TickMerged_hurdle.R

melissachen

2020-04-27

```
######## GAM model with manual hurdle component ##########
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✔ ggplot2 3.2.1          ✔ purrr   0.3.3
## ✔ tibble  2.1.3          ✔ dplyr   0.8.3
## ✔ tidyr   1.0.0.9000     ✔ stringr 1.4.0
## ✔ readr   1.3.1          ✔ forcats 0.4.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(mgcv) # for GAMs
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

```
library(randomForest) # for random forest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library("lubridate") # for dates
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(boot) # for invlogit

tck <- read.csv("data_derived/MASTER_all_tck_data_merged.csv")

#### Filter tick dataset and adjust ####

## How many plots were tested, and how many times?
tck %>%
  mutate(tested=ifelse(is.na(testingID),"no","yes")) %>%
  select(plotID, tested) %>% table(useNA="ifany")
```

```
##           tested
## plotID        no   yes
##   ABBY_001    70     0
##   ABBY_002    63     0
##   ABBY_003    63     0
##   ABBY_005    67     0
##   ABBY_006    63     0
##   ABBY_023    68     0
##   BARR_021     3     0
##   BARR_030     9     0
##   BARR_031     9     0
##   BARR_034     9     0
##   BARR_037     9     0
##   BARR_084     9     0
##   BART_002    93     0
##   BART_010    87     0
##   BART_011    87     0
##   BART_015    90     0
##   BART_019    87     0
##   BART_029    66     0
##   BLAN_002   180     0
##   BLAN_004   123     0
##   BLAN_005  2193  2807
##   BLAN_008   126     0
##   BLAN_012   221     6
##   BLAN_015   798     2
##   BLAN_020     3     0
##   BONA_002    15     0
##   BONA_004    12     0
##   BONA_012    15     0
##   BONA_013    12     0
##   BONA_020    12     0
##   BONA_044    15     0
##   CLBJ_032    87     0
##   CLBJ_033    91     0
##   CLBJ_034    87     0
##   CLBJ_036    81     0
##   CLBJ_040    87     0
##   CLBJ_043    87     0
##   CPER_001    66     0
##   CPER_002    66     0
##   CPER_003    66     0
```

```
##   CPER_004    66     0
##   CPER_005    66     0
##   CPER_007    66     0
##   DCFS_001    62     0
##   DCFS_003    53     0
##   DCFS_004    39     0
##   DCFS_005    43     0
##   DCFS_007    42     0
##   DCFS_010    59     0
##   DEJU_001    24     0
##   DEJU_003    24     0
##   DEJU_009    24     0
##   DEJU_014    24     0
##   DEJU_015    24     0
##   DEJU_044    21     0
##   DELA_001    96     0
##   DELA_004    21     0
##   DELA_005   113     0
##   DELA_007   100     0
##   DELA_008   117     6
##   DELA_011    21     0
##   DELA_014   115     0
##   DELA_016    94     0
##   DSNY_001    90     0
##   DSNY_002     9     0
##   DSNY_004    87     0
##   DSNY_005    84     0
##   DSNY_006    90     0
##   DSNY_008    89     0
##   DSNY_014     6     0
##   DSNY_021    69     0
##   GRSM_003    84     0
##   GRSM_004    84     0
##   GRSM_005    42     0
##   GRSM_006    84     0
##   GRSM_009    84     0
##   GRSM_010    90     0
##   GRSM_015    42     0
##   GUAN_002    45     0
##   GUAN_004    12     0
##   GUAN_006    48     0
##   GUAN_031    45     0
##   GUAN_064    33     0
```

```
##   GUAN_065    48     0
##   GUAN_066    48     0
##   HARV_001   273   471
##   HARV_002   184     0
##   HARV_004  3185   352
##   HARV_006   771     8
##   HARV_020   152    46
##   HARV_022  1382    88
##   HARV_026   675    12
##   HEAL_001    24     0
##   HEAL_004    24     0
##   HEAL_010    27     0
##   HEAL_011    30     0
##   HEAL_014    27     0
##   HEAL_026    24     0
##   JERC_002    96     0
##   JERC_004    96     0
##   JERC_005    21     0
##   JERC_010    93     0
##   JERC_011    54     0
##   JERC_034    90     0
##   JERC_044    87     0
##   JORN_001    51     0
##   JORN_002    56     0
##   JORN_003    54     0
##   JORN_004    54     0
##   JORN_005    54     0
##   JORN_006    51     0
##   KONA_002    47    56
##   KONA_003    41    48
##   KONA_004    42     0
##   KONA_019    41     2
##   KONA_021    42     0
##   KONA_024    39     0
##   KONZ_001   116    14
##   KONZ_002   178    29
##   KONZ_004   154    32
##   KONZ_007   115     2
##   KONZ_009   116    35
##   KONZ_025  7938  3001
##   LAJA_001    45     0
##   LAJA_002    45     0
##   LAJA_003    33     0
```

```
##   LAJA_004    42     0
##   LAJA_005    30     0
##   LAJA_030    30     0
##   LENO_002   832     6
##   LENO_003   744   245
##   LENO_004  1206     0
##   LENO_015   124     0
##   LENO_023   159    24
##   LENO_042   102     8
##   MLBS_002    24     0
##   MLBS_003    24     0
##   MLBS_004    19     0
##   MLBS_005    24     0
##   MLBS_006    18     0
##   MLBS_009    24     0
##   MOAB_001    60     0
##   MOAB_002    60     0
##   MOAB_003    57     0
##   MOAB_004    60     0
##   MOAB_005    60     0
##   MOAB_006    60     0
##   NIWO_003    27     0
##   NIWO_005    27     0
##   NIWO_006    27     0
##   NIWO_011    27     0
##   NIWO_017    27     0
##   NIWO_021    27     0
##   NOGP_001    75     0
##   NOGP_002    76     0
##   NOGP_005    72     0
##   NOGP_006    72     0
##   NOGP_007    72     0
##   NOGP_008    72     0
##   OAES_003   110     0
##   OAES_004   122     0
##   OAES_005   110     0
##   OAES_007   109     0
##   OAES_008   124     0
##   OAES_021   133     0
##   ONAQ_003     6     0
##   ONAQ_004    63     0
##   ONAQ_005    63     0
##   ONAQ_008     6     0
```

```
##   ONAQ_009    63     0
##   ONAQ_010     6     0
##   ONAQ_011    57     0
##   ONAQ_016    57     0
##   ONAQ_023    58     0
##   ORNL_002  5278  8471
##   ORNL_003  3429   561
##   ORNL_007 10677  3160
##   ORNL_008  4779  6088
##   ORNL_009  7493   638
##   ORNL_040  5148  4196
##   OSBS_001  3521  3609
##   OSBS_002   231     0
##   OSBS_003  2012  7181
##   OSBS_004   144    26
##   OSBS_005  9193 28824
##   OSBS_022    72  1132
##   OSBS_048    41     0
##   RMNP_002    18     0
##   RMNP_003    18     0
##   RMNP_004    18     0
##   RMNP_007    18     0
##   RMNP_012    18     0
##   RMNP_013    18     0
##   SCBI_002  2696  1036
##   SCBI_005   608    60
##   SCBI_006   609    16
##   SCBI_007  4085  1093
##   SCBI_013  1215  4714
##   SCBI_039  1951   147
##   SERC_001  3300   567
##   SERC_002  6861  3164
##   SERC_005  1537  1105
##   SERC_006   634   874
##   SERC_012  1806   565
##   SERC_023   125     8
##   SJER_001    39     0
##   SJER_002    39     0
##   SJER_007    39     0
##   SJER_008    39     0
##   SJER_009    39     0
##   SJER_025    39     0
##   SOAP_003    12     0
```

```
##   SOAP_004    12     0
##   SOAP_005    12     0
##   SOAP_006    12     0
##   SOAP_009    12     0
##   SOAP_026    12     0
##   SRER_002    42     0
##   SRER_003    45     0
##   SRER_004    39     0
##   SRER_006    45     0
##   SRER_007    39     0
##   SRER_008    42     0
##   STEI_001    66     0
##   STEI_002    70     0
##   STEI_003    69     0
##   STEI_005    70     0
##   STEI_023    65     0
##   STEI_029    81     0
##   STER_006    42     0
##   STER_026    51     0
##   STER_027    54     0
##   STER_028    51     0
##   STER_029    51     0
##   STER_033    54     0
##   TALL_001  2703   923
##   TALL_002  5996 18767
##   TALL_003   951   738
##   TALL_006   675   260
##   TALL_008  1819  8798
##   TALL_016   208     2
##   TOOL_009     9     0
##   TOOL_028     9     0
##   TOOL_031     9     0
##   TOOL_032     9     0
##   TOOL_035     9     0
##   TOOL_071     9     0
##   TREE_005    95     0
##   TREE_007   376   373
##   TREE_015   271    23
##   TREE_017   741   712
##   TREE_019  1599  1607
##   TREE_022   152   188
##   UKFS_001 48606 22766
##   UKFS_002  8765   315
```

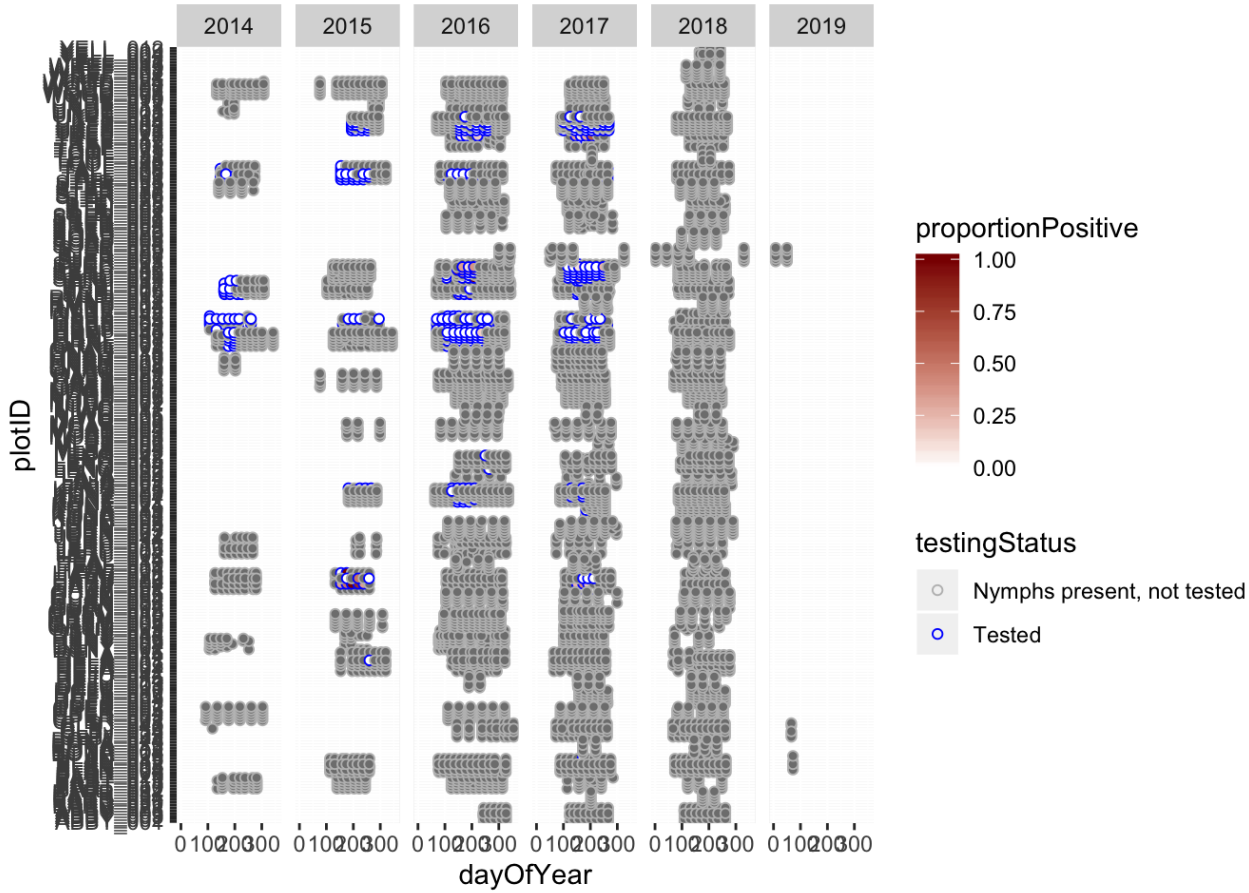
```
##   UKFS_003 63104  8448
##   UKFS_004 36999  6667
##   UKFS_018   128     0
##   UKFS_030   157    32
##   UNDE_002    69     0
##   UNDE_003     9     0
##   UNDE_011    66     0
##   UNDE_013    75     0
##   UNDE_017    94     0
##   UNDE_019    76     0
##   UNDE_999    69     0
##   WOOD_001   248     0
##   WOOD_002   163     0
##   WOOD_006   225     0
##   WOOD_007   290     0
##   WOOD_009   187     0
##   WOOD_024   111     0
##   WREF_001    12     0
##   WREF_004    12     0
##   WREF_007    12     0
##   WREF_008    12     0
##   WREF_009    12     0
##   WREF_013    12     0
##   YELL_001     9     0
##   YELL_002     9     0
##   YELL_003    12     0
##   YELL_004    13     0
##   YELL_009    12     0
##   YELL_012     9     0
```

```
## What type of ticks did they test?
tck %>%
  mutate(tested=ifelse(is.na(testingID),"no","yes")) %>%
  select(lifeStage, tested) %>% table(useNA="ifany")
```

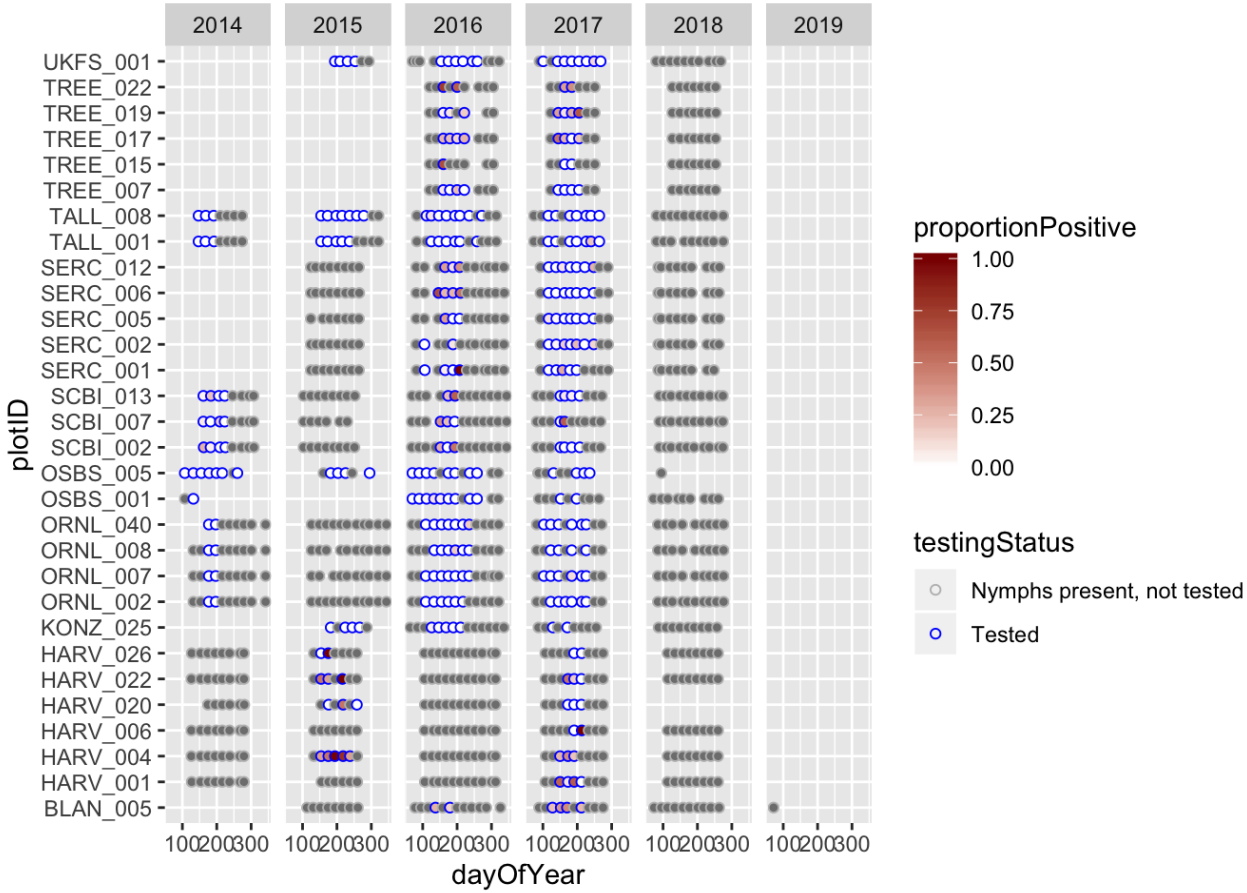
```
##          tested
## lifeStage     no    yes
##     Adult   9815      0
##     Larva 260512      0
##     Nymph  13821 155154
```

```
## Aggregate by plotID:dayOfYear:year. Count number of different life stages; create proportiontested/positive et
c
tck_allsamples_borr <- tck %>%
  mutate(tested=ifelse(is.na(Borrelia_sp.),0,1) # Was each tick  ever tested for borrellia?
         ,isPositive=ifelse(Borrelia_sp.=="Positive", 1,0) # And if it was tested, was it positive or negative?
  ) %>%
  group_by(domainID, siteID, nlcdClass, plotID, elevation, collectDate, dayOfYear, year, month, totalSampledArea)
%>% # collapse by sample-- summing all tck counts together
  summarize(numberTested=sum(tested, na.rm=TRUE) # number of tested ticks in that sample
            ,n=n() # total ticks in that sample
            , numberPositive=sum(isPositive,na.rm = TRUE) # number of positive ticks in that sample
            , nAdult=sum(lifeStage=="Adult")
            , nNymph=sum(lifeStage=="Nymph")
            , nLarva=sum(lifeStage=="Larva")) %>%
  ungroup() %>%
  mutate(proportionTested=numberTested/nNymph # proportion of all nymph ticks tested-- only nymphs were ever test
ed.
         , proportionPositive=numberPositive/numberTested
         , tested = ifelse(numberTested > 0 , TRUE, FALSE) # new true/false tested, which is summed across ticks
         , testingStatus = ifelse(numberTested > 0, "Tested", ifelse(nNymph>0, "Nymphs present, not tested", "No
 nymphs"))
         ) %>%
  mutate(nlcdClass=factor(nlcdClass, levels=c("emergentHerbaceousWetlands","cultivatedCrops","pastureHay","grassl
andHerbaceous"
                                              ,"dwarfScrub","shrubScrub","sedgeHerbaceous"
                                              ,"woodyWetlands","deciduousForest","evergreenForest","mixedForest"
))) %>%
  mutate(borrPresent = ifelse(numberPositive>0,1,0)
         , domainID = factor(as.character(domainID))
         , siteID = factor(as.character(siteID))
         , plotID = factor(as.character(plotID))
         , year = factor(year)
         , tckDensity = sum(c(nLarva, nNymph, nAdult))/totalSampledArea
         , ntckDensity = nNymph/totalSampledArea
         , NLtckDensity = sum(c(nNymph, nAdult), na.rm = TRUE)/totalSampledArea
         , atckDensity = nAdult/totalSampledArea)  %>%
  mutate(lognymphDensity=log(ntckDensity)
         , logNLtckDensity=log(NLtckDensity)
         , logtckDensity = log(tckDensity)
         , logadultDensity=log(atckDensity))

## Plot over time, by year, by plot, to see how zero-inflated it is.
```

```
tck_allsamples_borr %>%
  ggplot() + geom_point(aes(x=dayOfYear, y=plotID, fill=proportionPositive, col=testingStatus), pch=21) +
  scale_fill_gradient(low="white", high="darkred") +
  scale_color_manual(values=c(Tested="blue", `Nymphs present, not tested`="grey", `No nymphs`="white")) +
  facet_grid(.~year)
```



```
## Let's filter out plots that NEVER had any positive borrelia
infectedPlots <- tck %>%
  filter(Borrelia_sp.=="Positive") %>%
  select(plotID) %>%pull() %>% unique()

## Filtering same data as above; only now removing non-infected plots.
tck_borrelia_positivePlots <- tck_allsamples_borr %>%
  filter(plotID %in% as.character(infectedPlots))

## Re-assess the apperance of plots
tck_borrelia_positivePlots %>%
  ggplot() + geom_point(aes(x=dayOfYear, y=plotID, fill=proportionPositive, col=testingStatus), pch=21) +
  scale_fill_gradient(low="white", high="darkred") +
  scale_color_manual(values=c(Tested="blue", `Nymphs present, not tested`="grey", `No nymphs`="white")) +
  facet_grid(.~year)
```

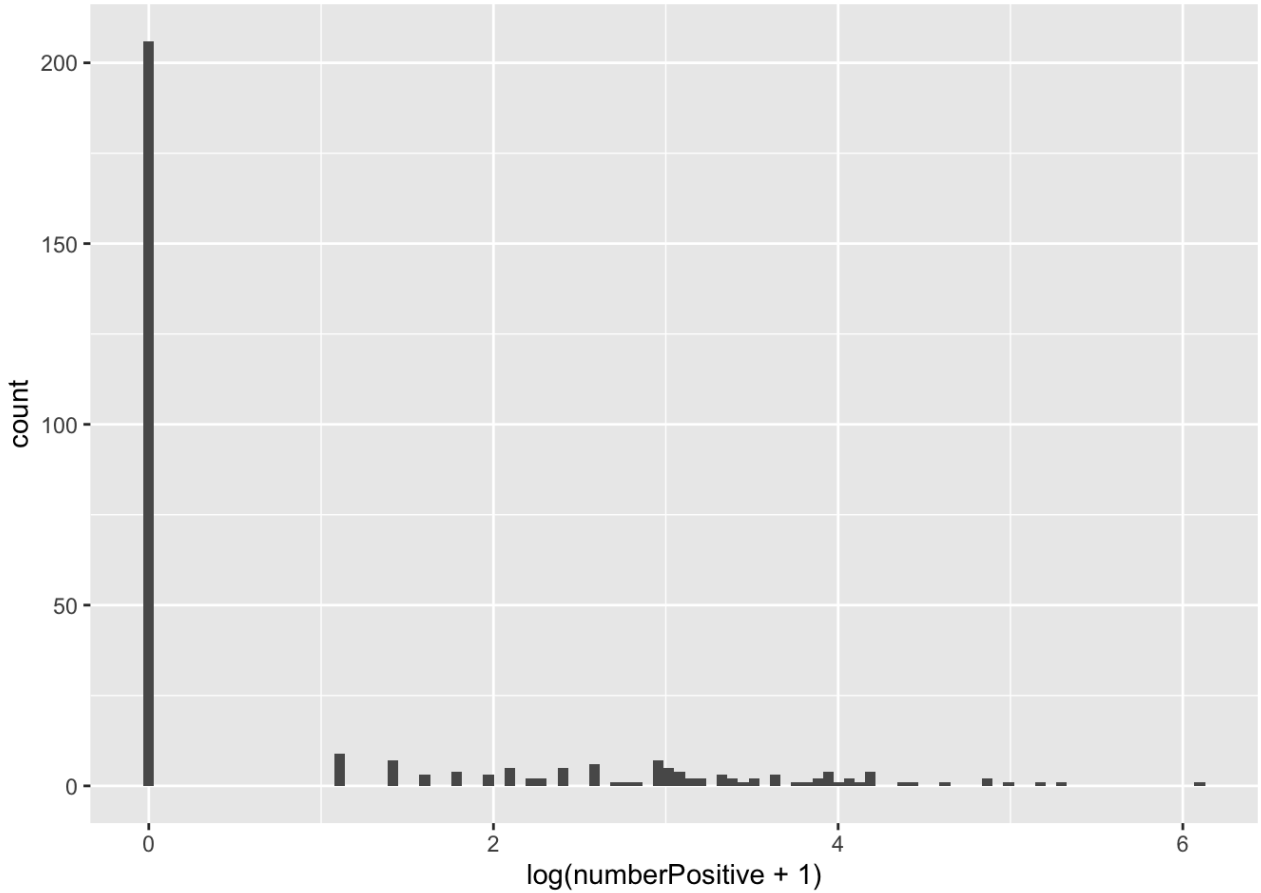


```
## There are STILL a lot of zeros-- zeros where there are nymphs but they were never tested.
# Let's remove those.
tck_borrelia <- tck_borrelia_positivePlots %>%
  mutate(numberPositive=ifelse(numberTested==0,NA,numberPositive)) %>% # Make sure that numberPositive is not art
ificially zero-- if there were no tests, it should be NA
  filter(numberTested!=0) # get rid of all samples where the didn't actually test.

tck_borrelia %>%
  ggplot() + geom_point(aes(x=dayOfYear, y=plotID, fill=proportionPositive, col=testingStatus), pch=21) +
  scale_fill_gradient(low="white", high="darkred") +
  scale_color_manual(values=c(Tested="blue", `Nymphs present, not tested`="grey", `No nymphs`="white")) +
  facet_grid(.~year)
```

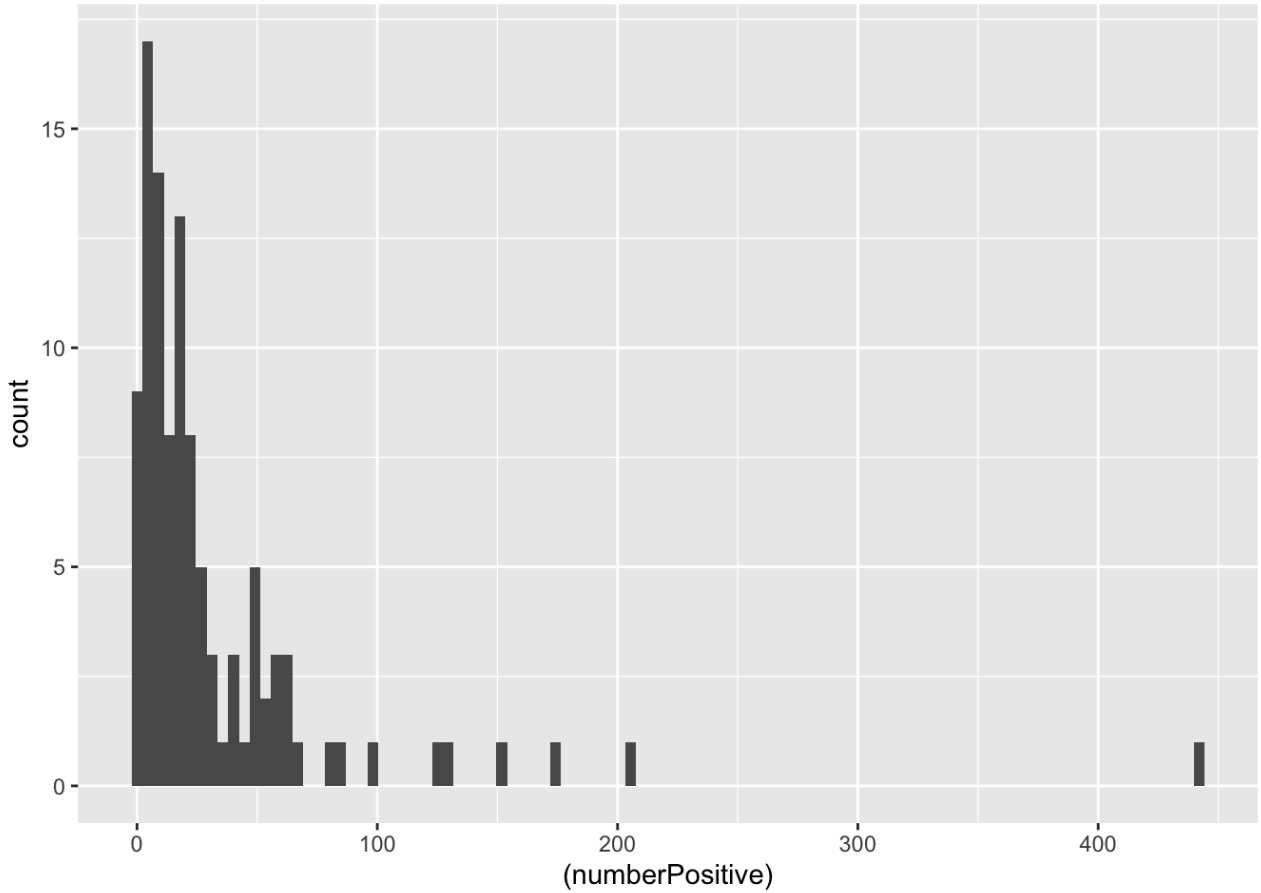

```
nrow(tck_borrelia)
```

```
## [1] 311
```

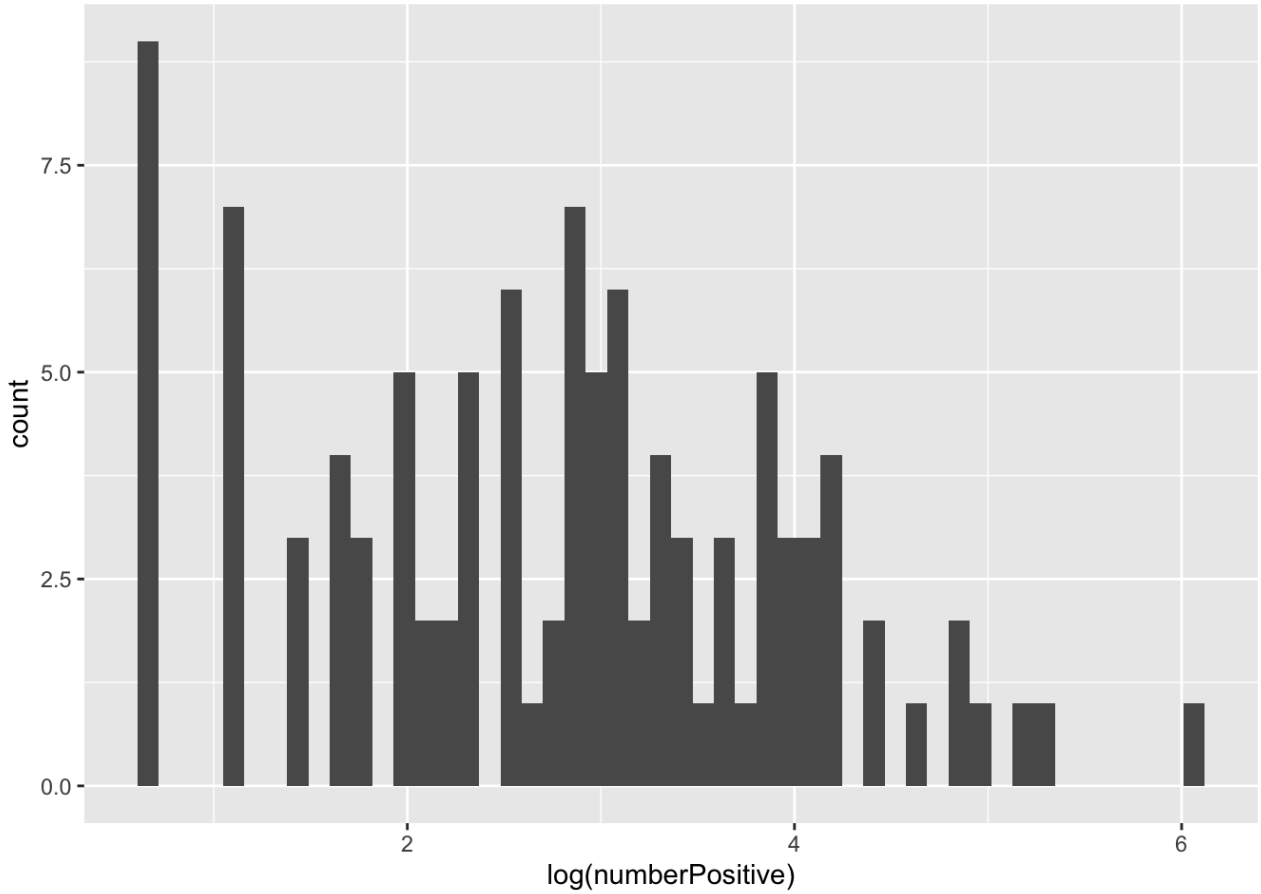
```
# There are only 311 samples left-- still zero inflated, but slightly better.

#### Preliminary Plotting ####

# First, what is the distribution of Borrelia prevalence?
tck_borrelia %>%
  ggplot() +geom_histogram(aes(x=log(numberPositive+1)), bins=100)
```



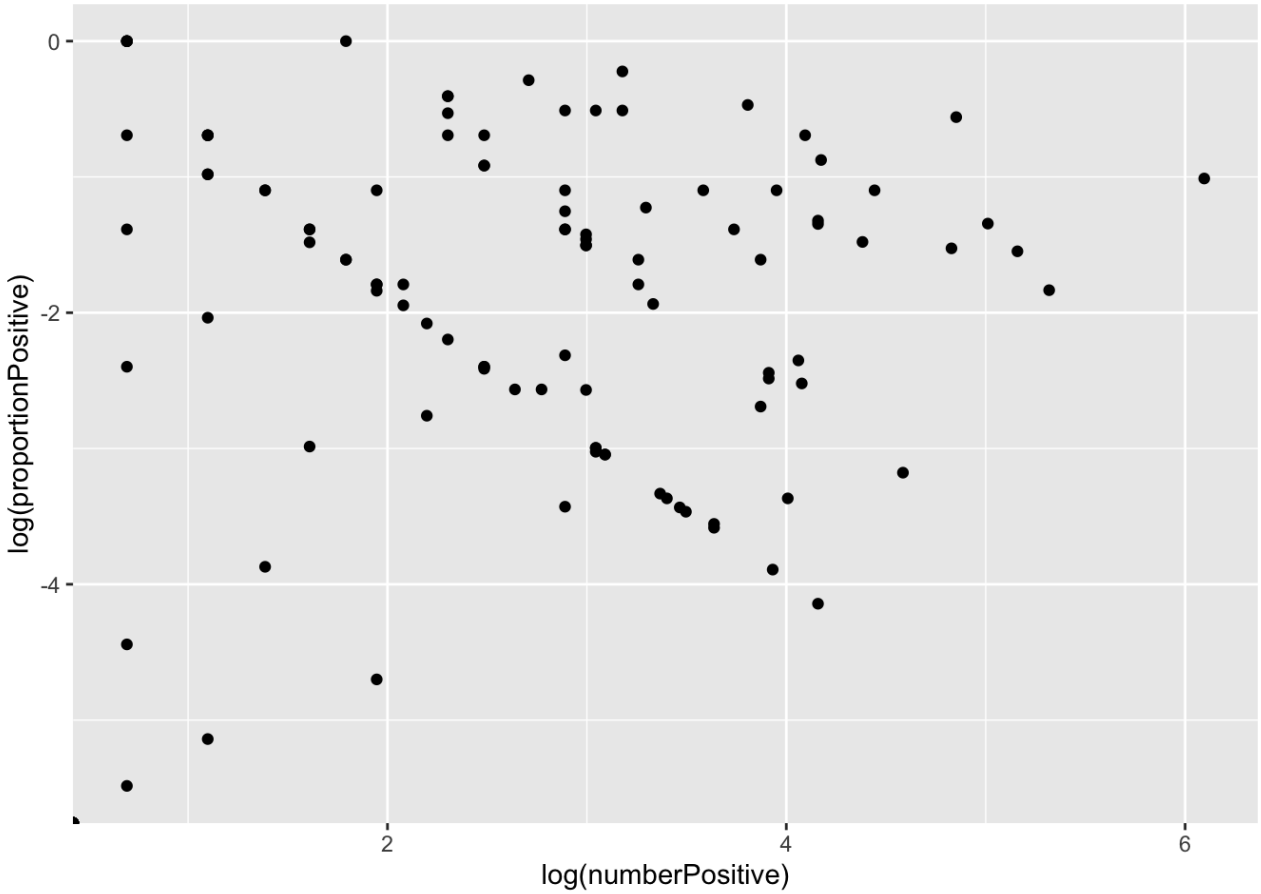
```
# histogram without any zeros
tck_borrelia %>%
  filter(numberPositive>0) %>%
  ggplot() +geom_histogram(aes(x=(numberPositive)), bins=100)
```



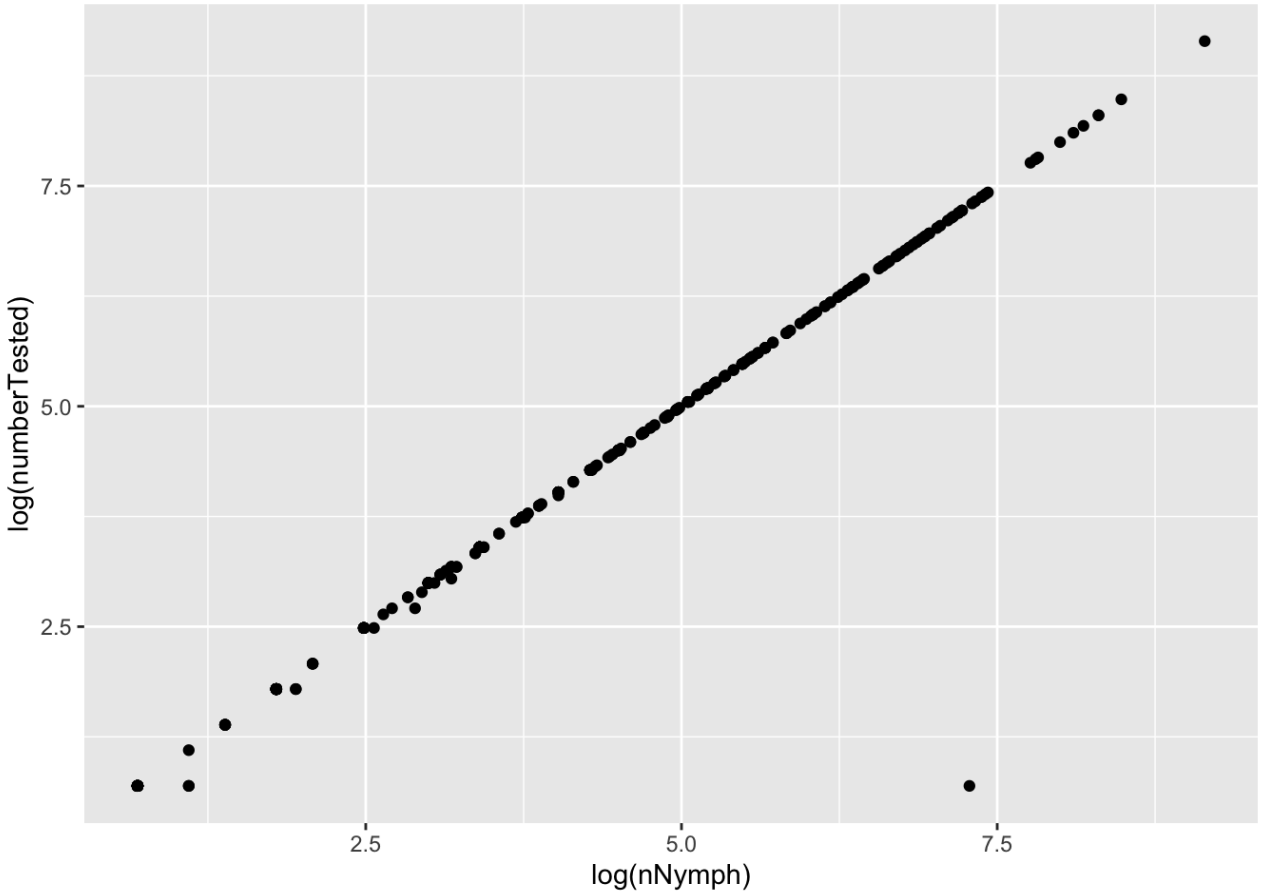

```
# histogram log abundance, no zeros
tck_borrelia %>%
  filter(numberPositive>0) %>%
  ggplot() +geom_histogram(aes(x=log(numberPositive)), bins=50)
```




```
# What is relationship between proportion positive and number positive
tck_borrelia %>%
  ggplot() +geom_point(aes(x=log(numberPositive), y=log(proportionPositive)))
```



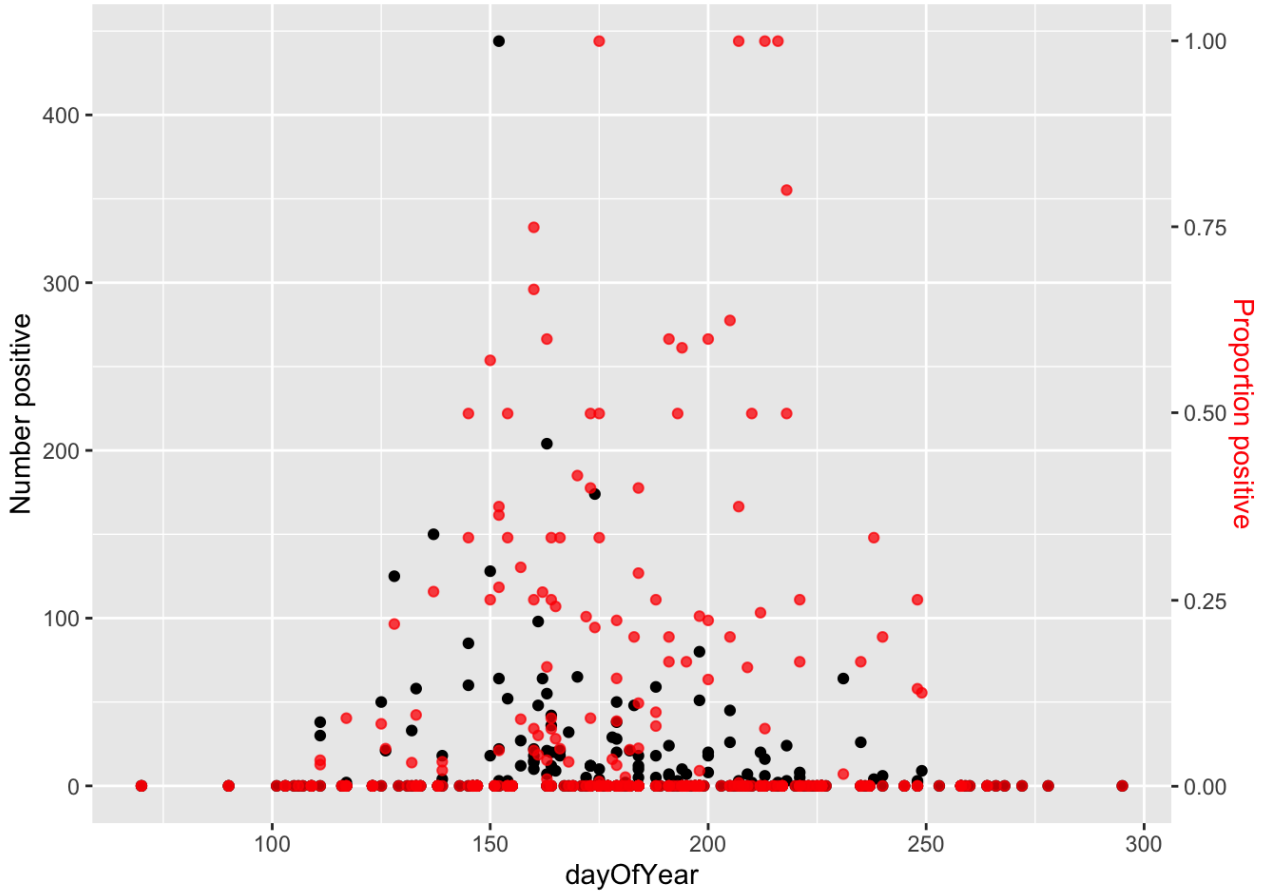

```
# Now, what is the relationship between number tested and nNymph? Might be a binning artifact?
tck_borrelia %>%
  ggplot() +geom_point(aes(x=log(nNymph), y=log(numberTested)))
```



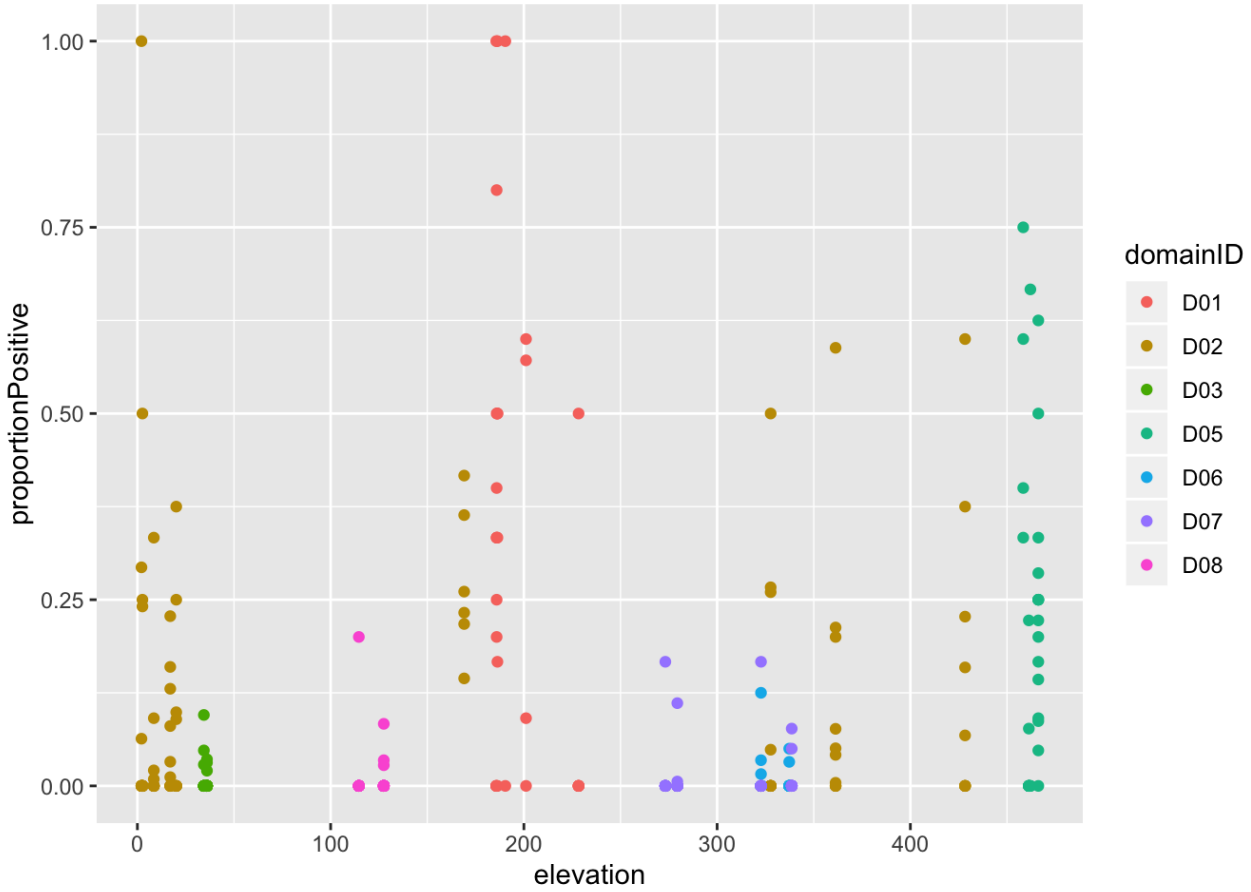

```
# Just get rid of all data where they didn't test most nymphs-- see how many that is
tck_borrelia %>%
  filter(abs(nNymph-numberTested)>0) %>% select(plotID, collectDate, nNymph, numberTested)
```

```
## # A tibble: 36 x 4
##    plotID   collectDate       nNymph numberTested
##    <fct>    <fct>              <int>        <dbl>
##  1 HARV_004 2015-07-13T18:22Z   1451            2
##  2 BLAN_005 2016-05-16T15:37Z    577          575
##  3 BLAN_005 2017-06-01T17:59Z   1222         1221
##  4 BLAN_005 2017-06-19T14:51Z    158          156
##  5 SCBI_002 2016-05-31T14:40Z    241          240
##  6 SCBI_002 2017-07-03T16:34Z      7            6
##  7 SCBI_007 2017-05-31T17:32Z     13           12
##  8 SCBI_013 2017-07-26T17:29Z    483          482
##  9 SERC_001 2017-06-28T15:00Z     31           30
## 10 SERC_001 2017-07-17T20:38Z      7            6
## # … with 26 more rows
```

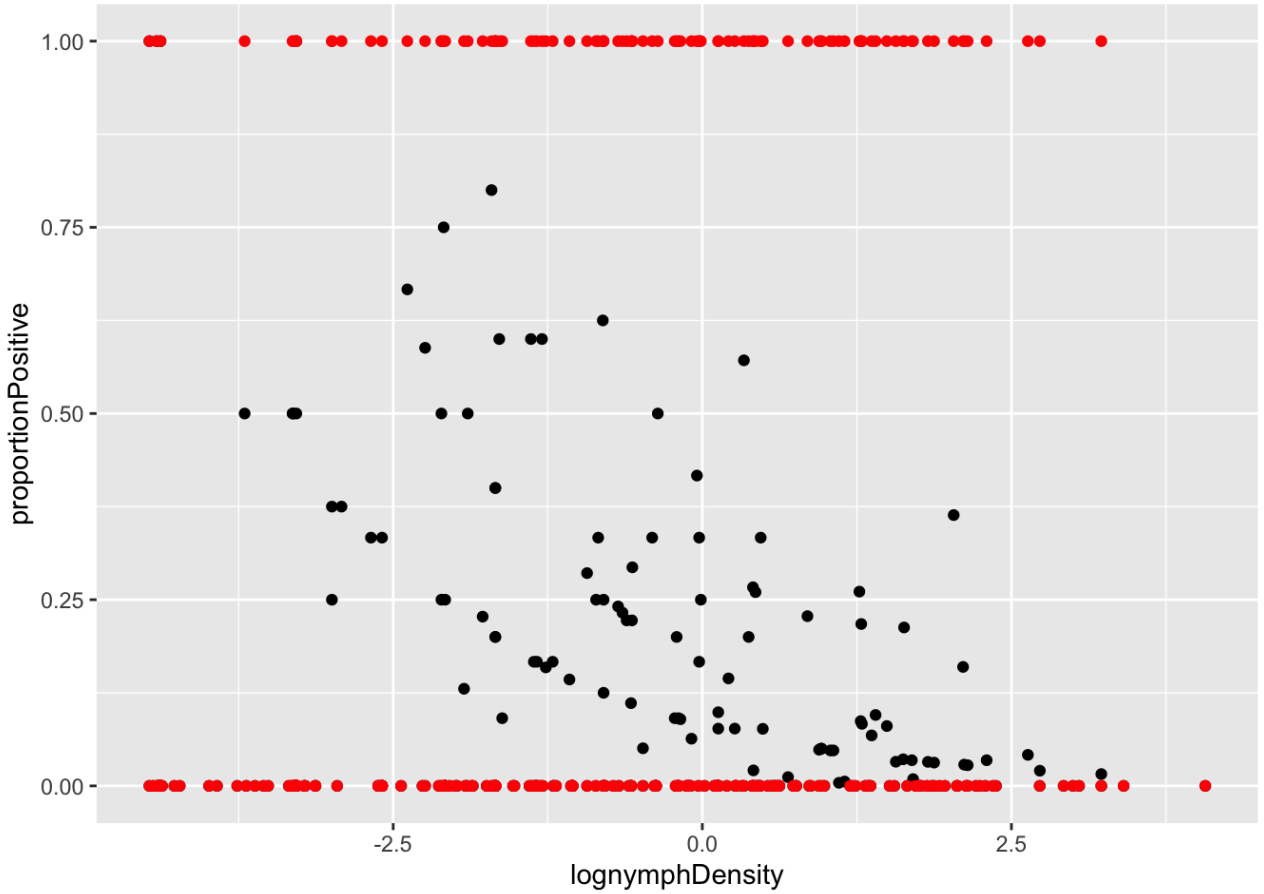
```
# Most of these samples are not a problem, except HARV_004 (2015) and maybe SCI_002 (2017) and SERC_001 (2017-0
7).
# Let's get rid of only HARV_004; I think all the rest are fine.
tck_borrelia_adj <- tck_borrelia %>%
  filter(abs(nNymph-numberTested)<3) # HARV_004 is the only site that differs nNymph and numberTested by more tha
n 3

## Inspecting possible predictors of borrelia
# dayOfYear vs borrelia
tck_borrelia_adj %>%
  ggplot() +geom_point(aes(x=dayOfYear, y=numberPositive))+
  geom_point(aes(x=dayOfYear, y=proportionPositive*max(tck_borrelia_adj$numberPositive)), col="red", alpha=0.75)
 +
  scale_y_continuous(sec.axis=sec_axis(trans=~./max(tck_borrelia_adj$numberPositive), name="Proportion positive"
))+
  theme(axis.title.y.right = element_text(color="red")) + ylab("Number positive")
```



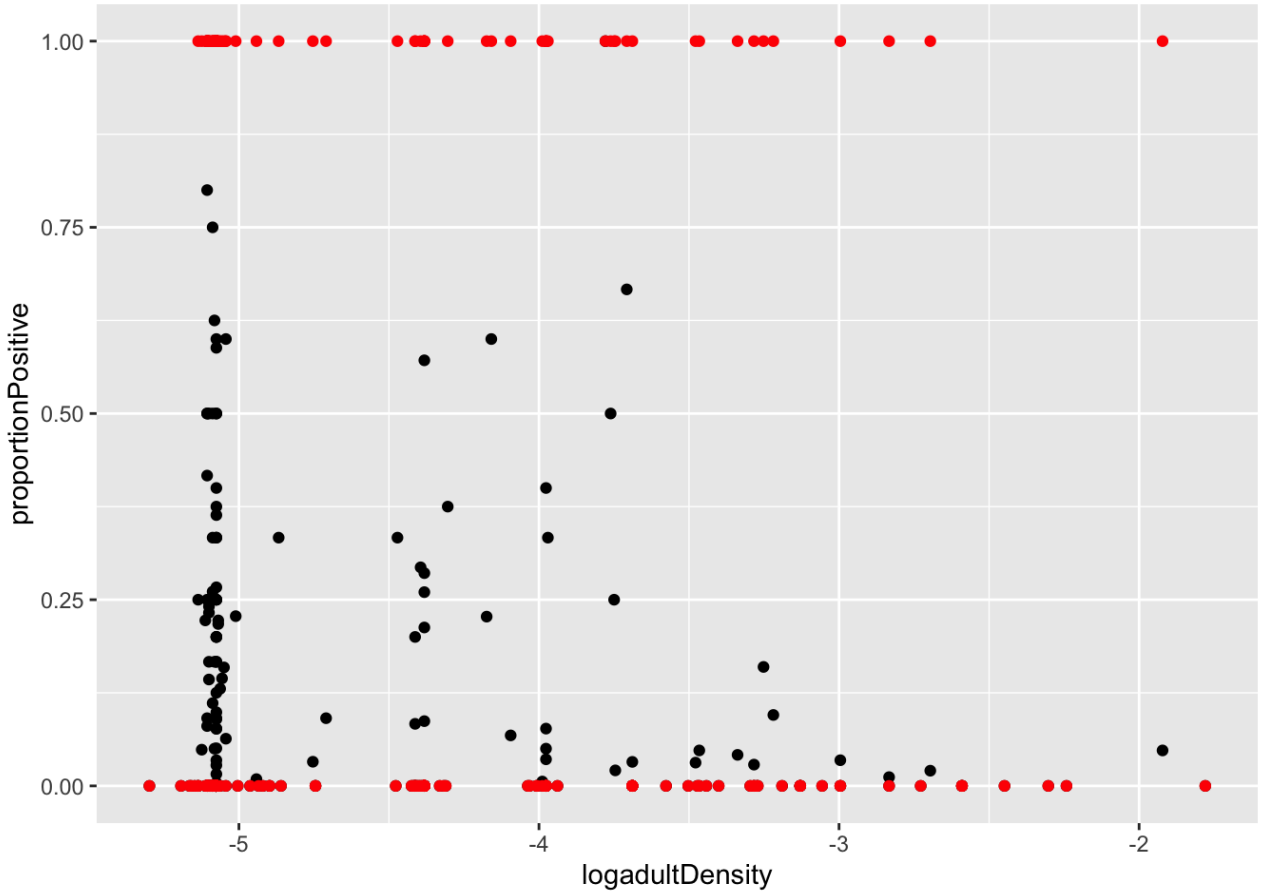

```
# elevation vs borrelia
tck_borrelia_adj %>%
  ggplot() +geom_point(aes(x=elevation, y=proportionPositive, col=domainID))
```



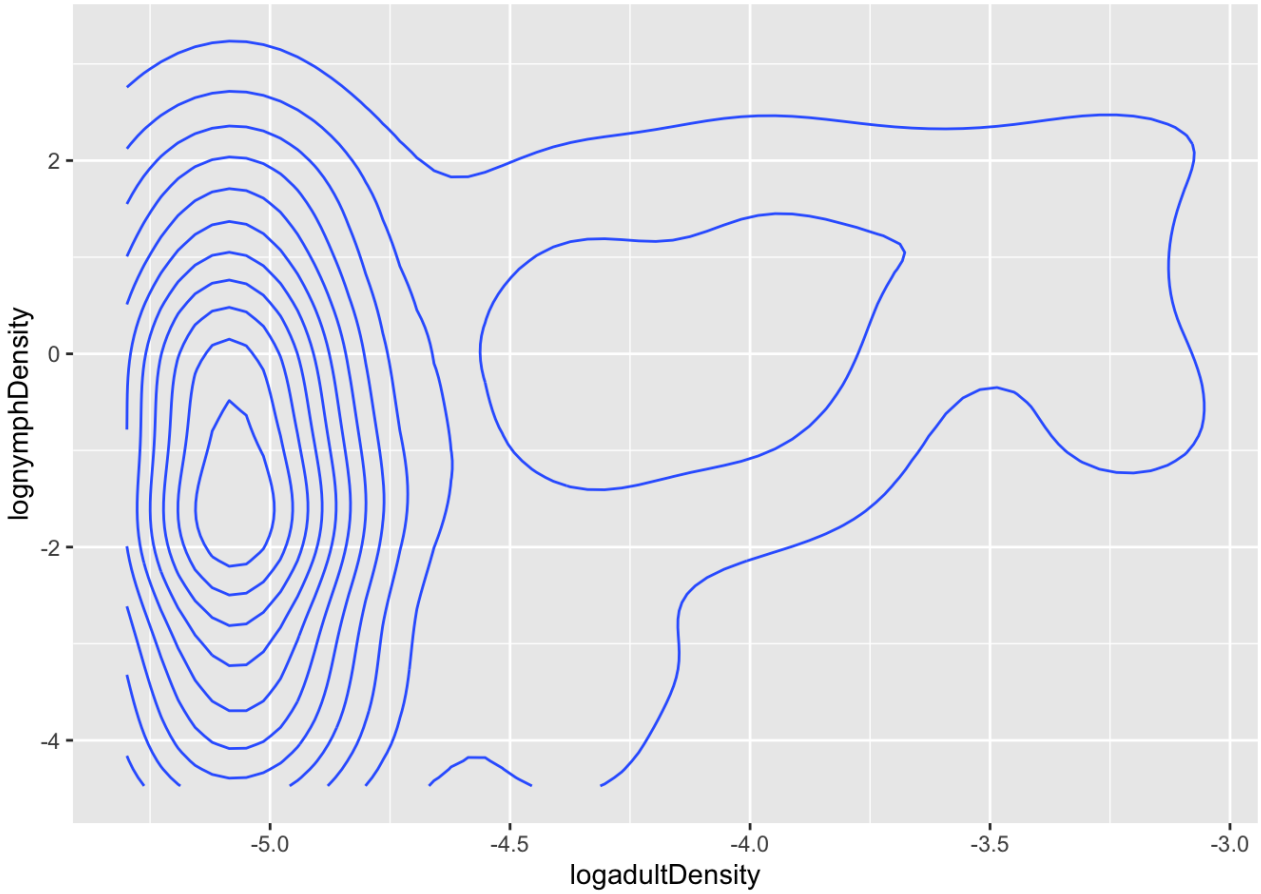

```
# nymph density vs borrelia
tck_borrelia_adj %>%
  ggplot() +geom_point(aes(x=lognymphDensity, y=proportionPositive)) +
  geom_point(aes(x=lognymphDensity, y=borrPresent), col="red")
```



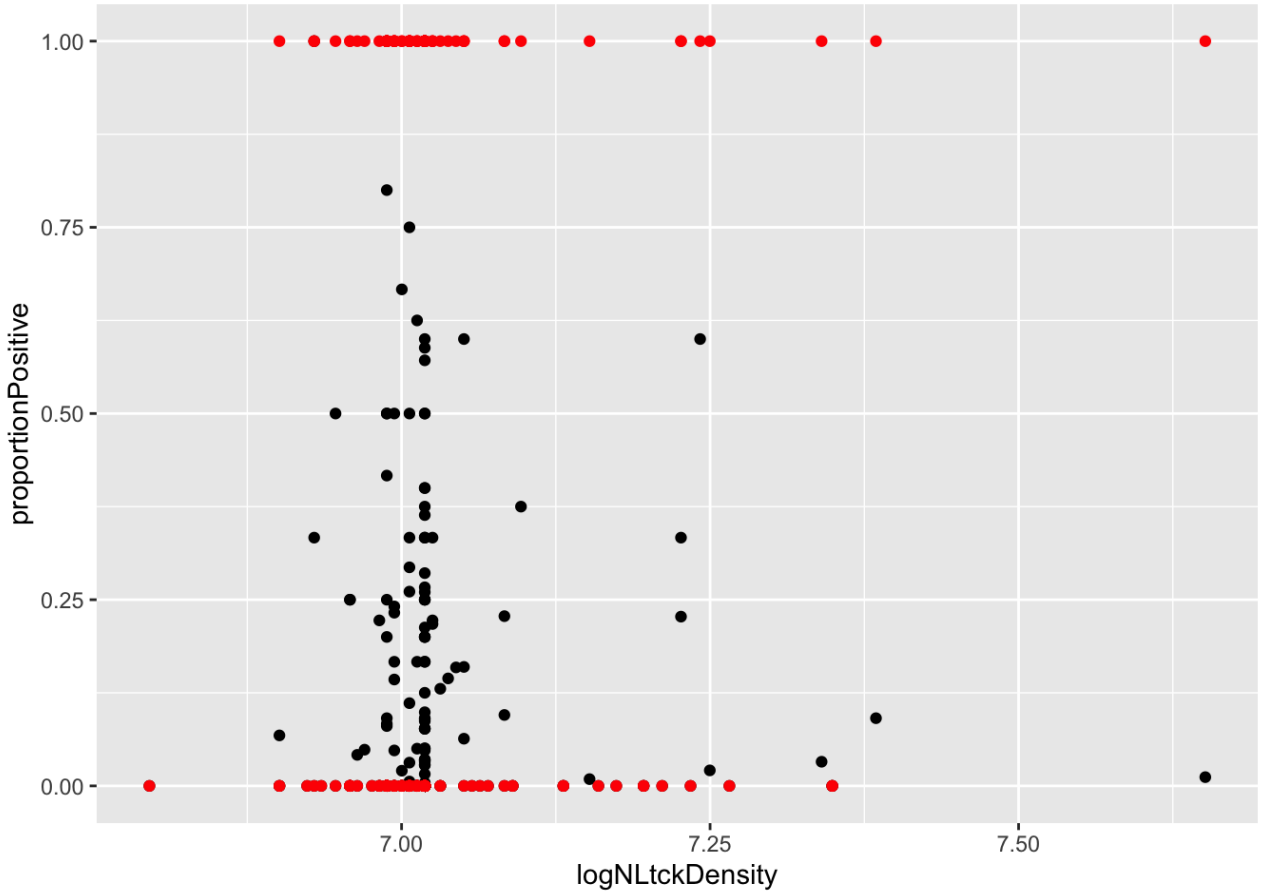

```
# adult density vs borrelia
tck_borrelia_adj %>%
  ggplot() +geom_point(aes(x=logadultDensity, y=proportionPositive)) +
  geom_point(aes(x=logadultDensity, y=borrPresent), col="red")
```




```
# nymph and adult density vs borrelia-- first, are they correlated?
tck_borrelia_adj %>%
  ggplot() + geom_density2d(aes(x=logadultDensity, y=lognymphDensity))
```




```
tck_borrelia_adj %>%
  ggplot() +geom_point(aes(x=logNLtckDensity, y=proportionPositive)) +
  geom_point(aes(x=logNLtckDensity, y=borrPresent), col="red")
```


1.00
0.75
0.50
0.25
0.00
proportionPositive
7.00
7.25
7.50
logNLtckDensity

```
#### Fitting a two-step GAM ####

stepGAM <- function(dat, predictors, response, constant_pred=NULL, family, ignore.combos) {
  ## For troubleshooting
  # dat <- tck_borrelia_adj
  # predictors = allPred
  # response = "borrPresent"
  # constant_pred <- "offset(log(numberTested))"
  # family=binomial
  # ignore.combos=list(c("domainID","s(domainID, bs='re')"), c("s(logNLtckDensity)","s(logadultDensity)","s(logny
mphDensity)")
  #                    , c("s(logNLtckDensity)","s(logadultDensity)"),c("s(logNLtckDensity)","s(lognymphDensit
y)"))
  n_rows <- sum(sapply(1:length(predictors),FUN=function(x){ncol(combn(predictors,m=x))} ))
  allAIC <- setNames(data.frame(matrix(nrow=n_rows, ncol=(4+length(predictors)))),nm = c("AIC","REML","Dev.expl",
"formula",predictors))
  i <- 1
  pb <- txtProgressBar(title = "progress bar", min = 0,
                       max = n_rows, style=3)
  for ( p in 1:length(predictors) ) {
    combos <- combn(predictors, m=p)
    for ( c in 1:ncol(combos) ) {
      # skip any combinations that are meant to be ignored
      if (any(sapply(ignore.combos, FUN=function(ic){sum(ic %in% combos[,c]) == length(ic)}))) {
        i = i + 1
        next
      }
      if ( length(constant_pred)>0 ) {
        frml.temp <- paste(c(paste0(response," ~ ",constant_pred),combos[,c]), collapse="+")
      } else {
        frml.temp <- paste0(paste0(response," ~ "),paste(combos[,c], collapse=" + "), sep="")
      }
      # If more coefficients than data
      res <- try(gam(as.formula(frml.temp)
              , data=dat
              , method="REML"
              , family=family))
      if ( inherits(res, "try-error") ) {
        next
      } else {
        gam.temp <- gam(as.formula(frml.temp)
                        , data=dat
```

```
                        , method="REML"
                        , family=family)

        pred.temp <- (colnames(allAIC) %in% combos[,c])[-c(1,2,3,4)]
        allAIC[i,c("AIC","REML","Dev.expl","formula")]  <- as.vector(c(gam.temp$aic,gam.temp$gcv.ubre[1],summary
(gam.temp)$dev.expl,frml.temp))
        allAIC[i, predictors] <- pred.temp

      }

      i <- i+1
      Sys.sleep(0.1)
      setTxtProgressBar(pb, i, label=paste( round(i/total*100, 0),
                                            "% done"))
    }
  }
  close(pb)
  return(allAIC)
}

#### Subsetting data into training and testing sets ####

# We want to subset into multiple subsets over time. How far apart are sampling days?
tck_borrelia_adj %>%
  filter(year==2016, plotID=="OSBS_005") %>%
  arrange(dayOfYear) %>% select(dayOfYear) %>%pull() %>%diff()
```

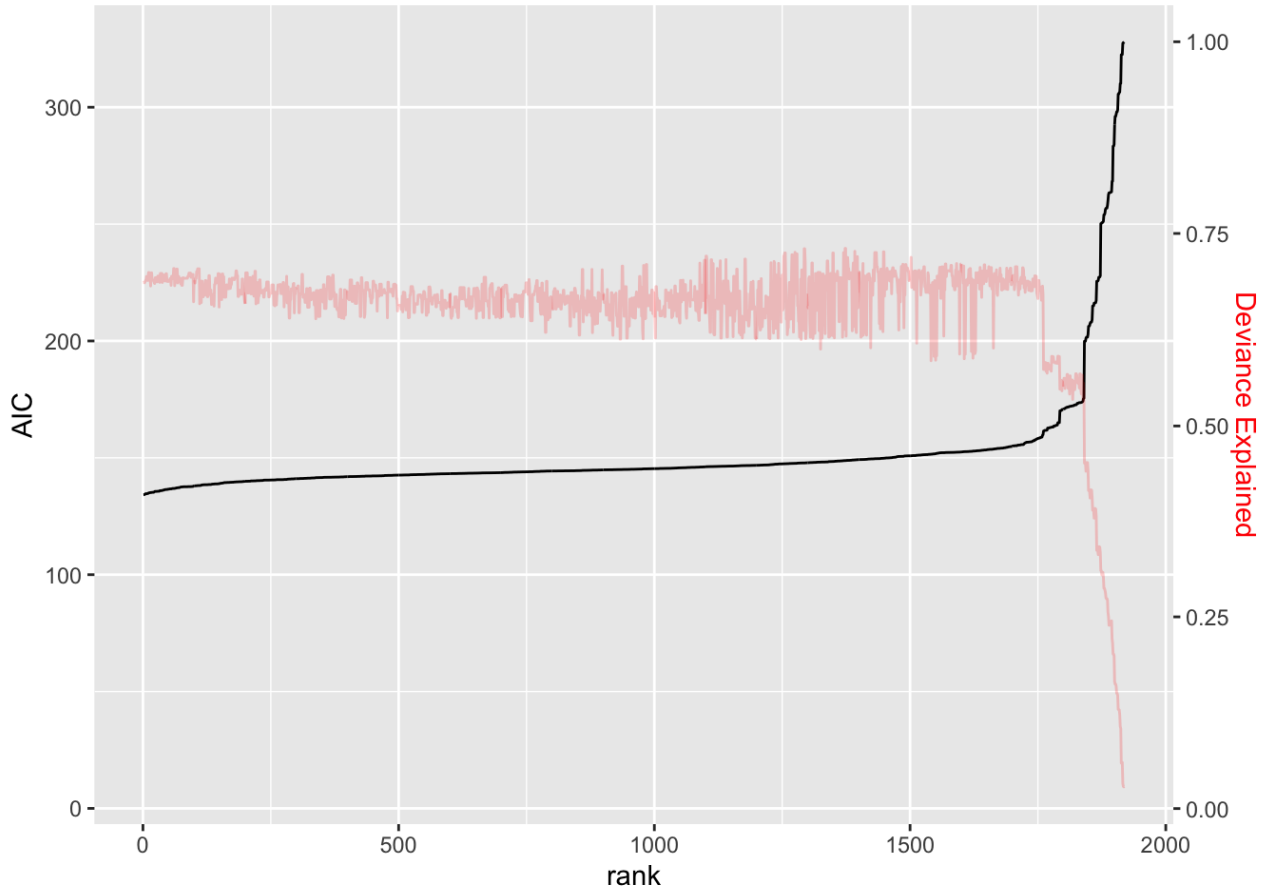
```
## [1] 20 21 22 42 19 43 21
```

```
# Around every 20-25 days, on average


## Get only 2014-2016 data for training
tck_borrelia_train <- tck_borrelia_adj %>% filter(year!=2017)
# Filter so test dataset so we only have those in training
inclPlots <- as.character(unique(tck_borrelia_train$plotID))
# make testing set with same plots
tck_borrelia_test <- tck_borrelia_adj %>% filter(year==2017, plotID %in% inclPlots)


## Subsetting multiple time subsets
subsetData <- list()
daysIn2017 <- tck_borrelia_adj %>% filter(year==2017, plotID %in% c(inclPlots)) %>% select(dayOfYear) %>% pull()
 %>% unique() %>% sort()
for ( doy in daysIn2017 ) {
  subsetData[[paste(doy)]] <- list()
  subsetData[[paste(doy)]][['train']] <- tck_borrelia_adj %>% filter(!(year==2017 & dayOfYear>=doy), plotID %in%
 inclPlots) %>% mutate(year=factor(year))
  subsetData[[paste(doy)]][['test']] <- tck_borrelia_adj %>% filter((year==2017 & dayOfYear>=doy), plotID %in% in
clPlots)%>% mutate(year=factor(year))
}

#### Part I: binomial hurdle component ####
# First, model a binomial component using only 2017 data
# Plot is a random effect, which I include as a varying-intercept, varying-slope model.
allPred <- c("s(dayOfYear)","nlcdClass","s(elevation)"
             ,"s(logNLtckDensity)","s(logadultDensity)","s(lognymphDensity)"
             ,"s(plotID, bs='re')", "s(plotID, dayOfYear, bs='re')"
             ,"s(year, bs='re')", "s(year, dayOfYear, bs='re')"
             , "domainID", "s(domainID, bs='re')"
)


if (FALSE) {
  allAIC <- stepGAM(dat = tck_borrelia_train, predictors = allPred, response = "borrPresent", constant_pred = "of
fset(log(numberTested))", family = binomial, ignore.combos=list(c("domainID","s(domainID, bs='re')"), c("s(logNLt
ckDensity)","s(logadultDensity)","s(lognymphDensity)")
, c("s(logNLtckDensity)","s(logadultDensity)"),c("s(logNLtckDensity)","s(lognymphDensity)")))
  allAIC_filt <- allAIC %>% filter(!is.na(AIC)) %>% arrange(AIC) %>%mutate(rank=seq(1:length(AIC))) %>%
    mutate(AIC=as.numeric(AIC)
           , Dev.expl = as.numeric(Dev.expl)
```

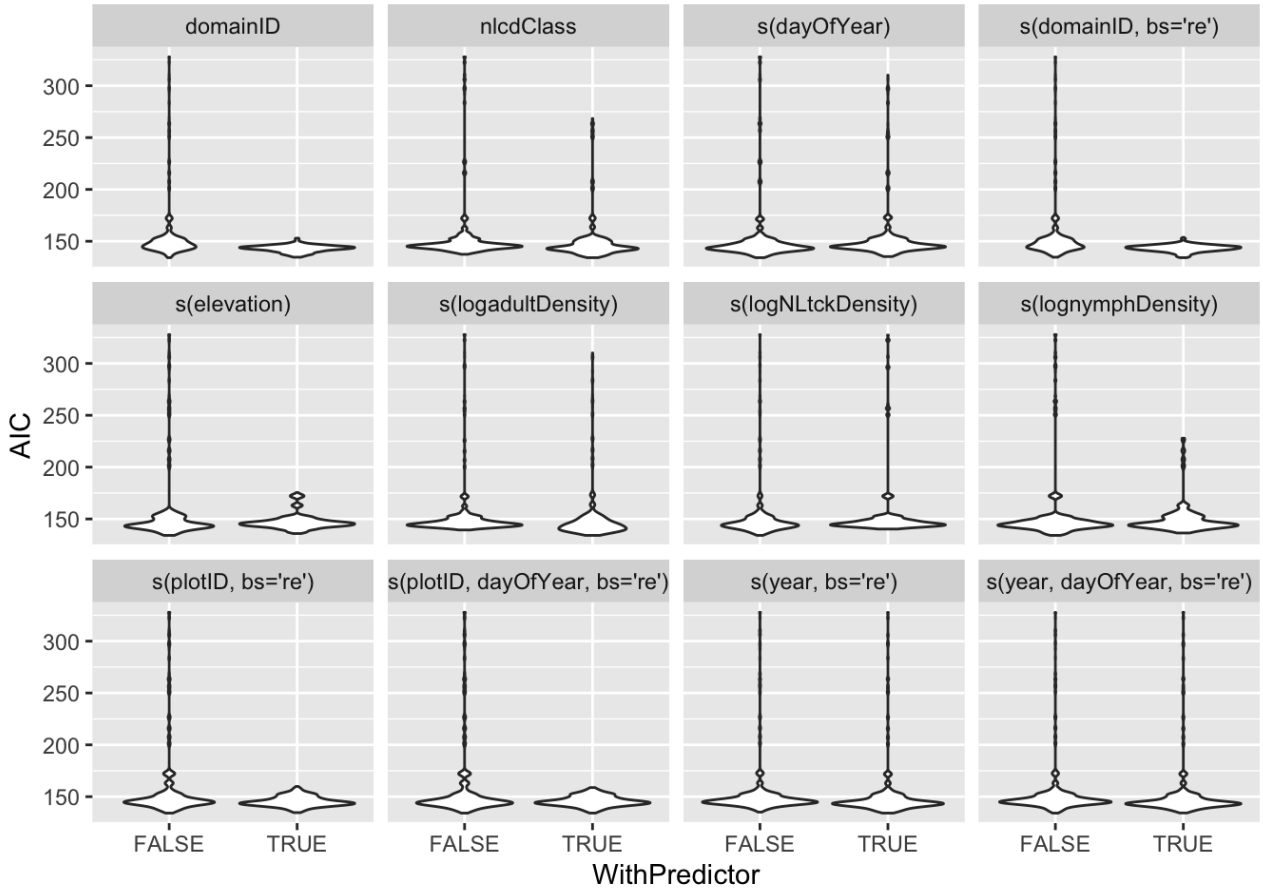
```
           , REML = as.numeric(REML)) %>%
    filter(AIC<500)
  save(allAIC, file="allAIC.RData")
  save(allAIC_filt, file="allAIC_filt.RData")

} else {
  load("allAIC.RData")
  load("allAIC_filt.RData")

}
# allAIC %>%
#   as_tibble() %>%
#   arrange(AIC) %>% View()

nrow(allAIC)
```

```
## [1] 4095
```

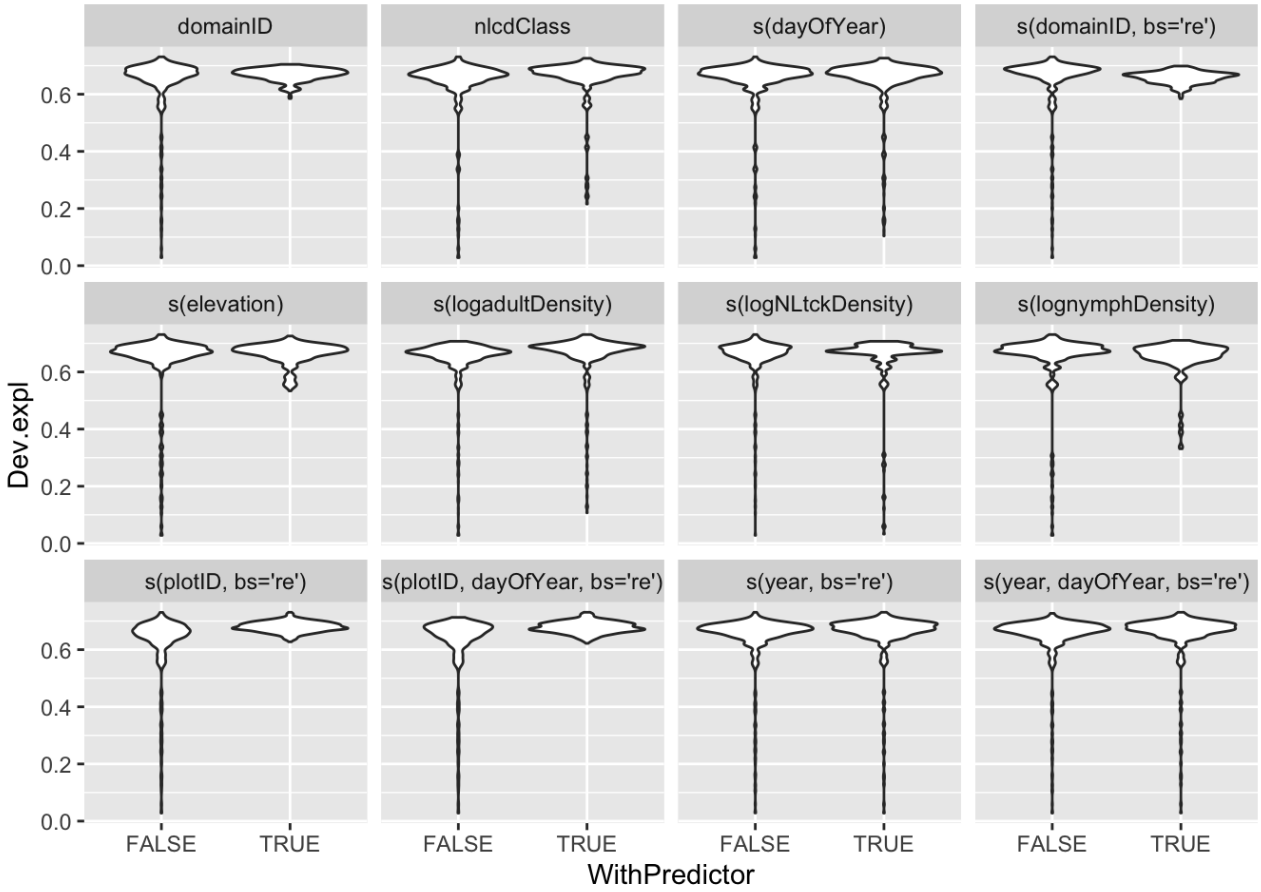
```
# Let's see distribution of AIC values
allAIC_filt %>%
  ggplot() +geom_line(aes(x=rank, y=AIC))+
  geom_line(aes(x=rank, y=Dev.expl*(max(allAIC_filt$AIC))), col="red", alpha=0.2) + scale_y_continuous(sec.axis=s
ec_axis(~./(max(allAIC_filt$AIC)), name="Deviance Explained")) +
  theme(axis.title.y.right = element_text(colour = "red"))
```




```
# Compare AIC for models with and without each predictor
allAIC_filt %>% gather(-c(AIC, REML, Dev.expl, formula,rank), key=Predictor, value=WithPredictor) %>%
  select(AIC, REML, Dev.expl, Predictor, WithPredictor) %>%
  mutate(WithPredictor = ifelse(WithPredictor==0, FALSE, TRUE)) %>%
  ggplot() + geom_violin(aes(x=WithPredictor, y=AIC))  +  facet_wrap(.~Predictor)
```



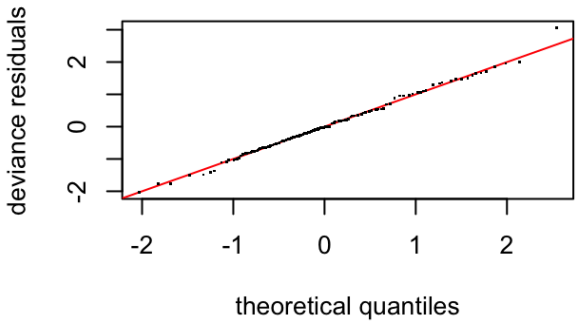

```
# Compare deviance explained for models with and without each predictor
allAIC_filt %>% gather(-c(AIC, REML, Dev.expl, formula,rank), key=Predictor, value=WithPredictor) %>%
  select(AIC, REML, Dev.expl, Predictor, WithPredictor) %>%
  mutate(WithPredictor = ifelse(WithPredictor==0, FALSE, TRUE)) %>%
  ggplot() + geom_violin(aes(x=WithPredictor, y=Dev.expl))  +  facet_wrap(.~Predictor)
```

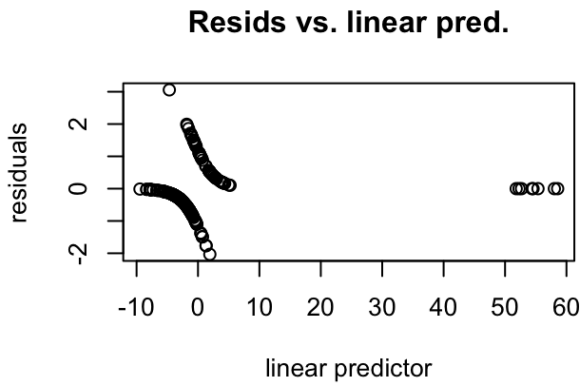


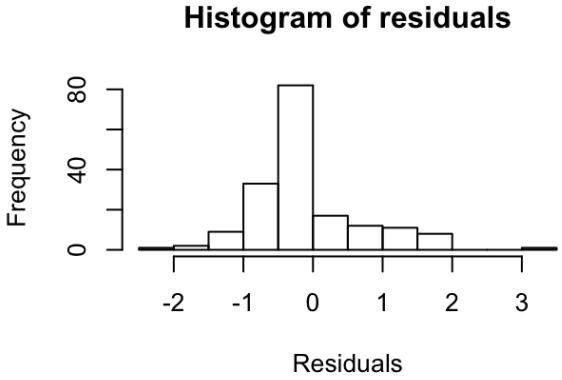

```
frml_bin1_bestAIC <- allAIC_filt[allAIC_filt$AIC==min(allAIC_filt$AIC),"formula"]
frml_bin1_bestDevexpl <- allAIC_filt[allAIC_filt$Dev.expl==max(allAIC_filt$Dev.expl),"formula"]

# Look at the model with the best AIC value
mod.gambin_bestAIC <- gam(as.formula(frml_bin1_bestAIC)
                      # , offset=log(numberTested) # I include the offset in the formula instead, so that it is i
ncluded in predictions. Including it here does NOT incorporate numberTested in predictiosn.
                      , data=tck_borrelia_train
                      , method="REML"
                      , family=binomial)
summary(mod.gambin_bestAIC)
```

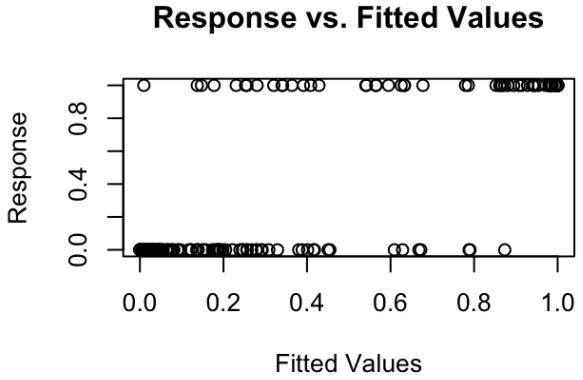
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## borrPresent ~ offset(log(numberTested)) + nlcdClass + s(logadultDensity) +
##     s(year, bs = "re") + s(domainID, bs = "re")
##
## Parametric coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -5.768e+00  2.083e+00  -2.769  0.00562 **
## nlcdClassdeciduousForest -7.696e-01  1.943e+00  -0.396  0.69203
## nlcdClassevergreenForest  2.806e+00  1.638e+00   1.713  0.08669 .
## nlcdClassmixedForest      5.261e+01  2.373e+07   0.000  1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df Chi.sq  p-value
## s(logadultDensity) 4.149  5.049  10.58   0.0621 .
## s(year)            1.462  2.000  19.95   0.0816 .
## s(domainID)        5.372  6.000 102.12 8.22e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.54   Deviance explained = 68.6%
## -REML = 51.939  Scale est. = 1         n = 176
```

```
gam.check(mod.gambin_bestAIC)
```


deviance residuals
-2
0
2
-2
-1
0
1
2
theoretical quantiles

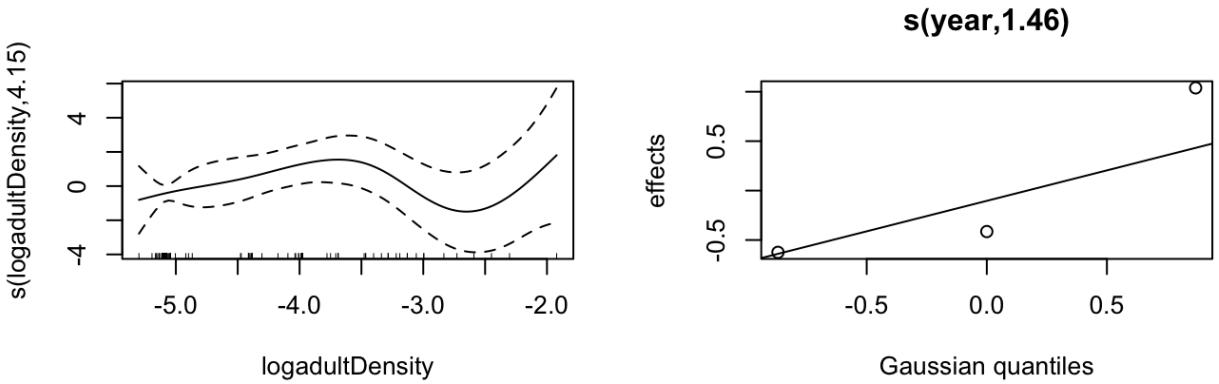

Resids vs. linear pred.
residuals
-2
0
2
-10
0
10
20
30
40
50
60
linear predictor

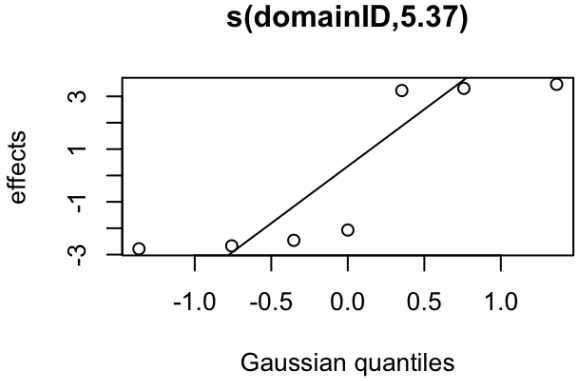

Histogram of residuals
Frequency
0
40
80
-2
-1
0
1
2
3
Residuals


Response vs. Fitted Values
Response
0.0
0.4
0.8
0.0
0.2
0.4
0.6
0.8
1.0
Fitted Values

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 9 iterations.
## Gradient range [-2.658227e-08,2.705552e-09]
## (score 51.939 & scale 1).
## Hessian positive definite, eigenvalue range [0.2856262,2.501879].
## Model rank =  23 / 23
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                      k'  edf k-index p-value
## s(logadultDensity) 9.00 4.15    0.89   0.095 .
## s(year)            3.00 1.46      NA      NA
## s(domainID)        7.00 5.37      NA      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(mod.gambin_bestAIC, pages=1)
```

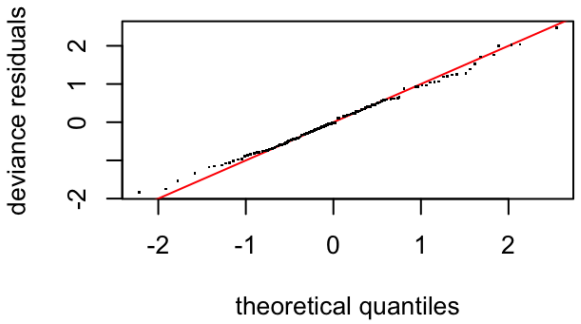


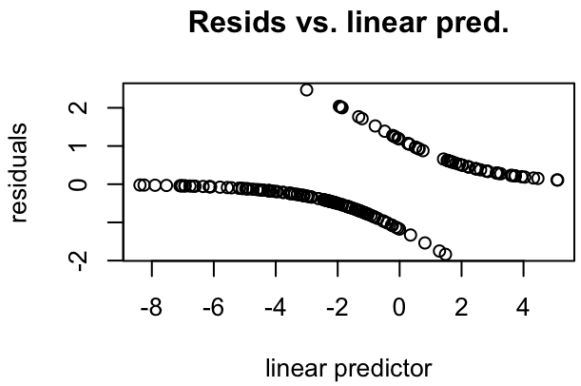


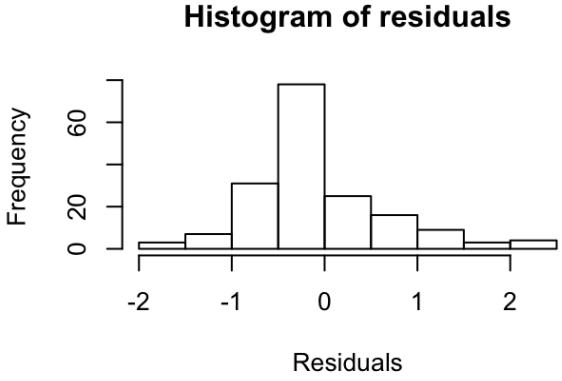

```
# Look at the model with the best deviance explained
mod.gambin_bestDevExpl <- gam(as.formula(frml_bin1_bestDevexpl)
                          # , offset=log(numberTested) # I include the offset in the formula instead, so that it
 is included in predictions. Including it here does NOT incorporate numberTested in predictiosn.
                          , data=tck_borrelia_train
                          , method="REML"
                          , family=binomial)
summary(mod.gambin_bestDevExpl)
```

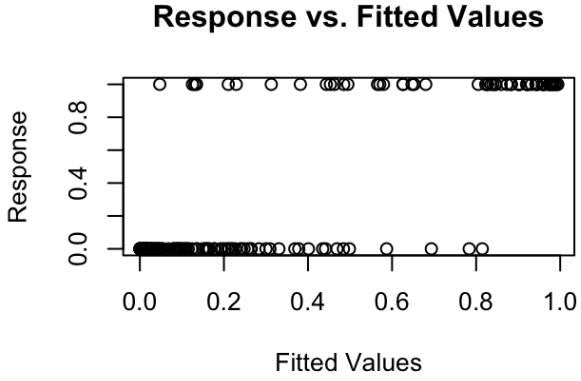
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## borrPresent ~ offset(log(numberTested)) + s(logadultDensity) +
##     s(plotID, bs = "re") + s(plotID, dayOfYear, bs = "re") +
##     s(year, bs = "re") + s(year, dayOfYear, bs = "re")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.9028     0.9139  -5.365  8.1e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                           edf Ref.df  Chi.sq p-value
## s(logadultDensity)  3.206e+00  3.881   4.797   0.254
## s(plotID)           6.472e+00 27.000  81.313   0.622
## s(plotID,dayOfYear) 1.752e+01 28.000 172.997   0.219
## s(year)             8.619e-05  2.000   0.000   0.763
## s(year,dayOfYear)   1.551e+00  3.000  27.936   0.117
##
## R-sq.(adj) =  0.584   Deviance explained = 73.1%
## -REML = 88.371  Scale est. = 1         n = 176
```

```
gam.check(mod.gambin_bestDevExpl)
```


deviance residuals
-2
0
1
2
-2
-1
0
1
2
theoretical quantiles

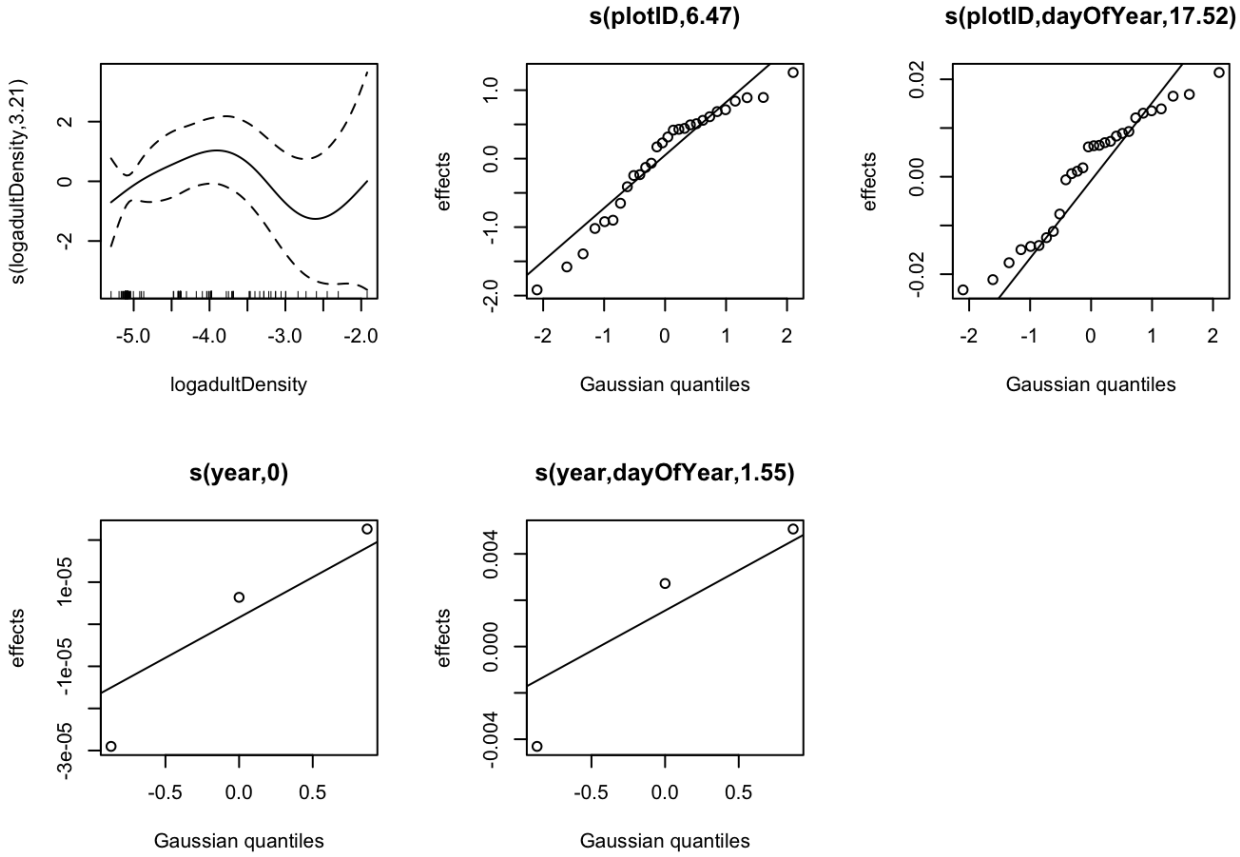

Resids vs. linear pred.
residuals
-2
0
1
2
-8
-6
-4
-2
0
2
4
linear predictor


Histogram of residuals
Frequency
0
20
60
-2
-1
0
1
2
Residuals


Response vs. Fitted Values
Response
0.0
0.4
0.8
0.0
0.2
0.4
0.6
0.8
1.0
Fitted Values

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-3.141881e-05,-1.537113e-06]
## (score 88.37141 & scale 1).
## Hessian positive definite, eigenvalue range [3.14177e-05,3.677544].
## Model rank =  72 / 72
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                           k'      edf k-index p-value
## s(logadultDensity)  9.00e+00 3.21e+00    0.93     0.2
## s(plotID)           2.80e+01 6.47e+00      NA      NA
## s(plotID,dayOfYear) 2.80e+01 1.75e+01      NA      NA
## s(year)             3.00e+00 8.62e-05      NA      NA
## s(year,dayOfYear)   3.00e+00 1.55e+00      NA      NA
```

```
plot(mod.gambin_bestDevExpl, pages=1)
```



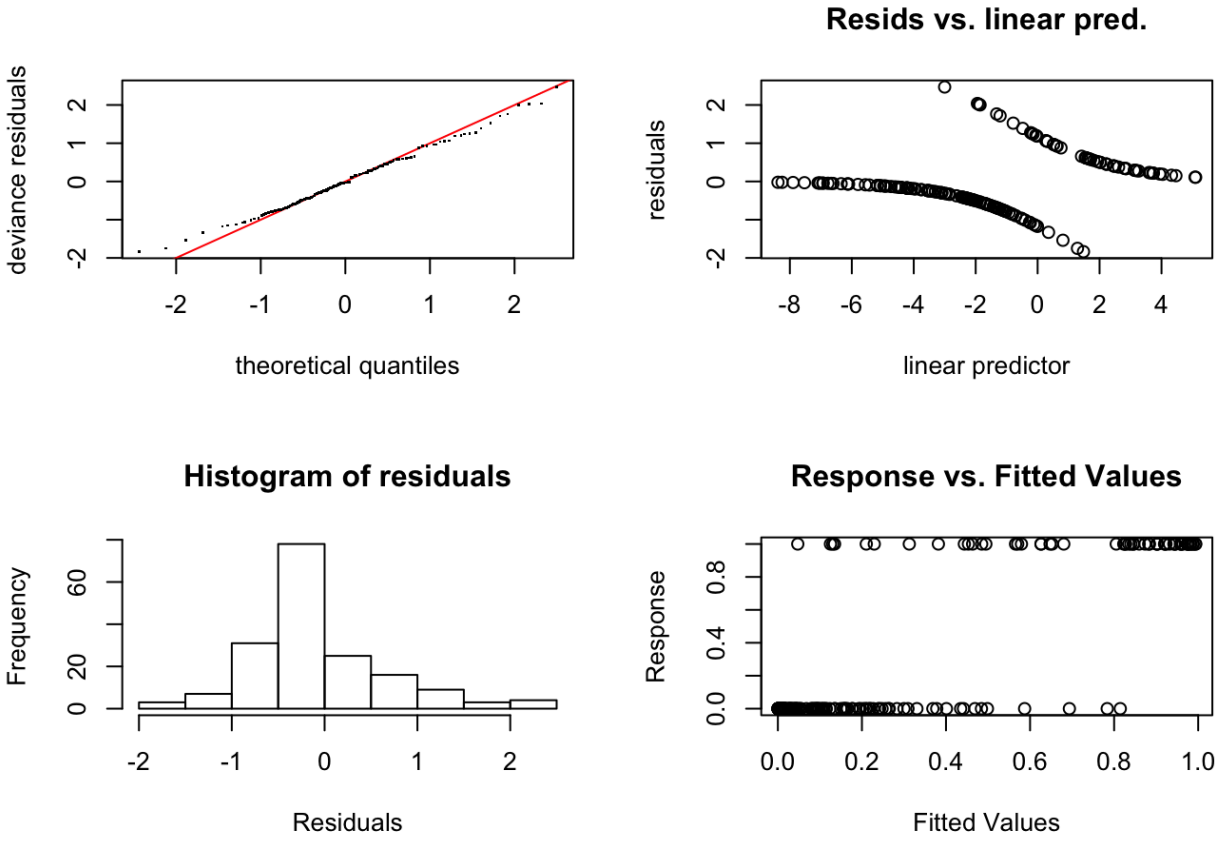

```
# Compare accuracy rate
mean((predict(mod.gambin_bestAIC, type = "response")>0.5) == tck_borrelia_train$borrPresent)
```

```
## [1] 0.875
```

```
mean((predict(mod.gambin_bestDevExpl, type = "response")>0.5) == tck_borrelia_train$borrPresent)
```

```
## [1] 0.9034091
```

```
# Honestly, the "deviance explained" looks better and makes more sense, in my opinion.

## FINAL MODEL:
frml.bin1 <- frml_bin1_bestDevexpl
mod.gambin<- gam(as.formula(frml.bin1)
                 # , offset=log(numberTested) # I include the offset in the formula instead, so that it is includ
ed in predictions. Including it here does NOT incorporate numberTested in predictiosn.
                 , data=tck_borrelia_train
                 , method="REML"
                 , family=binomial)
gam.check(mod.gambin)
```

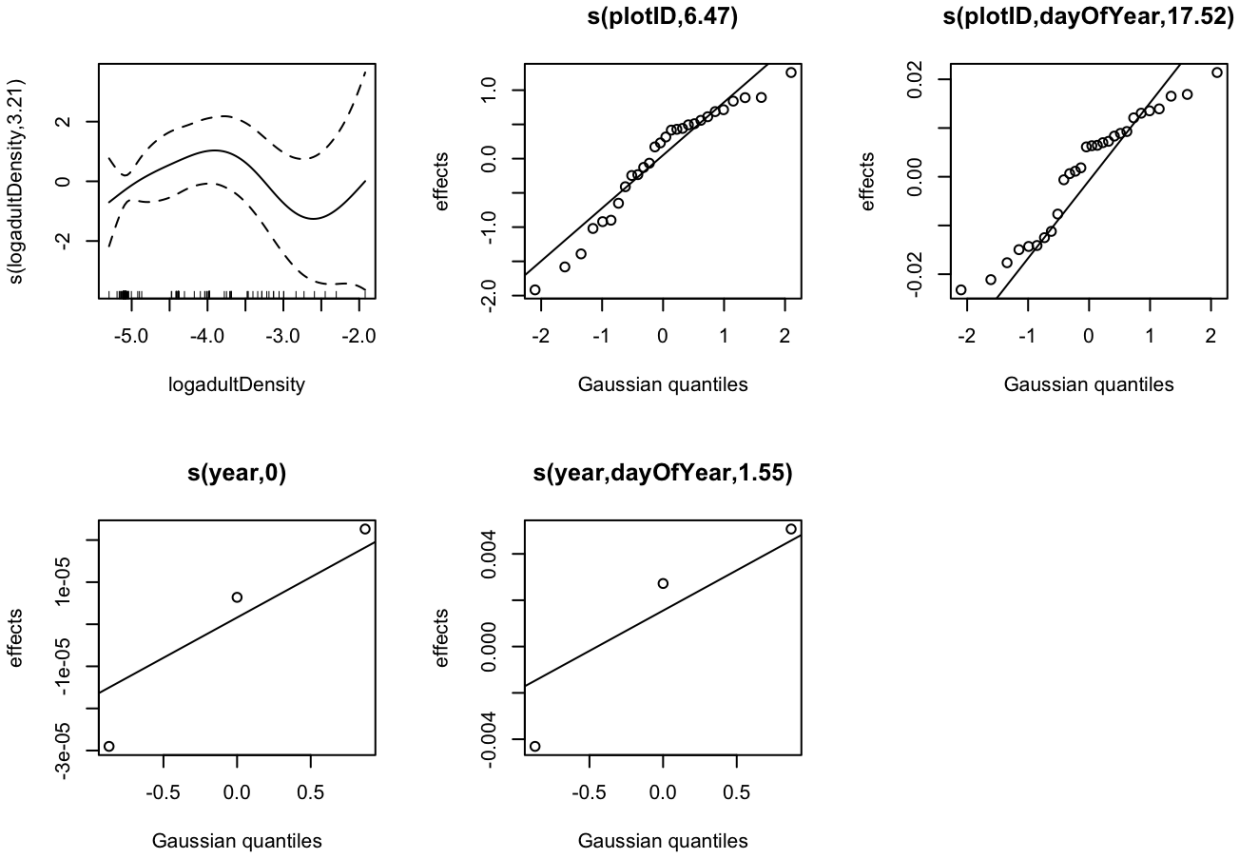


```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-3.141881e-05,-1.537113e-06]
## (score 88.37141 & scale 1).
## Hessian positive definite, eigenvalue range [3.14177e-05,3.677544].
## Model rank =  72 / 72
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                          k'      edf k-index p-value
## s(logadultDensity)  9.00e+00 3.21e+00    0.93    0.16
## s(plotID)           2.80e+01 6.47e+00      NA      NA
## s(plotID,dayOfYear) 2.80e+01 1.75e+01      NA      NA
## s(year)             3.00e+00 8.62e-05      NA      NA
## s(year,dayOfYear)   3.00e+00 1.55e+00      NA      NA
```

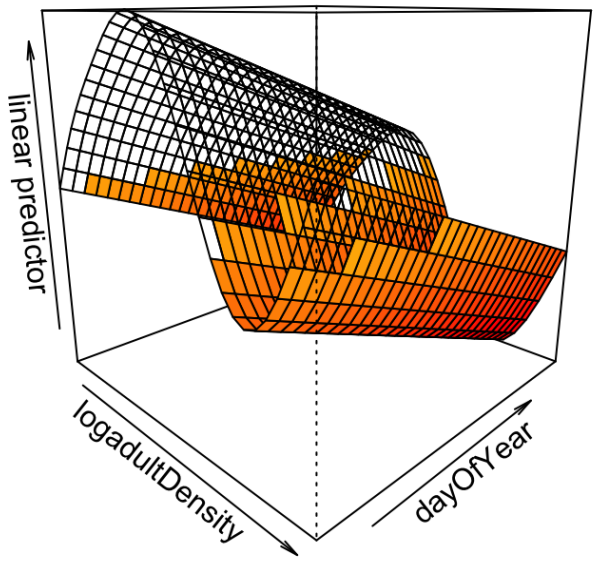
```
summary(mod.gambin)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## borrPresent ~ offset(log(numberTested)) + s(logadultDensity) +
##     s(plotID, bs = "re") + s(plotID, dayOfYear, bs = "re") +
##     s(year, bs = "re") + s(year, dayOfYear, bs = "re")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.9028     0.9139  -5.365  8.1e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                           edf Ref.df  Chi.sq p-value
## s(logadultDensity)  3.206e+00  3.881   4.797   0.254
## s(plotID)           6.472e+00 27.000  81.313   0.622
## s(plotID,dayOfYear) 1.752e+01 28.000 172.997   0.219
## s(year)             8.619e-05  2.000   0.000   0.763
## s(year,dayOfYear)   1.551e+00  3.000  27.936   0.117
##
## R-sq.(adj) =  0.584   Deviance explained = 73.1%
## -REML = 88.371  Scale est. = 1         n = 176
```

```
plot(mod.gambin, scale=0, pages=1)
```

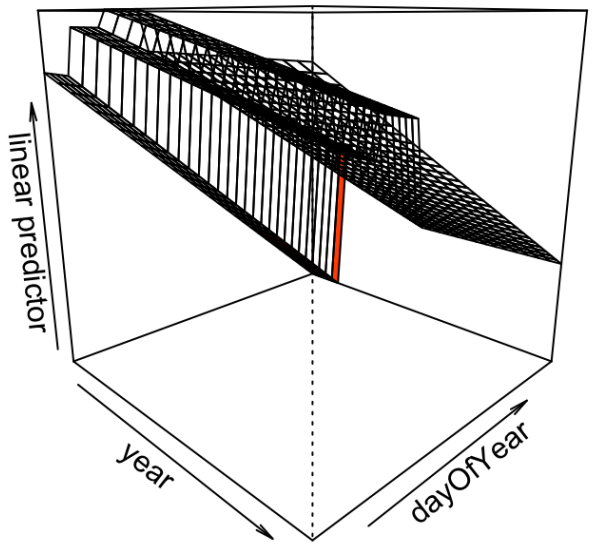



```
vis.gam(mod.gambin, view = c("logadultDensity","dayOfYear"), theta=45)
```



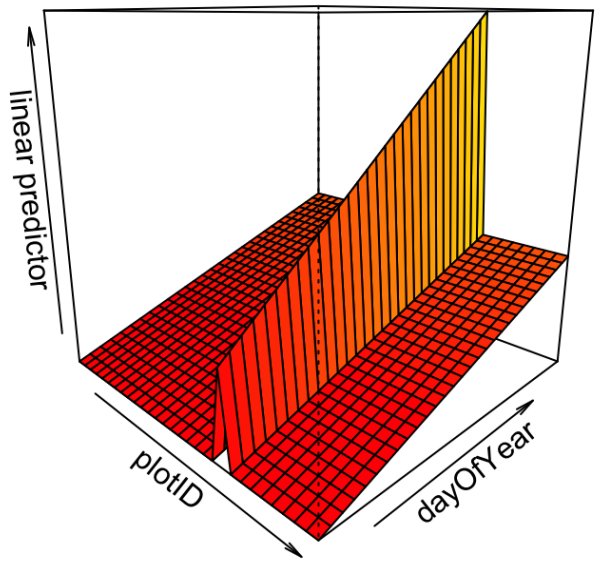

```
vis.gam(mod.gambin, view = c("year","dayOfYear"), theta=45)
```

```
## Warning in predict.gam(x, newdata = newd, se.fit = TRUE, type = type): factor
## levels 2017, 2018, 2019 not in original fit
```




```
vis.gam(mod.gambin, view = c("plotID","dayOfYear"), theta=45)
```

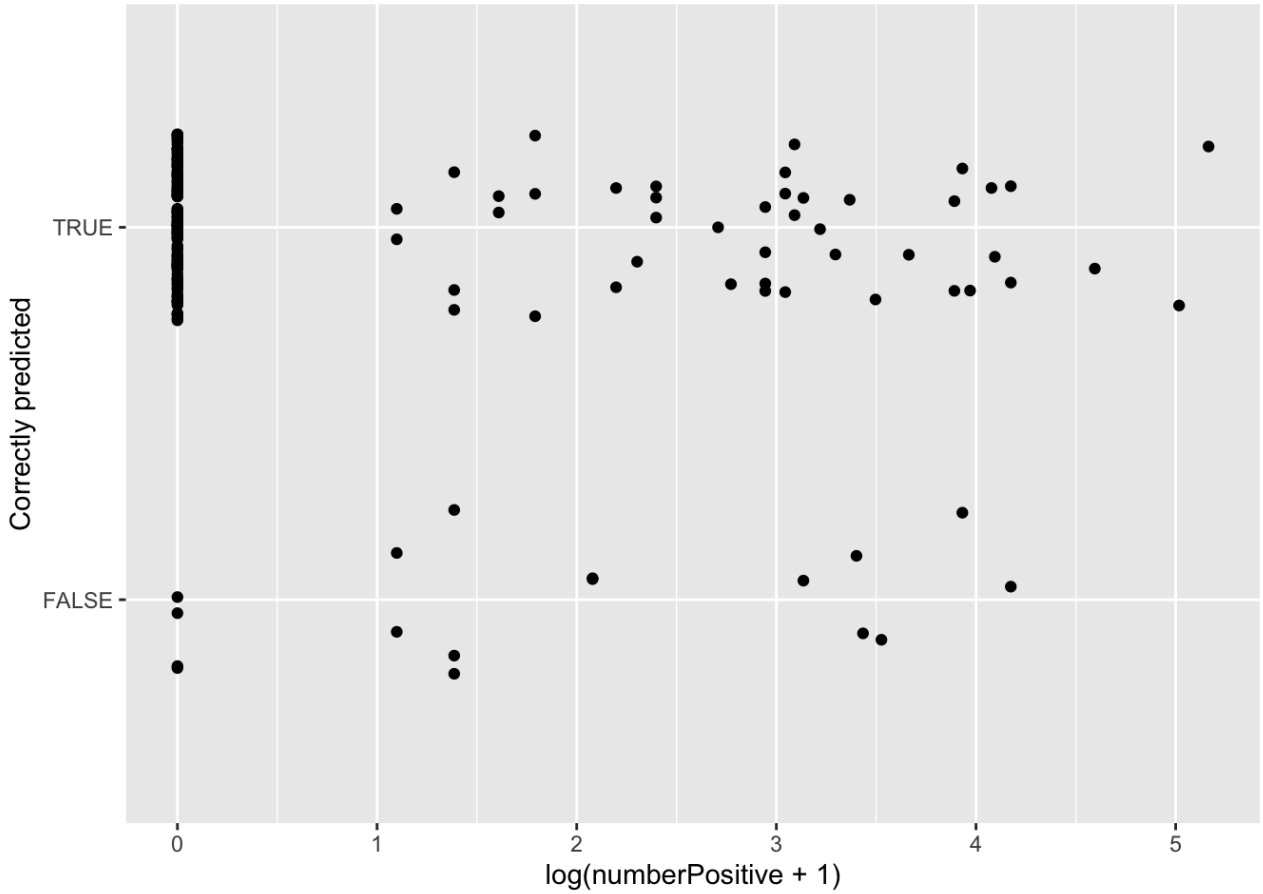
```
## Warning in predict.gam(x, newdata = newd, se.fit = TRUE, type = type): factor
## levels ABBY_001, ABBY_002, ABBY_003, ABBY_005, ABBY_006, ABBY_023, BARR_021,
## BARR_030, BARR_031, BARR_034, BARR_037, BARR_084, BART_002, BART_010, BART_011,
## BART_015, BART_019, BART_029, BLAN_002, BLAN_004, BLAN_008, BLAN_012, BLAN_015,
## BLAN_020, BONA_002, BONA_004, BONA_012, BONA_013, BONA_020 not in original fit
```



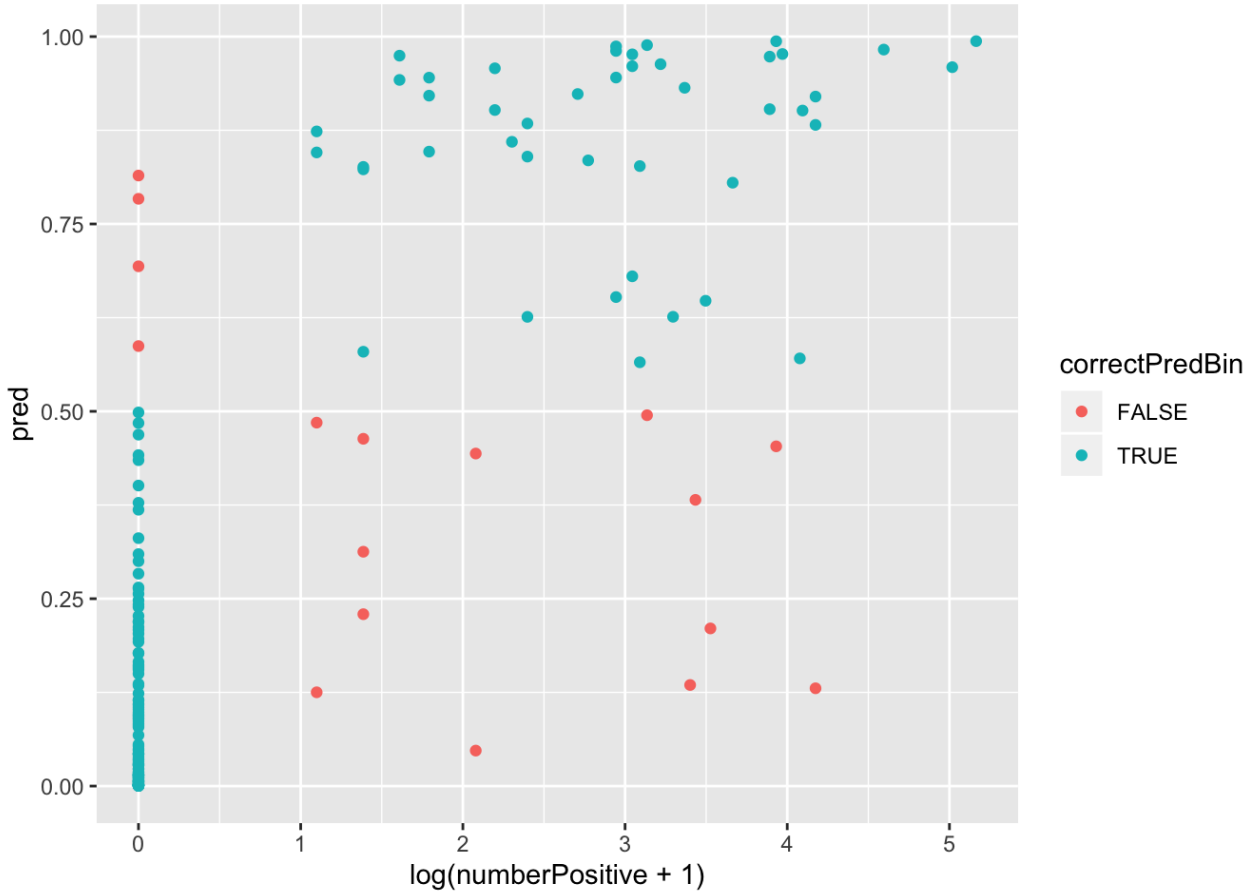

```
# Proportion of correct predictions
mean((predict(mod.gambin, type = "response")>0.5) == tck_borrelia_train$borrPresent)
```

```
## [1] 0.9034091
```

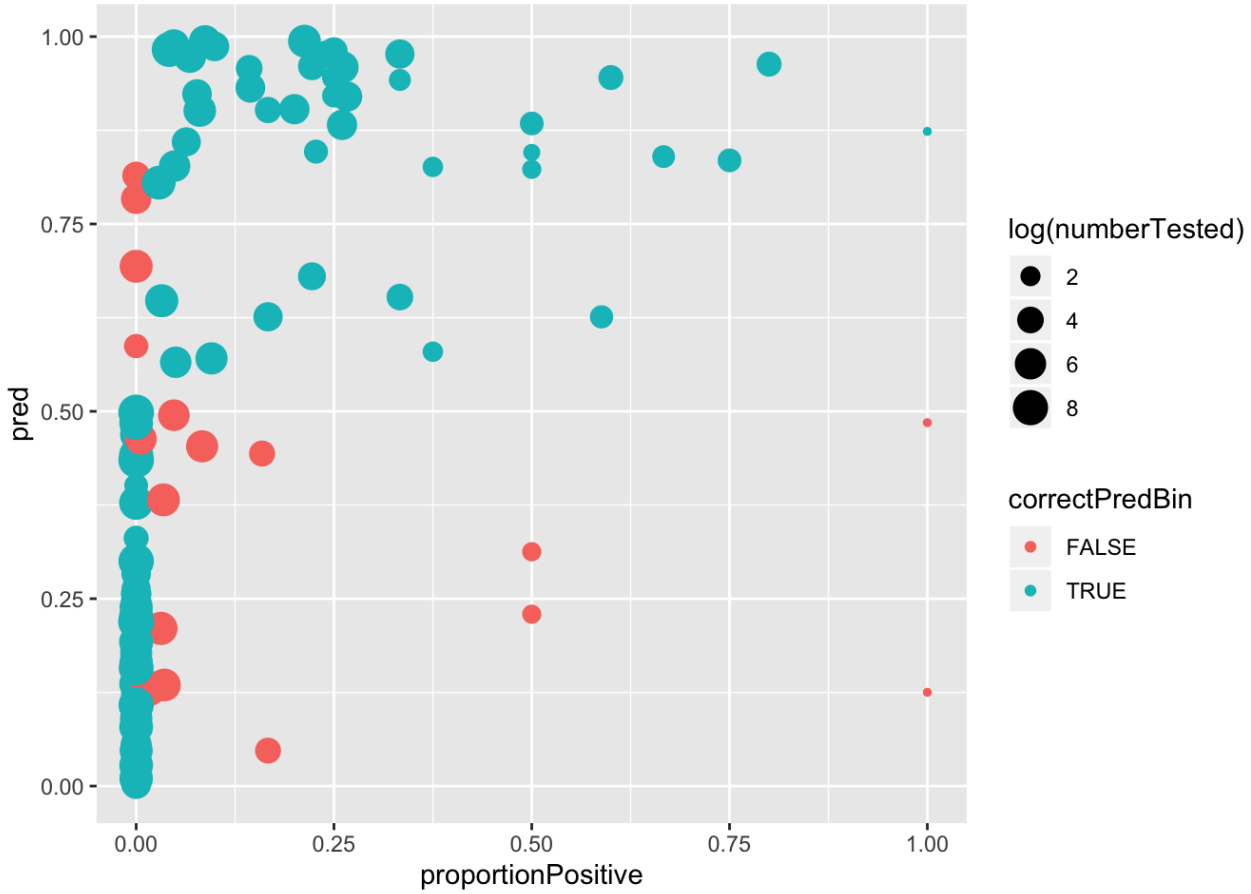
```
# Of those where they predicted presence/absence incorrectly, how "far" were they off?
# (i.e. were there are "negative" predictions that actually had a LOT of borrellia?)
tck_borrelia_train %>%
  mutate(pred =as.numeric(predict(mod.gambin, type="response"))
         ,correctPredBin = ((pred>0.5) == tck_borrelia_train$borrPresent) ) %>%
  ggplot() +geom_jitter(aes(x=log(numberPositive+1), y=correctPredBin), height=0.25, width=0) + ylab("Correctly p
redicted")
```



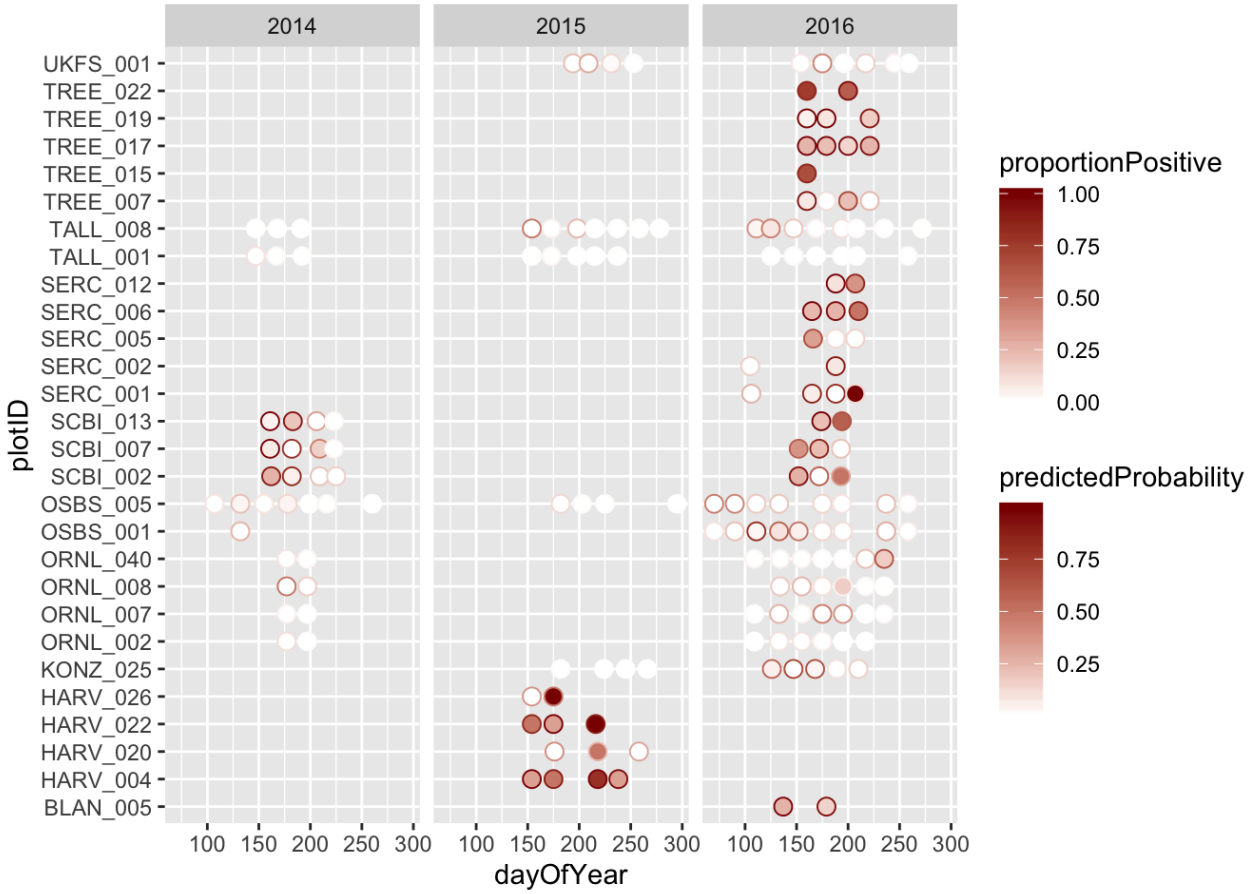

```
tck_borrelia_train %>%
  mutate(pred =as.numeric(plogis(predict(mod.gambin)))
         ,correctPredBin =((pred>0.5) == tck_borrelia_train$borrPresent) ) %>%
  ggplot() +geom_point(aes(x=log(numberPositive+1), y=pred, col=correctPredBin))
```




```
# Ideally, you have a positive, linear correlation between proportion positive and predicted probability. "Error
s" should be correlated with small sample size.
tck_borrelia_train %>%
  mutate(pred =as.numeric(plogis(predict(mod.gambin)))
         ,correctPredBin =((pred>0.5) == tck_borrelia_train$borrPresent) ) %>%
  ggplot() +geom_point(aes(x=proportionPositive, y=pred, col=correctPredBin, cex=log(numberTested)))
```




```
# What are the groups that they least accurately predicted?
tck_borrelia_train %>%
  mutate(predictedProbability =as.numeric(plogis(predict(mod.gambin)))
         ,correctPredBin =((predictedProbability>0.5) == tck_borrelia_train$borrPresent)
         # , proportionPositive = ifelse(proportionPositive==0, NA, proportionPositive)
         ) %>%
  ggplot() + geom_point(aes(x=dayOfYear, y=plotID, fill=proportionPositive, col=predictedProbability), pch=21,cex
=3) +
  scale_fill_gradient(low="white", high="darkred") +
  scale_color_gradient(low="white", high="darkred") +
  facet_grid(.~year)
```




```
# Is there a bias for predicting negative or positive results?
tck_borrelia_train %>%
  mutate(predictedProbability =as.numeric(plogis(predict(mod.gambin)))
         , predictedPA = (predictedProbability>0.5)
         ,correctPredBin =((predictedProbability>0.5) == tck_borrelia_train$borrPresent) ) %>%
  select(borrPresent,predictedPA) %>% table()
```

```
##            predictedPA
## borrPresent FALSE TRUE
##           0   115    4
##           1    13   44
```

```
4/(115+4) # False positive rate
```

```
## [1] 0.03361345
```

```
13/(13+44) # False negative rate
```

```
## [1] 0.2280702
```

```
# The false negative rate is actually a lot higher than the false positive rate.
# This means, on average, more samples are positive then you'd expect, given the model.


## So finally, we need to filter the dataset by the predicted "absent" and fit a poisson or negative binomial mod
el
# To include the maximum amount of samples as possible, I'm going to filter the dataset sort of un-usually--
# First, I am going to keep all positive borrelia results. Then, for each "negative" borrelia result,
# I will look at the predicted probability in our model and determine whether it is a "binomial/hurdle" negative,
or a "poisson/NB" negative.

tck_borrelia_filtBin <- tck_borrelia_train %>%
  mutate(pred=predict(mod.gambin, type="response")) %>%
  filter((borrPresent>0 | pred > 0.5))

# Double check I did the filtering correctly
tck_borrelia_filtBin %>%
  mutate(predictedPA = (pred > 0.5)) %>%
  select(borrPresent,predictedPA) %>% table()
```

```
##            predictedPA
## borrPresent FALSE TRUE
##           0     0    4
##           1    13   44
```

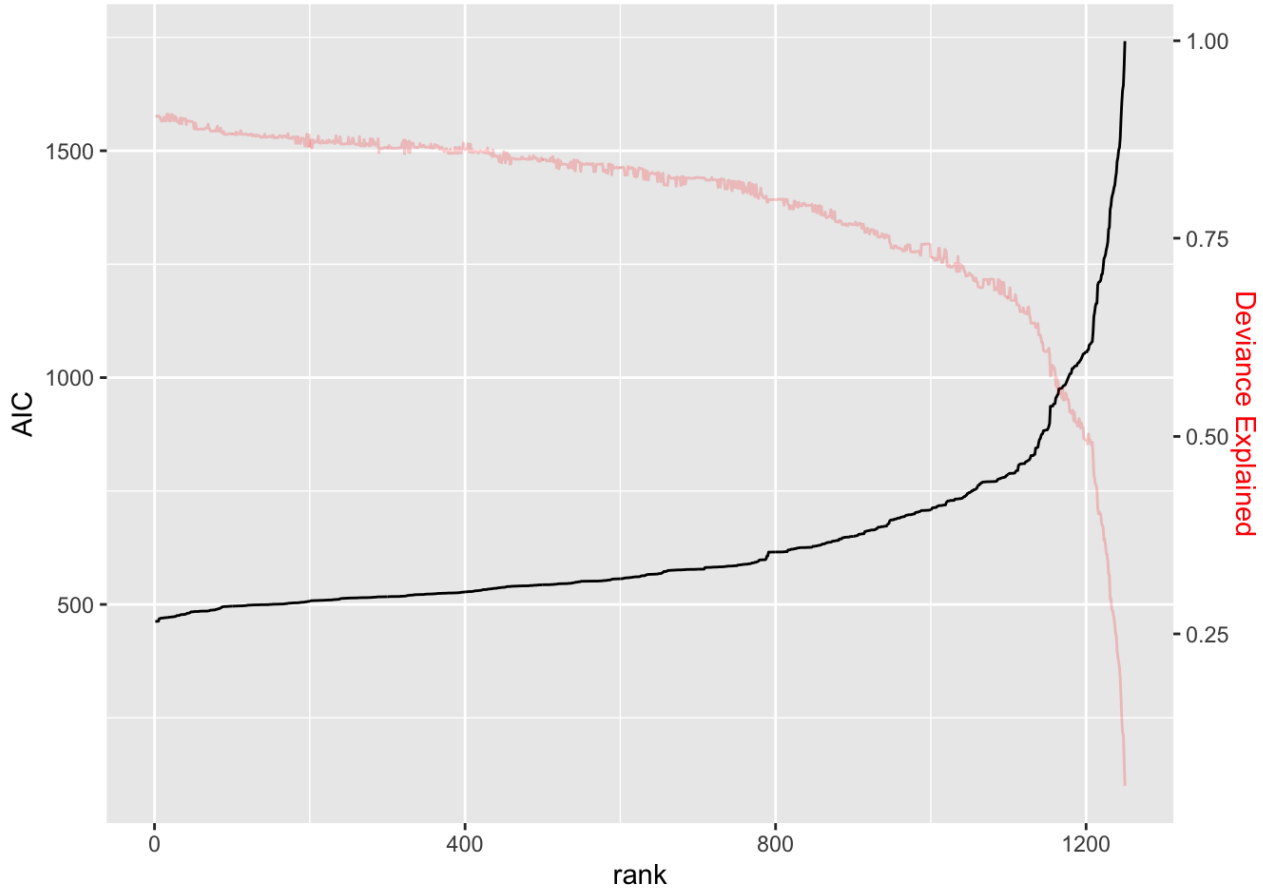
```
# make year a factor
tck_borrelia_filtBin <- tck_borrelia_filtBin %>%
  mutate(year=factor(year))
#### Part II: the second GAM model for abundance ####

### Try the whole model section process again with a binomial
# I'm increasing spline smoothness here, because the AIC values change drastically when I don't include them
allPred2 <- c("s(dayOfYear, sp=1)","nlcdClass","s(elevation)"
              ,"s(logNLtckDensity, sp=1)","s(logadultDensity, sp=1)","s(lognymphDensity, sp=1)"
              ,"s(plotID, bs='re')", "s(plotID, dayOfYear, bs='re')"
              ,"s(year, bs='re')", "s(year, dayOfYear, bs='re')"
              , "domainID", "s(domainID, bs='re')"
)
ignoreCombos <- list(c("domainID","s(domainID, bs='re')")
                     , c("s(logNLtckDensity, sp=1)","s(logadultDensity, sp=1)","s(lognymphDensity, sp=1)")
                     , c("s(logNLtckDensity, sp=1)","s(lognymphDensity, sp=1)")
                     , c("s(logNLtckDensity, sp=1)","s(logadultDensity, sp=1)")
)
nrow(tck_borrelia_filtBin)
```

```
## [1] 61
```

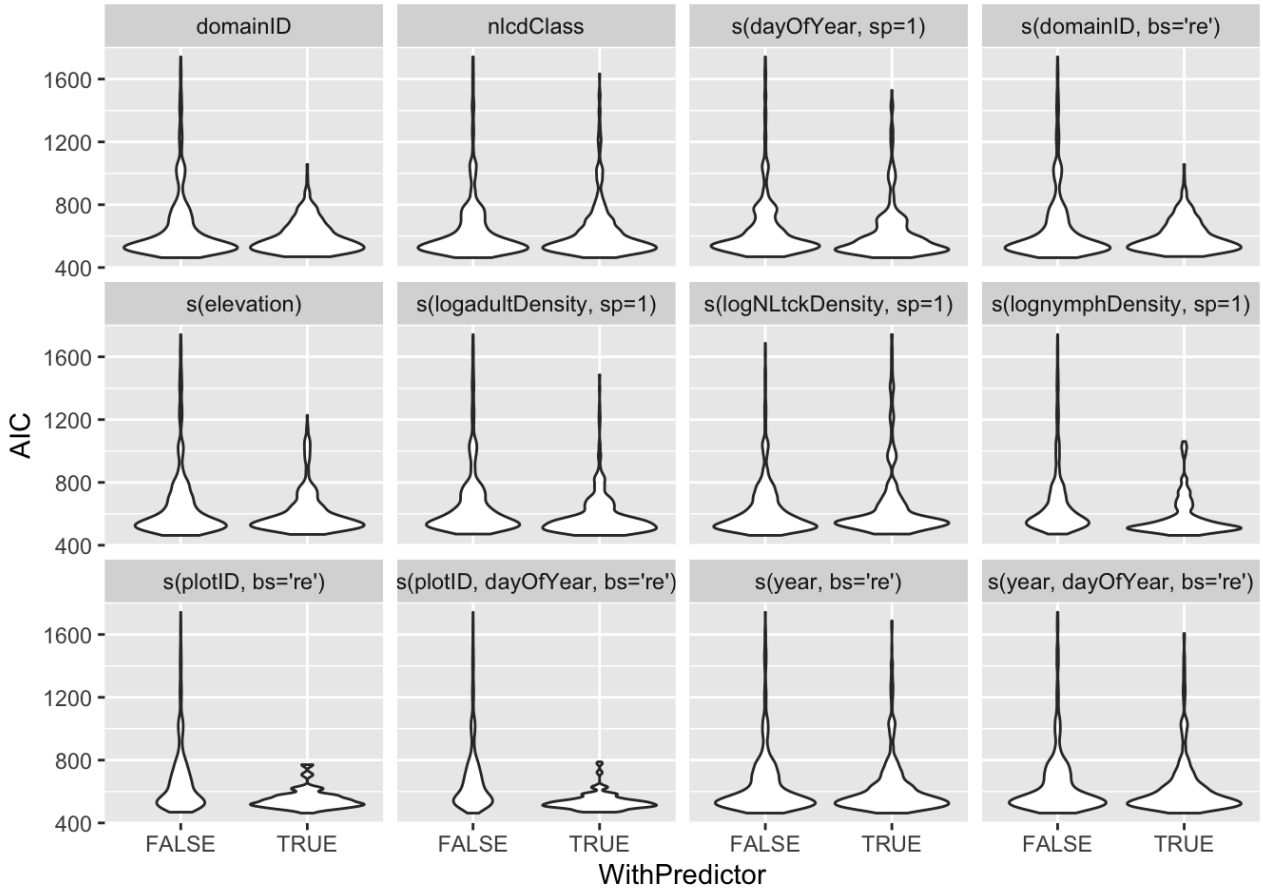
```
if ( FALSE ) {
  allAIC_2 <- stepGAM(dat=tck_borrelia_filtBin, predictors = allPred2, response = "cbind(numberPositive,numberTes
ted)", family = binomial, ignore.combos=ignoreCombos)
  allAIC_2_filt <- allAIC_2 %>% filter(!is.na(AIC)) %>%
    mutate(AIC=as.numeric(AIC)
           , Dev.expl = as.numeric(Dev.expl)
           , REML = as.numeric(REML)) %>%
    arrange(AIC) %>%mutate(rank=seq(1:n())) %>%
    filter(AIC<2000)
  save(allAIC_2, file="allAIC_2.RData")
  save(allAIC_2_filt, file="allAIC_2_filt.RData")
  } else {
  load("allAIC_2.RData")
  load("allAIC_2_filt.RData")
}
summary(allAIC_2)
```

```
##      AIC                REML             Dev.expl           formula
##  Length:4095        Length:4095        Length:4095        Length:4095
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##  s(dayOfYear, sp=1) nlcdClass       s(elevation)    s(logNLtckDensity, sp=1)
##  Mode :logical      Mode :logical   Mode :logical   Mode :logical
##  FALSE:698          FALSE:641       FALSE:698       FALSE:992
##  TRUE :552          TRUE :609       TRUE :552       TRUE :258
##  NA's :2845         NA's :2845      NA's :2845      NA's :2845
##  s(logadultDensity, sp=1) s(lognymphDensity, sp=1) s(plotID, bs='re')
##  Mode :logical            Mode :logical            Mode :logical
##  FALSE:812                FALSE:812                FALSE:860
##  TRUE :438                TRUE :438                TRUE :390
##  NA's :2845               NA's :2845               NA's :2845
##  s(plotID, dayOfYear, bs='re') s(year, bs='re') s(year, dayOfYear, bs='re')
##  Mode :logical                 Mode :logical    Mode :logical
##  FALSE:860                     FALSE:641        FALSE:641
##  TRUE :390                     TRUE :609        TRUE :609
##  NA's :2845                    NA's :2845       NA's :2845
##   domainID       s(domainID, bs='re')
##  Mode :logical   Mode :logical
##  FALSE:855       FALSE:855
##  TRUE :395       TRUE :395
##  NA's :2845      NA's :2845
```

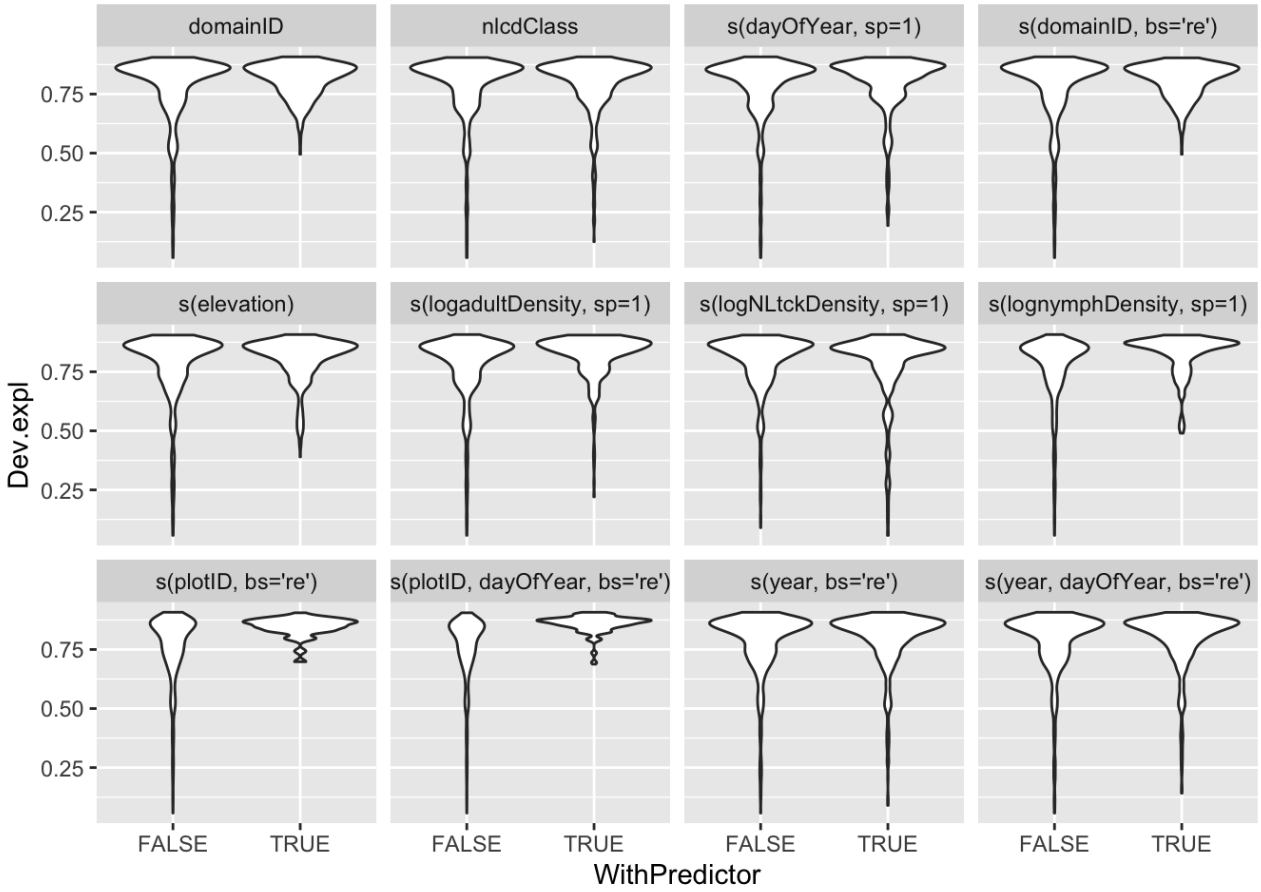
```
# allAIC_2%>%
#   arrange(AIC) %>%View()
#
# # Let's see distribution of AIC values
# allAIC_2 %>%
#   filter(!is.na(AIC), !is.na(Dev.expl)) %>%
#   arrange(AIC) %>% mutate(rank=seq(1,n())) %>%
#   ggplot() +geom_line(aes(x=rank, y=AIC))+
#   geom_line(aes(x=rank, y=Dev.expl*(max(allAIC_2$AIC))), col="red", alpha=0.2) + scale_y_continuous(sec.axis=se
c_axis(~./(max(allAIC_2$AIC)), name="Deviance Explained")) +
#   theme(axis.title.y.right = element_text(colour = "red"))
#

allAIC_2_filt %>%
  ggplot() +geom_line(aes(x=rank, y=AIC))+
  geom_line(aes(x=rank, y=Dev.expl*(max(allAIC_2_filt$AIC))), col="red", alpha=0.2) + scale_y_continuous(sec.axis
=sec_axis(~./(max(allAIC_2_filt$AIC)), name="Deviance Explained")) +
  theme(axis.title.y.right = element_text(colour = "red"))
```




```
# Compare AIC for models with and without each predictor
allAIC_2_filt %>% gather(-c(AIC, REML, Dev.expl, formula, rank), key=Predictor, value=WithPredictor) %>%
  select(AIC, REML, Dev.expl, Predictor, WithPredictor) %>%
  mutate(WithPredictor = ifelse(WithPredictor==0, FALSE, TRUE)) %>%
  ggplot() + geom_violin(aes(x=WithPredictor, y=AIC))  +  facet_wrap(.~Predictor)
```



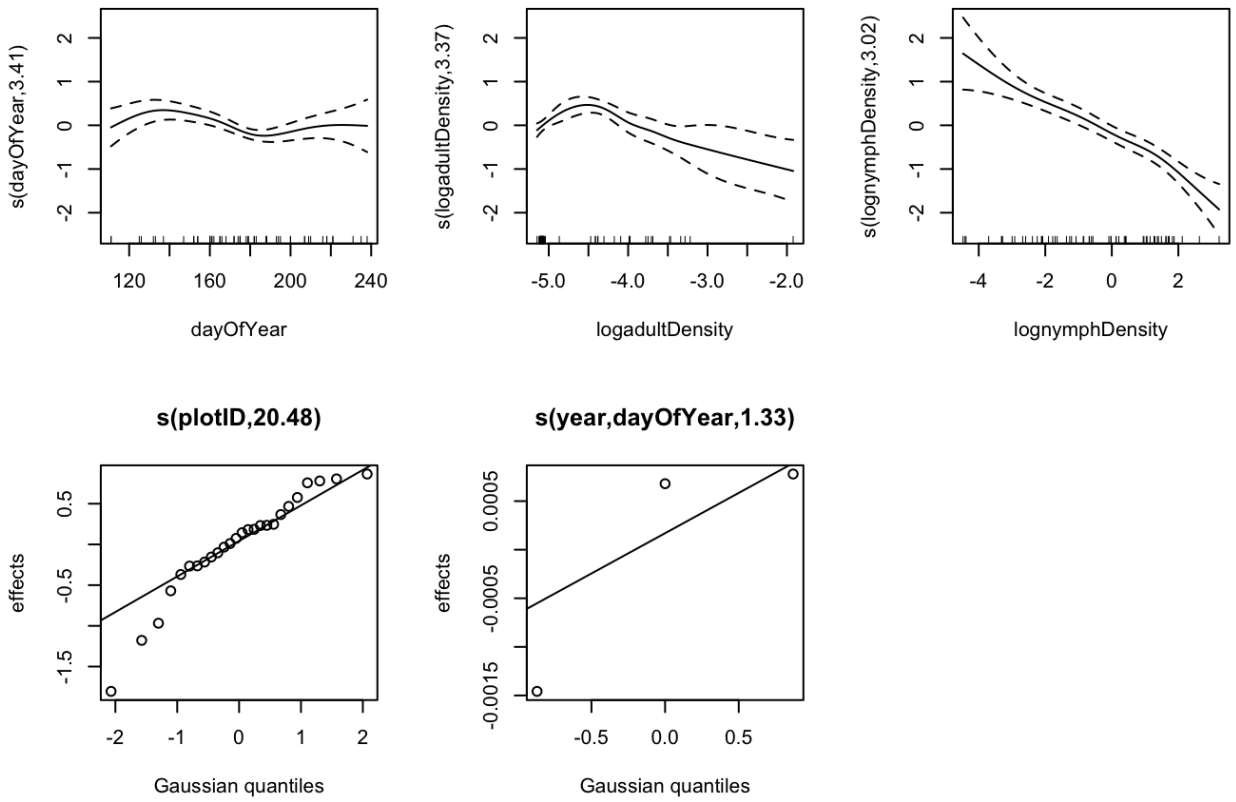

```
# Compare deviance explained for models with and without each predictor
allAIC_2_filt %>% gather(-c(AIC, REML, Dev.expl, formula,rank), key=Predictor, value=WithPredictor) %>%
  select(AIC, REML, Dev.expl, Predictor, WithPredictor) %>%
  mutate(WithPredictor = ifelse(WithPredictor==0, FALSE, TRUE)) %>%
  ggplot() + geom_violin(aes(x=WithPredictor, y=Dev.expl))  +  facet_wrap(.~Predictor)
```



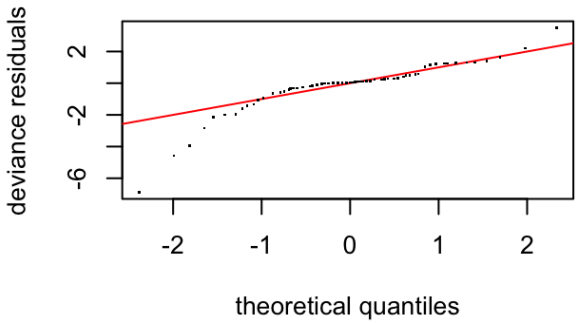

```
frml_bin2_bestAIC <- allAIC_2_filt[allAIC_2_filt$AIC==min(allAIC_2_filt$AIC),"formula"]
frml_bin2_bestDevexpl <- allAIC_2_filt[allAIC_2_filt$Dev.expl==max(allAIC_2_filt$Dev.expl),"formula"]
# frml_bin2_bestAIC_adjdayOfYear <- "cbind(numberPositive,numberTested) ~ s(dayOfYear, sp=1) + nlcdClass + s(logN
LtckDensity, sp=1) + s(plotID, bs='re') + s(plotID, dayOfYear, bs='re') + s(year, bs='re') + s(year, dayOfYear, b
s='re') + domainID"
# they're the same

## Best AIC model
mod.gambin2_bestAIC <- gam(as.formula(frml_bin2_bestAIC)
                           , data=tck_borrelia_filtBin
                           , method="REML"
                           , family=binomial)
summary(mod.gambin2_bestAIC)
```

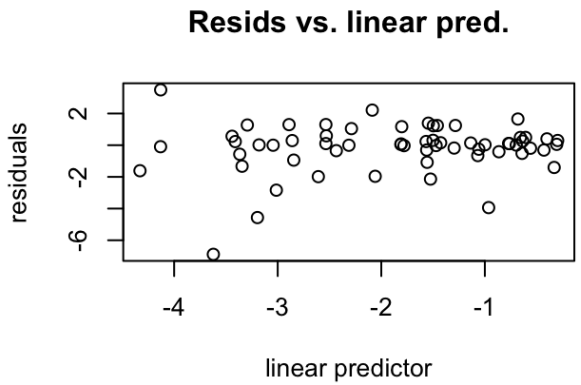
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(numberPositive, numberTested) ~ s(dayOfYear, sp = 1) +
##     s(logadultDensity, sp = 1) + s(lognymphDensity, sp = 1) +
##     s(plotID, bs = "re") + s(year, dayOfYear, bs = "re")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.9956     0.2245  -8.888   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df Chi.sq  p-value
## s(dayOfYear)        3.413  4.119  25.47 0.000184 ***
## s(logadultDensity)  3.372  3.758  44.58 1.11e-07 ***
## s(lognymphDensity)  3.020  3.672  80.15 3.80e-16 ***
## s(plotID)          20.484 25.000 293.13 2.59e-15 ***
## s(year,dayOfYear)   1.330  2.000 216.82 0.002659 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.823   Deviance explained = 90.5%
## -REML = 269.44  Scale est. = 1         n = 61
```

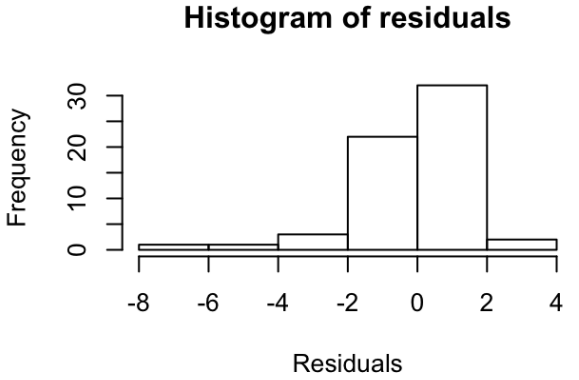
```
plot(mod.gambin2_bestAIC, pages=1)
```

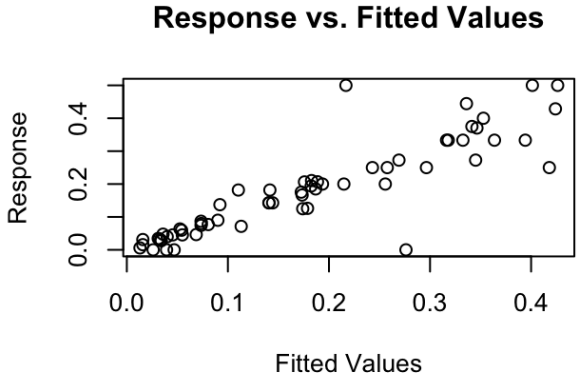



```
gam.check(mod.gambin2_bestAIC)
```


deviance residuals
2
-2
-6
-2
-1
0
1
2
theoretical quantiles


Resids vs. linear pred.
residuals
2
-2
-6
-4
-3
-2
-1
linear predictor


Histogram of residuals
Frequency
0
10
20
30
-8
-6
-4
-2
0
2
4
Residuals

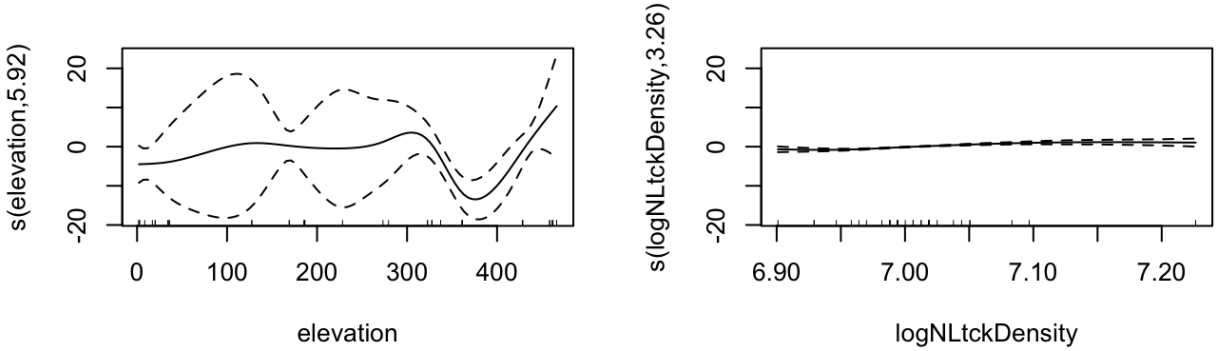

Response vs. Fitted Values
Response
0.0
0.2
0.4
0.0
0.1
0.2
0.3
0.4
Fitted Values

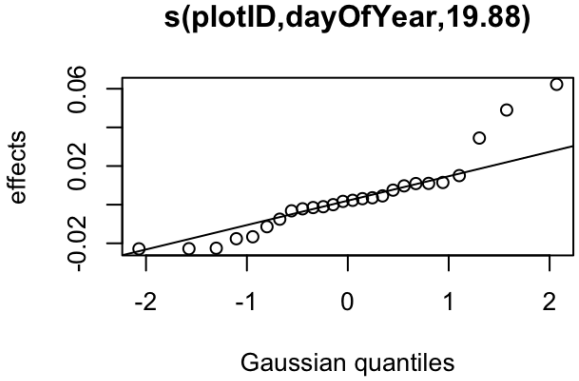
```
##
## Method: REML   Optimizer: outer newton
## full convergence after 5 iterations.
## Gradient range [-1.004244e-09,3.619618e-09]
## (score 269.4377 & scale 1).
## Hessian positive definite, eigenvalue range [0.542399,8.018056].
## Model rank =  57 / 57
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                       k'   edf k-index p-value
## s(dayOfYear)        9.00  3.41    1.20    0.95
## s(logadultDensity)  9.00  3.37    1.12    0.88
## s(lognymphDensity)  9.00  3.02    1.27    0.98
## s(plotID)          26.00 20.48      NA      NA
## s(year,dayOfYear)   3.00  1.33      NA      NA
```

```
## Best Dev expl model
mod.gambin2_bestdevexpl <- gam(as.formula(frml_bin2_bestDevexpl)
                           , data=tck_borrelia_filtBin
                           , method="REML"
                           , family=binomial)
summary(mod.gambin2_bestdevexpl)
```

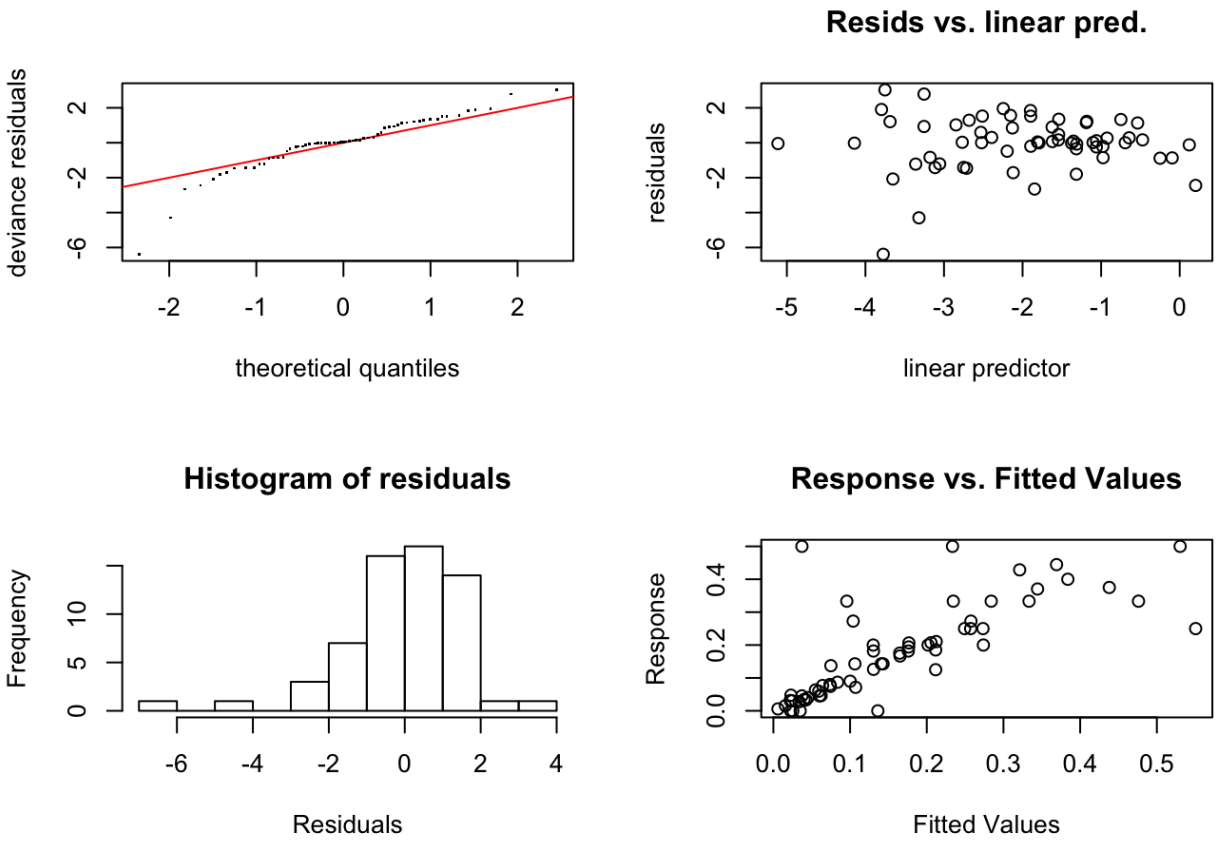
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(numberPositive, numberTested) ~ nlcdClass + s(elevation) +
##     s(logNLtckDensity, sp = 1) + s(plotID, dayOfYear, bs = "re") +
##     domainID
##
## Parametric coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -0.04672    9.09647  -0.005    0.996
## nlcdClassdeciduousForest  -0.05289    7.63411  -0.007    0.994
## nlcdClassevergreenForest   3.11336    7.53570   0.413    0.679
## nlcdClassmixedForest      -3.32665    7.46055  -0.446    0.656
## domainIDD02                1.63473    3.60013   0.454    0.650
## domainIDD03               -0.73083   10.88858  -0.067    0.946
## domainIDD05              -11.68637    8.51298  -1.373    0.170
## domainIDD06               -0.69852    3.89949  -0.179    0.858
## domainIDD07               -5.45374    5.01527  -1.087    0.277
## domainIDD08              -10.95766   10.90795  -1.005    0.315
##
## Approximate significance of smooth terms:
##                         edf Ref.df Chi.sq  p-value
## s(elevation)          5.923  6.211  53.36 9.81e-10 ***
## s(logNLtckDensity)    3.255  3.881  76.73 2.19e-15 ***
## s(plotID,dayOfYear) 19.882 24.000 207.78  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.746   Deviance explained = 90.8%
## -REML = 292.66  Scale est. = 1         n = 61
```

```
plot(mod.gambin2_bestdevexpl, pages=1)
```






```
gam.check(mod.gambin2_bestdevexpl)
```

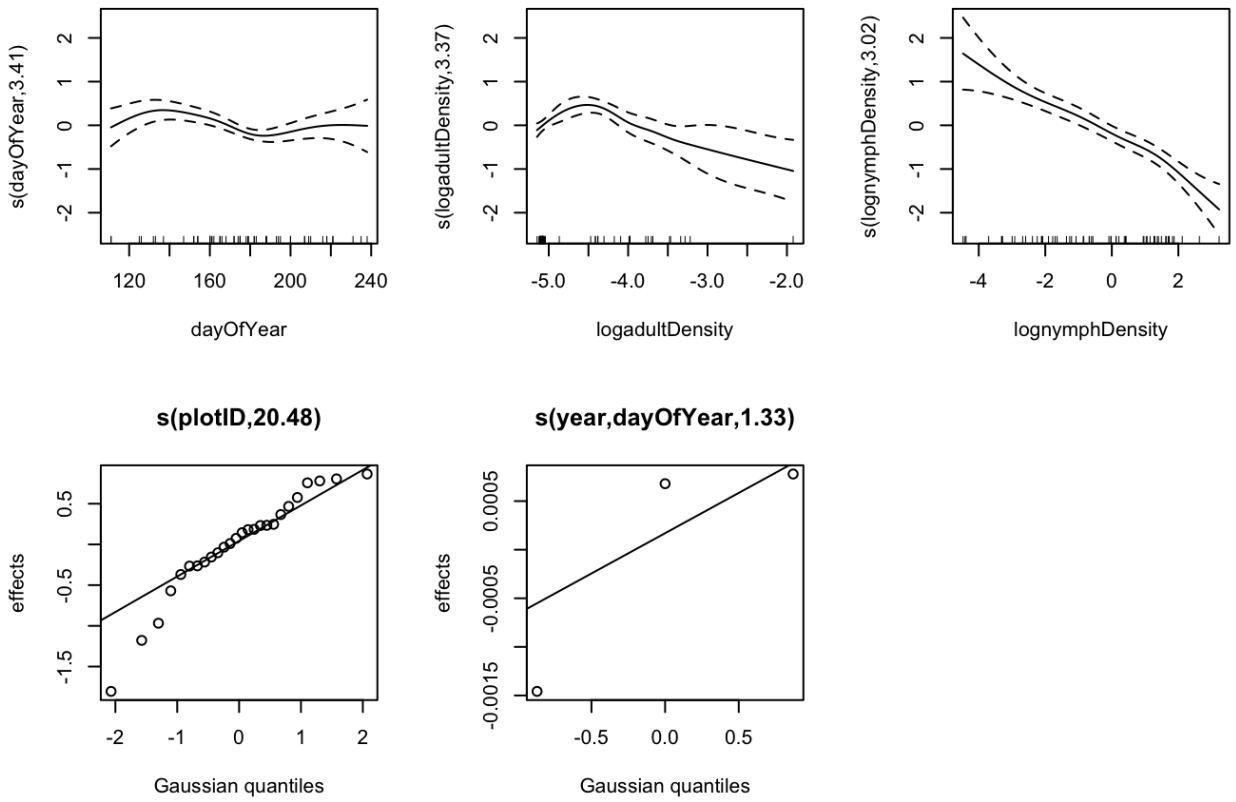



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-9.354166e-08,4.959567e-07]
## (score 292.655 & scale 1).
## Hessian positive definite, eigenvalue range [1.90978,6.107993].
## Model rank =  54 / 54
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                         k'   edf k-index p-value
## s(elevation)          9.00  5.92    1.40    1.00
## s(logNLtckDensity)    9.00  3.26    1.31    0.99
## s(plotID,dayOfYear) 26.00 19.88      NA      NA
```

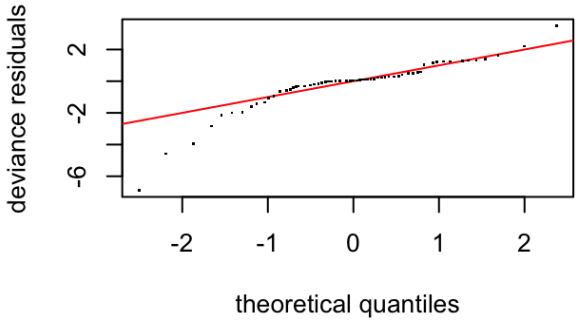
```
### BEST MODEL:
frml.bin2 <- frml_bin2_bestAIC
mod.gambin_2 <- gam(as.formula(frml.bin2)
                    , data=tck_borrelia_filtBin
                    , method="REML"
                    , family=binomial)
summary(mod.gambin_2)
```

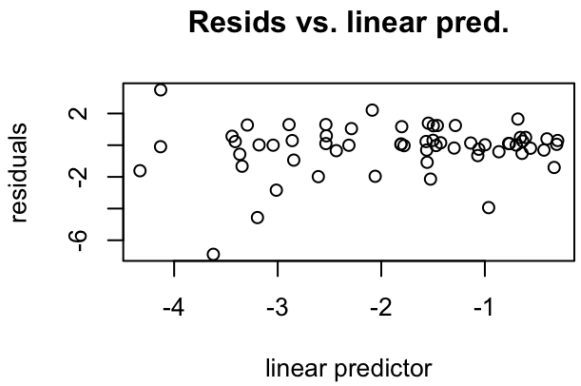
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(numberPositive, numberTested) ~ s(dayOfYear, sp = 1) +
##     s(logadultDensity, sp = 1) + s(lognymphDensity, sp = 1) +
##     s(plotID, bs = "re") + s(year, dayOfYear, bs = "re")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.9956     0.2245  -8.888   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df Chi.sq  p-value
## s(dayOfYear)        3.413  4.119  25.47 0.000184 ***
## s(logadultDensity)  3.372  3.758  44.58 1.11e-07 ***
## s(lognymphDensity)  3.020  3.672  80.15 3.80e-16 ***
## s(plotID)          20.484 25.000 293.13 2.59e-15 ***
## s(year,dayOfYear)   1.330  2.000 216.82 0.002659 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.823   Deviance explained = 90.5%
## -REML = 269.44  Scale est. = 1         n = 61
```

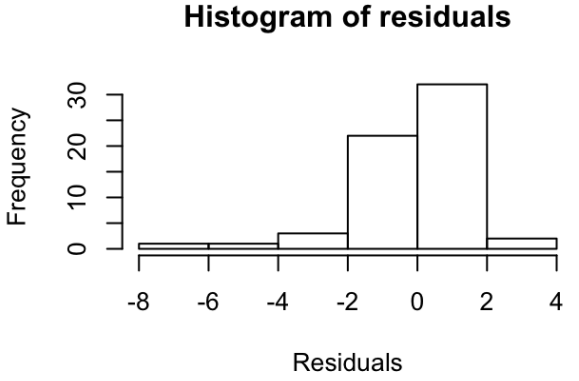
```
plot(mod.gambin_2, pages=1)
```

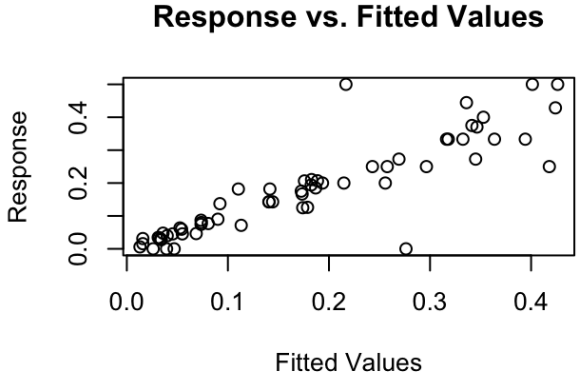



```
gam.check(mod.gambin_2)
```


deviance residuals
2
-2
-6
-2
-1
0
1
2
theoretical quantiles

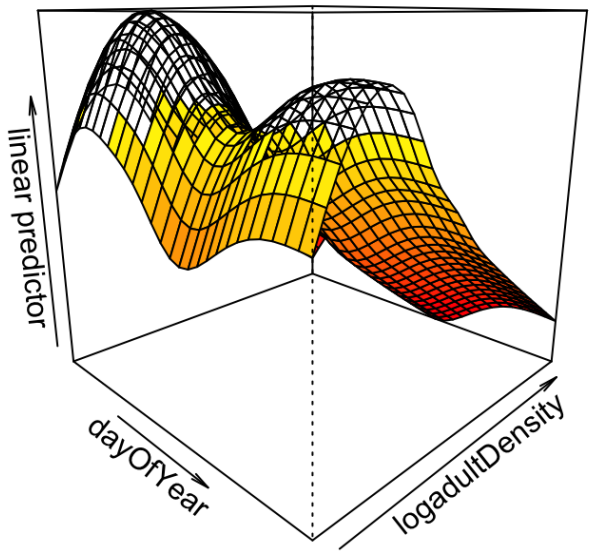

Resids vs. linear pred.
residuals
2
-2
-6
-4
-3
-2
-1
linear predictor


Histogram of residuals
Frequency
0
10
20
30
-8
-6
-4
-2
0
2
4
Residuals

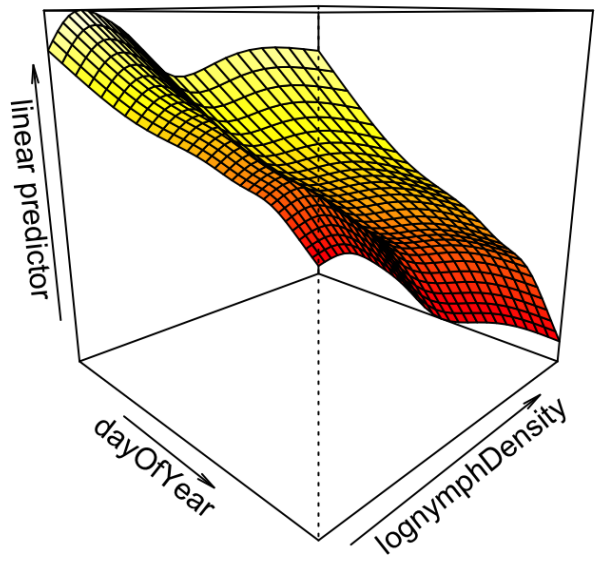

Response vs. Fitted Values
Response
0.0
0.2
0.4
0.0
0.1
0.2
0.3
0.4
Fitted Values

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 5 iterations.
## Gradient range [-1.004244e-09,3.619618e-09]
## (score 269.4377 & scale 1).
## Hessian positive definite, eigenvalue range [0.542399,8.018056].
## Model rank =  57 / 57
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                       k'   edf k-index p-value
## s(dayOfYear)        9.00  3.41    1.20    0.95
## s(logadultDensity)  9.00  3.37    1.12    0.82
## s(lognymphDensity)  9.00  3.02    1.27    0.99
## s(plotID)          26.00 20.48      NA      NA
## s(year,dayOfYear)   3.00  1.33      NA      NA
```

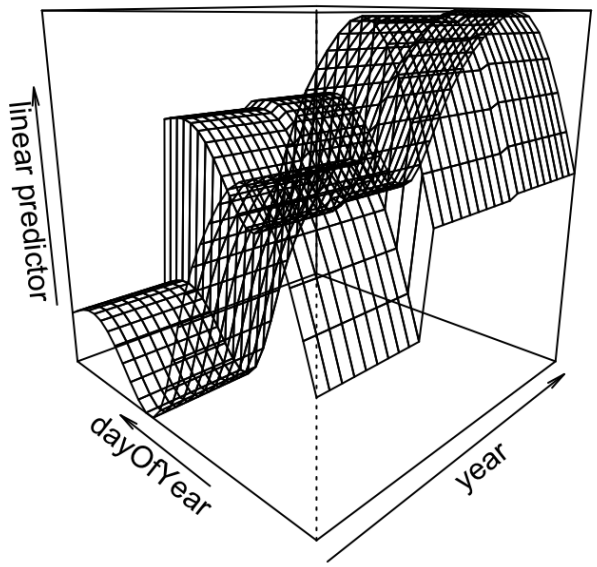
```
vis.gam(mod.gambin_2, view = c("dayOfYear","logadultDensity"), theta=45)
```



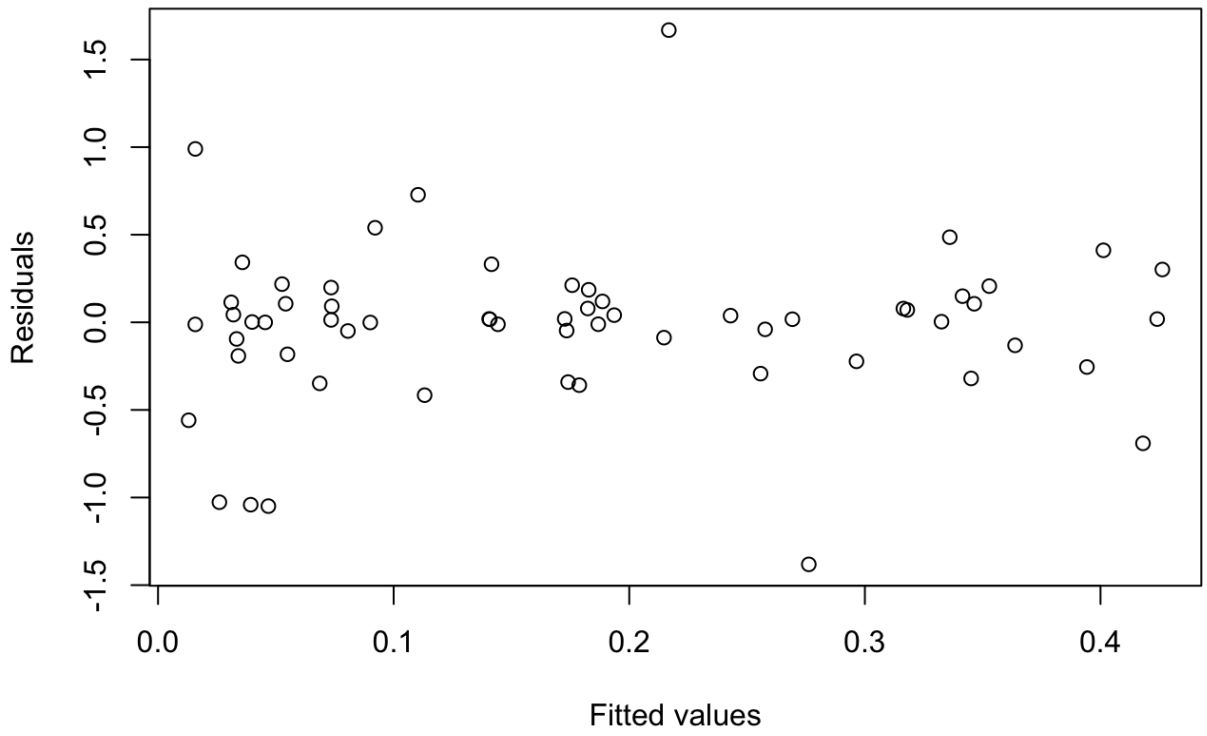

```
vis.gam(mod.gambin_2, view = c("dayOfYear","lognymphDensity"), theta=45)
```




```
vis.gam(mod.gambin_2, view = c("year","dayOfYear"), theta=-45)
```




```
# Residuals
plot(mod.gambin_2$residuals ~ mod.gambin_2$fitted.values, ylab="Residuals",xlab="Fitted values")
```




```
#### Part III: Trying to predict 2017 #####

predictions_over_doy_unprocessed <- data.frame(matrix(ncol=35, dimnames = list(NULL, c("pred_type","pred_bin","pr
ed_bin2","se","se2","doy_cutoff",colnames(tck_borrelia_test)))))
pb <- txtProgressBar(title = "progress bar", min = 0,
                     max = length(daysIn2017), style=3)
```

```
##
  |
  |                                                                      |   0%
```

```
## Should adjust model to NOT include an interac
# frml.bin1 <- "borrPresent ~ offset(log(numberTested))+s(logadultDensity)+s(plotID, bs='re')+s(plotID, dayOfYea
r, bs='re')+s(year, bs='re')"
# frml.bin2 <- "cbind(numberPositive,numberTested) ~ s(dayOfYear, sp=1) + s(logadultDensity, sp=1) + s(lognymphDe
nsity, sp=1) + s(plotID, bs='re') + s(year, bs='re')"
for ( i in 1:length(daysIn2017) ) {
  doy <- daysIn2017[i]
  tempTrain <- subsetData[[paste0(doy)]][['train']]
  tempTrain2017 <- tempTrain %>% filter(year==2017) %>% arrange(dayOfYear)
  tempTest <- subsetData[[paste0(doy)]][['test']]
  ### Part I: Presence/absence
  tempFit <- gam(as.formula(frml.bin1), data=tempTrain, family = binomial(), method = "REML")
  tempNewPredict <- predict(tempFit, newdata = tempTest, type = "link", se.fit = TRUE)
  tempOldPredict <- predict(tempFit, newdata = tempTrain2017, type="link", se.fit = TRUE)
  tempTrainPredict <- predict(tempFit, type='link', se.fit=TRUE)

  # #store predictions
  # predictions_over_doy_unprocessed <-rbind(predictions_over_doy_unprocessed, cbind(data.frame(pred_type=c(rep
("train_pred",length(tempOldPredict$fit)),rep("test_pred",length(tempNewPredict$fit)))
  #                                 ,  pred=c(tempOldPredict$fit, tempNewPredict$fit)
  #                                 , se = c(tempOldPredict$se.fit, tempNewPredict$se.fit)
  #                                 , doy_cutoff = rep(doy, nrow(tck_borrelia_test))
  #                                 ),rbind(tempTrain2017,tempTest)))

  ### Part II: prevalence
  # Filter trainig dataset to exclude zeros
  tempTrain2 <- tempTrain %>%
    cbind(data.frame(pred_bin1=tempTrainPredict$fit)) %>%
    filter(pred_bin1>0.5 | borrPresent>0)

  tempFit2 <- gam(as.formula(frml.bin2), data=tempTrain2, family = binomial(), method = "REML")
  tempNewPredict2 <- predict(tempFit2, newdata = tempTest, type = "link", se.fit = TRUE)
  tempOldPredict2 <- predict(tempFit2, newdata = tempTrain2017, type="link", se.fit = TRUE)


  Sys.sleep(0.1)
  setTxtProgressBar(pb, i, label=paste( round(i/length(daysIn2017)*100, 0),
                                        "% done"))

  #store predictions
  predictions_over_doy_unprocessed <-rbind(predictions_over_doy_unprocessed, cbind(data.frame(pred_type=c(rep("tr
ain_pred",length(tempOldPredict$fit)),rep("test_pred",length(tempNewPredict$fit)))
```

```
                                                                                , pred_bin=c(tempOldPredict$fit, tempNewPre
dict$fit)
                                                                                , pred_bin2=c(tempOldPredict2$fit, tempNewP
redict2$fit)
                                                                                , se = c(tempOldPredict$se.fit, tempNewPred
ict$se.fit)
                                                                                , se2 = c(tempOldPredict2$se.fit, tempNewPr
edict2$se.fit)
                                                                                , doy_cutoff = rep(doy, nrow(tck_borrelia_t
est))
  ),rbind(tempTrain2017,tempTest)))
}
```

```
## Warning in predict.gam(tempFit, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels 2017 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels 2017 not in original fit
```

```
##
  |
  |==                                                                   |   2%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels 2017 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTrain2017, type = "link", :
## factor levels 2017 not in original fit
```

```
##
  |
  |===                                                                   |   5%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels 2017 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTrain2017, type = "link", :
## factor levels 2017 not in original fit
```

```
##
  |
  |=====                                                                 |   7%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels 2017 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTrain2017, type = "link", :
## factor levels TALL_001 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTrain2017, type = "link", :
## factor levels 2017 not in original fit
```

```
##
  |
  |======                                                                |   9%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTrain2017, type = "link", :
## factor levels TALL_001 not in original fit
```

```
##
  |
  |========                                                              |  11%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTrain2017, type = "link", :
## factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |==========                                                            |  14%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |===========                                                           |  16%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |=============                                                         |  18%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |==============                                                  |  20%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |================                                                |  23%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |==================                                              |  25%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |===================                                                   |  27%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |=====================                                                 |  30%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |======================                                                |  32%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |========================                                              |  34%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |=========================                                             |  36%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |==========================                                            |  39%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |===========================                                           |  41%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |==============================                                       |  43%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels ORNL_002, TALL_001 not in original fit
```

```
##
  |
  |===============================                                      |  45%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
## Warning in predict.gam(tempFit2, newdata = tempTrain2017, type = "link", :
## factor levels TALL_001 not in original fit
```

```
##
  |
  |================================                                     |  48%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=================================                                    |  50%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |====================================                                  |  52%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=====================================                                 |  55%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |======================================                                |  57%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=======================================                               |  59%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |========================================                              |  61%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=========================================                             |  64%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=========================================                             |  66%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |================================================                      |  68%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |================================================                      |  70%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |================================================                      |  73%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=================================================                 |  75%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |==================================================                |  77%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |====================================================              |  80%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=====================================================             |  82%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=============================================================          |  84%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=============================================================          |  86%
```

```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=============================================================          |  89%
```

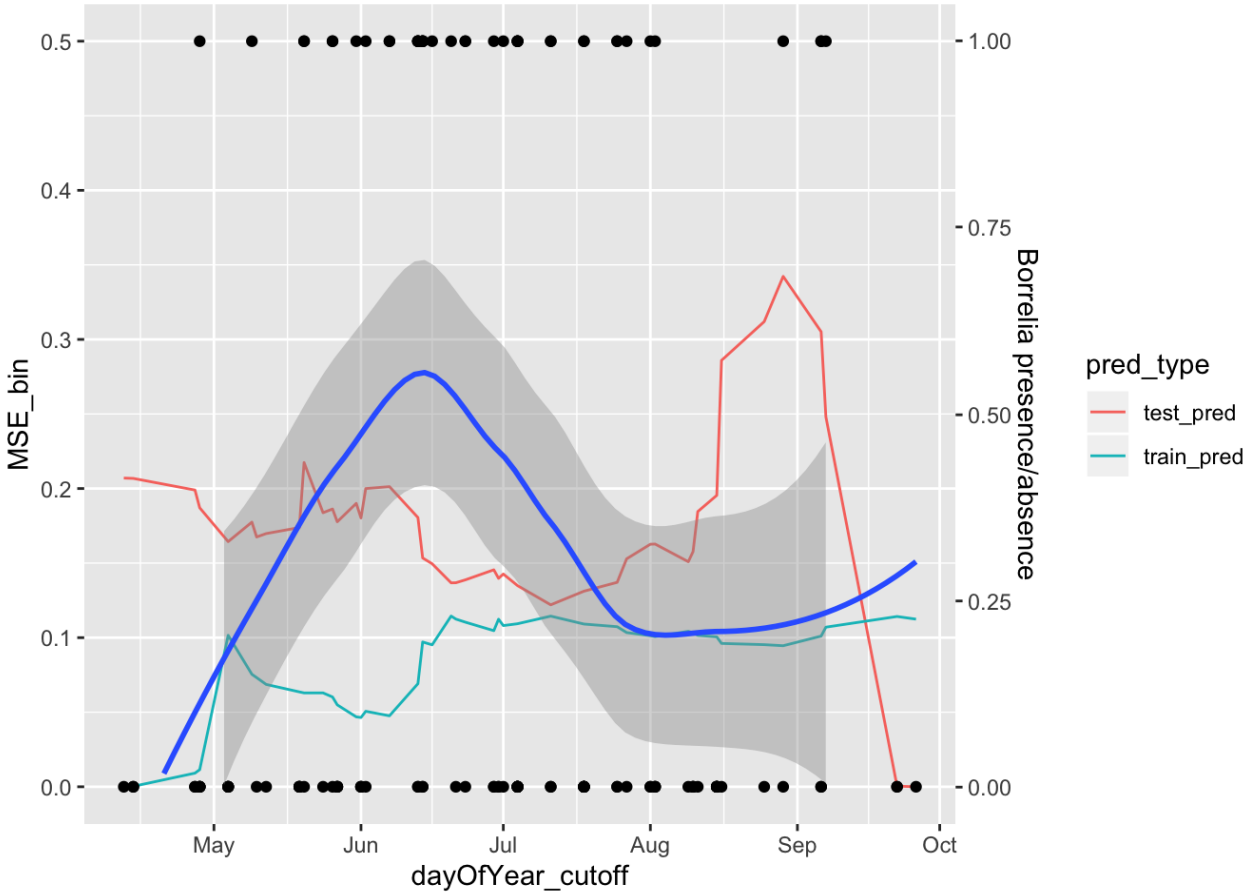
```
## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit

## Warning in predict.gam(tempFit2, newdata = tempTest, type = "link", se.fit =
## TRUE): factor levels TALL_001 not in original fit
```

```
##
  |
  |=============================================================      |  91%
  |
  |================================================================    |  93%
  |
  |==================================================================  |  95%
  |
  |==================================================================== |  98%
  |
  |======================================================================| 100%
```

```
predictions_over_doy <- predictions_over_doy_unprocessed %>%
  filter(!is.na(pred_bin))

# Summarize some MSE stats
MSE_summary <- predictions_over_doy %>%
  mutate(ressq_bin = (inv.logit(pred_bin)-borrPresent)^2
         , ressq_bin2 = (inv.logit(pred_bin2)-proportionPositive)^2
         , predCount_bin2 = inv.logit(pred_bin2)*numberTested
         , ressq_countbin2 = (inv.logit(predCount_bin2)-numberPositive)^2) %>%
  group_by(pred_type, doy_cutoff) %>%
  summarize(MSE_bin=sum(ressq_bin)/n()
            , MSE_bin2=sum(ressq_bin2)/n()
            , MSE_countbin2= sum(ressq_countbin2)/n()
            , predAve_bin = sum(borrPresent)/n()
            , predAve_bin2 = mean(inv.logit(pred_bin2))
            , predAveCount_bin2 = mean(inv.logit(predCount_bin2))) %>%
  ungroup()

## Get number of correct in predictions
summedCorrect <- predictions_over_doy %>%
  mutate(pred_borrPresent=ifelse(inv.logit(pred_bin)>0.5,1,0)
         , correct = borrPresent==pred_borrPresent
         , falsePositive = ifelse(borrPresent==0 & pred_borrPresent==1, 1, 0)
         , falseNegative = ifelse (borrPresent==1 & pred_borrPresent==0,1,0)) %>%
  group_by(doy_cutoff, pred_type) %>%
  summarize(nCorrect = sum(correct, na.rm=TRUE)
            , nFPositive = sum(falsePositive, na.rm=TRUE)
            , nFNegative = sum(falseNegative, na.rm=TRUE)
            , n = n()) %>%
  mutate(propCorrect = nCorrect/n
         , propFPositive = nFPositive/n
         , propFNegative = nFNegative/n)

## Binomial1
MSE_summary %>%
  mutate(dayOfYear_cutoff = as.Date(doy_cutoff, origin = "2017-01-01")
         , month = month(dayOfYear_cutoff)) %>%
  ggplot() + geom_line(aes(x=dayOfYear_cutoff, y=MSE_bin, col=pred_type)) +
  scale_y_continuous(sec.axis = sec_axis(trans=~./0.5, name = "Borrelia presence/absence"), limits = c(0,0.5)) +
  geom_point(data=tck_borrelia_test, aes(x=as.Date(dayOfYear, origin = "2017-01-01"), y=as.numeric(borrPresent)*
0.5)) +
  geom_smooth(data=tck_borrelia_test, aes(x=as.Date(dayOfYear, origin = "2017-01-01"), y=borrPresent*0.5))
```

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

```
## Warning: Removed 4 rows containing missing values (geom_smooth).
```



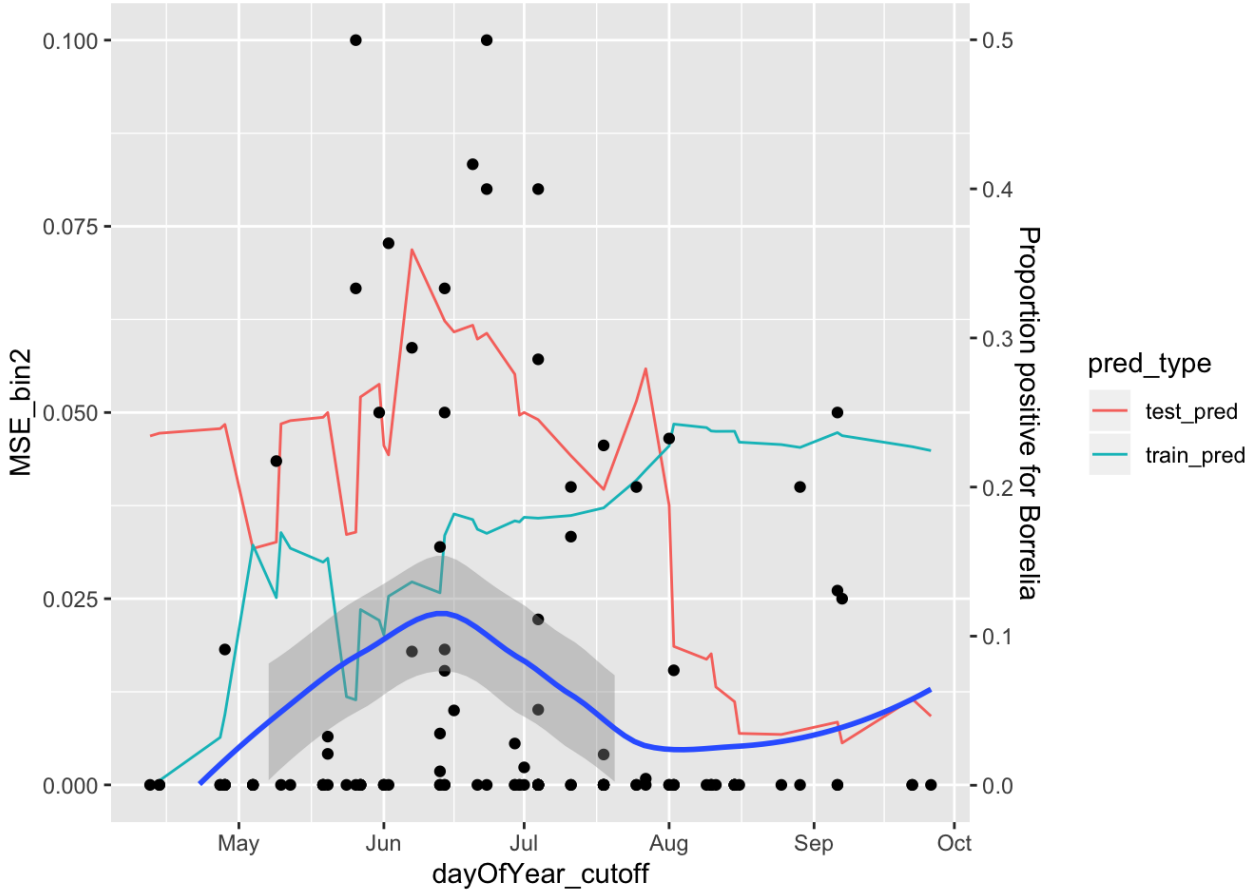

```
## Binomial 2- proportions
MSE_summary %>%
  mutate(dayOfYear_cutoff = as.Date(doy_cutoff, origin = "2017-01-01")
         , month = month(dayOfYear_cutoff)) %>%
  ggplot() + geom_line(aes(x=dayOfYear_cutoff, y=MSE_bin2, col=pred_type)) +
  geom_point(data=tck_borrelia_test, aes(x=as.Date(dayOfYear, origin = "2017-01-01"), y=proportionPositive/5)) +
  geom_smooth(data=tck_borrelia_test, aes(x=as.Date(dayOfYear, origin = "2017-01-01"), y=proportionPositive/5)) +
  scale_y_continuous(sec.axis = sec_axis(trans=~.*5, name = "Proportion positive for Borrelia"), limits=c(0,0.1))
```

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

```
## Warning: Removed 2 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```

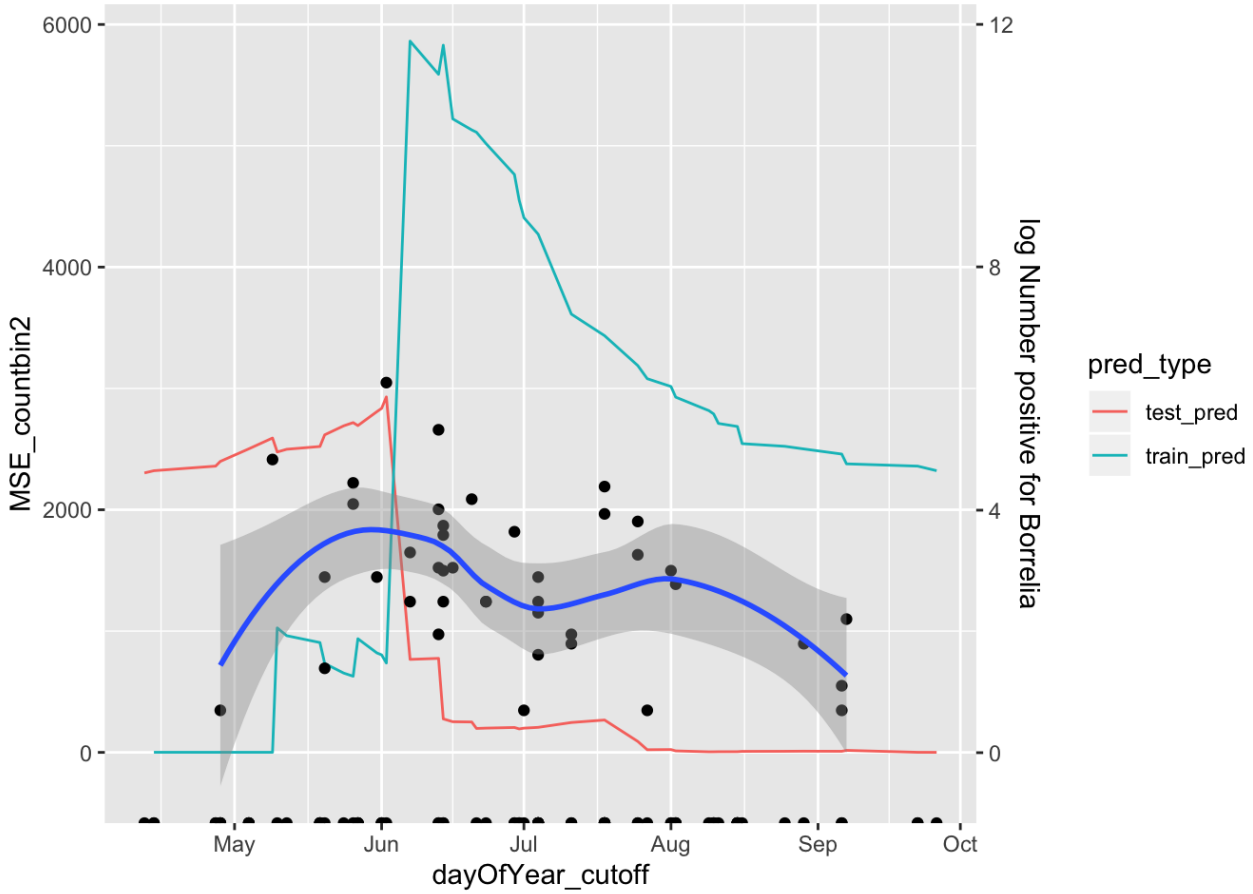
```
## Warning: Removed 5 rows containing missing values (geom_smooth).
```



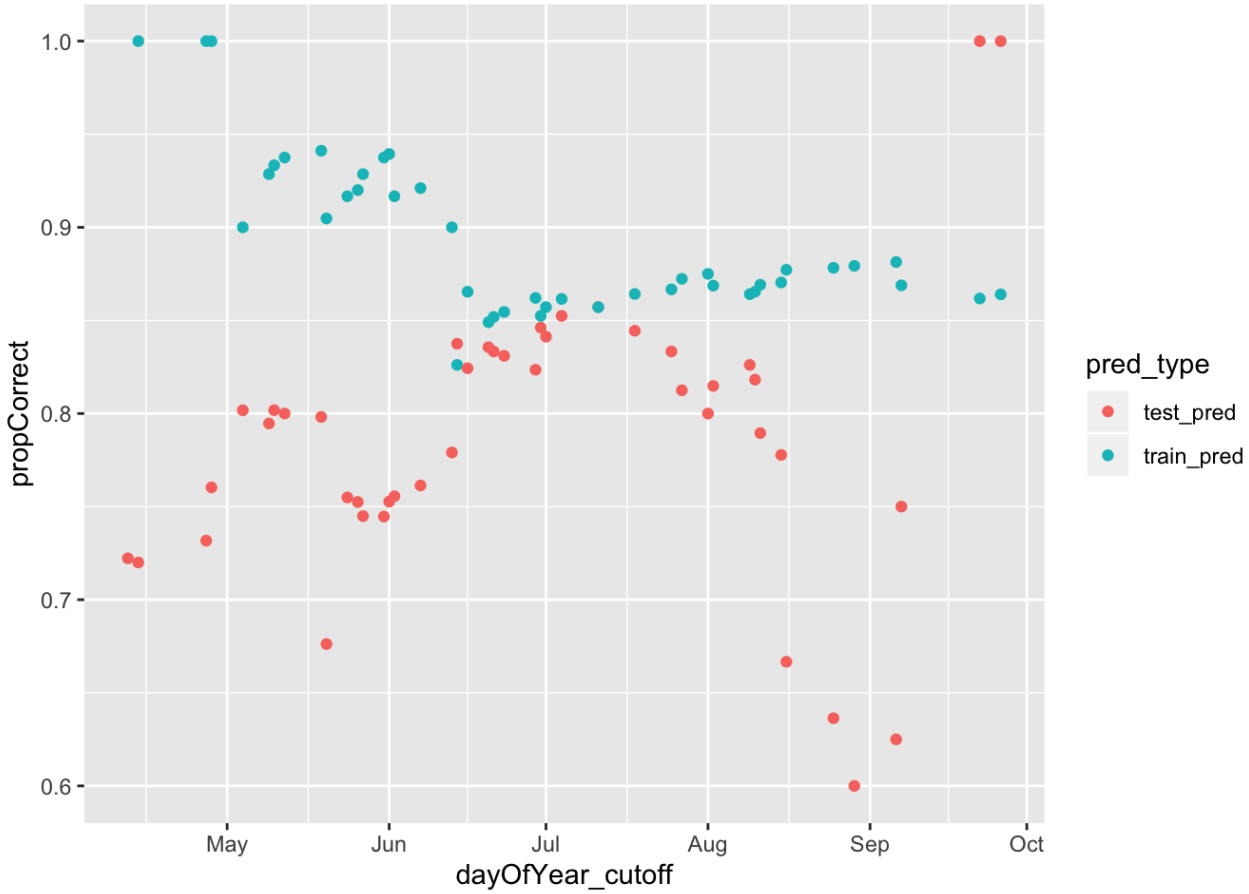
```
## Binomial 2- counts
MSE_summary %>%
  mutate(dayOfYear_cutoff = as.Date(doy_cutoff, origin = "2017-01-01")
         , month = month(dayOfYear_cutoff)) %>%
  ggplot() + geom_line(aes(x=dayOfYear_cutoff, y=MSE_countbin2, col=pred_type)) +
  geom_point(data=tck_borrelia_test, aes(x=as.Date(dayOfYear, origin = "2017-01-01"), y=log(numberPositive)*500))
+
  geom_smooth(data=tck_borrelia_test, aes(x=as.Date(dayOfYear, origin = "2017-01-01"), y=log(numberPositive)*500
)) +
  scale_y_continuous(sec.axis = sec_axis(trans=~./500, name = "log Number positive for Borrelia"))
```

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

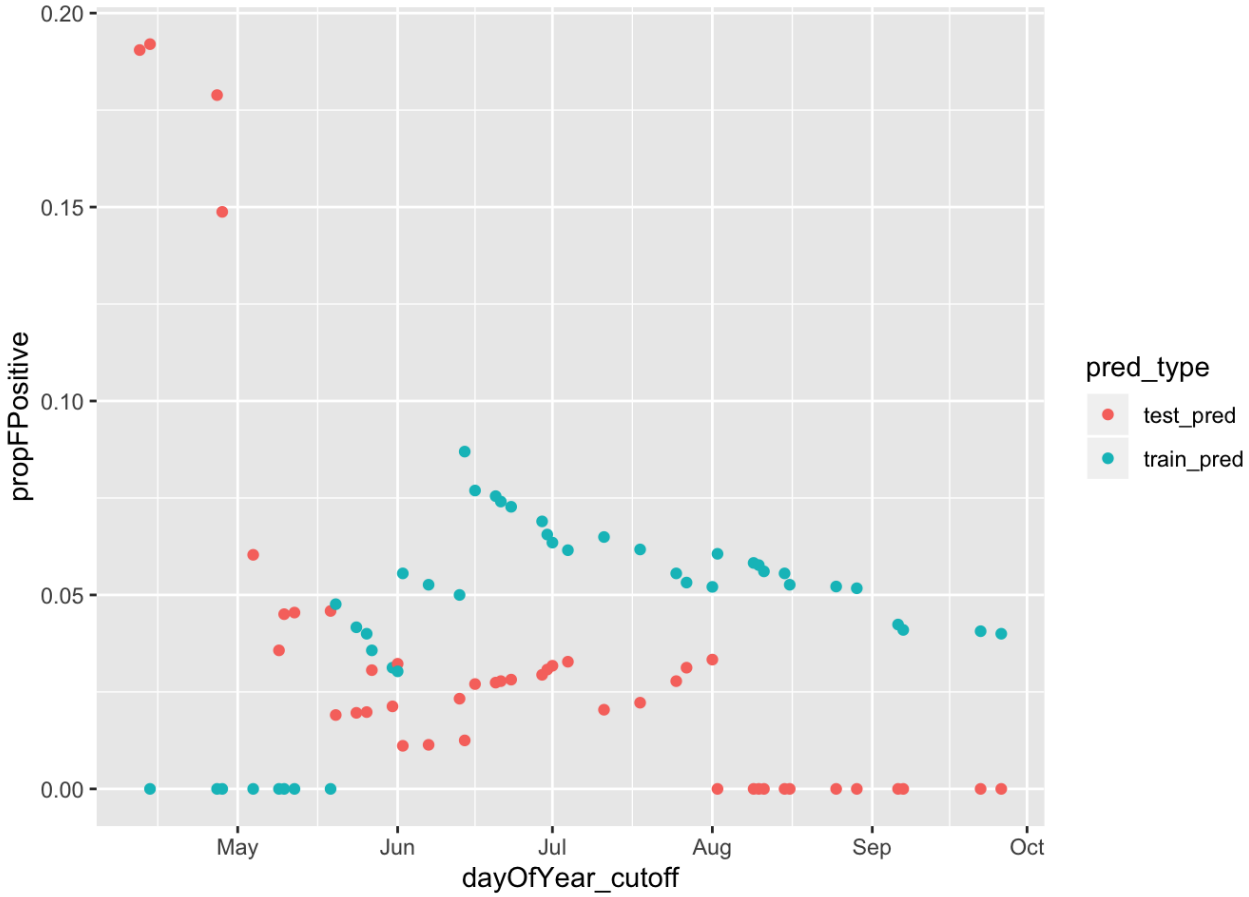
```
## Warning: Removed 85 rows containing non-finite values (stat_smooth).
```



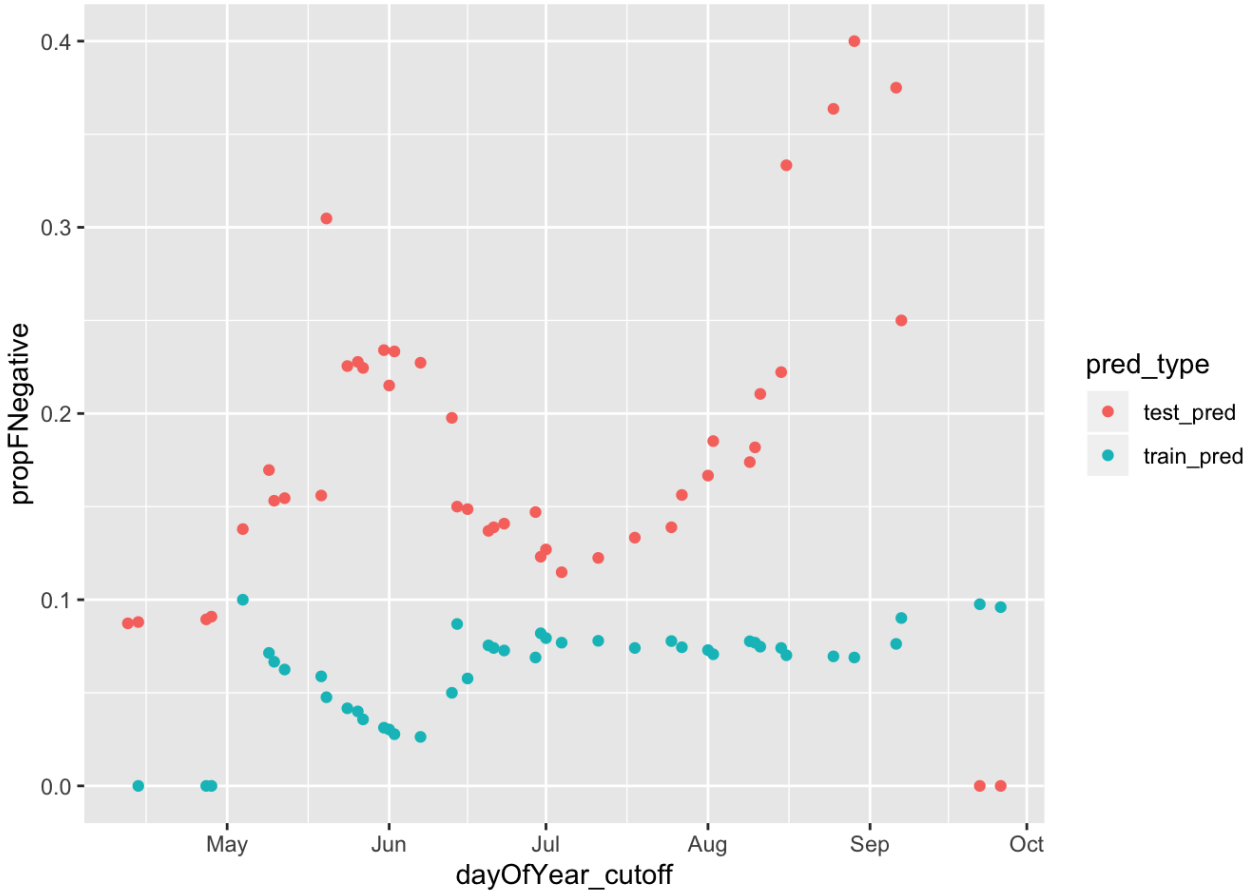

```
summedCorrect %>%
  mutate(dayOfYear_cutoff = as.Date(doy_cutoff, origin = "2017-01-01")
         , month = month(dayOfYear_cutoff)) %>%
  ggplot() + geom_point(aes(x=dayOfYear_cutoff, y=propCorrect, col=pred_type))
```




```
summedCorrect %>%
  mutate(dayOfYear_cutoff = as.Date(doy_cutoff, origin = "2017-01-01")
         , month = month(dayOfYear_cutoff)) %>%
  ggplot() + geom_point(aes(x=dayOfYear_cutoff, y=propFPositive, col=pred_type))
```

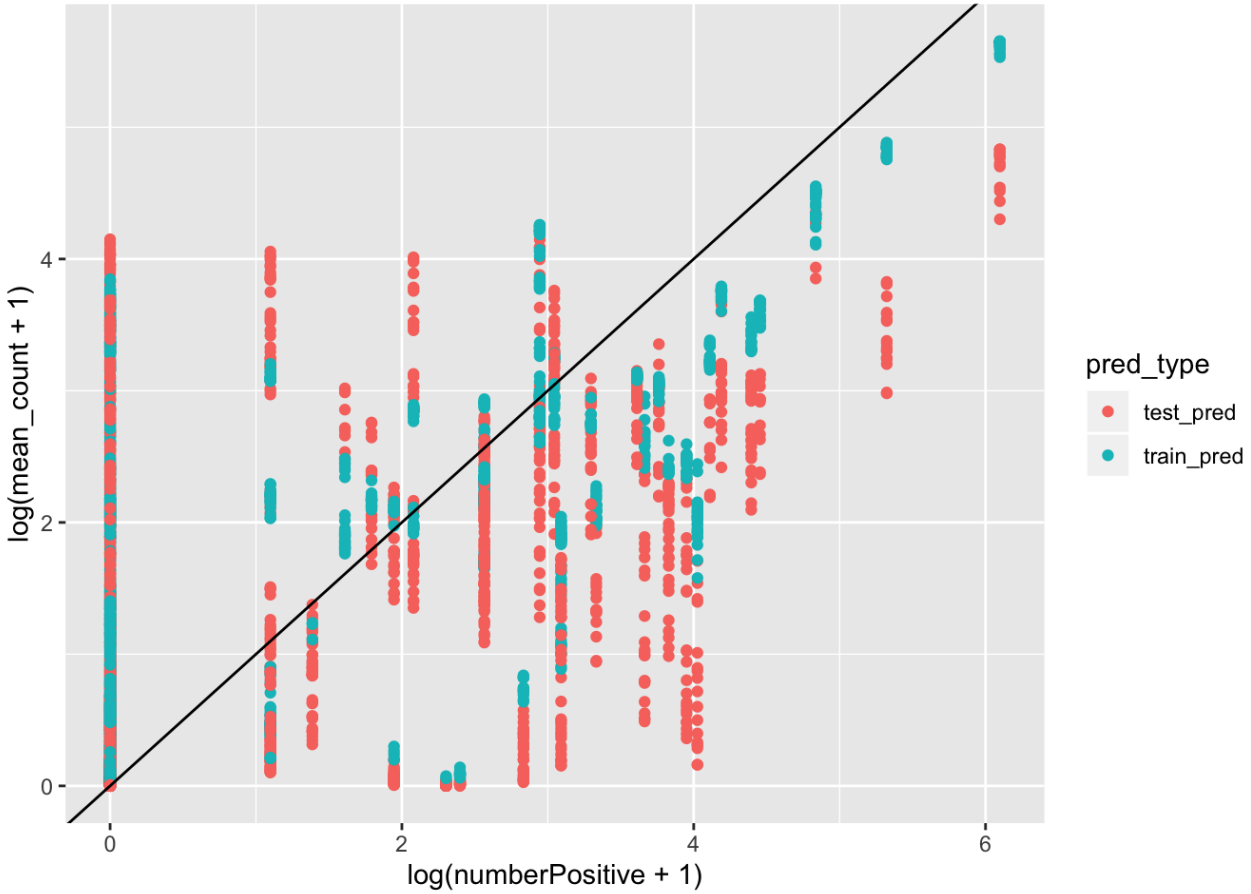



```
summedCorrect%>%
  mutate(dayOfYear_cutoff = as.Date(doy_cutoff, origin = "2017-01-01")
         , month = month(dayOfYear_cutoff)) %>%
  ggplot() + geom_point(aes(x=dayOfYear_cutoff, y=propFNegative, col=pred_type))
```

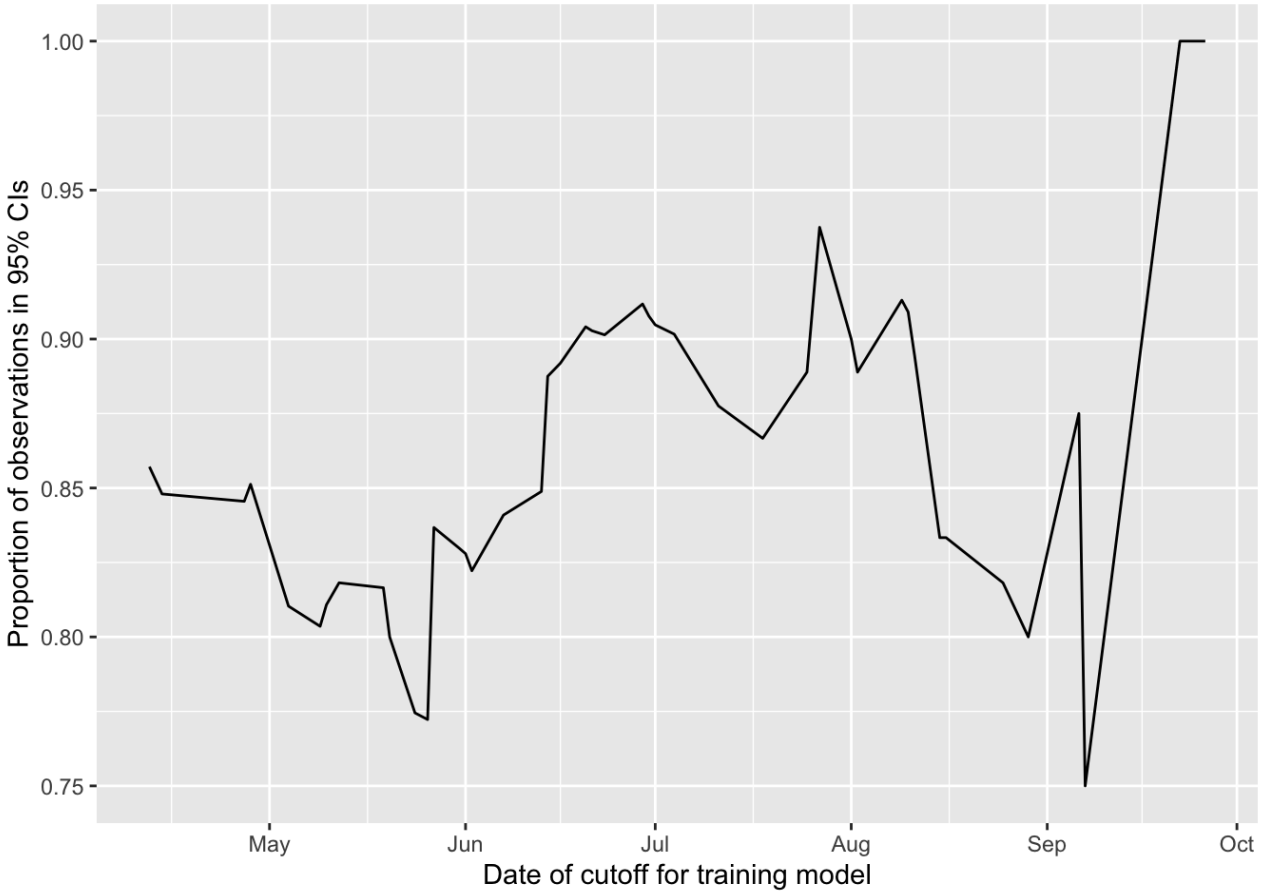

0.4
0.3
0.2
0.1
0.0
propFNegative
May
Jun
Jul
Aug
Sep
Oct
dayOfYear_cutoff
pred_type
test_pred
train_pred

```
#### Part IV: Simulate predictions and assess predictive accuracy ####

# First, make a  column of sample names for samples
predictions_over_doy <- predictions_over_doy %>%
  mutate(rown = 1:n()
           ,SimSample = paste0("V",rown))

nsim <- 1000
countPred_sim <- matrix(ncol=nrow(predictions_over_doy), nrow=nsim)
propPred_sim <- matrix(ncol=nrow(predictions_over_doy), nrow=nsim)
bin1_sim_all <- matrix(ncol=nrow(predictions_over_doy), nrow=nsim)
bin2_sim_all <- matrix(ncol=nrow(predictions_over_doy), nrow=nsim)
if ( FALSE ) {
  pb <- txtProgressBar(title = "progress bar", min = 0,
                       max = nrow(predictions_over_doy), style=3)

  for ( r in 1:nrow(predictions_over_doy)) {
    expbin_sim <- rnorm(n=nsim, mean=predictions_over_doy[r,"pred_bin"], sd=predictions_over_doy[r,"se"] )
    bin_sim <- rbinom(n=nsim,size=1,prob = inv.logit(expbin_sim) )
    expbin2_sim <- rnorm(n=nsim, mean=predictions_over_doy[r,"pred_bin2"], sd=predictions_over_doy[r,"se2"] )
    bin2_sim <- rbinom(n=nsim, size=predictions_over_doy[r,"numberTested"], prob=inv.logit(expbin2_sim))
    exp_count_combined <- bin_sim*bin2_sim
    exp_prop_combined <- exp_count_combined/predictions_over_doy[r,"numberTested"]
    # Save
    bin1_sim_all[,r] <- bin_sim
    bin2_sim_all[,r] <- bin2_sim
    countPred_sim[,r] <- exp_count_combined

    Sys.sleep(0.1)
    setTxtProgressBar(pb, r)
  }
  save(bin1_sim_all, file="bin1_sim_all.RData")
  save(bin2_sim_all, file="bin2_sim_all.RData")
  save(countPred_sim, file="countPred_sim.RData")
} else {
  load("bin1_sim_all.RData")
  load("bin2_sim_all.RData")
  load("countPred_sim.RData")
}
countPred_sim_long <- countPred_sim %>%
  as.data.frame() %>%
  mutate(SimID = 1:nsim) %>%
```

```
  gather(-SimID,key=SimSample, value=sim_count)
# propPred_sim_long <- propPred_sim %>%
#   as.data.frame() %>%
#   gather(key=SampleID, value=sim_prop)
if (FALSE) {
  sim_results <- predictions_over_doy %>%
    full_join(countPred_sim_long) %>%
    mutate(prop_sim = sim_count/numberTested
           ,SampleID = paste0(plotID,year,dayOfYear, sep="_"))

  save(sim_results, file="sim_results.RData")
} else {
  load("sim_results.RData")
}

sim_results_summarized <- sim_results %>%
  group_by(SampleID, numberPositive, numberTested, proportionPositive, pred_type, pred_bin, pred_bin2, doy_cutof
f) %>%
  summarise(CI97.5_count=quantile(sim_count, probs=0.975)
            , CI2.5_count=quantile(sim_count, probs=0.025)
            , sd_count=sd(sim_count)
            , mean_count=mean(sim_count)
            , CI97.5_prop=quantile(prop_sim, probs=0.975)
            , CI2.5_prop=quantile(prop_sim, probs=0.025)
            , sd_prop=sd(prop_sim)
            , mean_prop=mean(prop_sim)) %>%
  ungroup() %>%
  mutate(res_count = numberPositive-mean_count)
# count_sim_results$doy_cutoff

# No CI bars; logged
sim_results_summarized %>%
  ggplot() + geom_point(aes(x=log(numberPositive+1), y=log(mean_count+1), col=pred_type)) +
  geom_abline(aes(intercept=0,slope=1))
```




```
# Within 95% confidence intervals?
sim_results_summarized %>%
  filter(pred_type=="test_pred") %>%
  mutate(inCI = (numberPositive>=CI2.5_count & numberPositive<=CI97.5_count)) %>%
  select(doy_cutoff, inCI, numberPositive, CI2.5_count, CI97.5_count) %>%
  group_by(doy_cutoff) %>%
  summarize(propInCI95 = sum(inCI)/n()) %>%
  ungroup() %>%
  mutate(Cutoff_for_training = as.Date(doy_cutoff, origin="2017-01-01")) %>%
  ggplot()+ geom_line(aes(x=Cutoff_for_training, y=propInCI95)) +
  ylab("Proportion of observations in 95% CIs") +
  xlab("Date of cutoff for training model")
```


1.00
0.95
0.90
0.85
0.80
0.75
Proportion of observations in 95% CIs
May
Jun
Jul
Aug
Sep
Oct
Date of cutoff for training model

```
#
# library("brms")
#
# #### BRMS: Part I #####
# if (FALSE ) {
#   brm_bin1 <- brm(bf(frml.bin1)
#                   , seed=48
#                   , data=tck_borrelia_adj
#                   , family=bernoulli
#                   # , prior = gambin_priors
#                   , control=list(adapt_delta=0.99, max_treedepth=15)
#                   )
#   save(brm_bin1, file="brm_bin1.RData")
#   # save(brm_bin_noelev, file="brm_bin_noelev.RData")
#
# } else {
#   load("brm_bin1.RData")
#   # load("brm_bin_noelev.RData")
# }
# summary(brm_bin1)
# plot(brm_bin1)
# conditional_effects(brm_bin1, effects = "dayOfYear")
# conditional_effects(brm_bin1, effects = "lognymphDensity")
#
#
# # Look at how estimated probability maps to actual data for borrelia
# fitted(brm_bin1) %>%
#   cbind(brm_bin1$data) %>%
#   ggplot() +geom_point(aes(x=Estimate, y=borrPresent))
#
# # Inspect error rate (false positive and false negative)
# predict(brm_bin1) %>%
#   cbind(brm_bin1$data) %>%
#   mutate(Pred = ifelse(Estimate>0.5, 1, 0)) %>%
#   select(borrPresent, Pred) %>% table()
# 12/(12+194) # false positive rate
# 29/(29+73) # false negative rate
#
# # See if there is correlation between probability and how many positives there actually were
# fitted(brm_bin1) %>%
#   cbind(tck_borrelia_adj[,c("proportionPositive","numberPositive", "numberTested")]) %>%
#   ggplot() + geom_point(aes(x=numberPositive, y=Estimate, cex=numberTested))
```

```
#
# ## Look at effect of various predictors
# # Effect of day of year
# fitted(brm_bin1) %>%
#   cbind(brm_bin1$data) %>%
#   left_join(tck_borrelia_adj) %>%
#   ggplot() + geom_point(aes(x=dayOfYear, y=Estimate, col=nlcdClass)) + geom_point(aes(x=dayOfYear, y=borrPresen
t), col="red",alpha=0.2) + geom_smooth(aes(x=dayOfYear, y=borrPresent))
# conditional_smooths(brm_bin1, smooths="s(dayOfYear)")
#
# # Effect of tck density
# fitted(brm_bin1) %>%
#   cbind(brm_bin1$data) %>%
#   left_join(tck_borrelia_adj) %>%
#   ggplot() + geom_point(aes(x=lognymphDensity, y=Estimate, col=domainID)) + geom_point(aes(x=lognymphDensity, y
=borrPresent), col="red",alpha=0.2) + geom_smooth(aes(x=lognymphDensity, y=borrPresent))
# conditional_smooths(brm_bin1, smooths="s(lognymphDensity)")
#
# # Since each sample is IID, we can include all positive results in poisson component of model, and use fitted p
robabilities to determine
# # whether each zero is a "binomial" zero or a "poisson" zero.
# tck_borrelia_filtbin_brm <- tck_borrelia_adj %>%
#   cbind(predict(brm_bin1)) %>%
#   filter(borrPresent>0 | Estimate>0.5)
#
# ## Now try to fit a second distribution?
# tck_borrelia_filtbin_brm %>%
#   ggplot() + geom_histogram(aes(x=proportionPositive), bins=20)
# tck_borrelia_filtbin_brm %>%
#   ggplot() + geom_histogram(aes(x=numberPositive), bins=20)
#
# ##### BRMS: Part II ####
# # Re-format frml for brm
# frml_bin2_adj  <- gsub(pattern="cbind(numberPositive,numberTested)",replacement="numberPositive | trials(number
Tested)",frml_bin2_bestDevexpl, fixed = TRUE)
#
# if (FALSE) {
#   brm_bin2 <- brm(bf(as.formula(frml_bin2_adj))
#                   , data=tck_borrelia_filtbin_brm
#                   , family=binomial()
#                   , seed=24
#                   , control = list(adapt_delta=0.95, max_treedepth=15)
#   )
```

```
#   save(brm_bin2, file = "brm_bin2.RData")
# } else {
#   load("brm_bin2.RData")
# }
# summary(brm_bin2)
#
# plot(brm_bin2)
# conditional_smooths(brm_bin2, smooths = "s(dayOfYear)")
# conditional_smooths(brm_bin2, smooths = "s(lognymphDensity, sp = 1)")
# conditional_smooths(brm_bin2, smooths = "s(logadultDensity, sp = 1)")
#
# conditional_effects(brm_bin2, effects = "dayOfYear")
# conditional_effects(brm_bin2, effects = "logadultDensity")
# conditional_effects(brm_bin2, effects = "lognymphDensity")
#
#
# predict(brm_bin2) %>%
#   cbind(tck_borrelia_filtbin_brm[,c("numberPositive","dayOfYear","numberTested","logNLtckDensity","year","plotI
D")]) %>%
#   ggplot() + geom_point(aes(x=log(numberPositive+1), y=log(Estimate+1))) +
#   geom_segment(aes(x=log(numberPositive+1), xend=log(numberPositive+1), y=log(Q2.5+1), yend=log(Q97.5+1)), col
="red")
#
#
#
#
#
#
```