Poxvirus Host Prediction

Katie Tseng, Dan Becker, Colin Carlson, etc.

Introduction

The following code reproduces the analysis from:

> etc.

Data Preparation

Load required packages and set system

```
#(1) libraries for preparing data for analysis
library(ape)
library(dplyr)
library(nlme)
library(tidyverse)
library(vroom)
## treespace dependencies include XQuartz v2.7.11 (https://www.xquartz.org/releases/XQuartz-2.7.11.html
library(rgl) # >install.packages("rgl"); >options(rgl.useNULL=TRUE)
library(treespace)

#(2) clean environment
rm(list=ls())
graphics.off()

#(3)set working directory
setwd("~/Library/CloudStorage/OneDrive-WashingtonStateUniversity(email.wsu.edu)/Fernandez Lab/Projects
```

Load raw data

```
#(1) load data
load("Data_raw.RData")

#(2) poxdata: host-OPV interactions detected via PCR/isolation from Virion database
##virion <- vroom('https://github.com/viralemergence/virion/blob/main/Virion/Virion.csv.gz')
poxdata <- virion %>% filter(VirusGenus == "orthopoxvirus" & (DetectionMethod %in% c("PCR/Sequencing","

#(3) taxa: mammal species taxonomy from vertlife
##vertlife <- read.csv(url('https://data.vertlife.org/mammaltree/taxonomy_mamPhy_5911species.csv'))
```

```
taxa <- vertlife

#(4) hostTraits: mammal traits from the COMBINE database <https://doi.org/10.1002/ecy.3344>
##path: ecy3344-sup-0001-datas1.zip > COMBINE_archives > trait_data_imputed.csv)
hostTraits <- combine

#(5) hostTree: mammal phylogeny tree from Dryad, <https://doi.org/10.5061/dryad.tb03d03>
##path: Data_S8_finalFigureFiles > _DATA > MamPhy_fullPosterior_BDvr_Completed_5911sp_topoCons_NDexp_MC
hostTree <- dryad

#(6) viralTraits: OPV accessory genes from ... (Steph to provide refined datatable)
viralTraits <- opvgenes

#(7) clean environment
rm(virion, vertlife, dryad, combine, opvgenes)
```

Aggregate poxdata to genus-level

```
#(1) exclude if host genus or virus is NA; exclude variola (smallpox) virus
poxdata <- poxdata[!is.na(poxdata$HostGenus),]
poxdata <- poxdata[!is.na(poxdata$Virus),]
poxdata <- poxdata[!(poxdata$Virus=="variola virus"),]

#(2) to dis-aggregate West African from Congo Basin MPXV clades, export MPXV interactions
mpxvdata <- poxdata %>% filter(Virus=="monkeypox virus" & (DetectionMethod %in% c("PCR/Sequencing","Isol
#write.csv(mpxvdata, "~/mpxvdata.csv")

#(3) merge clade-specific data
#TBD: Steph to share clade-specific data

#(4) extract PCR-positive data
pcr <- subset(poxdata[which(poxdata$DetectionMethod=="PCR/Sequencing"),], select=c("Host","HostGenus","V
pcr$Host <- ifelse(is.na(pcr$Host),"sp.",pcr$Host)
pcr$pcr <- 1
pcr <- aggregate(.~Host+HostGenus+Virus, data=pcr, sum)

#(5) extract isolation-positive data
competence <- subset(poxdata[which(poxdata$DetectionMethod=="Isolation/Observation"),], select=c("Host"
competence$Host <- ifelse(is.na(competence$Host),"sp.",competence$Host)
competence$competence <- 1
competence <- aggregate(.~Host+HostGenus+Virus, data=competence, sum)

#(6) merge PCR/isolation-positive data; create binary vars
poxdata <- merge(pcr, competence, by=c("Host","HostGenus","Virus"), all=TRUE)

#(7) create studies variable
poxdata$studies <- ifelse(is.na(poxdata$pcr),0,poxdata$pcr) + ifelse(is.na(poxdata$competence),0,poxdata

#(8) create binary variables for detection via pcr/competence
poxdata$pcr=ifelse(is.na(poxdata$pcr),0,1)
poxdata$competence=ifelse(is.na(poxdata$competence),0,1)
```

```
#(9) aggregate by genus and virus
agg_pcr <- aggregate(pcr~HostGenus+Virus, data=poxdata, max)
agg_competence <- aggregate(competence~HostGenus+Virus, data=poxdata, max)
agg_studies <- aggregate(studies~HostGenus+Virus, data=poxdata, sum)

#(10) merge pcr, competence and studies variables
poxdata <- merge(agg_pcr,agg_competence)
poxdata <- merge(poxdata,agg_studies)

#(11) rename variables
poxdata <- rename(poxdata,c('HostGenus'='gen','Virus'='virus'))
poxdata$gen <- str_to_title(poxdata$gen)

#(12) clean environment
rm(mpxvdata, pcr,competence,agg_pcr, agg_competence, agg_studies)
```

Merge poxdata with broader mammal taxa to create pseudoabsences

```
#(1) drop duplicate genera in taxa
gtaxa <- taxa[!duplicated(taxa$gen),]
gtaxa <- gtaxa[c('gen','fam','ord')]

#(2) check for mismatched names, then merge poxdata with taxa
poxdata$gen[!poxdata$gen %in% taxa$gen]
poxdata <- merge(gtaxa,poxdata,by='gen',all.x=TRUE)

#(3) keep only genera from orders in which positive associations exist
keep <- subset(poxdata, pcr==1 | competence==1)
poxdata$keep <- ifelse(poxdata$ord %in% keep$ord,TRUE,FALSE)
poxdata <- subset(poxdata,keep==TRUE)
poxdata$keep=NULL

#(6) create binary variable for sampled host-OPV pairs
poxdata$sampled=ifelse(is.na(poxdata$pcr) & is.na(poxdata$competence),0,1)

#(7) reclassify NAs as pseudo-absences for viral detection
poxdata$pcr=ifelse(is.na(poxdata$pcr),0,poxdata$pcr)
poxdata$competence=ifelse(is.na(poxdata$competence),0,poxdata$competence)
poxdata$studies=ifelse(is.na(poxdata$studies),0,poxdata$studies)

#(8) replace NA taxonomic values based on host genera
poxdata=merge(poxdata,gtaxa,by='gen',all.x=TRUE)
poxdata <- rename(poxdata,c('fam.y'='fam','ord.y'='ord'))
poxdata$fam.x=NULL
poxdata$ord.x=NULL

#(9) clean environment
rm(taxa,gtaxa,keep)
```

Aggregate hostTraits to genus-level

```
#(1) observe variable names
colnames(hostTraits)

#(2) to aggregate continuous/integer variables, use the median as the summary measure
hostTraits_continuous=aggregate(cbind(adult_mass_g,brain_mass_g,adult_body_length_mm,adult_forearm_leng
                                      max_longevity_d,maturity_d,female_maturity_d,male_maturity_d,
                                      age_first_reproduction_d,gestation_length_d,teat_number_n,
                                      litter_size_n,litters_per_year_n,interbirth_interval_d,
                                      neonate_mass_g,weaning_age_d,weaning_mass_g,generation_length_d,
                                      dispersal_km,density_n_km2,home_range_km2,social_group_n,
                                      dphy_invertebrate,dphy_vertebrate,dphy_plant,
                                      det_inv,det_vend,det_vect,det_vfish,det_vunk,det_scav,det_fruit,det_
                                      upper_elevation_m,lower_elevation_m,altitude_breadth_m,habitat_bread
                                ~ order+family+genus, data=hostTraits, FUN=median, na.action=na.pass, na.rm
##'na.action=na.pass, na.rm=TRUE' is specified such that if species w/in a genus has a combination of re

#(3) to aggregate binary variables, use the mean as the summary measure
hostTraits$fossoriality[hostTraits$fossoriality==2]<-0  #recode 0/1
hostTraits_binary=aggregate(cbind(hibernation_torpor,fossoriality,freshwater,marine,terrestrial_non.vola
                                  island_dwelling,disected_by_mountains,glaciation) ~ order+family+genus, c

#(4) to aggregate categorical variables transform into binary
hostTraits_cat <- hostTraits
hostTraits_cat$trophic_herbivores <- ifelse(hostTraits_cat$trophic_level==1,1,0)
hostTraits_cat$trophic_omnivores <- ifelse(hostTraits_cat$trophic_level==2,1,0)
hostTraits_cat$trophic_carnivores <- ifelse(hostTraits_cat$trophic_level==3,1,0)
hostTraits_cat$activity_nocturnal <- ifelse(hostTraits_cat$activity_cycle==1,1,0)
hostTraits_cat$activity_crepuscular <- ifelse(hostTraits_cat$activity_cycle==2,1,0) #nocturnal/crepuscu
hostTraits_cat$activity_diurnal <- ifelse(hostTraits_cat$activity_cycle==3,1,0)
hostTraits_cat$forager_marine <- ifelse(hostTraits_cat$foraging_stratum=="M",1,0)
hostTraits_cat$forager_ground <- ifelse(hostTraits_cat$foraging_stratum=="G",1,0)
hostTraits_cat$forager_scansorial <- ifelse(hostTraits_cat$foraging_stratum=="S",1,0)
hostTraits_cat$forager_arboreal <- ifelse(hostTraits_cat$foraging_stratum=="Ar",1,0)
hostTraits_cat$forager_aerial <- ifelse(hostTraits_cat$foraging_stratum=="A",1,0)
hostTraits_cat$island_end_marine <- ifelse(hostTraits_cat$island_endemicity=="Exclusively marine",1,0)
hostTraits_cat$island_end_mainland <- ifelse(hostTraits_cat$island_endemicity=="Occurs on mainland",1,0
hostTraits_cat$island_end_lgbridge <- ifelse(hostTraits_cat$island_endemicity=="Occurs on large land br
##hostTraits_cat$island_end_smbridge <- ifelse(hostTraits_cat$island_endemicity=="Occurs on small land
hostTraits_cat$island_end_isolated <- ifelse(hostTraits_cat$island_endemicity=="Occurs only on isolated
hostTraits_cat$biogeo_afrotropical <- ifelse(grepl("Afrotropical",hostTraits_cat$biogeographical_realm)
hostTraits_cat$biogeo_antarctic <- ifelse(grepl("Antarctic",hostTraits_cat$biogeographical_realm),1,0)
hostTraits_cat$biogeo_australasian <- ifelse(grepl("Australasian",hostTraits_cat$biogeographical_realm)
hostTraits_cat$biogeo_indomalayan <- ifelse(grepl("Indomalayan",hostTraits_cat$biogeographical_realm),1
hostTraits_cat$biogeo_nearctic <- ifelse(grepl("Nearctic",hostTraits_cat$biogeographical_realm),1,0)
hostTraits_cat$biogeo_neotropical <- ifelse(grepl("Neotropical",hostTraits_cat$biogeographical_realm),1
hostTraits_cat$biogeo_oceanian <- ifelse(grepl("Oceanian",hostTraits_cat$biogeographical_realm),1,0)
hostTraits_cat$biogeo_palearctic <- ifelse(grepl("Palearctic",hostTraits_cat$biogeographical_realm),1,0

#(5) to aggregate transformed categorical-to-binary variables, use the mean as the summary measure
hostTraits_cat=aggregate(cbind(trophic_herbivores,trophic_omnivores,trophic_carnivores,
                               activity_nocturnal,activity_crepuscular,activity_diurnal,
```

```
                              forager_marine,forager_ground,forager_scansorial,forager_arboreal,forager_a
                              island_end_marine,island_end_mainland,island_end_lgbridge,island_end_isolat
                              biogeo_afrotropical,biogeo_antarctic,biogeo_australasian,biogeo_indomalayan
                         ~ order+family+genus, data=hostTraits_cat, FUN=mean, na.action=na.pass, na.rm=TRU

#(6) merge continuous variables with binary variables and clean environment
hostTraits <- full_join(hostTraits_continuous, hostTraits_binary, by = c("order","family","genus"),keep=
hostTraits <- rename(hostTraits,c('order.x'='order','family.x'='family','genus.x'='genus'))
hostTraits=subset(hostTraits, select=-c(order.y,family.y,genus.y))

#(7) merge transformed categorical variables and clean environment
hostTraits <- full_join(hostTraits, hostTraits_cat, by = c("order","family","genus"),keep=TRUE)
hostTraits <- rename(hostTraits,c('order.x'='order','family.x'='family','genus.x'='genus'))
hostTraits <- subset(hostTraits, select=-c(order.y,family.y,genus.y))

#(8) clean environment
rm(hostTraits_binary,hostTraits_cat,hostTraits_continuous)
```

Collapse hostTree to genus-level

```
#(1) reformat
hostTree$tip.label[hostTree$tip.label=="_Anolis_carolinensis"] <- "Anolis_carolinensis"

#(2) create dataframe linking tip labels with their corresponding categories (genus and species)
tdata <- data.frame(matrix(NA,nrow=length(hostTree$tip.label),ncol=0))
tdata$genus <- sapply(strsplit(hostTree$tip.label,'_'),function(x) paste(x[1],sep='_'))
tdata$species <- hostTree$tip.label

#(3) collapse tree to genus level
hostTree <- makeCollapsedTree(tree=hostTree,df=tdata[c('genus','species')])

#(4) clean environment
rm(tdata)
```

Check for mismatched genera names in poxdata, hostTraits and hostTree

```
#(1) are all poxdata genera in hostTree?
poxdata$gtip <- poxdata$gen
hostTree$gtip <- hostTree$tip.label
poxdata$intree <- ifelse(poxdata$gtip%in%setdiff(poxdata$gtip,hostTree$gtip),'missing','upham')

#(2) are all poxdata genera in hostTraits?
hostTraits$gtip <- hostTraits$genus
poxdata$intraits <- ifelse(poxdata$gtip%in%setdiff(poxdata$gtip,hostTraits$gtip),'missing','traits')

#(3) create dataframe of just observations with mismatched names
fix <- poxdata[c('gtip','intree','intraits')]
fix <- fix[fix$intree=='missing'|fix$intraits=='missing',]
fix <- unique(fix)
```

```
#(4) identify homotypic synonyms or proxy species via IUCN (https://www.iucnredlist.org/) and NCBI (htt
fix$treename <- NA
fix$traitname <- NA
fix$proxy <- NA
fix$proxy <- ifelse(fix$gtip=="Calassomys","Delomys",fix$proxy)
  ##source: https://academic.oup.com/jmammal/article/95/2/201/860032
fix$traitname <- ifelse(fix$gtip=="Liomys","Heteromys",fix$traitname)
  ##source: https://www.iucnredlist.org/species/40768/22345036
fix$traitname <- ifelse(fix$gtip=="Oreonax","Lagothrix",fix$traitname)
  ##source: https://www.iucnredlist.org/species/39924/192307818
fix$traitname <- ifelse(fix$gtip=="Paralomys","Phyllotis",fix$traitname)
  ##source: https://www.iucnredlist.org/species/17226/22333354
fix$traitname <- ifelse(fix$gtip=="Pearsonomys","Geoxus",fix$traitname)
  ##source: https://www.iucnredlist.org/species/40768/22345036
fix$traitname <- ifelse(fix$gtip=="Pipanacoctomys","Tympanoctomys",fix$traitname)
  ##source: https://www.iucnredlist.org/species/136557/78324400#taxonomy
fix$traitname <- ifelse(fix$gtip=="Pseudalopex","Lycalopex",fix$traitname)
  ##source: https://www.iucnredlist.org/species/6926/87695615
## hostTraits$genus[which(grepl('Tympanoctomys',hostTraits$genus))]

#(5) merge revised names with poxdata
fix <- subset(fix, select=-c(intree,intraits))
poxdata <- merge(poxdata,fix,by='gtip',all.x=T)

#(6) treename will be used for merging poxdata & hostTree
poxdata$treename <- ifelse(poxdata$treename=='',NA,as.character(poxdata$treename))
poxdata$treename <- ifelse(is.na(poxdata$treename),as.character(poxdata$gtip),as.character(poxdata$tree

#(7) traitname will be used for merging poxdata & hostTraits
poxdata$traitname <- ifelse(poxdata$traitname=='',NA,as.character(poxdata$traitname))
poxdata$traitname <- ifelse(poxdata$intraits=='missing' & is.na(poxdata$traitname),as.character(poxdata$
                      ifelse(poxdata$intraits=='missing' & !is.na(poxdata$traitname),as.character(poxda
                             as.character(poxdata$gtip)))

#(8) simplify and clean environment
poxdata <- subset(poxdata, select=-c(intree,intraits,proxy))
rm(fix)
```

Merge poxdata with hostTraits and trim hostTree to mirror poxdata

```
#(2) merge traits with poxdata
hostTraits$traitname <- hostTraits$gtip
poxdata <- merge(poxdata,hostTraits,by=c('traitname'),all.x=T)

#(3) clean up poxdata
poxdata <- rename(poxdata,c('gtip.x'='gtip'))
poxdata <- subset(poxdata,select=-c(order, family, genus,gtip.y))

#(4) trim hostTree to mirror poxdata
hostTree <- keep.tip(hostTree,hostTree$tip.label[hostTree$tip.label%in%poxdata$treename])
hostTree$gtip <- NULL
hostTree=makeLabel(hostTree)
```

```
#(5) clean environment
rm(hostTraits)
```

Add PubMed citations and evolutionary distinctiveness measure

```
#(1) load library for PubMed citations
library(easyPubMed)

#(2) create function to count citations
counter=function(name){
  as.numeric(as.character(get_pubmed_ids(gsub('_','-',name))$Count))
}
citations=c()

#(3) extract unique genera from poxdata
treename <- unique(poxdata$treename)

#(4) apply counter function while looping through treenames
for(i in 1:length(treename)) {
  citations[i]=counter(treename[i])
  print(i)
}

#(5) compile citation numbers
cites <- data.frame(treename=treename,cites=citations)

#(6) merge cites with poxdata
poxdata <- merge(poxdata,cites,by='treename')

#(7) load library for evolutionary distinctiveness (ed) measure
library(picante)  #before loading picante, make sure latest version of nlme package is loaded
ed <- evol.distinct(hostTree,type='equal.splits') #calculates ed measures for a suite of species by equ

#(8) rename variables in ed
ed <- rename(ed,c('Species'='treename','w'='ed_equal'))

#(9) merge ed with poxdata
poxdata <- merge(poxdata,ed,by='treename')

#(10) clean environment
rm(cites,ed,citations,i,treename,counter)

## consider adding viral genome length, viral richness (number of virus detected in each genera), and h
```

Save simple dataset for phylogenetic analysis and Model #1

```
#(1) save poxdata containing only genera of taxonomic orders with known host-OPV associations for phylo
poxdataMin <- subset(poxdata,select=-c(virus))
```

```
#(2) remove duplicate genera: aggregate to genus-level taking the max value of pcr/comp and the sum of
agg_pcr <- aggregate(pcr~gen, data=poxdataMin, max)
agg_competence <- aggregate(competence~gen, data=poxdataMin, max)
agg_studies <- aggregate(studies~gen, data=poxdataMin, sum)

#(3) remove duplicate genera: merge pcr and competence data back in
poxdataMin$pcr=NULL
poxdataMin$competence=NULL
poxdataMin$studies=NULL
poxdataMin <- poxdataMin[!duplicated(poxdataMin$gen),]
poxdataMin <- list(poxdataMin,agg_pcr,agg_competence,agg_studies) %>% reduce(full_join, by='gen')

#(4) clean environment
rm(agg_competence,agg_pcr,agg_studies)
```

Add all possible host-OPV combinations for link prediction model

```
#(1) create separate dataframes for hostTraits and interaction data
hostTraits <- subset(poxdata, select=-c(virus,pcr,competence,studies,sampled))
hostTraits <- hostTraits[!duplicated(hostTraits$gen),]
interactions <- subset(poxdata, select=c(gen,virus,pcr,competence,studies,sampled))

#(1) create dataframe of all possible host-OPV combinations (for mammal genera that exist in orders w/
uniq_gen <- unique(poxdata$gen[!is.na(poxdata$gen)])
uniq_virus <- unique(poxdata$virus[!is.na(poxdata$virus)])
combinations <- expand.grid(uniq_gen,uniq_virus)
combinations <- rename(combinations,c('Var1'='gen','Var2'='virus'))

#(2) merge host-OPV interaction data with all possible combinations
poxdata <- merge(combinations,interactions,by=c("gen","virus"),all.x=TRUE)

#(3) merge host-related data
poxdata <- merge(poxdata,hostTraits,by=c("gen"),all.x=TRUE)

#(6) create binary variable for sampled host-OPV pairs
poxdata$sampled=ifelse(is.na(poxdata$pcr) & is.na(poxdata$competence),0,1)

#(7) reclassify NAs as pseudo-absences for viral detection
poxdata$pcr=ifelse(is.na(poxdata$pcr),0,poxdata$pcr)
poxdata$competence=ifelse(is.na(poxdata$competence),0,poxdata$competence)
poxdata$studies=ifelse(is.na(poxdata$studies),0,poxdata$studies)

#(8) clean environment
rm(hostTraits,interactions,uniq_gen,uniq_virus,combinations)
```

Merge poxdata with viral accessory genes

```r
#(1) simplify data
viralTraits <- head(viralTraits, -2)

#(2) rename column names
viralTraits <- viralTraits[,-2]
colnames(viralTraits) <- paste("ag" ,colnames(viralTraits),sep="_")
names(viralTraits)[1] <- c("virus")

#(3) to assess variation in viralTraits, create mode function
mode.prop <- function(x) {
  ux <- unique(x[is.na(x)==FALSE])        # creates array of unique values
  tab <- tabulate(match(na.omit(x), ux))  # creates array of the frequency a unique value appears in a
  max(tab)/length(x[is.na(x)==FALSE])     # max-frequency / number of elements in each column that are
}

#(4) assess variation across columns (2 indicates columns)
vars=data.frame(apply(viralTraits,2,function(x) mode.prop(x)),
                apply(viralTraits,2,function(x) length(unique(x)))) # number of unique elements in each
vars$variables=rownames(vars)
colnames(vars) <- c("var","uniq","column")

## trim
#vars <- vars[-c(1,2), ]

#(5) drop variables with no variation
vars <- subset(vars,vars$var<1)

# ## visualize distribution of NA
# png("/Users/katietseng/Downloads/virus_ag_variation.png", width=4,height=4,units="in",res=600)
# ggplot(vars,
#        aes(var))+
#   geom_histogram(bins=50)+
#   geom_vline(xintercept=0.70,linetype=2,size=0.5)+
#   theme_bw()+
#   theme(panel.grid.major=element_blank(),panel.grid.minor=element_blank())+
#   theme(axis.title.x=element_text(margin=margin(t=10,r=0,b=0,l=0)))+
#   theme(axis.title.y=element_text(margin=margin(t=0,r=10,b=0,l=0)))+
#   labs(y="frequency",
#        x="trait coverage across viral species")+
#   scale_x_continuous(labels=scales::percent)
# dev.off()

# ## drop based on threshold
# vars$keep=ifelse(vars$var>=0.7,"keep","cut")
# keeps=vars[-which(vars$keep=="cut"),]$column
# keeps <- append("virus",keeps)
# viralTraits=viralTraits[keeps]

#(6) edit virus names
viralTraits$virus_new <- NA
viralTraits$virus_new <- ifelse(grepl("Abatino",viralTraits$virus)==TRUE,"abatino macacapox virus",viral
viralTraits$virus_new <- ifelse(grepl("Akhmeta",viralTraits$virus)==TRUE,"akhmeta virus",viralTraits$vi
viralTraits$virus_new <- ifelse(grepl("Alaskapox",viralTraits$virus)==TRUE,"alaskapox virus",viralTrait
```

```
viralTraits$virus_new <- ifelse(grepl("Camelpox",viralTraits$virus)==TRUE,"camelpox virus",viralTraits$v
viralTraits$virus_new <- ifelse(grepl("Cetacean poxvirus 1",viralTraits$virus)==TRUE,"cetacean poxvirus
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"cetacean poxvirus 2",viralTraits$v
viralTraits$virus_new <- ifelse(grepl("Cowpox",viralTraits$virus)==TRUE,"cowpox virus",viralTraits$virus
viralTraits$virus_new <- ifelse(grepl("Ectromelia",viralTraits$virus)==TRUE,"ectromelia virus",viralTrai
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"feline poxvirus ita2_bc",viralTrai
viralTraits$virus_new <- ifelse(grepl("Monkeypox",viralTraits$virus)==TRUE,"monkeypox virus",viralTraits
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"orthopoxvirus gcp2010",viralTraits
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"orthopoxvirus gcp2013",viralTraits
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"orthopoxvirus sp.",viralTraits$vir
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"orthopoxvirus tena dona",viralTrai
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"raccoonpox virus",viralTraits$viru
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"skunkpox virus",viralTraits$virus_
# viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"steller sea lion poxvirus",viralTr
viralTraits$virus_new <- ifelse(grepl("Taterapox",viralTraits$virus)==TRUE,"taterapox virus",viralTraits
viralTraits$virus_new <- ifelse(grepl("Vaccinia",viralTraits$virus)==TRUE,"vaccinia virus",viralTraits$v
#viralTraits$virus_new <- ifelse(grepl("",viralTraits$virus)==TRUE,"volepox virus virus",viralTraits$vi

#(7) reformat and drop dups
viralTraits$virus <- viralTraits$virus_new
viralTraits$virus_new = NULL
viralTraits <- subset(viralTraits,!is.na(viralTraits$virus))
viralTraits <- viralTraits[!duplicated(viralTraits$virus),]

#(8) identify rows with duplicate values (i.e., hosts with identical presence/absence of accessory gene
which(duplicated(viralTraits[,-c(1)])| duplicated(viralTraits[,-c(1)], fromLast = TRUE))
viralTraits$dup <- duplicated(viralTraits[,-c(1)])

#(9) merge with poxdata; full join returns only rows found in both poxdata and viralTraits
poxdata <- merge(poxdata,viralTraits,by=c('virus'))

#(10) clean environment
rm(viralTraits,vars,keeps,original_cols,mode.prop)
```

Save cleaned data

```
poxdataMin <- poxdataMin %>%
  relocate(gen,fam,ord,gtip,treename,traitname,pcr,competence,studies,sampled,cites,ed_equal)

poxdata <- poxdata %>%
  relocate(virus,gen,fam,ord,gtip,treename,traitname,pcr,competence,studies,sampled,cites,ed_equal)

save(poxdataMin, poxdata, hostTree, file='/Users/katietseng/Downloads/Data_clean.RData')
save(poxdata, file='/Users/katietseng/Downloads/poxdata_temp.RData')
```

Phylogenetic analysis

Load required packages and set system

```
#(1) libraries for phylogenetic analysis
library(ape)
library(caper)
library(data.table)
library(BiocManager)  ## BiocManager::install(c("Biostrings","ggtree"))
library(phylofactor)  ## devtools::install_github('reptalex/phylofactor'); more info at: https://reptal
library(treeio)       ## BiocManager::install("treeio")
library(ggtree)

#(2) clean environment
rm(list=ls())
graphics.off()

#(3) set working directory
setwd("~/Library/CloudStorage/OneDrive-WashingtonStateUniversity(email.wsu.edu)/Fernandez Lab/Projects
```

Phylogenetic patterns

```
#(1) load data and trim unnecessary columns
load("Data_clean.RData")
data <- poxdataMin

#(2) check that genus name in poxdata is also in hostTree
which(data$treename%in%setdiff(data$treename,hostTree$tip.label))

#(3) create variables label and Species (required in later functions)
data$label <- data$treename
data$Species <- data$treename

#(4) merge phylogeny w/ data ensuring consistent structure & ordering (caper::comparative.data)
cdata=comparative.data(phy=hostTree,data=data,names.col=treename,vcv=T,na.omit=F,warn.dropped=T)
cdata$data$tree=NULL

#(5) what proportion of genera have evidence of infection?
nrow(data)
count(data$pcr==1)
round(prop.table(table(data$pcr)),4)*100
count(data$competence==1)
round(prop.table(table(data$competence)),4)*100
##values in each cell divided by the sum of the 4 cells

#(6) Does the raw data display a phylogenetic signal in response?
## D of 0 = Brownian model, D of 1 = random (no phylogenetic signal)
set.seed(1)
mod1 <- phylo.d(cdata,binvar=pcr,permut=10000); mod1
set.seed(1)
mod2 <- phylo.d(cdata,binvar=competence,permut=10000); mod2
```

Phylofactorization

```
#(1) create dataframe of taxonomy
cdata$data$taxonomy=paste(cdata$data$ord,cdata$data$fam,cdata$data$gen,sep='; ')
taxonomy <- data.frame(cdata$data$taxonomy)
names(taxonomy) <- "taxonomy"
taxonomy$Species <- rownames(cdata$data)
taxonomy <- taxonomy[c("Species","taxonomy")]
taxonomy$taxonomy <- as.character(taxonomy$taxonomy)

#(2) Holm rejection procedure: pf=phylofactor and FWER=family-wise error rate (alpha .05)
HolmProcedure <- function(pf,FWER=0.05){
  ## get split variable
  cs=names(coef(pf$models[[1]]))[-1]
      ### returns names of model coefficients (var names) extracted by 'coef' in
      ### the 1st list element of 'pf$models' minus the 1st element among those
      ### names; double brackets access a list element
  split=ifelse(length(cs)>1,cs[3],cs[1])
      ### returns 3rd element in 'cs' if length of the number of elements in
      ### 'cs' >1; else returns 1st element

  ## obtain p values
  if (pf$models[[1]]$family$family%in%c('gaussian',"Gamma","quasipoisson")){
      ### if fam$fam of 1st list element of pf$models is in columns 'gaussian'...
    pvals <- sapply(pf$models,FUN=function(fit) summary(fit)$coefficients[split,'Pr(>|t|)'])
      ### then to each element of pf$models, apply summary function w/ argument
      ### 'fit' and assign output to 'pvals';
      ### specifically, we use 'summary(fit)' to call the output of 'pf$models',
      ### extracting the 'coefficients' section, whereby we index the column
      ### named 'Pr(>|t\)' and split the data in that column; see sample output
      ### of linear model of R for reference (https://feliperego.github.io/blog/2015/10/23/Interpreting
  } else {
    pvals <- sapply(pf$models,FUN=function(fit) summary(fit)$coefficients[split,'Pr(>|z|)'])
      ### else extract p-val based on z statistic
  }
  D <- length(pf$tree$tip.label)
      ### returns number of elements in pf$tree$tip.label

  ## this is the line for Holm's sequentially rejective cutoff, where HB = Target alpha / (n - rank + 1
  keepers <- pvals<=(FWER/(2*D-3 - 2*(0:(pf$nfactors-1))))
      ### returns TRUE/FALSE if p-values are <= to 0.05/(n-rank+1)

  if (!all(keepers)){
      ### if not all pvals were keepers (i.e., all items in keepers were true)...
    nfactors <- min(which(!keepers))-1
      ### then assign nfactors to minimum/earliest position of items in keepers that were false, minus
  } else {
    nfactors <- pf$nfactors
      ###:else, assign nfactors as the value of pf$nfactors
  }
  return(nfactors)
}
```

```
## get species in a clade
cladeget=function(pf,factor){
  ### creates function 'cladeget' w/ arguments 'pf' and 'factor'
  spp=pf$tree$tip.label[pf$groups[[factor]][[1]]]
    ### returns n'th element of the pf$tree$tip.label based on the value of
    ### the first component inside the n'th ('factor') component of  'pf$groups'
  return(spp)
}

#(3) summarize pf object
pfsum=function(pf){

  ## get formula
  chars=as.character(pf$frmla.phylo)[-1]  ### returns pf$frmla.phylo minus 1st element

  ## response
  resp=chars[1]               ###returns 1st element of chars

  ## holm
  hp=HolmProcedure(pf)

  ## save model
  model=chars[2]

  ## set key
  setkey(pf$Data,'Species')   ### creates key on sorted pf$Datacolumn 'Species'

  ## make data
  dat=data.frame(pf$Data)

  ## make clade columns in data
  for(i in 1:hp){

    dat[,paste0(resp,'_pf',i)]=ifelse(dat$Species%in%cladeget(pf,i),'factor','other')
    ### paste0 concatenates all elements w/o a separator

  }

  ## make data frame to store taxa name, response, mean, and other
  results=data.frame(matrix(ncol=6, nrow = hp))
  colnames(results)=c('factor','taxa','tips','node',"clade",'other')

  ## set taxonomy
  taxonomy=dat[c('Species','taxonomy')]
  taxonomy$taxonomy=as.character(taxonomy$taxonomy)

  ## loop
  for(i in 1:hp){

    ## get taxa
    tx=pf.taxa(pf,taxonomy,factor=i)$group1            #gets taxonomic order

    ## get tail
```

```
    tx=sapply(strsplit(tx,'; '),function(x) tail(x,1)) #gets tax family as list

    ## combine
    tx=paste(tx,collapse=', ')         #collapses tax family into single string

    # save
    results[i,'factor']=i         #returns index number in 'factor' column
    results[i,'taxa']=tx          #returns string element (tx) in 'taxa' column

    ## get node
    tips=cladeget(pf,i)
    node=ggtree::MRCA(pf$tree,tips)
    ### MRCA = finds Most Recent Common Ancestor among a vector of tips
    results[i,'tips']=length(tips)
    results[i,'node']=ifelse(is.null(node) & length(tips)==1,'species',
                             ifelse(is.null(node) & length(tips)!=1,NA,node))

    ## get means
    ms=(tapply(dat[,resp],dat[,paste0(resp,'_pf',i)],FUN=mean))
    ### tapply takes mean of '1 vs. 0' (dat[,resp]) by 'other'/'factor' type (dat[,paste...]

    ## add in
    results[i,'clade']=ms['factor']
    results[i,'other']=ms['other']

  }

  ## return
  return(list(set=dat,results=results))       #returns number of clades with significantly greater prop
}

#(4) phylofactorization of infection data
set.seed(1)
pcr_pf=gpf(Data=cdata$data,tree=cdata$phy,
           frmla.phylo=pcr~phylo,
           family=binomial,algorithm='phylo',nfactors=10,min.group.size=5)

#(5) summarize infection PF results
HolmProcedure(pcr_pf)
pcr_pf_results=pfsum(pcr_pf)$results

#(6) phylofactorization of competence data
set.seed(1)
hc_pf=gpf(Data=cdata$data,tree=cdata$phy,
          frmla.phylo=competence~phylo,
          family=binomial,algorithm='phylo',nfactors=2,min.group.size=5)

#(7) summarize competence PF results
HolmProcedure(hc_pf)
hc_pf_results=pfsum(hc_pf)$results
```

Plot results of phylofactorization

```
#(1) save tree for plotting
cdata$data$infect=factor(cdata$data$pcr)
cdata$data$comp=factor(cdata$data$competence)
dtree=treeio::full_join(as.treedata(cdata$phy),cdata$data,by="label")

#(2) fix palette
AlberPalettes <- c("YlGnBu","Reds","BuPu", "PiYG")
AlberColours <- sapply(AlberPalettes, function(a) RColorBrewer::brewer.pal(5, a)[4])
afun=function(x){
  a=AlberColours[1:x]
  return(a)
}

#(3) make low and high, and set x max
pcols=afun(2)
plus=1
pplus=plus+1

#(4) fix taxa font formatting
pcr_pf_results$taxa
pcr_pf_results$taxa[1]="Rodentia"
hc_pf_results$taxa
hc_pf_results$taxa[1]="italic(Felidae)"

#(5) plot pcr infection w/ ggtree
pcr_gg=ggtree(dtree,size=0.25)+
  geom_tippoint(aes(colour=infect),shape=15)+
  scale_colour_manual(values=c("grey80","black"))+
  guides(colour="none")

#(6) add clades to plot
for(i in 1:nrow(pcr_pf_results)){

  pcr_gg=pcr_gg+
    geom_hilight(node=pcr_pf_results$node[i],
                 alpha=0.25,
                 fill=ifelse(pcr_pf_results$clade>
                               pcr_pf_results$other,pcols[2],pcols[1])[i])+
    geom_cladelabel(node=pcr_pf_results$node[i],
                    label=pcr_pf_results$taxa[i],
                    offset=pplus,
                    hjust=0.75,
                    offset.text=pplus*2,
                    parse=T,
                    angle=90)
}
pcr_gg=pcr_gg

#(7) plot competence
comp_gg=ggtree(dtree,size=0.25)+
  geom_tippoint(aes(colour=comp),shape=15)+
```

```
  scale_colour_manual(values=c("grey80","black"))+
  guides(colour=F)

#(8) add clades to plot
for(i in 1:nrow(hc_pf_results)){

  comp_gg=comp_gg+
    geom_hilight(node=hc_pf_results$node[i],
                 alpha=0.25,
                 fill=ifelse(hc_pf_results$clade>
                               hc_pf_results$other,pcols[2],pcols[1])[i])+
    geom_cladelabel(node=hc_pf_results$node[i],
                    label=hc_pf_results$taxa[i],
                    offset=pplus,
                    hjust=0.75,
                    offset.text=pplus*2,
                    parse=T,
                    angle=90)
}
comp_gg=comp_gg

#(9) print tree figures for infection and competence
library(ggpubr)
png("Results/Figure 1.png",width=6,height=6,units="in",res=300)
ggarrange(pcr_gg,comp_gg,ncol=2,widths=c(1.2,1),
          labels=c("(a) RT-PCR","(b) virus isolation"),
          label.x=c(-0.1,-0.2),
          font.label=list(face="plain",size=12))
dev.off()
```

Additional phylofactorization models

```
#(1) log1p pubmed cites
cdata$data$logcites=log1p(cdata$data$cites)

#(2) model PCR with pubmed cites as weight variable
set.seed(1)
pcr_pf_pm=gpf(Data=cdata$data,tree=cdata$phy,
                frmla.phylo=pcr~phylo,
                weights=cdata$data$logcites,
                family=binomial,algorithm='phylo',nfactors=10,min.group.size=5)

#(3) summarize
HolmProcedure(pcr_pf_pm)
pcr_pf_pm_results=pfsum(pcr_pf_pm)$results

#(4) model competence with pubmed cites as weight variable
set.seed(1)
hc_pf_pm=gpf(Data=cdata$data,tree=cdata$phy,
               frmla.phylo=competence~phylo,
               weights=cdata$data$logcites,
               family=binomial,algorithm='phylo',nfactors=10,min.group.size=5)
```

```
#(5) summarize
HolmProcedure(hc_pf_pm)
hc_pf_pm_results=pfsum(hc_pf_pm)$results

#(6) model cites themselves (not log1pm-transformed)
set.seed(1)
pm_pf=gpf(Data=cdata$data,tree=cdata$phy,
             frmla.phylo=cites~phylo,
             family=poisson,algorithm='phylo',nfactors=10,min.group.size=5)
HolmProcedure(pm_pf)
pm_pf_results=pfsum(pm_pf)$results
```

Boosted regression trees (BRT)

Load required packages and set system

```
#(1) libraries for BRT model
library(gbm)
library(fastDummies)
library(rsample)
library(ROCR)
library(sciplot)
library(ggplot2)
library(pdp)
library(PresenceAbsence)
library(tidyr)
library(viridis)
library(caper)
library(phylofactor)
library(ggtree)
library(treeio)
library(caret)
library(InformationValue)
library(mgcv)

#(2) clean environment
rm(list=ls())
graphics.off()
```

MODEL 1: Load data and set working directory for appropriate model

```
load("~/Library/CloudStorage/OneDrive-WashingtonStateUniversity(email.wsu.edu)/Fernandez Lab/Projects (A
data <- poxdataMin
rm(poxdata,poxdataMin)
setwd("/Users/katietseng/Downloads/Results/Model 1")
```

MODEL 2: Load data and set working directory for appropriate model

```
#(1) load data and clean environment
load("/Users/katietseng/Downloads/poxdata_temp.RData")
data <- poxdata
setwd("/Users/katietseng/Downloads/Results/Model 2")

#(2) ensure all accessory gene vars are numeric
ag_columns <- colnames(data[which(grepl("ag_",names(data)))])
data[,c(ag_columns)] <- lapply(data[c(ag_columns)],as.numeric)
```

Create taxonomic variables as predictors for the model

```
#(1) classify true negatives
data$type=ifelse(data$pcr==0 & data$competence==0,"true negative","other")

#(2) which species is competent but no PCR record?
set=data
set$treename[set$pcr==0 & set$competence==1]

#(3) tabulate PCR/infection and isolation
set$inf=ifelse(set$pcr==0,"PCR negative","PCR positive")
set$iso=ifelse(set$competence==0,"no isolation","isolation")
table(set$inf,set$iso)

#(4) make binary variables for each taxonomic family; remove any duplicates
dums=dummy_cols(data["fam"])
dums=dums[!duplicated(dums$fam),]

#(5) ensure all family vars are factor
for(i in 1:ncol(dums)){
  dums[,i]=factor(dums[,i])
}

#(6) merge family taxa variables with dataset as predictors
data=merge(data,dums,by="fam",all.x=T)

#(7) drop unnecessary columns and clean environment
data$traitname=NULL
rm(dums,set,ag_columns)
```

Assess variation and availability of data

```
#(1) mode function
mode.prop <- function(x) {
  ux <- unique(x[is.na(x)==FALSE])        # creates array of unique values
  tab <- tabulate(match(na.omit(x), ux))  # creates array of the frequency (number of times) a unique v
  max(tab)/length(x[is.na(x)==FALSE])     # max-frequency / number of elements in each column that are
}
```

```r
#(2) assess variation across columns (2 indicates columns)
vars=data.frame(apply(data,2,function(x) mode.prop(x)),
                apply(data,2,function(x) length(unique(x))))    # number of unique elements in each col

#(3) get names
vars$variables=rownames(vars)
names(vars)=c("var","uniq","column")

# ## round values
# vars$var=round(vars$var,2)

#(4)label variables "cut" if homogeneous (100%)
vars$keep=ifelse(vars$var<1,"keep","cut")
vars$keep=ifelse(vars$column%in%c('fam','virus','gen','pcr','competence','fam'),'keep',vars$keep) # ens
vars=vars[order(vars$keep),]

#(5) trim (creates array of column names to cut and removes from df)
keeps=vars[-which(vars$keep=="cut"),]$column

#(6) drop if no variation
data=data[keeps]
rm(keeps,vars)

#(7) assess missing values
mval=data.frame(apply(data,2,function(x) length(x[!is.na(x)])/nrow(data))) # proportion of values that

#(8) get names
mval$variables=rownames(mval)
names(mval)=c("comp","column")
#
# #(9) visualize distribution of NA
# png("Figure S1.png", width=4,height=4,units="in",res=600)
# ggplot(mval[!mval$column%in%c("gen","treename","pcr","competence","tip.label","fam"),],
#        aes(comp))+
#   geom_histogram(bins=50)+
#   geom_vline(xintercept=0.70,linetype=2,size=0.5)+
#   theme_bw()+
#   theme(panel.grid.major=element_blank(),panel.grid.minor=element_blank())+
#   theme(axis.title.x=element_text(margin=margin(t=10,r=0,b=0,l=0)))+
#   theme(axis.title.y=element_text(margin=margin(t=0,r=10,b=0,l=0)))+
#   labs(y="frequency",
#        x="trait coverage across mammal species (genus)")+
#   scale_x_continuous(labels = scales::percent)
# dev.off()

#(10) label variables "cut" if >30% values are NA
mval$keep=ifelse(mval$comp>=0.70,"keep","cut")
table(mval$keep)
mval=mval[order(mval$keep),]

#(11) trim (creates array of column names to cut and removes from df)
keeps=mval[-which(mval$keep=="cut"),]$column
```

```
#(12) drop if not well represented
data=data[keeps]
rm(keeps,mval)

#(14) save list of covariates and their coverage as table S1
set <- subset(data,select=-c(virus,gen,fam,ord,gtip,treename,type,studies,sampled))
ts1=data.frame(apply(set,2,function(x) length(x[!is.na(x)])/nrow(set)))

#(15) rename and reorder columns
ts1$variables=rownames(ts1)
names(ts1)=c("coverage","feature")
rownames(ts1)=NULL
ts1=ts1[!ts1$feature%in%c("pcr","competence"),]
ts1 <- subset(ts1,select=c(feature,coverage))
#
# #(16) save Table S1 to results
# write.csv(ts1, "TableS1.csv")

#(17) check that binary variables are numeric and not factor
str(set)
```

Temp: create simple version w/o accessory genes

```
set <- subset(set, select=-c(which(grepl("ag_",colnames(set)))))
setwd("/Users/katietseng/Downloads/Results/Test")
```

Model tuning function [hfit]: assesses model performance for each combination of tuning parameters with

```
#(1) hyperparameter tuning ifelse
#hok="ok"
hok="notok"
if(hok!="ok"){

  ## hyperparameter grid
  hgrid=expand.grid(n.trees=5000,                                  #creates df from all combinations of fac
                    interaction.depth=c(2,3,4),
                    shrinkage=c(0.01,0.001,0.0005),
                    n.minobsinnode=4,
                    seed=seq(1,10,by=1))
  # hgrid=expand.grid(n.trees=500,                                  #creates df from all combinations of fa
  #                   interaction.depth=c(2,3,4),
  #                   shrinkage=c(0.1,0.01,0.005),
  #                   n.minobsinnode=4,
  #                   seed=seq(1,10,by=1))
  # fix trees
  hgrid$n.trees=ifelse(hgrid$shrinkage<0.001,hgrid$n.trees*3,hgrid$n.trees)

  ## trees, depth, shrink, min, prop
```

```r
hgrid$id=with(hgrid,paste(n.trees,interaction.depth,shrinkage,n.minobsinnode))   #creates var 'id' co

## sort by id then seed
hgrid=hgrid[order(hgrid$id,hgrid$seed),]

## now add rows
hgrid$row=1:nrow(hgrid)                                          #adds var 'row' based on row number in

## factor id
hgrid$id2=factor(as.numeric(factor(hgrid$id)))                   #creates 9-level factor var 'id2'


## function to assess each hyperpar combination
hfit=function(row,response){

  ## make new data
  ndata=set

  ## correct response
  ndata$response=ndata[response][,1]                             #creates var 'response'

  ## remove raw
  ndata$pcr=NULL
  ndata$competence=NULL

  ## use rsample to split
  set.seed(hgrid$seed[row])                                      #sets seed value of 1-10
  split=initial_split(ndata,prop=0.7,strata="response")          #creates single binary split of data i

  ## test and train
  dataTrain=training(split)
  dataTest=testing(split)

  ## yTest and yTrain
  yTrain=dataTrain$response                                      #create array of just response values
  yTest=dataTest$response

  ## BRT
  set.seed(1)
  gbmOut=gbm(response ~ . ,data=dataTrain,                       #y~x; gbmOut contains list of 29 eleme
             n.trees=hgrid$n.trees[row],                         #total number of trees to fit (number
             distribution="bernoulli",
             shrinkage=hgrid$shrinkage[row],                     #equiv to learning rate or step-size r
             interaction.depth=hgrid$interaction.depth[row],     #max depth of each tree (highest level
             n.minobsinnode=hgrid$n.minobsinnode[row],           #min. number of obs in terminal nodes
             cv.folds=5,class.stratify.cv=TRUE,                  #no. of cross-val folds to perform; fo
             bag.fraction=0.5,train.fraction=1,                  #fraction of training set obs randomly
             n.cores=5,                                          #no. of CPU cores to use
             verbose=F)
             # par.details=(gbmParallel(num_threads=5)),

  ## performance
  par(mfrow=c(1,1),mar=c(4,4,1,1))                               #sets graphical parameters such that s
```

```
  best.iter=gbm.perf(gbmOut,method="cv")                          #estimates optimal number of boosting

  ## predict with test data
  preds=predict(gbmOut,dataTest,n.trees=best.iter,type="response")  #number of trees based on the opt

  ## known
  result=dataTest$response

  # ##estimate threshold value for classification of predicted probability
  # #library(pROC)
  # analysis <- roc(result,preds)  #roc([actual values],[predicted values])
  # e <- cbind(analysis$thresholds,analysis$sensitivities+analysis$specificities) #pulls each array a
  #
  # ##optimum threshold value
  # opt_t <- subset(e,e[,2]==max(e[,2]))[,1] #subsets dataframe and returns the max (sens+spec) value
  # #threshold<-opt_t #set as threshold value
  # #threshold = 0.2

  ## sensitivity and specificity                                  #e.g., test run produced sensitivity of
  sen=InformationValue::sensitivity(result,preds)                  #calculates sensitivity (# of obs with
  spec=InformationValue::specificity(result,preds)                 #calculates specificity (# of obs w/o

  ## AUC on train
  auc_train=gbm.roc.area(yTrain,predict(gbmOut,dataTrain,n.trees=best.iter,type="response"))   #compu

  ## AUC on test
  auc_test=gbm.roc.area(yTest,predict(gbmOut,dataTest,n.trees=best.iter,type="response"))

  ## print
  print(paste("hpar row ",row," done; test AUC is ",auc_test,sep=""))  #prints "hpar row [x] done; te

  ## save outputs
  return(list(best=best.iter,                       #saves optimal number of iterations, AUC on training
              trainAUC=auc_train,
              testAUC=auc_test,
              spec=spec,
              sen=sen,
              wrow=row))
}

## run the function for PCR
hpars=lapply(1:nrow(hgrid),function(x) hfit(x,response="pcr"))

## get results
hresults=data.frame(sapply(hpars,function(x) x$trainAUC),
                    sapply(hpars,function(x) x$testAUC),
                    sapply(hpars,function(x) x$spec),
                    sapply(hpars,function(x) x$sen),
                    sapply(hpars,function(x) x$wrow),
                    sapply(hpars,function(x) x$best))
names(hresults)=c("trainAUC","testAUC",
                  "spec","sen","row","best")
```

```
  ## combine and save
  hsearch=merge(hresults,hgrid,by="row")

  ## save
  hsearch$type="PCR"

  ## rerun the function for competence
  hpars=lapply(1:nrow(hgrid),function(x) hfit(x,response="competence"))

  ## get results
  hresults=data.frame(sapply(hpars,function(x) x$trainAUC),
                      sapply(hpars,function(x) x$testAUC),
                      sapply(hpars,function(x) x$spec),
                      sapply(hpars,function(x) x$sen),
                      sapply(hpars,function(x) x$wrow),
                      sapply(hpars,function(x) x$best))
  names(hresults)=c("trainAUC","testAUC",
                    "spec","sen","row","best")

  ## combine and save
  csearch=merge(hresults,hgrid,by="row")

  ## assign data type
  csearch$type="competence"

  ## combine
  search=rbind.data.frame(csearch,hsearch)
  search$type=factor(search$type,levels=c("PCR","competence"))

  ## export
  write.csv(search,"par tuning data summary.csv")

}else{

  ## load
  search=read.csv("par tuning data summary.csv")

}
```

Model tuning results: Figure S2

```
#(1) factor parameters
search$shrinkage=factor(search$shrinkage)
lvl=rev(sort(unique(search$shrinkage)))  #sorts unique shrinkage parameters in order of largest to smal
search$shrinkage=factor(search$shrinkage,levels=lvl); rm(lvl)  #applies as factor

#(2) factor other
search$interaction.depth=factor(search$interaction.depth)

#(3) fix type
search$type=plyr::revalue(search$type,    #replace specified values w/ new values
                          c("PCR"="RT-PCR",
```

```
                              "competence"="virus isolation"))

#(4) PCR beta regression for AUC
mod=gam(testAUC~interaction.depth*shrinkage,   #gam: Generalized additive models with integrated smooth
        data=search[search$type=="RT-PCR",],method="REML",family=betar)
anova(mod)

#(5) competence beta regression for AUC
mod=gam(testAUC~interaction.depth*shrinkage,
        data=search[search$type=="virus isolation",],method="REML",family=betar)
anova(mod)

#(6) PCR beta regression for sensitivity
mod=gam(sen~interaction.depth*shrinkage,
        data=search[search$type=="RT-PCR",],method="REML",family=betar)
anova(mod)

#(7) competence beta regression for sensitivity
mod=gam(sen~interaction.depth*shrinkage,
        data=search[search$type=="virus isolation",],method="REML",family=betar)
anova(mod)


#(8) PCR beta regression for specificity
mod=gam(spec~interaction.depth*shrinkage,
        data=search[search$type=="RT-PCR",],method="REML",family=betar)
anova(mod)

#(9) competence beta regression for specificity
mod=gam(spec~interaction.depth*shrinkage,
        data=search[search$type=="virus isolation",],method="REML",family=betar)
anova(mod)

#(10) recast from wide to long
search2=gather(search,measure,value,testAUC:sen)

#(11) revalue and factor (relabel values and change to factor)
search2$measure=plyr::revalue(search2$measure,
                              c("sen"="sensitivity",
                                "spec"="specificity",
                                "testAUC"="test AUC"))
search2$measure=factor(search2$measure,
                       levels=c("test AUC","sensitivity","specificity"))

#(12) visualize - Figure S2
png("Figure S2.png",width=5,height=8,units="in",res=600)
set.seed(1)
ggplot(search2,aes(shrinkage,value,
                   colour=interaction.depth,fill=interaction.depth))+
  geom_boxplot(alpha=0.25)+
  geom_point(alpha=0.75,
             position = position_jitterdodge(dodge.width=0.75))+
  theme_bw()+
```

```
  theme(panel.grid.major=element_blank(),panel.grid.minor=element_blank())+
  theme(axis.title.x=element_text(margin=margin(t=10,r=0,b=0,l=0)))+
  theme(axis.title.y=element_text(margin=margin(t=0,r=10,b=0,l=0)))+
  facet_grid(measure~type,scales="free_y",switch="y")+
  theme(strip.placement="outside",
        strip.background=element_blank())+
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12),
        strip.text=element_text(size=12))+
  theme(legend.position="top")+
  scale_color_brewer(palette="Pastel2")+
  scale_fill_brewer(palette="Pastel2")+
  guides(colour=guide_legend(title="interaction depth"),
         fill=guide_legend(title="interaction depth"))+
  labs(y=NULL,
       x="learning rate")+
  scale_y_continuous(n.breaks=4)
dev.off()

#(11) clean
rm(search,search2,hok,mod)
```

```
## brt function to use different data partitions
brt_part=function(seed,response){

  ## make new data
  ndata=set

  ## correct response
  ndata$response=ndata[response][,1]

  ## remove raw
  ndata$pcr=NULL
  ndata$competence=NULL

  ## fix cites if response
  if(response=="cites"){

    ## plus 1 for 0
    ndata$cites=ifelse(ndata$cites==0,1,ndata$cites)

  }else{

    ndata=ndata

  }

  ## use rsample to split
  set.seed(seed)
  split=initial_split(ndata,prop=0.7,strata="response")
```

```
## test and train
dataTrain=training(split)
dataTest=testing(split)

## yTest and yTrain
yTrain=dataTrain$response
yTest=dataTest$response

## dist
dist=ifelse(response=="cites","poisson","bernoulli")

## n.trees
nt=ifelse(response=="cites",10000,5000)

## BRT
set.seed(1)
gbmOut=gbm(response ~ . ,data=dataTrain,
           n.trees=nt,
           distribution=dist,
           shrinkage=0.001,
           interaction.depth=3,
           n.minobsinnode=4,
           cv.folds=5,class.stratify.cv=TRUE,
           bag.fraction=0.5,train.fraction=1,
           n.cores=5,
           verbose=F)
          # par.details=(gbmParallel(num_threads=5)),

## performance
par(mfrow=c(1,1),mar=c(4,4,1,1))
best.iter=gbm.perf(gbmOut,method="cv")  #estimates optimal number of boosting iterations for a gbm ob

## predict with test data
preds=predict(gbmOut,dataTest,n.trees=best.iter,type="response")

## known
result=dataTest$response

## sensitivity and specificity
sen=InformationValue::sensitivity(result,preds)
spec=InformationValue::specificity(result,preds)

## AUC on train
auc_train=gbm.roc.area(yTrain,predict(gbmOut,dataTrain,n.trees=best.iter,type="response"))

## AUC on test
auc_test=gbm.roc.area(yTest,predict(gbmOut,dataTest,n.trees=best.iter,type="response"))

## skip if poisson
if(response=="cites"){

  perf=NA
```

```
}else{

  ## inner loop if yTest is all 0
  if(var(yTest)==0){

    perf=NA
  }else{

    ## ROC
    pr=prediction(preds,dataTest$response)
    perf=performance(pr,measure="tpr",x.measure="fpr")         #pr=prediction object; measure=perform
    perf=data.frame(perf@x.values,perf@y.values)
    names(perf)=c("fpr","tpr")

    ## add seed
    perf$seed=seed

  }
}

## relative importance
bars=summary(gbmOut,n.trees=best.iter,plotit=F)
# bars$rel.inf=round(bars$rel.inf,2)
bars$rel_inf=round(bars$rel_inf,2)

## predict with cites
preds=predict(gbmOut,data,n.trees=best.iter,type="response")
pred_data=data[c("gtip",'treename',"fam","ord","pcr","competence")]
pred_data$pred=preds
pred_data$type=response

## predict with mean cites
pdata=data
pdata$cites=mean(pdata$cites)
pred_data$cpred=predict(gbmOut,pdata,n.trees=best.iter,type="response")

## sort
pred_data=pred_data[order(pred_data$pred,decreasing=T),]

## print
print(paste("BRT ",seed," done; test AUC = ",auc_test,sep=""))

## save outputs
return(list(mod=gbmOut,
            best=best.iter,
            trainAUC=auc_train,
            testAUC=auc_test,
            spec=spec,
            sen=sen,
            roc=perf,
            rinf=bars,
            predict=pred_data,
            traindata=dataTrain,
```

```
              testdata=dataTest,
              seed=seed))
}
```

```
## apply across 100 splits each
# smax=101
smax=100
pcr_brts=lapply(1:smax,function(x) brt_part(seed=x,response="pcr"))
comp_brts=lapply(1:smax,function(x) brt_part(seed=x,response="competence"))
#
# ## run wos brts
# pm_brts=lapply(1:(smax-1),function(x) brt_part(seed=x,response="cites"))

## save results to wd
# save(pcr_brts,comp_brts,pm_brts,file="Data_results.RData")
save(pcr_brts,comp_brts,file="Data_results.RData")
```

Principal components analysis of viral accessory genes

Load required packages and set system

```
#(1) libraries for PCA
library(ape)
library(vegan)
library(dplyr)
library(factoextra) #fviz_eig
#(2) clean environment
rm(list=ls())
graphics.off()
```

Format data for PCA

```
#(1) load data of viral accessory genes
load("~/Library/CloudStorage/OneDrive-WashingtonStateUniversity(email.wsu.edu)/Fernandez Lab/Projects (A
genes <- opvgenes
rm(combine,dryad,vertlife,virion)

#(2) trim
genes <- head(genes, -2)
genes <- genes[,-2]
colnames(genes) <- paste("ag" ,colnames(genes),sep="_")
names(genes)[1] <- c("virus")

#(3) reformat as numeric data matrix
mat <- as.matrix(genes[,-1])
```

```
rownames(mat) <- genes[,1] %>% pull()
class(mat) <- "numeric"

#(4) remove genes with no variation
mat <- mat[,-which(apply(mat, 2, var)==0)]
```

Run PCA

```
#(1) PCA using stats::prcomp
pca <- prcomp(mat)
pca <- prcomp(mat,scale=TRUE,center=TRUE)

pca_var <- pca$sdev^2 / sum(pca$sdev^2)
pca_var_per <- round(pca_var*100,1)
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.data <- data.frame(Sample=rownames(pca$x),
                       X=pca$x[,1],
                       Y=pca$x[,2])
pca.data

#(2) PCA coordinate plot
ggplot(data=pca.data, aes(x=X,y=Y,label=Sample)) +
  geom_text() +
  xlab(paste("PC1 - ", pca.var.per[1], "%", sep="")) +
  ylab(paste("PC2 - ", pca.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("PCA plot")

#(3) Screeplot
fviz_eig(pca)
screeplot(pca, type="l", npcs=10, main="Screeplot of the first 10 PCs")

#pca <- princomp(mat) #{stats}
```

Run PCoA

```
#(6) PCoA with cmdscale {stats} - standardized
dist.mat <- dist(scale((mat),center=TRUE,scale=TRUE),
                 method="euclidean")
mds <- cmdscale(dist.mat, eig=TRUE,x.ret=TRUE) #perform MDS on gendist
mds.var.per <- round(mds$eig/sum(mds$eig)*100,1)
mds.values <- mds$points
mds.data <- data.frame(Sample=rownames(mds.values),
                       X=mds.values[,1],
                       Y=mds.values[,2])
ggplot(data=mds.data, aes(x=X,y=Y,label=Sample)) +
  geom_text() +
  xlab(paste("MDS1 - ", mds.var.per[1], "%", sep="")) +
```

```
  ylab(paste("MDS2 - ", mds.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("MDS plot using Euclidean distance")

#(7) PCoA with pcoa {ape} - unstandardized
dist.mat <- vegdist(mat)
##distance matrix (measure of the pairwise similarity between each virus based on whether they have the
pcoa <- pcoa(dist.mat) #{ape}
pcoa$vectors
pcoa.var.per <- round(pcoa$values$Eigenvalues/sum(pcoa$values$Eigenvalues)*100,1)
pcoa.values <- pcoa$vectors
pcoa.data <- data.frame(Sample=rownames(pcoa.values),
                        X=pcoa.values[,1],
                        Y=mds.values[,2])
ggplot(data=pcoa.data, aes(x=X,y=Y,label=Sample)) +
  geom_text() +
  xlab(paste("PCoA1 - ", mds.var.per[1], "%", sep="")) +
  ylab(paste("PCoA2 - ", mds.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("PCoA graph using Euclidean distance")

#(6) plot coordinate pairs and projections
biplot(pcoa)
biplot(pcoa, mat, dir.axis1=-1)

#scree plot - elbow, saturation (eigenvalue)
```