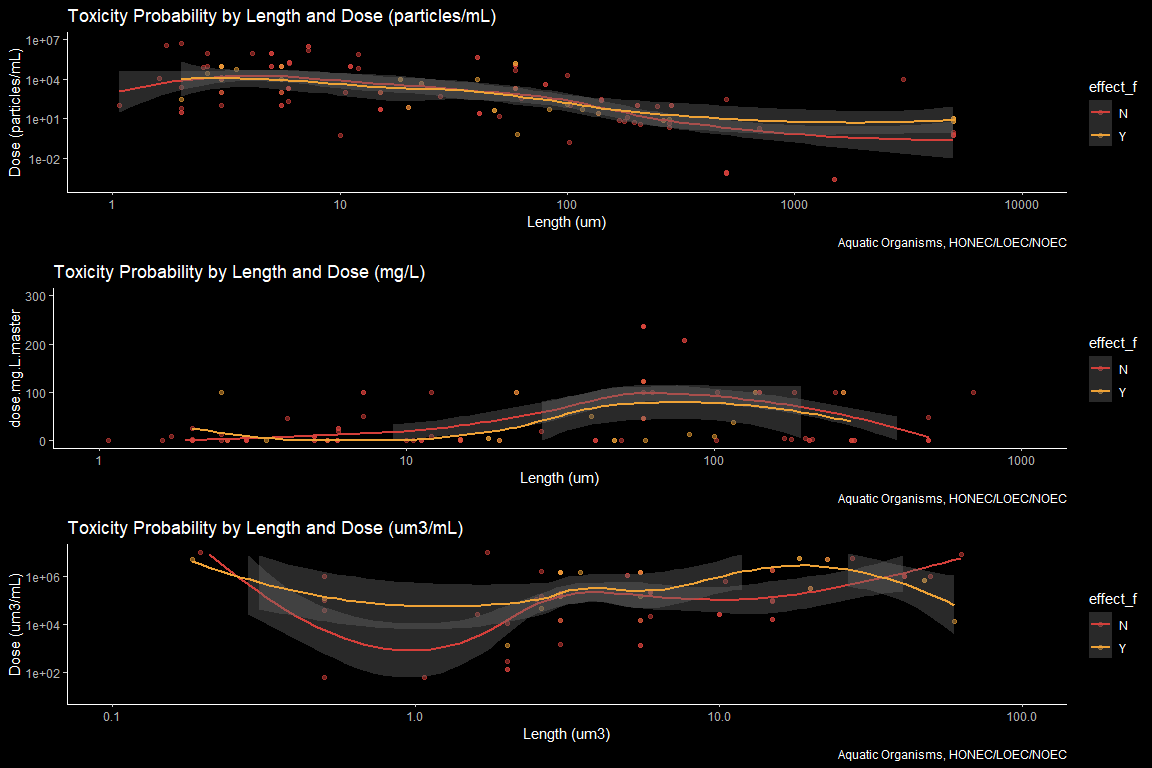
particles <- multiVar %>%   
 filter(!effect\_f == "NA") %>%   
 filter(effect.metric == c("HONEC", "LOEC", "NOEC")) %>%   
 filter(!acute.chronic\_f == "Unavailable") %>%   
 filter(bio.org == "organism") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 ggplot(aes(x = size.length.um.used.for.conversions, y = dose.particles.mL.master, color = effect\_f)) +   
 geom\_point(alpha = 0.5) +  
 geom\_smooth() +  
 scale\_y\_log10("Dose (particles/mL)",limits = c(1e-4, 1e7))+  
 scale\_x\_log10("Length (um)", limits = c(1, 1e4)) +  
 #coord\_trans(x = "log10") +  
 #scale\_x\_continuous("Length (um)", breaks = scales::trans\_breaks("log10", function(x) 10^x, n = 5),  
 # labels = trans\_format("log10", scales::math\_format(10^.x))) +  
 scale\_colour\_locuszoom() +  
 ggtitle("Toxicity Probability by Length and Dose (particles/mL)") +  
 labs(caption = "Aquatic Organisms, HONEC/LOEC/NOEC") +  
 dark\_theme\_classic() #+   
 #facet\_wrap(acute.chronic\_f ~.)  
particles



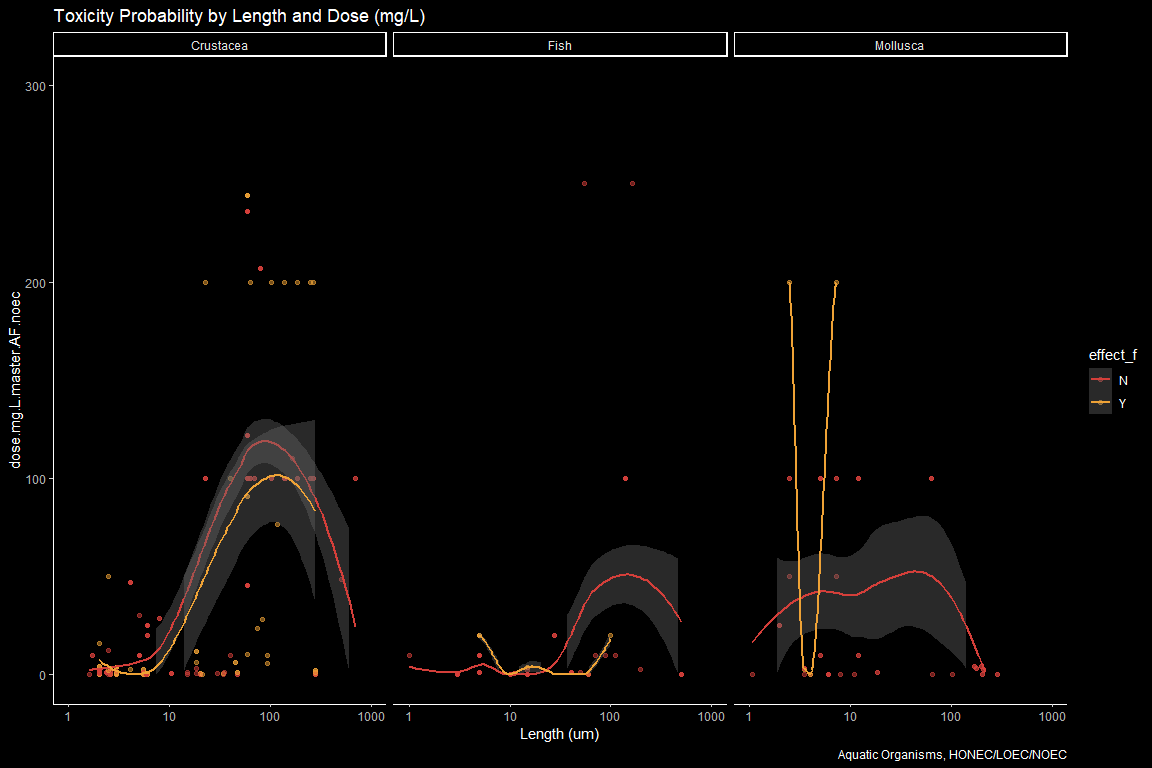
volume <- aoc\_z %>%   
 filter(!effect\_f == "NA") %>%   
 filter(effect.metric == c("HONEC", "LOEC", "NOEC")) %>%   
 filter(!acute.chronic\_f == "Unavailable") %>%   
 filter(bio.org == "organism") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 ggplot(aes(x = size.length.um.used.for.conversions, y = dose.um3.mL.master, color = effect\_f)) +   
 geom\_point(alpha = 0.5) +  
 geom\_smooth() +  
 scale\_y\_log10("Dose (um3/mL)",limits = c(1e+1, 1e7))+  
 scale\_x\_log10("Length (um3)", limits = c(1e-1, 100)) +  
 #coord\_trans(x = "log10") +  
 #scale\_x\_continuous("Length (um)", breaks = scales::trans\_breaks("log10", function(x) 10^x, n = 5),  
 # labels = trans\_format("log10", scales::math\_format(10^.x))) +  
 scale\_colour\_locuszoom() +  
 ggtitle("Toxicity Probability by Length and Dose (um3/mL)") +  
 labs(caption = "Aquatic Organisms, HONEC/LOEC/NOEC") +  
 dark\_theme\_classic() #+   
 #facet\_wrap(acute.chronic\_f ~.)  
volume



require(gridExtra)  
grid.arrange(particles, mass, volume, nrow = 3)



#scatterplot with size, dose and polymer  
taxa\_mass <-aoc\_z %>%   
 filter(!effect\_f == "NA") %>%   
 # filter(effect.metric == c("HONEC", "LOEC", "NOEC")) %>%   
 filter(!acute.chronic\_f == "Unavailable") %>%   
 filter(bio.org == "organism") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 ggplot(aes(x = size.length.um.used.for.conversions, y = dose.mg.L.master.AF.noec, color = effect\_f)) +   
 geom\_point(alpha = 0.5) +  
 geom\_smooth() +  
 scale\_y\_continuous(limits = c(0, 300))+  
 # scale\_y\_log10("Dose (mg/L)",limits = c(1e-4, 1e7))+  
 scale\_x\_log10("Length (um)", limits = c(1, 1e3)) +  
 #coord\_trans(x = "log10") +  
 #scale\_x\_continuous("Length (um)", breaks = scales::trans\_breaks("log10", function(x) 10^x, n = 5),  
 # labels = trans\_format("log10", scales::math\_format(10^.x))) +  
 scale\_colour\_locuszoom() +  
 ggtitle("Toxicity Probability by Length and Dose (mg/L)") +  
 labs(caption = "Aquatic Organisms, HONEC/LOEC/NOEC") +  
 dark\_theme\_classic() +   
 facet\_wrap(organism.group ~.)  
taxa\_mass



#Logistic Regression for acute fitness   
m1\_crust <-aoc\_z %>%   
 filter(!effect\_f == "NA") %>%   
 # filter(effect.metric == c("HONEC", "LOEC", "NOEC")) %>%   
# filter(organism.group == "Crustacea") %>%   
 filter(!acute.chronic\_f == "Unavailable") %>%   
 filter(bio.org == "organism") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 filter(!size\_f == "Not Reported") %>%   
 mutate(logdose.mg.L.master = log10(dose.mg.L.master))#%>%   
 # filter(acute.chronic\_f == "Acute")  
  
m1\_crust\_model <- glm(effect\_10 ~ (size.length.um.used.for.conversions + log10(dose.mg.L.master) +  
 log10(dose.particles.mL.master) + organism.group) ^ 2, #exponent gives all 2-way interactions  
 data = m1\_crust, na.action = "na.exclude", family = "binomial")  
  
summary(m1\_crust\_model)

##   
## Call:  
## glm(formula = effect\_10 ~ (size.length.um.used.for.conversions +   
## log10(dose.mg.L.master) + log10(dose.particles.mL.master) +   
## organism.group)^2, family = "binomial", data = m1\_crust,   
## na.action = "na.exclude")  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2294 -0.7369 -0.6168 -0.1817 2.8654   
##   
## Coefficients:  
## Estimate  
## (Intercept) -9.780e-01  
## size.length.um.used.for.conversions -1.785e-03  
## log10(dose.mg.L.master) 2.711e-02  
## log10(dose.particles.mL.master) -4.491e-02  
## organism.groupFish 5.462e-01  
## organism.groupMollusca -4.519e+00  
## size.length.um.used.for.conversions:log10(dose.mg.L.master) -9.672e-05  
## size.length.um.used.for.conversions:log10(dose.particles.mL.master) 2.284e-04  
## size.length.um.used.for.conversions:organism.groupFish -2.064e-02  
## size.length.um.used.for.conversions:organism.groupMollusca 3.543e-03  
## log10(dose.mg.L.master):log10(dose.particles.mL.master) 4.640e-02  
## log10(dose.mg.L.master):organism.groupFish 4.597e-01  
## log10(dose.mg.L.master):organism.groupMollusca -4.120e-01  
## log10(dose.particles.mL.master):organism.groupFish -3.123e-01  
## log10(dose.particles.mL.master):organism.groupMollusca 4.461e-01  
## Std. Error  
## (Intercept) 2.058e-01  
## size.length.um.used.for.conversions 1.346e-03  
## log10(dose.mg.L.master) 1.003e-01  
## log10(dose.particles.mL.master) 3.560e-02  
## organism.groupFish 9.960e-01  
## organism.groupMollusca 9.400e-01  
## size.length.um.used.for.conversions:log10(dose.mg.L.master) 4.184e-04  
## size.length.um.used.for.conversions:log10(dose.particles.mL.master) 4.093e-04  
## size.length.um.used.for.conversions:organism.groupFish 1.259e-02  
## size.length.um.used.for.conversions:organism.groupMollusca 1.997e-03  
## log10(dose.mg.L.master):log10(dose.particles.mL.master) 1.959e-02  
## log10(dose.mg.L.master):organism.groupFish 3.044e-01  
## log10(dose.mg.L.master):organism.groupMollusca 2.247e-01  
## log10(dose.particles.mL.master):organism.groupFish 2.055e-01  
## log10(dose.particles.mL.master):organism.groupMollusca 1.068e-01  
## z value  
## (Intercept) -4.752  
## size.length.um.used.for.conversions -1.326  
## log10(dose.mg.L.master) 0.270  
## log10(dose.particles.mL.master) -1.262  
## organism.groupFish 0.548  
## organism.groupMollusca -4.808  
## size.length.um.used.for.conversions:log10(dose.mg.L.master) -0.231  
## size.length.um.used.for.conversions:log10(dose.particles.mL.master) 0.558  
## size.length.um.used.for.conversions:organism.groupFish -1.639  
## size.length.um.used.for.conversions:organism.groupMollusca 1.774  
## log10(dose.mg.L.master):log10(dose.particles.mL.master) 2.369  
## log10(dose.mg.L.master):organism.groupFish 1.510  
## log10(dose.mg.L.master):organism.groupMollusca -1.834  
## log10(dose.particles.mL.master):organism.groupFish -1.520  
## log10(dose.particles.mL.master):organism.groupMollusca 4.176  
## Pr(>|z|)  
## (Intercept) 2.01e-06  
## size.length.um.used.for.conversions 0.1847  
## log10(dose.mg.L.master) 0.7869  
## log10(dose.particles.mL.master) 0.2071  
## organism.groupFish 0.5834  
## organism.groupMollusca 1.53e-06  
## size.length.um.used.for.conversions:log10(dose.mg.L.master) 0.8172  
## size.length.um.used.for.conversions:log10(dose.particles.mL.master) 0.5767  
## size.length.um.used.for.conversions:organism.groupFish 0.1013  
## size.length.um.used.for.conversions:organism.groupMollusca 0.0760  
## log10(dose.mg.L.master):log10(dose.particles.mL.master) 0.0179  
## log10(dose.mg.L.master):organism.groupFish 0.1309  
## log10(dose.mg.L.master):organism.groupMollusca 0.0667  
## log10(dose.particles.mL.master):organism.groupFish 0.1286  
## log10(dose.particles.mL.master):organism.groupMollusca 2.96e-05  
##   
## (Intercept) \*\*\*  
## size.length.um.used.for.conversions   
## log10(dose.mg.L.master)   
## log10(dose.particles.mL.master)   
## organism.groupFish   
## organism.groupMollusca \*\*\*  
## size.length.um.used.for.conversions:log10(dose.mg.L.master)   
## size.length.um.used.for.conversions:log10(dose.particles.mL.master)   
## size.length.um.used.for.conversions:organism.groupFish   
## size.length.um.used.for.conversions:organism.groupMollusca .   
## log10(dose.mg.L.master):log10(dose.particles.mL.master) \*   
## log10(dose.mg.L.master):organism.groupFish   
## log10(dose.mg.L.master):organism.groupMollusca .   
## log10(dose.particles.mL.master):organism.groupFish   
## log10(dose.particles.mL.master):organism.groupMollusca \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1311.5 on 1310 degrees of freedom  
## Residual deviance: 1190.7 on 1296 degrees of freedom  
## (293 observations deleted due to missingness)  
## AIC: 1220.7  
##   
## Number of Fisher Scoring iterations: 9

m1\_crust\_model$coef

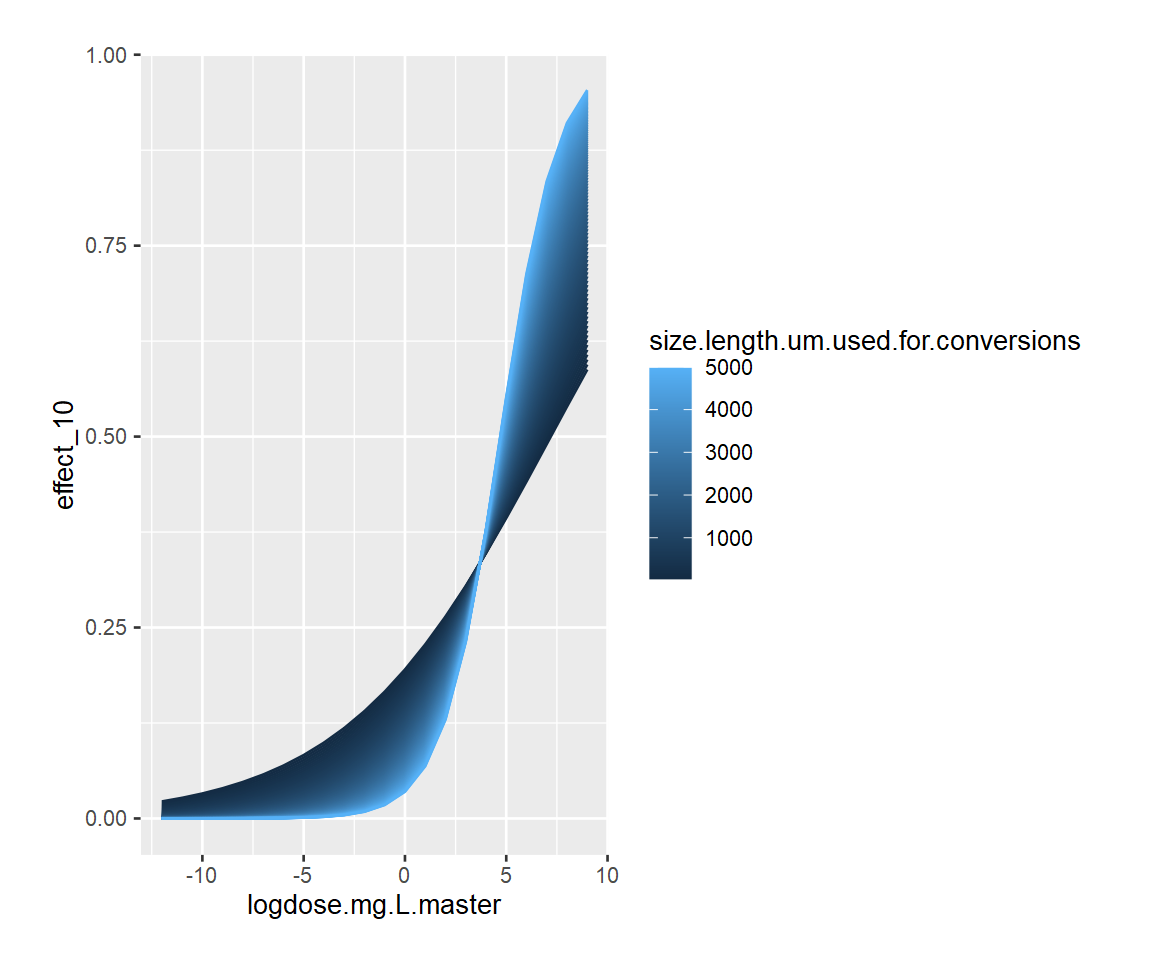
## (Intercept)   
## -9.780446e-01   
## size.length.um.used.for.conversions   
## -1.785247e-03   
## log10(dose.mg.L.master)   
## 2.710767e-02   
## log10(dose.particles.mL.master)   
## -4.490859e-02   
## organism.groupFish   
## 5.461589e-01   
## organism.groupMollusca   
## -4.519370e+00   
## size.length.um.used.for.conversions:log10(dose.mg.L.master)   
## -9.671817e-05   
## size.length.um.used.for.conversions:log10(dose.particles.mL.master)   
## 2.284465e-04   
## size.length.um.used.for.conversions:organism.groupFish   
## -2.063551e-02   
## size.length.um.used.for.conversions:organism.groupMollusca   
## 3.542676e-03   
## log10(dose.mg.L.master):log10(dose.particles.mL.master)   
## 4.640464e-02   
## log10(dose.mg.L.master):organism.groupFish   
## 4.597273e-01   
## log10(dose.mg.L.master):organism.groupMollusca   
## -4.120115e-01   
## log10(dose.particles.mL.master):organism.groupFish   
## -3.122596e-01   
## log10(dose.particles.mL.master):organism.groupMollusca   
## 4.460948e-01

## Plot binomial

m1\_crust\_simple <- m1\_crust %>%   
 dplyr::select(c(effect\_10, logdose.mg.L.master, size.length.um.used.for.conversions)) %>%   
 drop\_na  
#simple single parameter  
m1\_crust\_model\_dose <- glm(effect\_10 ~ logdose.mg.L.master \* size.length.um.used.for.conversions, #exponent gives all 2-way interactions  
 data = m1\_crust\_simple, na.action = "na.exclude", family = "binomial")  
  
summary(m1\_crust\_model\_dose)

##   
## Call:  
## glm(formula = effect\_10 ~ logdose.mg.L.master \* size.length.um.used.for.conversions,   
## family = "binomial", data = m1\_crust\_simple, na.action = "na.exclude")  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.8635 -0.7140 -0.6165 -0.5017 2.0663   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -1.418e+00 7.292e-02  
## logdose.mg.L.master 1.970e-01 4.744e-02  
## size.length.um.used.for.conversions -3.797e-04 1.787e-04  
## logdose.mg.L.master:size.length.um.used.for.conversions 1.016e-04 4.554e-05  
## z value Pr(>|z|)   
## (Intercept) -19.447 < 2e-16 \*\*\*  
## logdose.mg.L.master 4.152 3.29e-05 \*\*\*  
## size.length.um.used.for.conversions -2.125 0.0336 \*   
## logdose.mg.L.master:size.length.um.used.for.conversions 2.230 0.0257 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1358.5 on 1347 degrees of freedom  
## Residual deviance: 1316.5 on 1344 degrees of freedom  
## AIC: 1324.5  
##   
## Number of Fisher Scoring iterations: 4

#alternative plot with ggpredict  
#devtools::install\_github("cardiomoon/ggiraphExtra")  
require(ggiraphExtra)  
  
ggPredict(m1\_crust\_model\_dose,interactive=TRUE,colorn=100,jitter=FALSE)

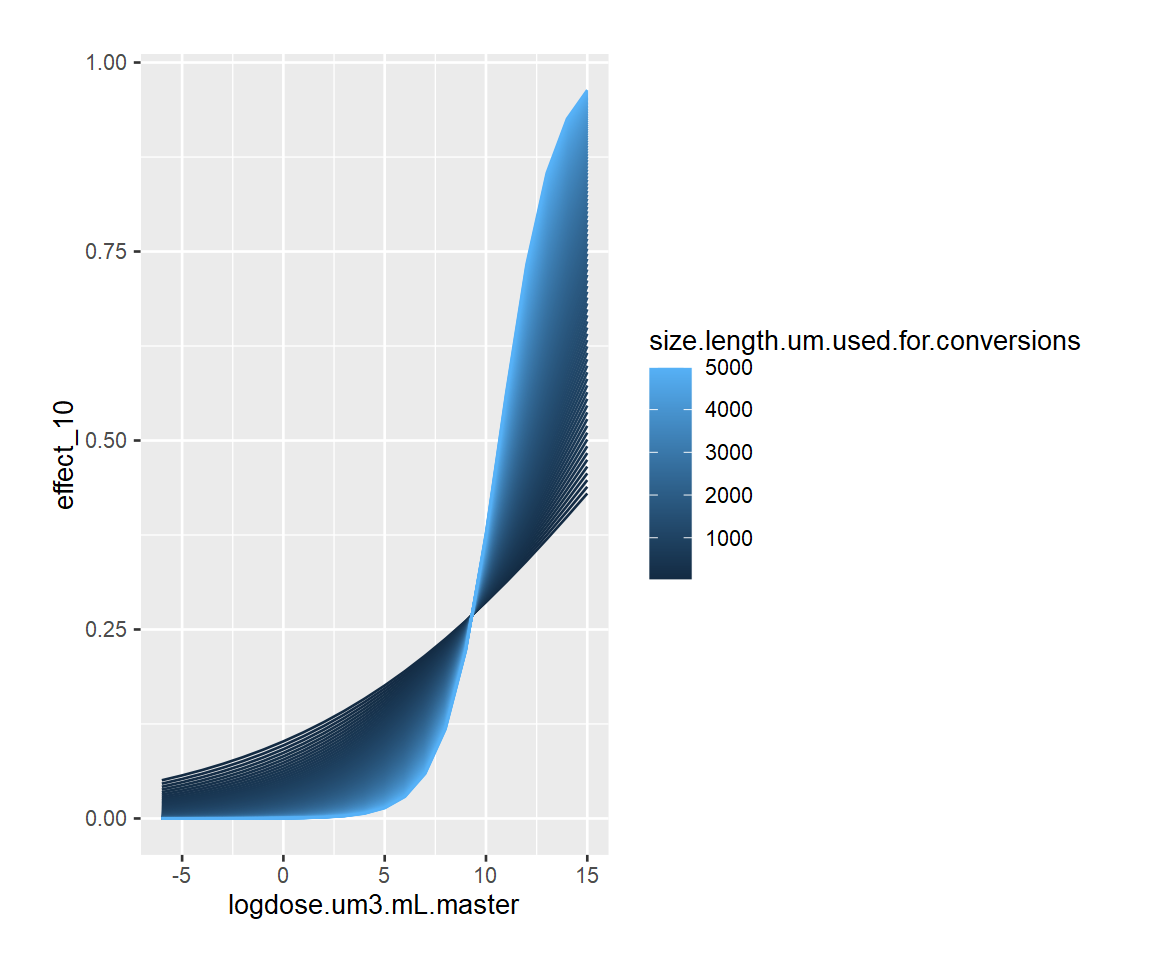
 What happens if we log-transfrom size?

What about volume?

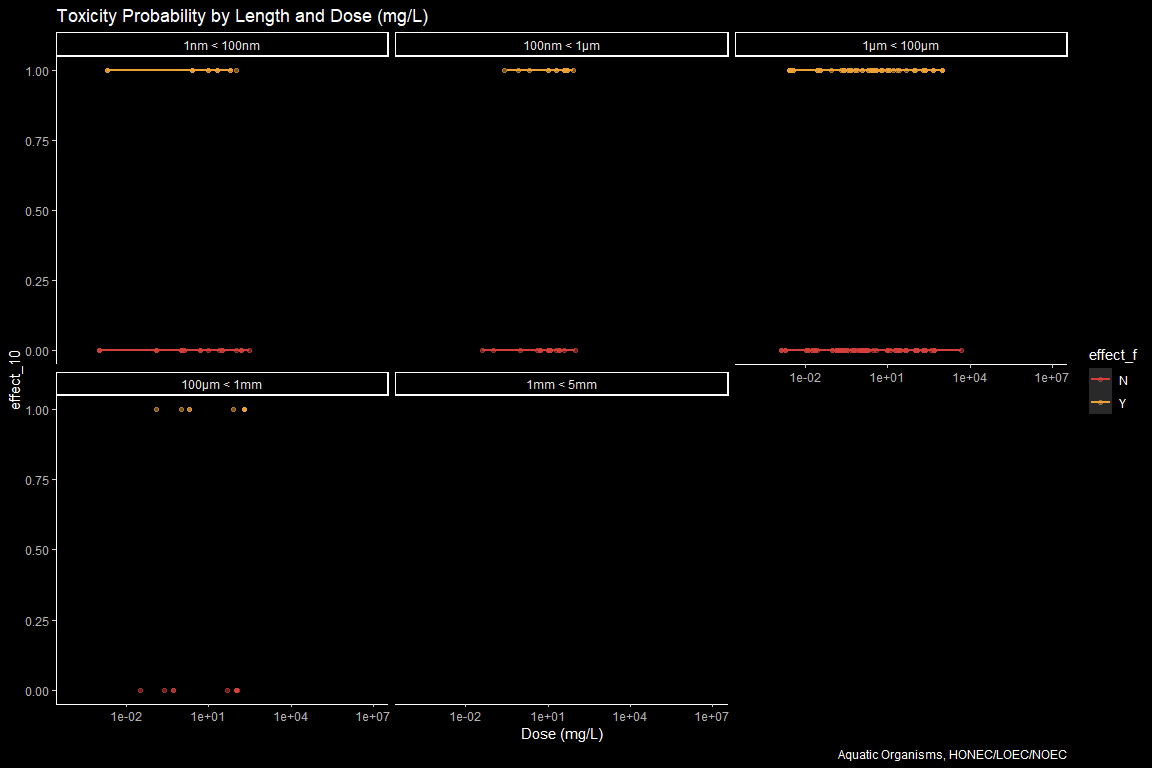
m1\_crust\_simple\_volume <- m1\_crust %>%   
 mutate(logdose.um3.mL.master = log10(dose.um3.mL.master)) %>%   
 dplyr::select(c(effect\_10, logdose.um3.mL.master, size.length.um.used.for.conversions)) %>%   
 drop\_na  
#simple single parameter  
m1\_crust\_model\_volume<- glm(effect\_10 ~ logdose.um3.mL.master \* size.length.um.used.for.conversions, #exponent gives all 2-way interactions  
 data = m1\_crust\_simple\_volume, na.action = "na.exclude", family = "binomial")  
  
summary(m1\_crust\_model\_volume)

##   
## Call:  
## glm(formula = effect\_10 ~ logdose.um3.mL.master \* size.length.um.used.for.conversions,   
## family = "binomial", data = m1\_crust\_simple\_volume, na.action = "na.exclude")  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.8116 -0.6866 -0.6282 -0.5292 2.0869   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -2.176e+00 2.574e-01  
## logdose.um3.mL.master 1.262e-01 4.049e-02  
## size.length.um.used.for.conversions -1.176e-03 4.464e-04  
## logdose.um3.mL.master:size.length.um.used.for.conversions 1.259e-04 4.533e-05  
## z value Pr(>|z|)   
## (Intercept) -8.451 < 2e-16 \*\*\*  
## logdose.um3.mL.master 3.118 0.00182 \*\*   
## size.length.um.used.for.conversions -2.635 0.00842 \*\*   
## logdose.um3.mL.master:size.length.um.used.for.conversions 2.777 0.00549 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1311.5 on 1310 degrees of freedom  
## Residual deviance: 1277.6 on 1307 degrees of freedom  
## AIC: 1285.6  
##   
## Number of Fisher Scoring iterations: 4

#plot logistic volume  
require(ggiraphExtra)  
ggPredict(m1\_crust\_model\_volume,interactive=TRUE,colorn=100,jitter=FALSE)



#scatterplot with size, dose and polymer  
mass\_probability\_crustacea <-aoc\_z %>%   
 filter(!effect\_f == "NA") %>%   
 # filter(effect.metric == c("HONEC", "LOEC", "NOEC")) %>%   
 filter(organism.group == "Crustacea") %>%   
 filter(!acute.chronic\_f == "Unavailable") %>%   
 filter(bio.org == "organism") %>%   
 filter(!environment == "Terrestrial") %>%   
 filter(lvl1\_f == "Fitness") %>%   
 filter(!size\_f == "Not Reported") %>%   
 droplevels() %>%   
 ggplot(aes(x = dose.mg.L.master.AF.noec, y = effect\_10, color = effect\_f)) +   
 geom\_point(alpha = 0.5) +  
 geom\_smooth() +  
 scale\_y\_continuous(limits = c(0, 1))+  
 scale\_x\_log10("Dose (mg/L)",limits = c(1e-4, 1e7))+  
 #scale\_x\_log10("Length (um)", limits = c(1, 1e3)) +  
 #coord\_trans(x = "log10") +  
 #scale\_x\_continuous("Length (um)", breaks = scales::trans\_breaks("log10", function(x) 10^x, n = 5),  
 # labels = trans\_format("log10", scales::math\_format(10^.x))) +  
 scale\_colour\_locuszoom() +  
 ggtitle("Toxicity Probability by Length and Dose (mg/L)") +  
 labs(caption = "Aquatic Organisms, HONEC/LOEC/NOEC") +  
 dark\_theme\_classic() +   
 facet\_wrap(size\_f ~.)  
mass\_probability\_crustacea



# Works Cited

======= >>>>>>> 1a6a3e4a7fded4e841412cd86dae11eed4ce0abe # Non-linear Dose Response If we hold all paramaters constant except for dose, we should observe a non-linear dose-response behavior that can be best predicted by a 3- or 4-parameter Hill Equation. The most available information is likely for polystyrene spheres in crustaceans.There are 21 studies available.