Point count data analysis: How to violate assumptions and get away with it

Peter Solymos

2019-06-06

Contents

List of Tables

List of Figures

Preface

This book provides material for the workshop *Analysis of point-count data in the presence of variable survey methodologies and detection error* at the AOS 2019 conference by Peter Solymos.

The book and related materials in this repository is the basis of a full day workshop (8 hours long with 3 breaks).

Prior exposure to R language is necessary (i.e. basic R object types and their manipulation, such as arrays, data frames, indexing) because this is not covered as part of the course. Check this intro.

About the book and the course

You'll learn

- how to analyze your point count data when it combines different methodologies/protocols/technologies,
- how to violate assumptions and get away with it.

This book/course is aimed towards ornithologists analyzing field observations, who are often faced by data heterogeneities due to field sampling protocols changing from one project to another, or through time over the lifespan of projects, or trying to combine 'legacy' data sets with new data collected by recording units. Such heterogeneities can bias analyses when data sets are integrated inadequately, or can lead to information loss when filtered and standardized to common standards. Accounting for these issues is important for better inference regarding status and trend of bird species and communities.

Analysts of such 'messy' data sets need to feel comfortable with manipulating the data, need a full understanding the mechanics of the models being used (i.e. critically interpreting the results and acknowledging assumptions and limitations), and should be able to make informed choices when faced with methodological challenges.

The course emphasizes critical thinking and active learning. Participants will be asked to take part in the analysis: first hand analytics experience from start to finish. We will use publicly available data sets to demonstrate the data manipulation and analysis. We will use freely available and open-source R packages.

The expected outcome of the course is a solid foundation for further professional development via increased confidence in applying these methods for field observations.

About the author

Peter Solymos is an ecologist (molluscs, birds), he is pretty good at stats (modeling, detectability, data cloning, multivariate), an R programmer (vegan, detect, ResourceSelection, pbapply), sometimes he teaches (like the contents of this book).

Installing

The **bookdown** package can be installed from CRAN or Github:

```r
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")

## clean up
bookdown::clean_book(TRUE)
## rendering the book
bookdown::render_book('index.Rmd', 'bookdown::pdf_book')
bookdown::render_book('index.Rmd', 'bookdown::gitbook')
bookdown::render_book('index.Rmd', 'bookdown::epub_book')
```

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.name/tinytex/.

How this works

This is an exercise.

This is a note.

This is a warning.

Acknowledgments

These are just reminders, to be deleted later

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 1. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```r
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 1.

```r
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

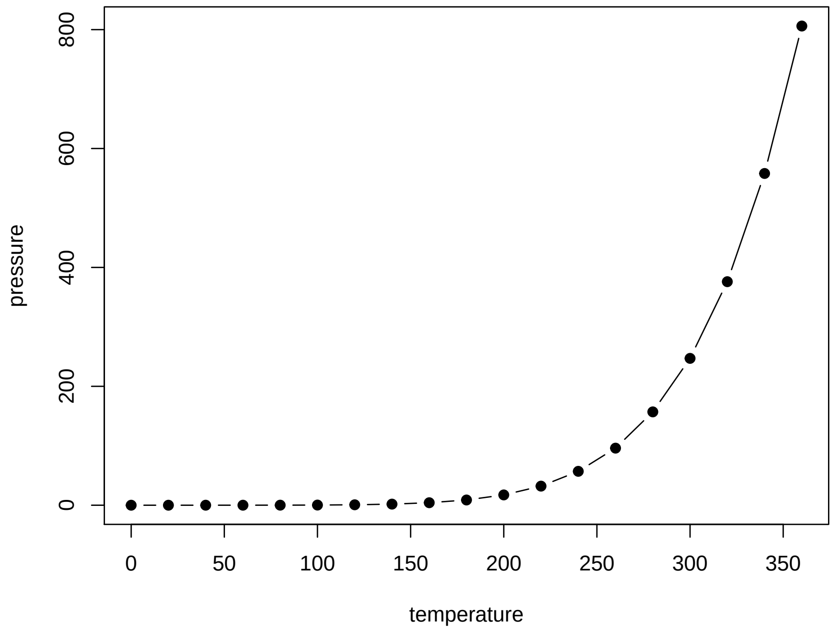


Figure 1: Here is a nice figure!

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2019) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Table 1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Chapter 1

Introduction

All assumptions are violated, but some are more than others

A comparison of apples and oranges occurs when two items or groups of items are compared that cannot be practically compared (Wikipedia). The way we measure things can have a big impact on the outcome of that measurement. For example, you might say that "I saw 5 robins walking down the road", while I might say that "I only saw one robin while sitting on my porch". Who say more robins? If looking at only the numeric results, you saw more robins than me. But this seems like an apples to oranges comparison.

To compare apples to apples, we need to agree on a comparable measurement scheme, or at least figure out how does *effort* affect the observations.

Effort in our example can depend on, e.g. the *area* of the physical space searched, the amount of *time* spent, etc. The outcome might further affected by weather, time of year, time of day, location, experience and skill level of the observer.

All these factors can affect the observed count. Which brings us to the definition of a *point count*: a trained observer records all the birds seen and heard from a point count station for a set period of time within a defined distance radius.

Point count duration and distance have profound effect on the counts, as shown in Figure 1.1 showing that a 10-min unlimited distance count is roughly 300% increased compared to 3-min 50-m counts (averaged across 54 species

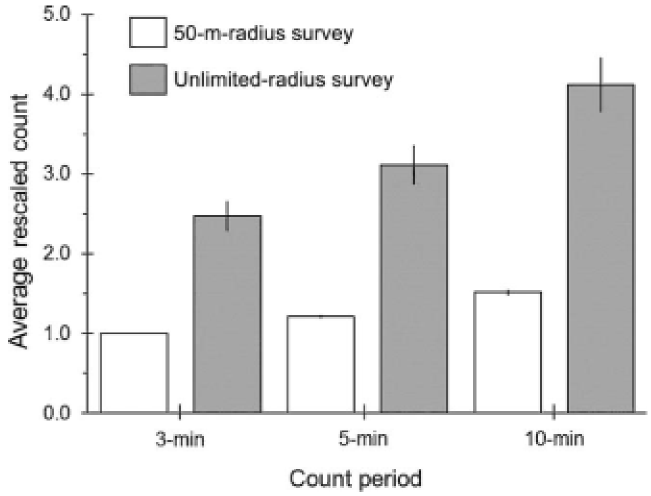


Figure 1.1: Effects of duration and distance on mean counts (Matsuoka et al. 2014).

of boreal songbirds, Matsuoka et al. 2014).

Point counts are commonly used to answer questions like:

- How many? (Abundance, density, population size)
- Is this location part of the range? (0/1)
- How is abundance changing in space? (Distribution)
- How is abundance changing in time? (Trend)
- What is the effect of a treatment on abundance?

1.1 Design-based approaches

Standards and recommendations can maximize efficiency in the numbers of birds and species counted, minimize extraneous variability in the counts.

But programs started to deviate from standards: *"For example, only 3% of 196,000 point counts conducted during the period 1992–2011 across Alaska and Canada followed the standards recommended for the count period and count radius"* (Matsuoka et al. 2014). Figure 1.2 show how point count protocol varies across the boreal region of North America.

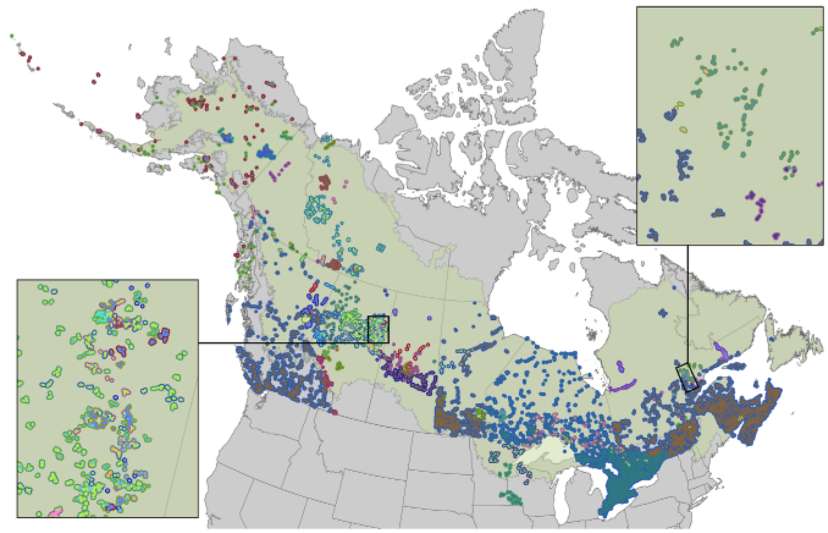


Figure 1.2: Survey methodology variation (colors) among contributed projects in the Boreal Avian Modelling (BAM) data base (Barker et al. 2015).

Exercise
In what regard can protocols differ?
What might drive protocol variation among projects?
Why have we abandoned following protocols?

1.2 Model-based approaches

Detection probabilities might vary even with fixed effort (we'll cover this more later), and programs might have their own goals and constraints (access, training, etc). These constraints would make it almost impossible, and potentially costly to set up very specific standards.

Labour intensive methods for unmarked populations have come to the forefront, and computing power of personal computers opened the door for model-based approaches, that can accomodate more variation given enough information in the observed data. These methods often rely on ancillary information and often some sort of replication.

Some of the commonly used model-based approaches are:

- double observer (Nichols et al. 2000),
- distance sampling (Buckland et al. 2001),
- removal sampling (Farnsworth et al. 2002),
- multiple visit occupancy (MacKenzie et al. 2002),
- multiple visit abundance (Royle 2004).

Models come with assumptions, such as:

- population is closed during multiple visits,
- observers are independent,
- all individuals emit cues with identical rates,
- spatial distribution of individuals is uniform,
- etc.

Although assumptions are everywhere, we are really good at ignoring and violating them.

Exercise
Can you mention some assumptions from everyday life?
Can you explain why we neglect/violate assumptions in these situations?

Assumptions are violated, because we seek simplicity. The main question we have to ask: *does it matter in practice* if we violate the assumptions?

1.3 Our approach

In this book and course, we will critically evaluate common assumptions made when analyzing point count data using the following approach:

1. we will introduce a concept,
2. understand how we can infer it from data,
3. then we recreate the situation *in silico*,
4. and see how the outcome changes as we make different assumptions.

It is guaranteed that we will violate every assumption we make. To get away with it, we need to understand how much is too much, and whether it has an impact in practice. If there is a practical consequence, we will look at ways to minimize that effects – so that we can safely ignore the assumption.

1.4 R basics

This short document is intended to help you brush up your R skills. If you feel that these R basics are not very familiar, I suggest to take a look at some introductory R books, sich as this preprint version of Norman Matloff's *The Art of R Programming* book: `http://heather.cs.ucdavis.edu/~matloff/132/NSPpart.pdf`, check out Chapters 1–6.

R is a great calculator:

```
1 + 2
```

```
## [1] 3
```

Assign a value and print an object using `=` or `<-` (preferred in this book):

```
(x = 2) # shorthand for print
```

```
## [1] 2
```

```
print(x)
```

```
## [1] 2
```

```
x == 2 # logical operator, not assignment
```

```
## [1] TRUE
```

```
y <- x + 0.5
y # another way to print
```

```
## [1] 2.5
```

Logical operators come handy:

```
x == y # equal
```

```
## [1] FALSE
```

```
x != y # not eaqual
```

```
## [1] TRUE
```

```r
x < y # smaller than
```

```
## [1] TRUE
```

```r
x >= y # greater than or equal
```

```
## [1] FALSE
```

Vectors and sequences are created most often by the functions `c`, `:`, `seq`, and `rep`:

```r
x <- c(1, 2, 3)
x
```

```
## [1] 1 2 3
```

```r
1:3
```

```
## [1] 1 2 3
```

```r
seq(1, 3, by = 1)
```

```
## [1] 1 2 3
```

```r
rep(1, 5)
```

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

```r
rep(1:2, 5)
```

```
##  [1] 1 2 1 2 1 2 1 2 1 2
```

```r
rep(1:2, each = 5)
```

```
##  [1] 1 1 1 1 1 2 2 2 2 2
```

When doing operations with vectors remember that values of the shorter object are recycled:

```r
x + 0.5
```

```
## [1] 1.5 2.5 3.5
```

```r
x * c(10, 11, 12, 13)
```

```
## Warning in x * c(10, 11, 12, 13): longer object length is not a
## multiple of shorter object length
```

```
## [1] 10 22 36 13
```

Indexing and ordering vectors is a a fundamental skill:

```r
x[1]
```

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

```r
x[c(1, 1, 1)] # a way of repeatig values
```

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

```r
x[1:2]
```

```
## [1] 1 2
```

```r
x[x != 2]
```

```
## [1] 1 3
```

```r
x[x == 2]
```

```
## [1] 2
```

```r
x[x > 1 & x < 3]
```

```
## [1] 2
```

```r
order(x, decreasing=TRUE)
```

```
## [1] 3 2 1
```

```r
x[order(x, decreasing=TRUE)]
```

```
## [1] 3 2 1
```

```r
rev(x) # reverse
```

```
## [1] 3 2 1
```

See how `NA` values can influence sorting character vectors:

```
z <- c("b", "a", "c", NA)
z[z == "a"]
```

```
## [1] "a" NA
```

```
z[!is.na(z) & z == "a"]
```

```
## [1] "a"
```

```
z[is.na(z) | z == "a"]
```

```
## [1] "a" NA
```

```
is.na(z)
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
which(is.na(z))
```

```
## [1] 4
```

```
sort(z)
```

```
## [1] "a" "b" "c"
```

```
sort(z, na.last=TRUE)
```

```
## [1] "a" "b" "c" NA
```

There are a few special values:

```
as.numeric(c("1", "a")) # NA: not available (missing or invalid)
```

```
## Warning: NAs introduced by coercion
```

```
## [1]  1 NA
```

```
0/0 # NaN: not a number
```

```
## [1] NaN
```

```
1/0 # Inf
```

```
## [1] Inf
```

```r
-1/0 # -Inf
```

```
## [1] -Inf
```

Matrices and arrays are vectors with dimensions, elements are in same mode:

```r
(m <- matrix(1:12, 4, 3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```r
matrix(1:12, 4, 3, byrow=TRUE)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```r
array(1:12, c(2, 2, 3))
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
##
## , , 3
##
##      [,1] [,2]
## [1,]    9   11
```

```
## [2,]   10   12
```

Many objects have attributes:

```r
dim(m)
```

```
## [1] 4 3
```

```r
dim(m) <- NULL
m
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```r
dim(m) <- c(4, 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```r
dimnames(m) <- list(letters[1:4], LETTERS[1:3])
m
```

```
##   A B  C
## a 1 5  9
## b 2 6 10
## c 3 7 11
## d 4 8 12
```

```r
attributes(m)
```

```
## $dim
## [1] 4 3
##
## $dimnames
## $dimnames[[1]]
## [1] "a" "b" "c" "d"
##
## $dimnames[[2]]
## [1] "A" "B" "C"
```

Matrice and indices:

```
m[1:2,]
```

```
##   A B  C
## a 1 5  9
## b 2 6 10
```

```
m[1,2]
```

```
## [1] 5
```

```
m[,2]
```

```
## a b c d
## 5 6 7 8
```

```
m[,2,drop=FALSE]
```

```
##   B
## a 5
## b 6
## c 7
## d 8
```

```
m[2]
```

```
## [1] 2
```

```
m[rownames(m) == "c",]
```

```
##  A  B  C
##  3  7 11
```

```
m[rownames(m) != "c",]
```

```
##   A B  C
## a 1 5  9
## b 2 6 10
## d 4 8 12
```

```
m[rownames(m) %in% c("a", "c", "e"),]
```

```
##   A B  C
```

```
## a 1 5  9
## c 3 7 11
```

```
m[!(rownames(m) %in% c("a", "c", "e")),]
```

```
##   A B  C
## b 2 6 10
## d 4 8 12
```

Lists and indexing:

```
l <- list(m = m, x = x, z = z)
l
```

```
## $m
##   A B  C
## a 1 5  9
## b 2 6 10
## c 3 7 11
## d 4 8 12
##
## $x
## [1] 1 2 3
##
## $z
## [1] "b" "a" "c" NA
```

```
l$ddd <- sqrt(l$x)
l[2:3]
```

```
## $x
## [1] 1 2 3
##
## $z
## [1] "b" "a" "c" NA
```

```
l[["ddd"]]
```

```
## [1] 1.000 1.414 1.732
```

Data frames are often required for statistical modeling. A data frame is a list where length of elements match and elements can be in different mode.

```
d <- data.frame(x = x, sqrt_x = sqrt(x))
d
```

Inspect structure of R objects:

```
str(x)
```

```
##  num [1:3] 1 2 3
```

```
str(z)
```

```
##  chr [1:4] "b" "a" "c" NA
```

```
str(m)
```

```
##  int [1:4, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:4] "a" "b" "c" "d"
##   ..$ : chr [1:3] "A" "B" "C"
```

```
str(l)
```

```
## List of 4
##  $ m  : int [1:4, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "a" "b" "c" "d"
##   .. ..$ : chr [1:3] "A" "B" "C"
##  $ x  : num [1:3] 1 2 3
##  $ z  : chr [1:4] "b" "a" "c" NA
##  $ ddd: num [1:3] 1 1.41 1.73
```

```
str(d)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x     : num  1 2 3
##  $ sqrt_x: num  1 1.41 1.73
```

```
str(as.data.frame(m))
```

```
## 'data.frame':    4 obs. of  3 variables:
##  $ A: int  1 2 3 4
##  $ B: int  5 6 7 8
##  $ C: int  9 10 11 12
```

```r
str(as.list(d))
```

```
## List of 2
##  $ x     : num [1:3] 1 2 3
##  $ sqrt_x: num [1:3] 1 1.41 1.73
```

Get summaries of these objects:

```r
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     1.0     1.5     2.0     2.0     2.5     3.0
```

```r
summary(z)
```

```
##    Length     Class      Mode 
##         4 character character
```

```r
summary(m)
```

```
##        A              B              C        
##  Min.   :1.00   Min.   :5.00   Min.   : 9.00  
##  1st Qu.:1.75   1st Qu.:5.75   1st Qu.: 9.75  
##  Median :2.50   Median :6.50   Median :10.50  
##  Mean   :2.50   Mean   :6.50   Mean   :10.50  
##  3rd Qu.:3.25   3rd Qu.:7.25   3rd Qu.:11.25  
##  Max.   :4.00   Max.   :8.00   Max.   :12.00
```

```r
summary(l)
```

```
##     Length Class  Mode     
## m   12     -none- numeric  
## x    3     -none- numeric  
## z    4     -none- character
## ddd  3     -none- numeric
```

```r
summary(d)
```

```
##        x           sqrt_x    
##  Min.   :1.0   Min.   :1.00  
##  1st Qu.:1.5   1st Qu.:1.21  
##  Median :2.0   Median :1.41
```

```
## Mean   :2.0   Mean   :1.38
## 3rd Qu.:2.5   3rd Qu.:1.57
## Max.   :3.0   Max.   :1.73
```

Chapter 2

Organizing and Processing Point Count Data

> All data are messy, but some are missing

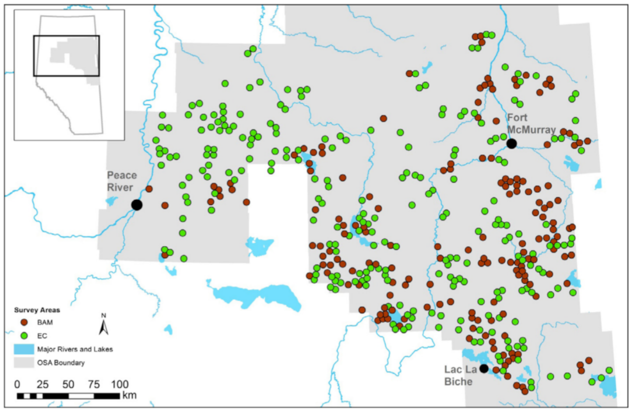
It is often called *data processing*, *data munging*, *data wrangling*, *data cleaning*. None of these expressions capture the dread associated with the actual activity.

Luckily, there are only 4 things that can get messed up:

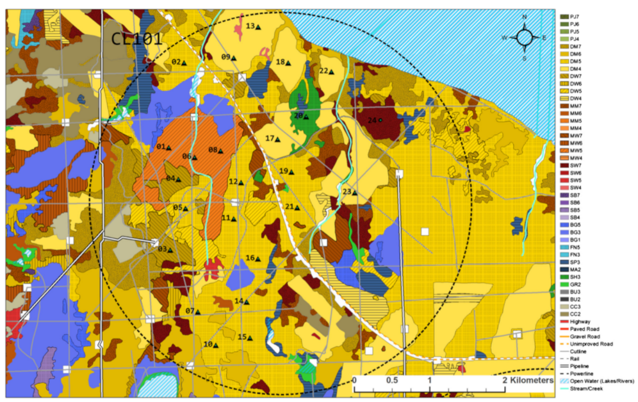
1. space (e.g. wrong UTM zones),
2. time (ISO format please),
3. taxonomy (UNK, mis-ID),
4. something else (if there were no errors, check again).

2.1 JOSM data set

Look at the source code in the `_data/josm` directory of the book if you are interested in data processing details. We skip that for now.



Cause-Effect Monitoring Migratory Landbirds at Regional Scales: understand how boreal songbirds are affected by human activity in the oil sands area.



Survey area boundary (r=2.5 km circle), habitat type and human footprint mapping, and clustered point count site locations.

Surveys were spatially replicated because:

- we want to make inferences about a population,
- full census is out of reach,
- thus we take a sample of the population
- that is representative and random.
- Ideally, sample size should be as large as possible,
- it reduces variability and
- increases statistical power.

Survey locations were pucked based on various criteria:

- stratification (land cover),
- gradients (disturbance levels),
- random location (control for unmeasured effects),
- take into account historical surveys (avoid, or revisit),
- access, cost (clusters).

The `josm` obejct is a list with 3 elements:

- `surveys`: data frame with survey specific information,
- `species`: lookup table for species,
- `counts`: individual counts by survey and species.

```
library(mefa4)
load("./_data/josm/josm.rda")
names(josm)
```

```
## [1] "surveys" "species" "counts"
```

Species info: species codes, common and scientific names. The table could also contain taxonomic, trait, etc. information as well.

```
head(josm$species)
```

At the survey level, we have coordinates, date/time info, variables capturing survey conditions, and land cover info extracted from 1 km^2 resolution rasters.

```
colnames(josm$surveys)
```

```
##  [1] "SiteID"        "SurveyArea"    "Longitude"
##  [4] "Latitude"      "Date"          "StationID"
##  [7] "ObserverID"    "TimeStart"     "VisitID"
## [10] "WindStart"     "PrecipStart"   "TempStart"
## [13] "CloudStart"    "WindEnd"       "PrecipEnd"
## [16] "TempEnd"       "CloudEnd"      "TimeFin"
## [19] "Noise"         "OvernightRain" "DateTime"
## [22] "SunRiseTime"   "SunRiseFrac"   "TSSR"
## [25] "OrdinalDay"    "DAY"           "Open"
## [28] "Water"         "Agr"           "UrbInd"
## [31] "SoftLin"       "Roads"         "Decid"
## [34] "OpenWet"       "Conif"         "ConifWet"
```

The count table contains one row for each unique individual of a species

(`SpeciesID` links to the species lookup table) observed during a survey (`StationID` links to the survey attribute table). Check the data dictionary in `_data/josm` folder for a detailed explanation of each column.

```
str(josm$counts)
```

```
## 'data.frame':    52372 obs. of  18 variables:
##  $ ObservationID: Factor w/ 57024 levels "CL10102-130622-001",..: 1 2 3 4 5
##  $ SiteID       : Factor w/ 4569 levels "CL10102","CL10106",..: 1 1 1 1 1 1
##  $ StationID    : Factor w/ 4569 levels "CL10102-1","CL10106-1",..: 1 1 1 1
##  $ TimeInterval : int  1 1 1 1 5 5 1 1 1 1 ...
##  $ Direction    : int  1 2 2 2 1 4 4 4 1 1 ...
##  $ Distance     : int  1 2 2 1 3 3 2 1 1 1 ...
##  $ DetectType1  : Factor w/ 3 levels "C","S","V": 2 2 2 2 1 1 2 2 2 2 ...
##  $ DetectType2  : Factor w/ 3 levels "C","S","V": NA NA NA NA NA NA NA NA N
##  $ DetectType3  : Factor w/ 3 levels "C","S","V": NA NA NA NA NA NA NA NA N
##  $ Sex          : Factor w/ 4 levels "F","M","P","U": 2 2 2 2 4 4 2 2 2 2 .
##  $ Age          : Factor w/ 6 levels "A","F","J","JUV",..: 1 1 1 1 1 1 1 1
##  $ Activity1    : Factor w/ 17 levels "BE","CF","CH",..: 5 5 5 5 NA NA NA 5
##  $ Activity2    : Factor w/ 17 levels "48","BE","CF",..: NA NA NA NA NA NA
##  $ Activity3    : Factor w/ 7 levels "CF","DC","DR",..: NA NA NA NA NA NA N
##  $ ActivityNote : Factor w/ 959 levels "AGITATED","AGITATED CALLING",..: NA
##  $ Dur          : Factor w/ 3 levels "0-3min","3-5min",..: 1 1 1 1 3 3 1 1
##  $ Dis          : Factor w/ 3 levels "0-50m","50-100m",..: 1 2 2 1 3 3 2 1
##  $ SpeciesID    : Factor w/ 150 levels "ALFL","AMBI",..: 107 95 95 107 46 4
```

2.2 Cross tabulating species counts

Take the following dummy data frame (long format):

```
(d <- data.frame(
  sample=factor(paste0("S", c(1,1,1,2,2)), paste0("S", 1:3)),
  species=c("BTNW", "OVEN", "CANG", "AMRO", "CANG"),
  abundance=c(1, 1, 2, 1, 1),
  behavior=rep(c("heard","seen"), c(4, 1))))
str(d)
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ sample   : Factor w/ 3 levels "S1","S2","S3": 1 1 1 2 2
```

```
##  $ species  : Factor w/ 4 levels "AMRO","BTNW",..: 2 4 3 1 3
##  $ abundance: num  1 1 2 1 1
##  $ behavior : Factor w/ 2 levels "heard","seen": 1 1 1 1 2
```

We want to add up the `abundance`s for each sample (rows) and species (column):

```
(y <- Xtab(abundance ~ sample + species, d))
```

```
## 3 x 4 sparse Matrix of class "dgCMatrix"
##    AMRO BTNW CANG OVEN
## S1    .    1    2    1
## S2    1    .    1    .
## S3    .    .    .    .
```

`y` is a sparse matrix, that is a very compact representation:

```
object.size(d[,1:3])
```

```
## 2328 bytes
```

```
object.size(y)
```

```
## 2160 bytes
```

Notice that we have 3 rows, but `d$sample` did not have an `S3` value, but it was a level. We can drop such unused levels, but it is generally not recommended, and we need to be careful not to drop samples where no species was detected (this can happen quite often depending on timing of surveys)

```
Xtab(abundance ~ sample + species, d, drop.unused.levels = TRUE)
```

```
## 2 x 4 sparse Matrix of class "dgCMatrix"
##    AMRO BTNW CANG OVEN
## S1    .    1    2    1
## S2    1    .    1    .
```

A sparse matrix can be converted to ordinary matrix

```
as.matrix(y)
```

```
##    AMRO BTNW CANG OVEN
## S1    0    1    2    1
```

```
## S2    1    0    1    0
## S3    0    0    0    0
```

The nice thing about this cross tabulation is that we can finter the records without changing the structure (rows, columns) of the table:

```r
Xtab(abundance ~ sample + species, d[d$behavior == "heard",])
```

```
## 3 x 4 sparse Matrix of class "dgCMatrix"
##    AMRO BTNW CANG OVEN
## S1    .    1    2    1
## S2    1    .    .    .
## S3    .    .    .    .
```

```r
Xtab(abundance ~ sample + species, d[d$behavior == "seen",])
```

```
## 3 x 4 sparse Matrix of class "dgCMatrix"
##    AMRO BTNW CANG OVEN
## S1    .    .    .    .
## S2    .    .    1    .
## S3    .    .    .    .
```

Now let's do this for the real data. We have no abundance column, because each row stands for exactly one individual. We can add a column with 1's, or we can just count the number of rows by using only the right-hand-side of the formula in `Xtab`. `ytot` will be our total count matrix for now.

We also want to filter the records to contain only `S`ongs and `C`alls, without `V`visual detections:

```r
table(josm$counts$DetectType1, useNA="always")
```

```
##
##     C     S     V  <NA>
##  9180 41808  1384     0
```

We use `SiteID` for row names, because only 1 station and visit was done at each site:

```r
ytot <- Xtab(~ SiteID + SpeciesID , josm$counts[josm$counts$DetectType1 != "V'
```

See how not storing 0's affect size compared to the long formar and an ordinary wide matrix

```
## 2-column data frame as reference
tmp <- as.numeric(object.size(
  josm$counts[josm$counts$DetectType1 != "V", c("StationID", "SpeciesID")]))
## spare matrix
as.numeric(object.size(ytot)) / tmp
```

```
## [1] 0.1366
```

```
## dense matrix
as.numeric(object.size(as.matrix(ytot))) / tmp
```

```
## [1] 1.106
```

```
## matrix fill
sum(ytot > 0) / prod(dim(ytot))
```

```
## [1] 0.04911
```

Check if counts are as expected:

```
max(ytot) # this is interesting
```

```
## [1] 200
```

```
sort(apply(as.matrix(ytot), 2, max)) # it is CANG
```

```
## BUFF BWTE COGO COHA DCCO GWTE HOLA NHOW NSHO RTHU WWSC CANV NOPI
##    0    0    0    0    0    0    0    0    0    0    0    0    0
## AMBI AMCO AMGO BAEA BAOR BEKI BOWA CONI CSWA EAPH GBHE GCTH GGOW
##    1    1    1    1    1    1    1    1    1    1    1    1    1
## GHOW HOWR LEOW MERL NESP NOGO NOHA NSWO PBGR RBGU RTHA SAVS SPSA
##    1    1    1    1    1    1    1    1    1    1    1    1    1
## WBNU BRBL CAGU MYWA SNBU VEER AMKE AMWI BADO BARS BBWO BHCO BLBW
##    1    1    1    1    1    1    2    2    2    2    2    2    2
## BLPW BLTE BWHA COGR DOWO EAKI HAWO KILL LEYE NAWA NOPO OSFL OSPR
##    2    2    2    2    2    2    2    2    2    2    2    2    2
## PIWO PUFI RNDU SORA SSHA COSN AMCR AMRO ATTW BHVI BOCH BRCR BTNW
##    2    2    2    2    2    2    3    3    3    3    3    3    3
## CMWA FOSP FRGU GCKI MAWR MOWA NOFL PHVI SACR SOSA SOSP SPGR TRES
##    3    3    3    3    3    3    3    3    3    3    3    3    3
```

```
## WETA WIWA WIWR YBSA FOTE BAWW BBWA BCCH BLJA CAWA CONW COTE GRYE
##    3    3    3    3    3    4    4    4    4    4    4    4    4
## NOWA NRWS OCWA REVI RNGR RUBL RWBL WAVI WEWP WISN YBFL YWAR ALFL
##    4    4    4    4    4    4    4    4    4    4    4    4    5
## AMRE CHSP CORA EVGR HETH LCSP RBGR RBNU RCKI SWSP CCSP COYE DEJU
##    5    5    5    5    5    5    5    5    5    5    6    6    6
## LEFL LISP MAWA OVEN RUGR SWTH BOGU MALL GRAJ PAWA WTSP YRWA COLO
##    6    6    6    6    6    6    7    7    8    8    8    8    9
## TEWA AMPI WWCR CEDW PISI RECR CANG
##   12   12   20   23   50   51  200
```

```
## lyover (FO) flock (FL) beyond 100m distance
head(josm$counts[
  josm$counts$SiteID == rownames(ytot)[which(ytot[,"CANG"] == 200)] &
  josm$counts$SpeciesID == "CANG",])
```

We can check overall mean counts

```
round(sort(colMeans(ytot)), 4)
```

```
##   BUFF   BWTE   COGO   COHA   DCCO   GWTE   HOLA   NHOW   NSHO
## 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
##   RTHU   WWSC   CANV   NOPI   GBHE   GCTH   GHOW   LEOW   NOHA
## 0.0000 0.0000 0.0000 0.0000 0.0002 0.0002 0.0002 0.0002 0.0002
##   RBGU   BRBL   CAGU   AMCO   BAEA   BARS   NESP   NOGO   NOPO
## 0.0002 0.0002 0.0002 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004
##   NSWO   RNDU   SNBU   VEER   BEKI   CSWA   MERL   SAVS   SSHA
## 0.0004 0.0004 0.0004 0.0004 0.0007 0.0007 0.0007 0.0007 0.0007
##   MYWA   AMKE   BAOR   OSPR   SPGR   WBNU   AMGO   AMWI   BOWA
## 0.0007 0.0009 0.0009 0.0009 0.0009 0.0009 0.0011 0.0011 0.0011
##   CONI   EAPH   HOWR   NRWS   BLTE   COGR   EAKI   GGOW   NAWA
## 0.0011 0.0011 0.0011 0.0011 0.0013 0.0013 0.0013 0.0013 0.0013
##   COSN   COTE   FRGU   MAWR   FOTE   KILL   RTHA   BADO   BLBW
## 0.0013 0.0015 0.0015 0.0015 0.0015 0.0018 0.0020 0.0024 0.0024
##   AMBI   PBGR   SPSA   AMPI   BHCO   BWHA   SOSP   RUBL   MALL
## 0.0028 0.0028 0.0028 0.0028 0.0031 0.0037 0.0042 0.0044 0.0046
##   PUFI   DOWO   SORA   LEYE   ATTW   HAWO   RNGR   BBWO   BLJA
## 0.0048 0.0059 0.0068 0.0094 0.0096 0.0101 0.0101 0.0107 0.0134
##   BOGU   AMCR   EVGR   RWBL   OSFL   LCSP   TRES   FOSP   WEWP
```

```
## 0.0140 0.0166 0.0169 0.0169 0.0186 0.0193 0.0201 0.0217 0.0232
##   WIWA   PIWO   RECR   SOSA   YWAR   GCKI   BLPW   CAWA   SACR
## 0.0236 0.0256 0.0269 0.0269 0.0291 0.0304 0.0306 0.0315 0.0322
##   BTNW   NOWA   OCWA   BRCR   CCSP   COLO   PHVI   CONW   CEDW
## 0.0335 0.0341 0.0359 0.0381 0.0385 0.0387 0.0394 0.0429 0.0449
##   RUGR   MOWA   WAVI   BCCH   BOCH   NOFL   SWSP   GRYE   WWCR
## 0.0475 0.0477 0.0582 0.0593 0.0593 0.0622 0.0659 0.0685 0.0751
##   AMRO   RBNU   BBWA   CMWA   BHVI   COYE   YBFL   YBSA   AMRE
## 0.0757 0.0766 0.0810 0.0812 0.0814 0.0814 0.0873 0.0878 0.0889
##   BAWW   LEFL   WETA   WISN   CORA   WIWR   ALFL   MAWA   PISI
## 0.0963 0.0974 0.1086 0.1280 0.1401 0.1466 0.1582 0.1727 0.1775
##   RBGR   LISP   DEJU   GRAJ   CANG   PAWA   REVI   RCKI   HETH
## 0.1832 0.2169 0.2725 0.2898 0.3018 0.3053 0.3344 0.3898 0.4344
##   CHSP   SWTH   WTSP   OVEN   YRWA   TEWA
## 0.4460 0.7402 0.8091 0.8831 0.8934 1.2221
```

2.3 Joining species data with predictors

Let's join the species counts with the survey attributes. This is how we can prepare the input data for regression analysis.

```
spp <- "OVEN" # which species
josm$species[spp,]

compare_sets(rownames(josm$surveys),rownames(ytot))
```

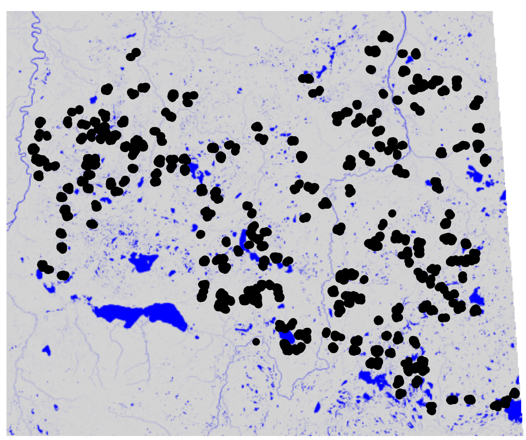
```
##        xlength ylength intersect union xbutnoty ybutnotx
## labels    4569    4569      4569  4569        0        0
## unique    4569    4569      4569  4569        0        0
```

```
x <- josm$surveys
x$y <- as.numeric(ytot[rownames(x), spp])
```

2.4 Explore predictor variables

Locations

```
library(raster)
library(sp)
rr <- stack("./_data/josm/landcover-hfi2016.grd")
#' Define CRS NAD83 for our sites
xy <- x[,c("Longitude", "Latitude")]
coordinates(xy) <- ~ Longitude + Latitude
proj4string(xy) <- "+proj=longlat +ellps=GRS80 +datum=NAD83 +no_defs"
xy <- spTransform(xy, proj4string(rr))
col <- colorRampPalette(c("lightgrey", "blue"))(100)
plot(rr[["Water"]], col=col, axes=FALSE, box=FALSE, legend=FALSE)
plot(xy, add=TRUE, pch=19, cex=0.5)
```



```
cn <- c("Open", "Water", "Agr", "UrbInd", "SoftLin", "Roads",
  "Decid", "OpenWet", "Conif", "ConifWet")
#plot(x[,cn])
```

Exercise
Play with the data to understand the distributions and associations. Use `summary`, `table`, `hist`, `plot`, etc.

Chapter 3

A Primer in Regression Techniques

> All models are wrong, but some are useful – Box

3.1 Introduction

This chapter will provide all the foundations we need for the coming chapters. It is not intended as a general and all-exhaustive introduction to regression techniques, but rather the minimum requirement moving forwards. We will also hone our data processing and plotting skills.

3.2 Prerequisites

```
library(mefa4)    # Data manipulation
library(mgcv)     # GAMs
library(pscl)     # zero-inflated models
library(lme4)     # GLMMs
library(MASS)     # Negative Binomial GLM
library(partykit) # regression trees
library(intrval)  # interval magic
library(opticut)  # optimal partitioning
library(visreg)   # regression visualization
```

```
library(MuMIn)    # multi-model inference
source("functions.R")
```

3.3 Poisson null model

```
load("./_data/josm/josm.rda")

spp <- "OVEN" # which species

ytot <- Xtab(~ SiteID + SpeciesID , josm$counts[josm$counts$DetectType1 != "V'
ytot <- ytot[,colSums(ytot > 0) > 0]
x <- data.frame(
  josm$surveys,
  y=as.numeric(ytot[rownames(x), spp]))

table(x$y)
```

```
##
##    0    1    2    3    4    5    6
## 2493  883  656  363  132   29   13
```

$E[Y_i] = \lambda_i = \lambda$, $(Y_i \mid \lambda) \sim Poisson(\lambda)$, $log(\lambda) = \beta_0$, $\lambda = e^{\beta_0}$

Null model

```
mP0 <- glm(y ~ 1, data=x, family=poisson)
mean(x$y)
```

```
## [1] 0.8831
```

```
mean(fitted(mP0))
```

```
## [1] 0.8831
```

```
exp(coef(mP0))
```

```
## (Intercept)
##      0.8831
```

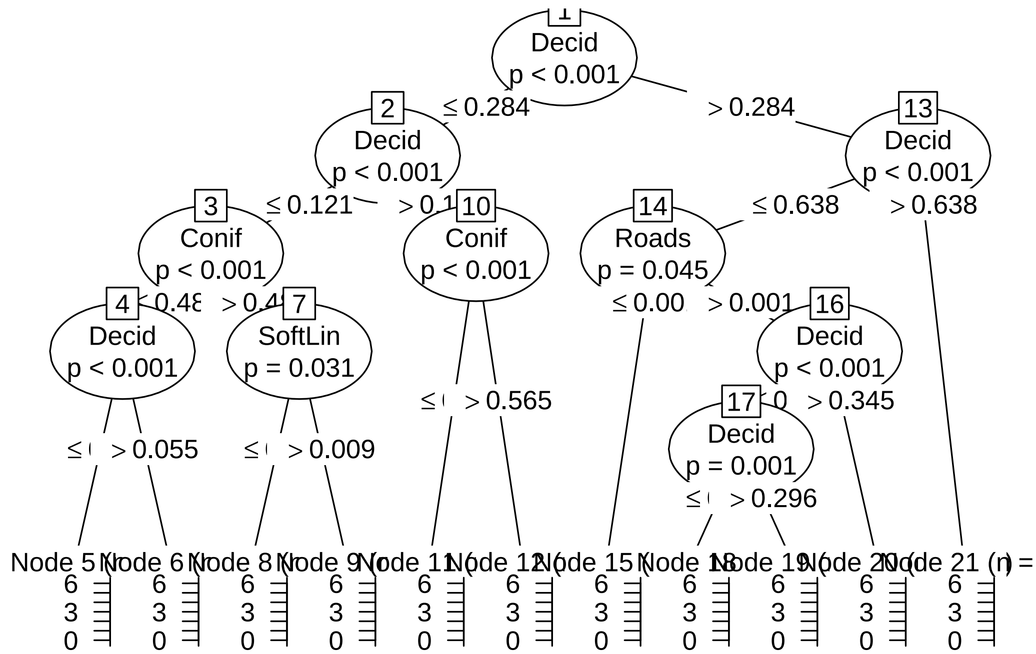
```
summary(mP0)
```

```
##
## Call:
## glm(formula = y ~ 1, family = poisson, data = x)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -1.33   -1.33   -1.33    1.02    3.57
##
## Coefficients:
##             Estimate Std. Error z value           Pr(>|z|)
## (Intercept)  -0.1243     0.0157   -7.89 0.0000000000000029 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 7424.8  on 4568  degrees of freedom
## Residual deviance: 7424.8  on 4568  degrees of freedom
## AIC: 12573
##
## Number of Fisher Scoring iterations: 6
```

3.4 Exploring covariates

What is a useful covariate?

```
mCT <- ctree(y ~ Open + Water + Agr + UrbInd + SoftLin + Roads +
  Decid + OpenWet + Conif + ConifWet, data=x)
plot(mCT, cex=0.5)
```



3.5 Poisson GLM with one covariate

```
mP1 <- glm(y ~ Decid, data=x, family=poisson)
mean(x$y)
```

```
## [1] 0.8831
```

```
mean(fitted(mP0))
```

```
## [1] 0.8831
```

```
summary(mP1)
```

```
##
## Call:
## glm(formula = y ~ Decid, family = poisson, data = x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.291  -0.977  -0.790   0.469   4.197
##
## Coefficients:
##             Estimate Std. Error z value            Pr(>|z|)
## (Intercept)  -1.1643     0.0352   -33.1 <0.0000000000000002 ***
## Decid         2.1338     0.0537    39.7 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 7424.8  on 4568  degrees of freedom
## Residual deviance: 5736.9  on 4567  degrees of freedom
## AIC: 10887
##
## Number of Fisher Scoring iterations: 6
```

```
AIC(mP0, mP1)

round(rbind(mP0=R2dev(mP0), mP1=R2dev(mP1)), 4)
```

```
##         R2  R2adj Deviance Dev0 DevR  df0  dfR p_value
## mP0 0.0000 0.0000        0 7425 7425 4568 4568       0
## mP1 0.2273 0.2272     1688 7425 5737 4568 4567       0
```

```
xnew <- data.frame(Decid=seq(0, 1, 0.01))
CI0 <- predict_sim(mP0, xnew, interval="confidence", level=0.95, B=999)
PI0 <- predict_sim(mP0, xnew, interval="prediction", level=0.95, B=999)
CI1 <- predict_sim(mP1, xnew, interval="confidence", level=0.95, B=999)
PI1 <- predict_sim(mP1, xnew, interval="prediction", level=0.95, B=999)

## nominal coverage is 95%
sum(x$y %[]% predict_sim(mP0, interval="prediction", level=0.95, B=999)[,c("lwr", "up
```

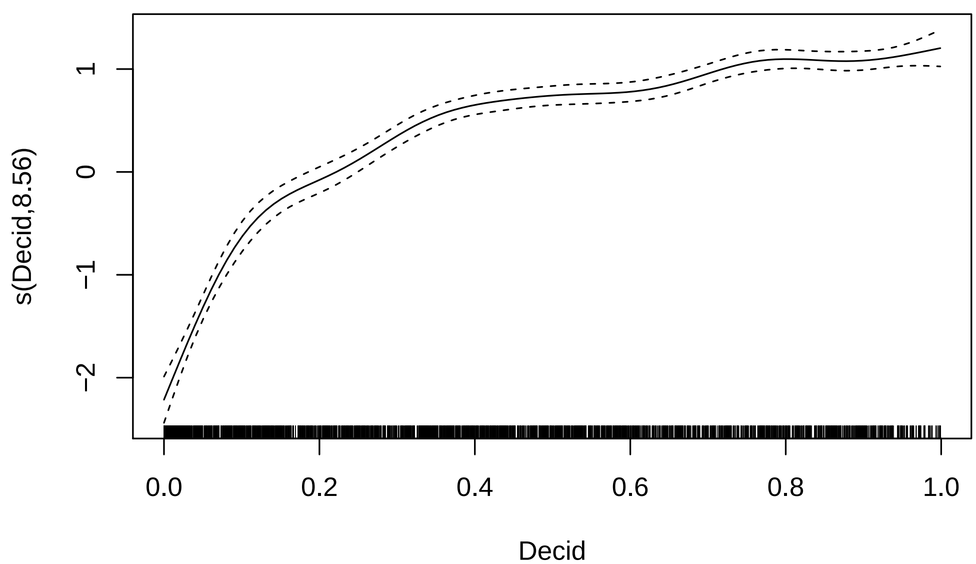
```
## [1] 0.9619
```

```
sum(x$y %[]% predict_sim(mP1, interval="prediction", level=0.95, B=999)[,c("lwr", "up
```

```
## [1] 0.9709
```

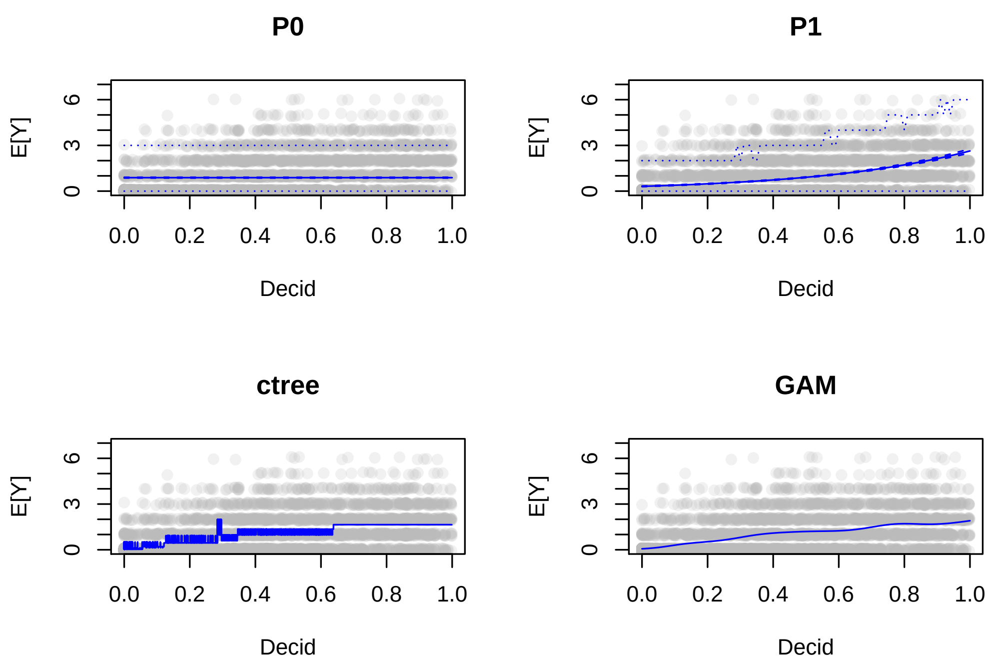
3.6 Additive model

```
mGAM <- mgcv::gam(y ~ s(Decid), x, family=poisson)
plot(mGAM)
```




```
fitCT <- predict(mCT, x[order(x$Decid),])
fitGAM <- predict(mGAM, xnew, type="response")

op <- par(mfrow=c(2,2))
plot(jitter(y, 0.5) ~ Decid, x, xlab="Decid", ylab="E[Y]",
  ylim=c(0, max(PI1$upr)+1), pch=19, col="#bbbbbb33", main="P0")
lines(CI0$fit ~ xnew$Decid, lty=1, col=4)
lines(CI0$lwr ~ xnew$Decid, lty=2, col=4)
lines(CI0$upr ~ xnew$Decid, lty=2, col=4)
lines(PI0$lwr ~ xnew$Decid, lty=3, col=4)
lines(PI0$upr ~ xnew$Decid, lty=3, col=4)

plot(jitter(y, 0.5) ~ Decid, x, xlab="Decid", ylab="E[Y]",
```

```
  ylim=c(0, max(PI1$upr)+1), pch=19, col="#bbbbbb33", main="P1")
lines(CI1$fit ~ xnew$Decid, lty=1, col=4)
lines(CI1$lwr ~ xnew$Decid, lty=2, col=4)
lines(CI1$upr ~ xnew$Decid, lty=2, col=4)
lines(PI1$lwr ~ xnew$Decid, lty=3, col=4)
lines(PI1$upr ~ xnew$Decid, lty=3, col=4)

plot(jitter(y, 0.5) ~ Decid, x, xlab="Decid", ylab="E[Y]",
  ylim=c(0, max(PI1$upr)+1), pch=19, col="#bbbbbb33", main="ctree")
lines(fitCT ~ x$Decid[order(x$Decid)], lty=1, col=4)

plot(jitter(y, 0.5) ~ Decid, x, xlab="Decid", ylab="E[Y]",
  ylim=c(0, max(PI1$upr)+1), pch=19, col="#bbbbbb33", main="GAM")
lines(fitGAM ~ xnew$Decid, lty=1, col=4)
```




```
par(op)
```

Exercise
Play with GAM and other variables to understand responses: plot(mgcv::gam(y ~ s(<variable_name>), data=x, family=poisson))

3.7 Multiple main effects

```r
mP2 <- step(glm(y ~ Open + Agr + UrbInd + SoftLin + Roads +
  Decid + OpenWet + Conif + ConifWet +
  OvernightRain + TSSR + DAY + Longitude + Latitude,
  data=x, family=poisson), trace=0)

summary(mP2)
```

```
##
## Call:
## glm(formula = y ~ Open + UrbInd + Decid + OpenWet + Conif + ConifWet +
##     TSSR + DAY + Longitude + Latitude, family = poisson, data = x)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.763  -0.986  -0.674   0.451   4.624
##
## Coefficients:
##             Estimate Std. Error z value          Pr(>|z|)
## (Intercept) -5.88293    1.30223   -4.52      0.0000062546826 ***
## Open        -3.47428    0.65867   -5.27      0.0000001330000 ***
## UrbInd      -1.66883    0.54216   -3.08              0.00208 **
## Decid        0.83372    0.25957    3.21              0.00132 **
## OpenWet     -0.74076    0.30238   -2.45              0.01430 *
## Conif       -0.88558    0.26566   -3.33              0.00086 ***
## ConifWet    -1.89423    0.27170   -6.97      0.0000000000031 ***
## TSSR        -1.23416    0.24984   -4.94      0.0000007818641 ***
## DAY         -2.87970    0.52686   -5.47      0.0000000460898 ***
## Longitude    0.03831    0.00877    4.37      0.0000124210771 ***
```

```
## Latitude     0.20930    0.02309    9.06 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 7424.8  on 4568  degrees of freedom
## Residual deviance: 5501.1  on 4558  degrees of freedom
## AIC: 10669
##
## Number of Fisher Scoring iterations: 6
```

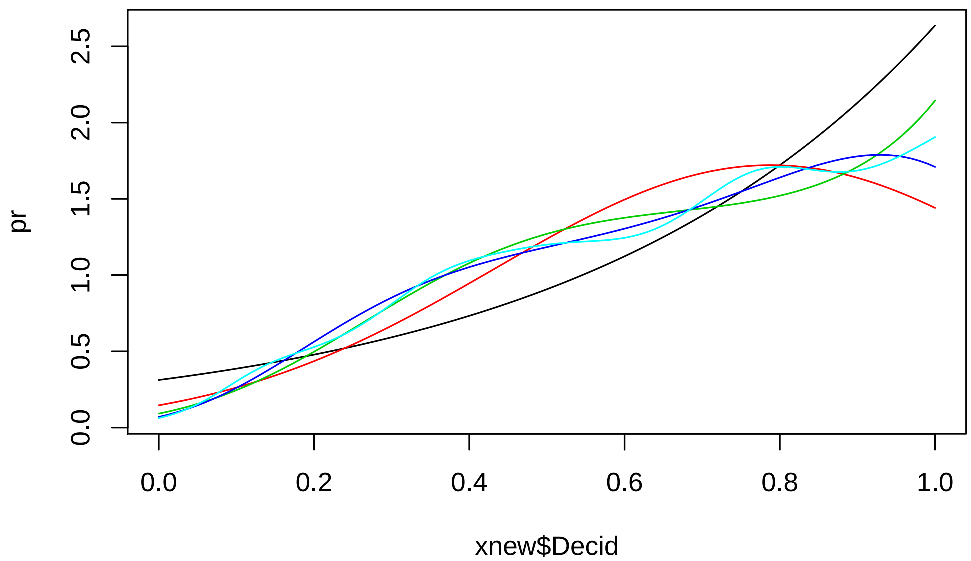
```
AIC(mP0, mP1, mP2)

round(rbind(mP0=R2dev(mP0), mP1=R2dev(mP1), mP2=R2dev(mP2)), 4)
```

```
##         R2  R2adj Deviance Dev0 DevR  df0  dfR p_value
## mP0 0.0000 0.0000        0 7425 7425 4568 4568       0
## mP1 0.2273 0.2272     1688 7425 5737 4568 4567       0
## mP2 0.2591 0.2575     1924 7425 5501 4568 4558       0
```

3.8 Nonlinear terms

Polynomials

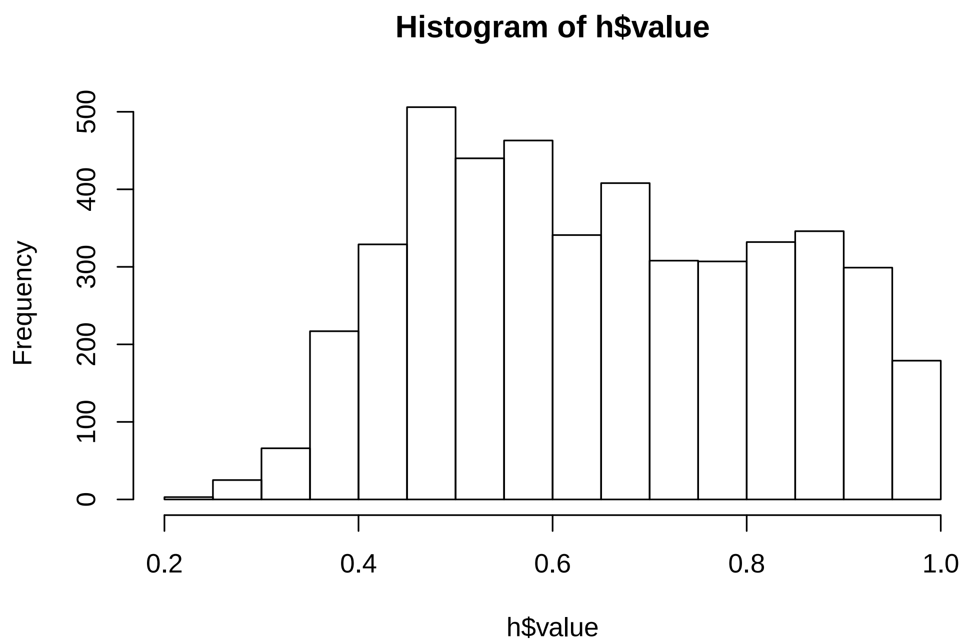
```
mP12 <- glm(y ~ Decid + I(Decid^2), data=x, family=poisson)
mP13 <- glm(y ~ Decid + I(Decid^2) + I(Decid^3), data=x, family=poisson)
mP14 <- glm(y ~ Decid + I(Decid^2) + I(Decid^3) + I(Decid^4), data=x, family=poisson)
AIC(mP1, mP12, mP13, mP14)

pr <- cbind(
  predict(mP1, xnew, type="response"),
  predict(mP12, xnew, type="response"),
  predict(mP13, xnew, type="response"),
  predict(mP14, xnew, type="response"),
  fitGAM)
matplot(xnew$Decid, pr, lty=1, type="l")
```



3.9 Categorical variables

Categories

```
cn <- c("Open", "Water", "Agr", "UrbInd", "SoftLin", "Roads", "Decid",
  "OpenWet", "Conif", "ConifWet")
h <- find_max(x[,cn])
hist(h$value)
```




```
table(h$index)
```

```
##
##      Open     Water       Agr    UrbInd   SoftLin     Roads     Decid
##        12        10         4        14         0         2      2084
##   OpenWet     Conif  ConifWet
##       160       745      1538
```

```
x$hab <- droplevels(h$index)

mP3 <- glm(y ~ hab, data=x, family=poisson)

summary(mP3)
```

```
##
## Call:
## glm(formula = y ~ hab, family = poisson, data = x)
##
## Deviance Residuals:
```

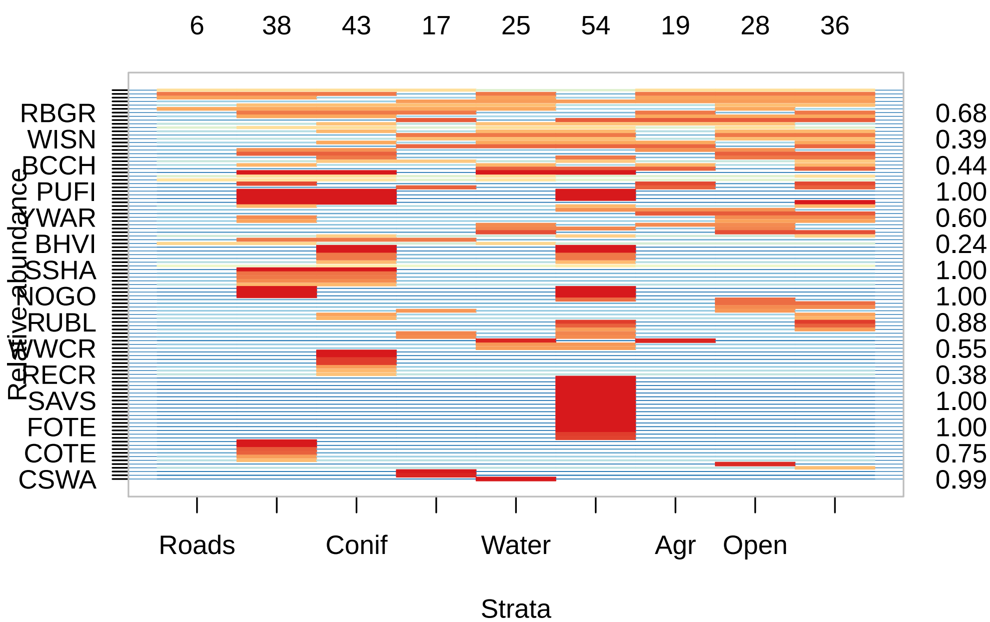
```
##    Min      1Q  Median      3Q     Max
## -1.691  -0.873  -0.817   0.449   4.832
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.386      0.577   -2.40   0.0163 *
## habWater       1.030      0.690    1.49   0.1357
## habAgr         0.693      0.913    0.76   0.4477
## habUrbInd      0.134      0.764    0.17   0.8612
## habRoads     -10.916    201.285   -0.05   0.9567
## habDecid       1.744      0.578    3.02   0.0025 **
## habOpenWet     0.422      0.591    0.71   0.4755
## habConif       0.913      0.579    1.58   0.1150
## habConifWet    0.288      0.579    0.50   0.6185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 7424.8  on 4568  degrees of freedom
## Residual deviance: 5997.2  on 4560  degrees of freedom
## AIC: 11161
##
## Number of Fisher Scoring iterations: 10
```

```r
AIC(mP0, mP1, mP2, mP3)

round(rbind(mP0=R2dev(mP0), mP1=R2dev(mP1), mP2=R2dev(mP2), mP3=R2dev(mP3)), 4
```

```
##         R2  R2adj Deviance Dev0 DevR  df0  dfR p_value
## mP0 0.0000 0.0000        0 7425 7425 4568 4568       0
## mP1 0.2273 0.2272     1688 7425 5737 4568 4567       0
## mP2 0.2591 0.2575     1924 7425 5501 4568 4558       0
## mP3 0.1923 0.1909     1428 7425 5997 4568 4560       0
```

3.9.1 Optimal partitioning

```
oc <- opticut(as.matrix(ytot) ~ 1, strata = x$hab, dist="poisson")
plot(oc)
```



3.9.2 Finding optimal combinations of factor levels

Categorical and compositional data

```
# dominant hab
M <- model.matrix(~hab-1, x)
colnames(M) <- levels(x$hab)
ol1 <- optilevels(x$y, M, dist="poisson")
sort(exp(coef(bestmodel(ol1))))
```

```
## `Open+UrbInd+Roads+OpenWet+ConifWet`
##                               0.3366
##                     `Water+Agr+Conif`
##                               0.6232
```

```
##                                Decid
##                               1.4304
```

```
## estimates
exp(ol1$coef)
```

```
##         Open  Water    Agr UrbInd      Roads Decid OpenWet
##  [1,] 0.2500 0.7000 0.5000 0.2857 0.00000454  1.43  0.3813
##  [2,] 0.2692 0.7000 0.5000 0.2692 0.00000454  1.43  0.3813
##  [3,] 0.2692 0.6238 0.5000 0.2692 0.00000454  1.43  0.3812
##  [4,] 0.2692 0.6232 0.6232 0.2692 0.00000454  1.43  0.3813
##  [5,] 0.3325 0.6232 0.6232 0.3325 0.00000454  1.43  0.3813
##  [6,] 0.3370 0.6232 0.6232 0.3370 0.00000454  1.43  0.3370
##  [7,] 0.3366 0.6232 0.6232 0.3366 0.33661645  1.43  0.3366
##  [8,]     NA     NA     NA     NA         NA    NA      NA
##  [9,]     NA     NA     NA     NA         NA    NA      NA
##        Conif ConifWet
##  [1,] 0.6228   0.3336
##  [2,] 0.6228   0.3336
##  [3,] 0.6238   0.3336
##  [4,] 0.6232   0.3336
##  [5,] 0.6232   0.3325
##  [6,] 0.6232   0.3370
##  [7,] 0.6232   0.3366
##  [8,]     NA       NA
##  [9,]     NA       NA
```

```
## optimal classification
ol1$rank
```

```
##       Open Water Agr UrbInd Roads Decid OpenWet Conif ConifWet
##  [1,]    2     8   6      3     1     9       5     7        4
##  [2,]    2     7   5      2     1     8       4     6        3
##  [3,]    2     6   5      2     1     7       4     6        3
##  [4,]    2     5   5      2     1     6       4     5        3
##  [5,]    2     4   4      2     1     5       3     4        2
##  [6,]    2     3   3      2     1     4       2     3        2
##  [7,]    1     2   2      1     1     3       1     2        1
##  [8,]   NA    NA  NA     NA    NA    NA      NA    NA       NA
```

```
##  [9,]   NA    NA  NA     NA    NA    NA      NA    NA       NA
```

```r
ol1$levels[[length(ol1$levels)]]
```

```
##                                    Open
## "Open+UrbInd+Roads+OpenWet+ConifWet"
##                                   Water
##                      "Water+Agr+Conif"
##                                     Agr
##                      "Water+Agr+Conif"
##                                  UrbInd
## "Open+UrbInd+Roads+OpenWet+ConifWet"
##                                   Roads
## "Open+UrbInd+Roads+OpenWet+ConifWet"
##                                   Decid
##                                 "Decid"
##                                 OpenWet
## "Open+UrbInd+Roads+OpenWet+ConifWet"
##                                   Conif
##                      "Water+Agr+Conif"
##                                ConifWet
## "Open+UrbInd+Roads+OpenWet+ConifWet"
```

```r
# composition
ol2 <- optilevels(x$y, x[,cn], dist="poisson")
sort(exp(coef(bestmodel(ol2))))
```

```
##                                  Open
##                               0.04936
##                              ConifWet
##                               0.18363
## `Agr+UrbInd+SoftLin+OpenWet+Conif`
##                               0.50444
##                        `Water+Roads`
##                               1.15427
##                                 Decid
##                               2.49578
```

```r
## estimates
exp(ol2$coef)
```

```
##          Open Water    Agr UrbInd SoftLin Roads Decid OpenWet
##  [1,] 0.04790 1.174 0.5579 0.4313  0.5102 1.038 2.496  0.5627
##  [2,] 0.04791 1.174 0.5625 0.4313  0.5104 1.038 2.496  0.5625
##  [3,] 0.04795 1.174 0.5628 0.4324  0.4944 1.046 2.497  0.5628
##  [4,] 0.04790 1.156 0.5628 0.4291  0.4940 1.156 2.494  0.5628
##  [5,] 0.04782 1.146 0.5632 0.4917  0.4917 1.146 2.493  0.5632
##  [6,] 0.04936 1.154 0.5044 0.5044  0.5044 1.154 2.496  0.5044
##  [7,]      NA    NA     NA     NA      NA    NA    NA      NA
##  [8,]      NA    NA     NA     NA      NA    NA    NA      NA
##  [9,]      NA    NA     NA     NA      NA    NA    NA      NA
## [10,]      NA    NA     NA     NA      NA    NA    NA      NA
##        Conif ConifWet
##  [1,] 0.4941   0.1827
##  [2,] 0.4941   0.1827
##  [3,] 0.4944   0.1828
##  [4,] 0.4940   0.1828
##  [5,] 0.4917   0.1826
##  [6,] 0.5044   0.1836
##  [7,]     NA       NA
##  [8,]     NA       NA
##  [9,]     NA       NA
## [10,]     NA       NA
```

```r
## optimal classification
ol2$rank
```

```
##       Open Water Agr UrbInd SoftLin Roads Decid OpenWet Conif
##  [1,]    1     9   6      3       5     8    10       7     4
##  [2,]    1     8   6      3       5     7     9       6     4
##  [3,]    1     7   5      3       4     6     8       5     4
##  [4,]    1     6   5      3       4     6     7       5     4
##  [5,]    1     5   4      3       3     5     6       4     3
##  [6,]    1     4   3      3       3     4     5       3     3
##  [7,]   NA    NA  NA     NA      NA    NA    NA      NA    NA
##  [8,]   NA    NA  NA     NA      NA    NA    NA      NA    NA
```

```
##  [9,]   NA    NA  NA     NA      NA    NA    NA      NA    NA
## [10,]   NA    NA  NA     NA      NA    NA    NA      NA    NA
##       ConifWet
##  [1,]        2
##  [2,]        2
##  [3,]        2
##  [4,]        2
##  [5,]        2
##  [6,]        2
##  [7,]       NA
##  [8,]       NA
##  [9,]       NA
## [10,]       NA
```

```
head(mefa4::groupSums(as.matrix(x[,cn]), 2, ol2$levels[[length(ol2$levels)]]))
```

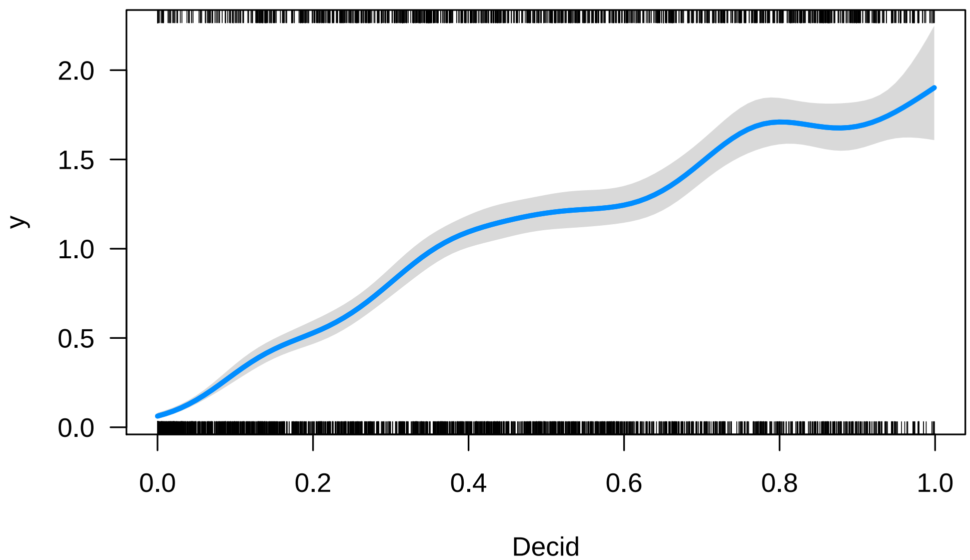
```
##         Open Water+Roads Agr+UrbInd+SoftLin+OpenWet+Conif
## CL10102    0    0.073010                          0.03326
## CL10106    0    0.008431                          0.60166
## CL10108    0    0.008431                          0.60166
## CL10109    0    0.036146                          0.08157
## CL10111    0    0.050734                          0.13353
## CL10112    0    0.050734                          0.13353
##           Decid ConifWet
## CL10102 0.85691 0.036819
## CL10106 0.04429 0.345625
## CL10108 0.04429 0.345625
## CL10109 0.88228 0.000000
## CL10111 0.81394 0.001797
## CL10112 0.81394 0.001797
```

3.10 Interactions

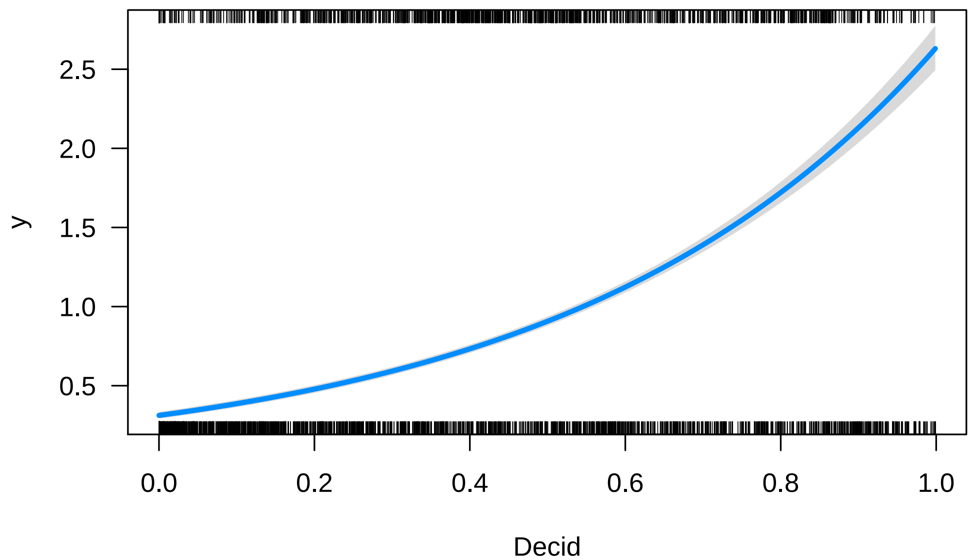
```
mP4 <- glm(y ~ Decid + ConifWet, data=x, family=poisson)
mP5 <- glm(y ~ Decid * ConifWet, data=x, family=poisson)
AIC(mP0, mP1, mP4, mP5)
summary(mP5)
```

```
##
## Call:
## glm(formula = y ~ Decid * ConifWet, family = poisson, data = x)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.081  -1.022  -0.484   0.374   4.321
##
## Coefficients:
##                 Estimate Std. Error z value            Pr(>|z|)
## (Intercept)      -0.5604     0.0566    -9.9 <0.0000000000000002
## Decid             1.2125     0.0782    15.5 <0.0000000000000002
## ConifWet         -2.3124     0.1490   -15.5 <0.0000000000000002
## Decid:ConifWet    5.3461     0.3566    15.0 <0.0000000000000002
##
## (Intercept)    ***
## Decid          ***
## ConifWet       ***
## Decid:ConifWet ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 7424.8  on 4568  degrees of freedom
## Residual deviance: 5395.2  on 4565  degrees of freedom
## AIC: 10549
##
## Number of Fisher Scoring iterations: 6
```

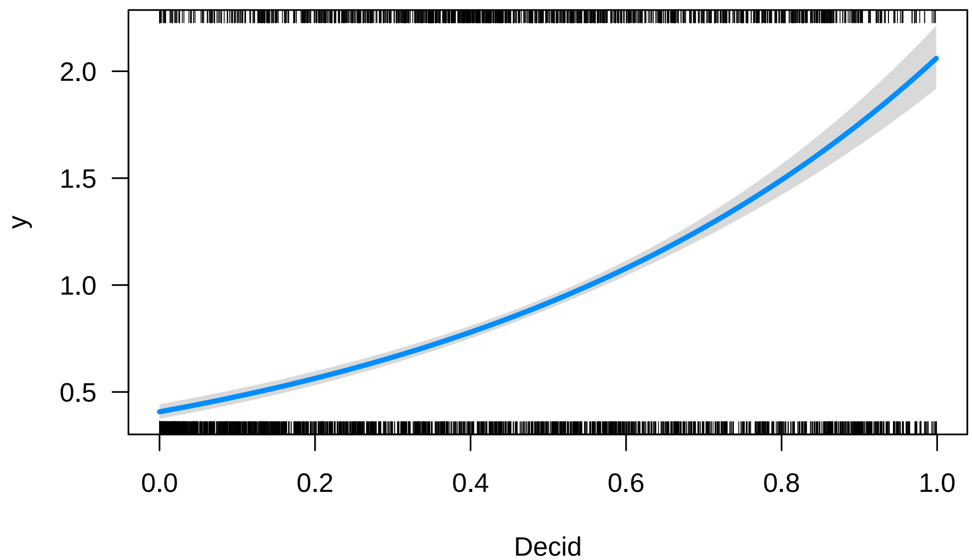
```
visreg(mGAM, scale="response")
```

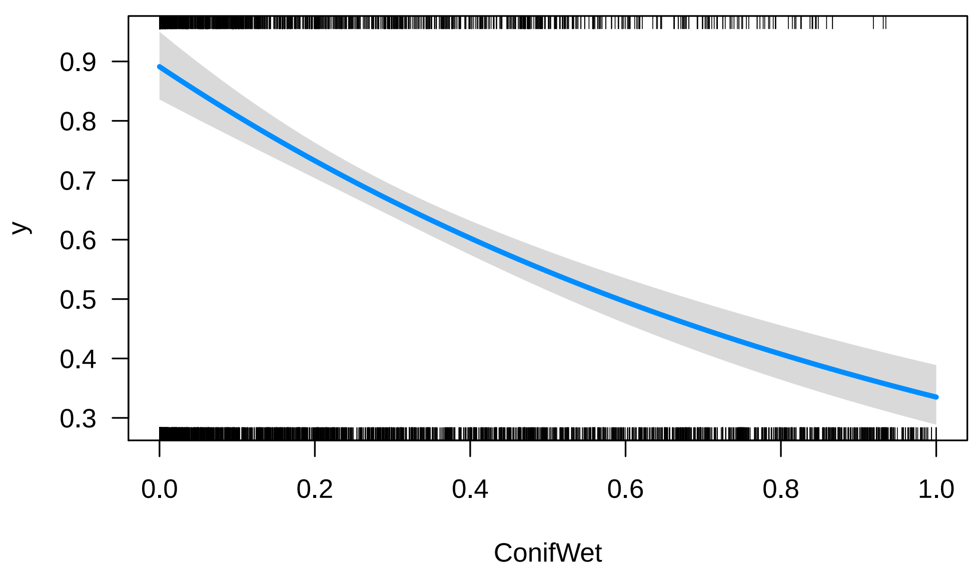



```
visreg(mP1, scale="response")
```

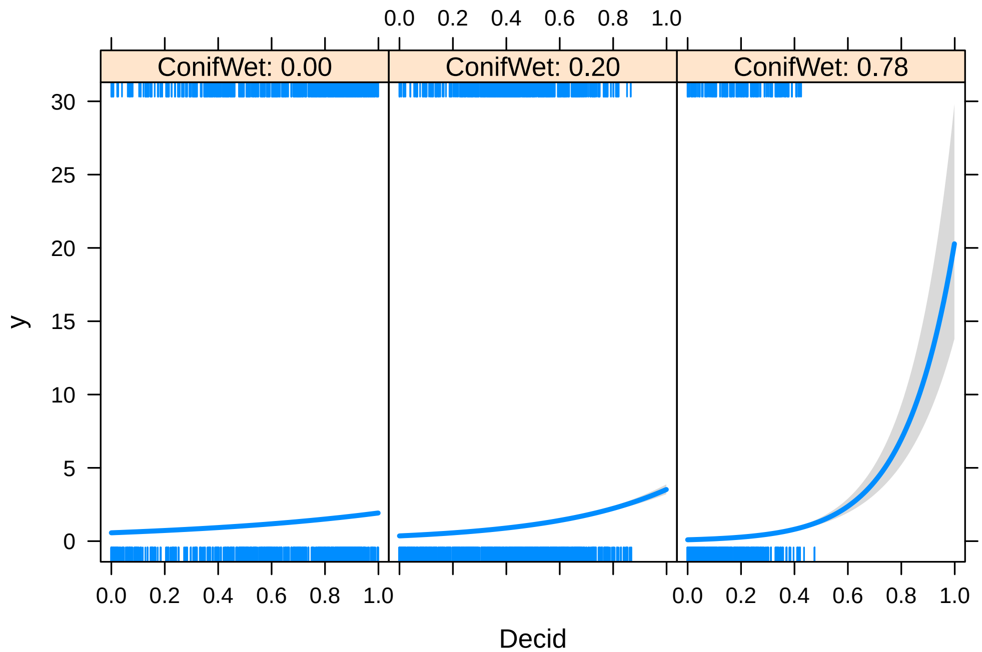



```
visreg(mP4, scale="response")
```


y
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Decid


y
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0
ConifWet

```
visreg(mP5, scale="response", xvar="Decid", by="ConifWet")
```




```
round(rbind(mP0=R2dev(mP0), mP1=R2dev(mP1), mP2=R2dev(mP2), mP3=R2dev(mP3),
  mP4=R2dev(mP4), mP5=R2dev(mP5)), 4)
```

```
##         R2  R2adj Deviance Dev0 DevR  df0  dfR p_value
## mP0 0.0000 0.0000        0 7425 7425 4568 4568       0
## mP1 0.2273 0.2272     1688 7425 5737 4568 4567       0
## mP2 0.2591 0.2575     1924 7425 5501 4568 4558       0
## mP3 0.1923 0.1909     1428 7425 5997 4568 4560       0
## mP4 0.2406 0.2403     1786 7425 5638 4568 4566       0
## mP5 0.2734 0.2729     2030 7425 5395 4568 4565       0
```

```
model.sel(mP0, mP1, mP2, mP3, mP4, mP5)
```

```
## Model selection table
##       (Int)    Dcd     Cnf     CnW   DAY    Ltt     Lng    Opn
## mP5 -0.5604 1.2130         -2.3120
## mP2 -5.8830 0.8337 -0.8856 -1.8940 -2.88 0.2093 0.03831 -3.474
```

```
## mP4 -0.7014 1.6220         -0.9785
## mP1 -1.1640 2.1340
## mP3 -1.3860
## mP0 -0.1243
##         OpW    TSS    UrI hab CnW:Dcd df  logLik  AICc  delta
## mP5                             5.346  4   -5271 10549    0.0
## mP2 -0.7408 -1.234 -1.669             11   -5324 10669  120.0
## mP4                                    3   -5392 10791  241.2
## mP1                                    2   -5442 10887  337.7
## mP3                        +           9   -5572 11162  612.1
## mP0                                    1   -6285 12573 2023.6
##     weight
## mP5      1
## mP2      0
## mP4      0
## mP1      0
## mP3      0
## mP0      0
## Models ranked by AICc(x)
```

3.11 Different error distributions

```
mP <- mP5 # best Poisson
mNB <- glm.nb(y ~ Decid * ConifWet, data=x)
mZIP <- zeroinfl(y ~ Decid * ConifWet | 1, x, dist="poisson")
mZINB <- zeroinfl(y ~ Decid * ConifWet | 1, x, dist="negbin")

AIC(mP, mNB, mZIP, mZINB)
summary(mZINB)
```

```
##
## Call:
## zeroinfl(formula = y ~ Decid * ConifWet | 1, data = x, dist = "negbin")
##
## Pearson residuals:
##    Min     1Q Median     3Q    Max
## -1.190 -0.689 -0.338  0.361  8.956
```

```
##
## Count model coefficients (negbin with log link):
##                 Estimate Std. Error z value             Pr(>|z|)
## (Intercept)      -0.3873     0.0732   -5.29           0.00000012
## Decid             1.1195     0.0911   12.29 < 0.0000000000000002
## ConifWet         -2.4230     0.1589  -15.25 < 0.0000000000000002
## Decid:ConifWet    5.9227     0.3963   14.94 < 0.0000000000000002
## Log(theta)        2.6504     0.5305    5.00           0.00000059
##
## (Intercept)    ***
## Decid          ***
## ConifWet       ***
## Decid:ConifWet ***
## Log(theta)     ***
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value            Pr(>|z|)
## (Intercept)   -1.861      0.193   -9.64 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 14.159
## Number of iterations in BFGS optimization: 22
## Log-likelihood: -5.2e+03 on 6 Df
```

```r
plogis(coef(mZINB, "zero")) # P of 0
```

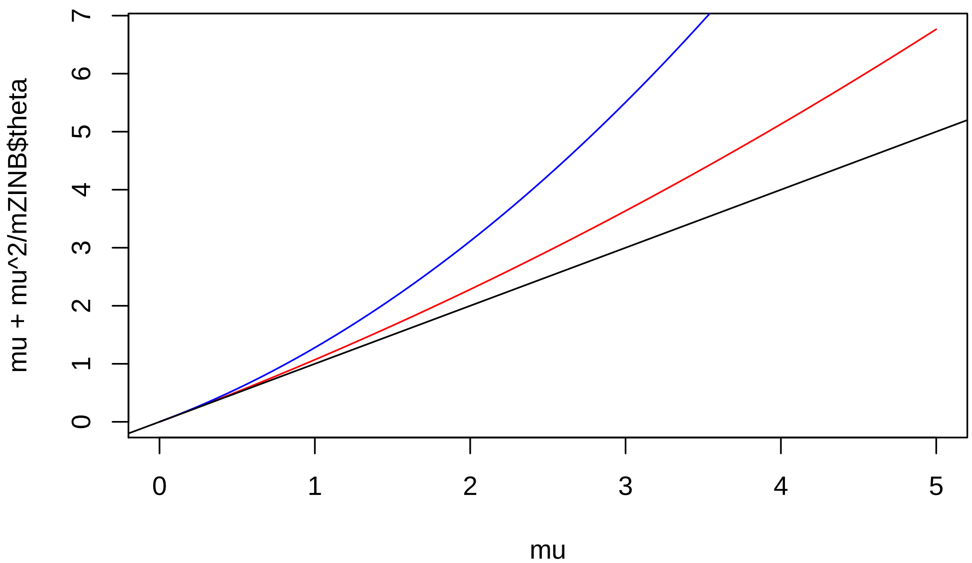
```
## (Intercept)
##      0.1346
```

```r
mZINB$theta # V(mu) = mu + mu^2/theta, ~inverse of variance
```

```
## [1] 14.16
```

```r
# Variance function, 1:1 is Poisson
mu <- seq(0, 5, 0.01)
theta <- mZINB$theta
plot(mu, mu + mu^2/mZINB$theta, type="l", col=2)
lines(mu, mu + mu^2/mNB$theta, type="l", col=4)
```

```
abline(0,1)
```



Poisson-Lognormal random effects, iid. and clustered:

```
mPLN1 <- glmer(y ~ Decid * ConifWet + (1 | SiteID), data=x, family=poisson)
mPLN2 <- glmer(y ~ Decid * ConifWet + (1 | SurveyArea), data=x, family=poisson)
AIC(mP, mNB, mZIP, mZINB, mPLN1, mPLN2)
summary(mPLN2)
```

```
## Generalized linear mixed model fit by maximum likelihood
##   (Laplace Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: y ~ Decid * ConifWet + (1 | SurveyArea)
##    Data: x
##
##      AIC      BIC   logLik deviance df.resid
##    10021    10053    -5006    10011     4564
##
## Scaled residuals:
```

```
##    Min     1Q Median     3Q    Max 
## -1.739 -0.643 -0.320  0.355  6.535 
## 
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  SurveyArea (Intercept) 0.295    0.543   
## Number of obs: 4569, groups:  SurveyArea, 271
## 
## Fixed effects:
##                Estimate Std. Error z value            Pr(>|z|)
## (Intercept)     -0.7459     0.0783   -9.53 <0.0000000000000002
## Decid            1.1967     0.0984   12.16 <0.0000000000000002
## ConifWet        -2.3213     0.1687  -13.76 <0.0000000000000002
## Decid:ConifWet   5.5346     0.3978   13.91 <0.0000000000000002
##                   
## (Intercept)    ***
## Decid          ***
## ConifWet       ***
## Decid:ConifWet ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Correlation of Fixed Effects:
##             (Intr) Decid  ConfWt
## Decid       -0.808              
## ConifWet    -0.610  0.628       
## Decid:CnfWt  0.162 -0.325 -0.670
```

3.12 Counting time effects

```
spp <- "OVEN" # which species

ydur <- Xtab(~ SiteID + Dur + SpeciesID , josm$counts[josm$counts$DetectType1

y <- as.matrix(ydur[[spp]])
head(y)
```

```
##         0-3min 3-5min 5-10min
## CL10102      3      0       0
## CL10106      0      0       0
## CL10108      0      0       0
## CL10109      2      0       1
## CL10111      2      0       0
## CL10112      2      0       0
```

```r
colMeans(y)
```

```
##  0-3min  3-5min 5-10min
## 0.67367 0.09346 0.11600
```

```r
cumsum(colMeans(y))
```

```
##  0-3min  3-5min 5-10min
##  0.6737  0.7671  0.8831
```

```r
x <- data.frame(
  josm$surveys,
  y3=y[,"0-3min"],
  y5=y[,"0-3min"]+y[,"3-5min"],
  y10=rowSums(y))

table(x$y3)
```

```
##
##    0    1    2    3    4    5    6
## 2768  922  576  226   61   14    2
```

```r
table(x$y5)
```

```
##
##    0    1    2    3    4    5    6
## 2643  894  632  285   87   24    4
```

```r
table(x$y10)
```

```
##
##    0    1    2    3    4    5    6
## 2493  883  656  363  132   29   13
```

```r
m3 <- glm(y3 ~ Decid, data=x, family=poisson)
m5 <- glm(y5 ~ Decid, data=x, family=poisson)
m10 <- glm(y10 ~ Decid, data=x, family=poisson)
mean(fitted(m3))
```

```
## [1] 0.6737
```

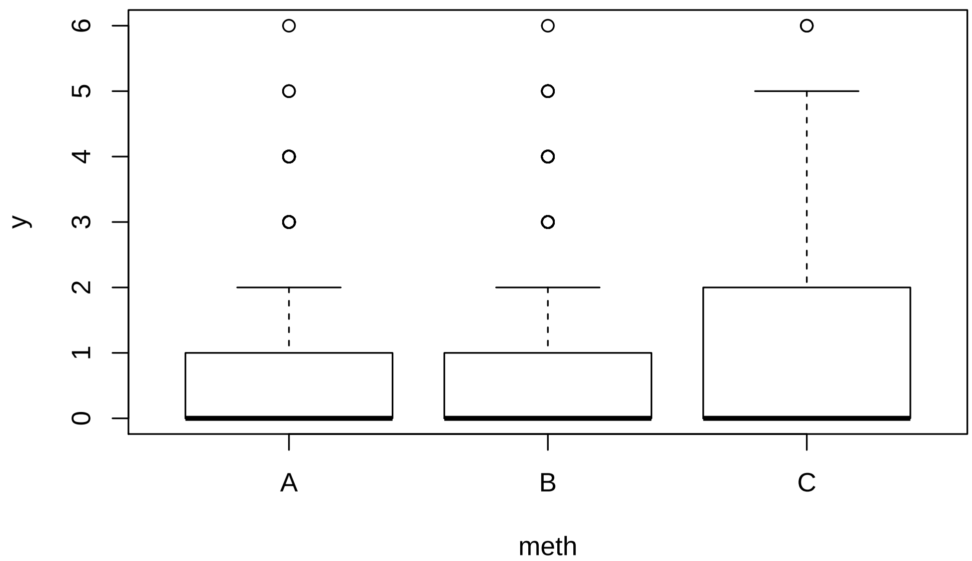
```r
mean(fitted(m5))
```

```
## [1] 0.7671
```

```r
mean(fitted(m10))
```

```
## [1] 0.8831
```

```r
set.seed(1)
x$meth <- sample(c("A", "B", "C"), nrow(x), replace=TRUE)
x$y <- x$y3
x$y[x$meth == "B"] <- x$y5[x$meth == "B"]
x$y[x$meth == "C"] <- x$y10[x$meth == "C"]
boxplot(y ~ meth, x)
```




```
mm <- glm(y ~ meth - 1, data=x, family=poisson)
summary(mm)
```

```
##
## Call:
## glm(formula = y ~ meth - 1, family = poisson, data = x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.312  -1.273  -1.147   0.391   3.980
##
## Coefficients:
##        Estimate Std. Error z value             Pr(>|z|)
## methA   -0.4186     0.0314  -13.32 < 0.0000000000000002 ***
## methB   -0.2100     0.0282   -7.44      0.0000000000001 ***
## methC   -0.1502     0.0280   -5.37      0.0000000794370 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 7233.2  on 4569  degrees of freedom
## Residual deviance: 6938.6  on 4566  degrees of freedom
## AIC: 11646
##
## Number of Fisher Scoring iterations: 6
```

```
exp(coef(mm))
```

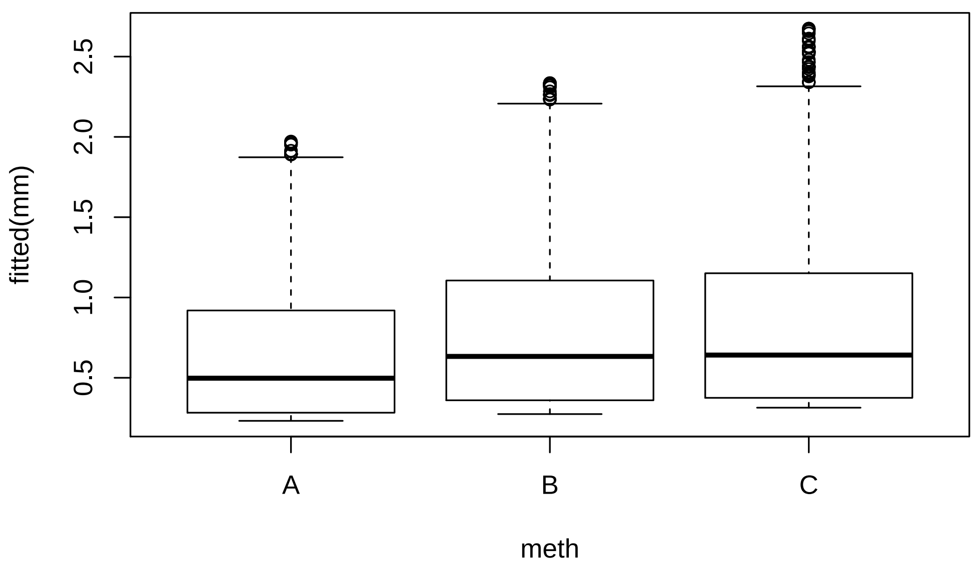
```
##  methA  methB  methC
## 0.6580 0.8106 0.8605
```

```
mm <- glm(y ~ Decid + meth, data=x, family=poisson)
summary(mm)
```

```
##
## Call:
## glm(formula = y ~ Decid + meth, family = poisson, data = x)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.279  -0.953  -0.740   0.447   4.568
##
## Coefficients:
##             Estimate Std. Error z value             Pr(>|z|)
## (Intercept)  -1.4616     0.0460  -31.80 < 0.0000000000000002 ***
## Decid         2.1450     0.0573   37.41 < 0.0000000000000002 ***
## methB         0.1669     0.0423    3.95     0.00007896414416 ***
## methC         0.3027     0.0421    7.19     0.00000000000064 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 6983.3  on 4568  degrees of freedom
## Residual deviance: 5443.8  on 4565  degrees of freedom
## AIC: 10153
```

```
##
## Number of Fisher Scoring iterations: 6
```

```r
boxplot(fitted(mm) ~ meth, x)
```




```r
exp(coef(mm))
```

```
## (Intercept)       Decid       methB       methC
##      0.2319      8.5421      1.1816      1.3535
```

```r
cumsum(colMeans(y))
```

```
##  0-3min  3-5min 5-10min
##  0.6737  0.7671  0.8831
```

```r
mean(y[,1]) * c(1, exp(coef(mm))[3:4])
```

```
##         methB  methC
## 0.6737 0.7960 0.9118
```

It is all relative, depends on reference methodology/protocol.

3.13 Counting radius effects

Use area subsets to demonstrate use of offsets

```
spp <- "OVEN" # which species

ydis <- Xtab(~ SiteID + Dis + SpeciesID , josm$counts[josm$counts$DetectType1

y <- as.matrix(ydis[[spp]])
head(y)
```

```
##         0-50m 50-100m 100+m
## CL10102     1       2     0
## CL10106     0       0     0
## CL10108     0       0     0
## CL10109     1       2     0
## CL10111     1       0     1
## CL10112     0       2     0
```

```
colMeans(y)
```

```
##   0-50m 50-100m   100+m
## 0.29241 0.49223 0.09849
```

```
cumsum(colMeans(y))
```

```
##   0-50m 50-100m   100+m
##  0.2924  0.7846  0.8831
```

```
x <- data.frame(
  josm$surveys,
  y50=y[,"0-50m"],
  y100=y[,"0-50m"]+y[,"50-100m"])

table(x$y50)
```

```
##
##    0    1    2    3    4    5
## 3521  792  228   25    2    1
```

```r
table(x$y100)
```

```
##
##    0    1    2    3    4    5    6
## 2654  833  647  316   92   20    7
```

```r
m50 <- glm(y50 ~ Decid, data=x, family=poisson)
m100 <- glm(y100 ~ Decid, data=x, family=poisson)
mean(fitted(m50))
```

```
## [1] 0.2924
```

```r
mean(fitted(m100))
```

```
## [1] 0.7846
```

```r
coef(m50)
```

```
## (Intercept)       Decid
##      -2.265       2.126
```

```r
coef(m100)
```

```
## (Intercept)       Decid
##      -1.327       2.209
```

3.14 Offsets

```r
m50 <- glm(y50 ~ Decid, data=x, family=poisson,
  offset=rep(log(0.5^2*pi), nrow(x)))
m100 <- glm(y100 ~ Decid, data=x, family=poisson,
  offset=rep(log(1^2*pi), nrow(x)))
coef(m50)
```

```
## (Intercept)       Decid
##      -2.024       2.126
```

```r
coef(m100)
```

```
## (Intercept)       Decid
##      -2.471       2.209
```

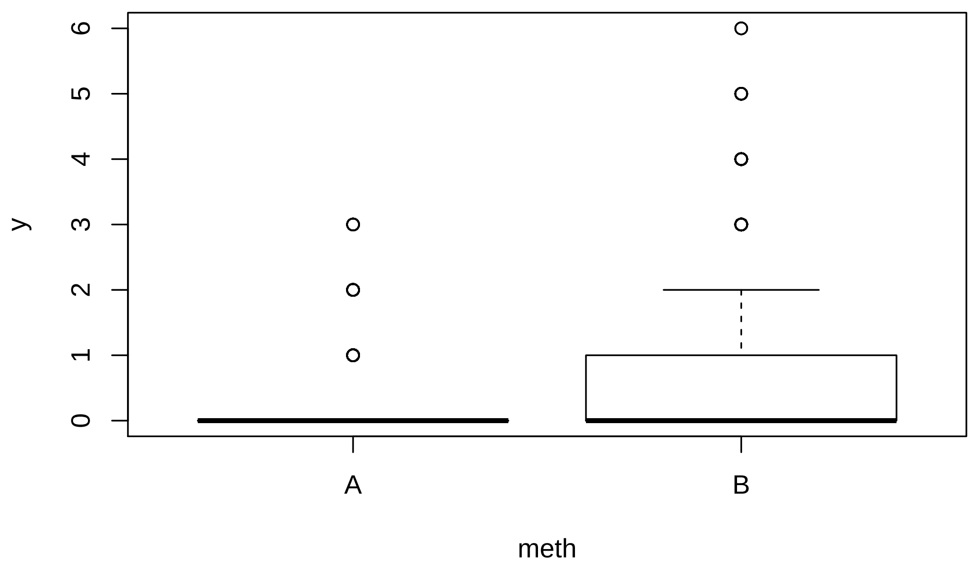
```
mean(exp(model.matrix(m50) %*% coef(m50)))
```

```
## [1] 0.3723
```

```
mean(exp(model.matrix(m100) %*% coef(m100)))
```

```
## [1] 0.2498
```

```
set.seed(1)
x$meth <- sample(c("A", "B"), nrow(x), replace=TRUE)
x$y <- x$y50
x$y[x$meth == "B"] <- x$y100[x$meth == "B"]
boxplot(y ~ meth, x)
```




```
mm <- glm(y ~ meth - 1, data=x, family=poisson)
summary(mm)
```

```
##
## Call:
## glm(formula = y ~ meth - 1, family = poisson, data = x)
```

```
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.261  -1.261  -0.759   0.221   3.720
##
## Coefficients:
##       Estimate Std. Error z value            Pr(>|z|)
## methA  -1.2444     0.0392  -31.77 <0.0000000000000002 ***
## methB  -0.2290     0.0234   -9.81 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 7387.0  on 4569  degrees of freedom
## Residual deviance: 5683.8  on 4567  degrees of freedom
## AIC: 9171
##
## Number of Fisher Scoring iterations: 6
```

```
exp(coef(mm))
```

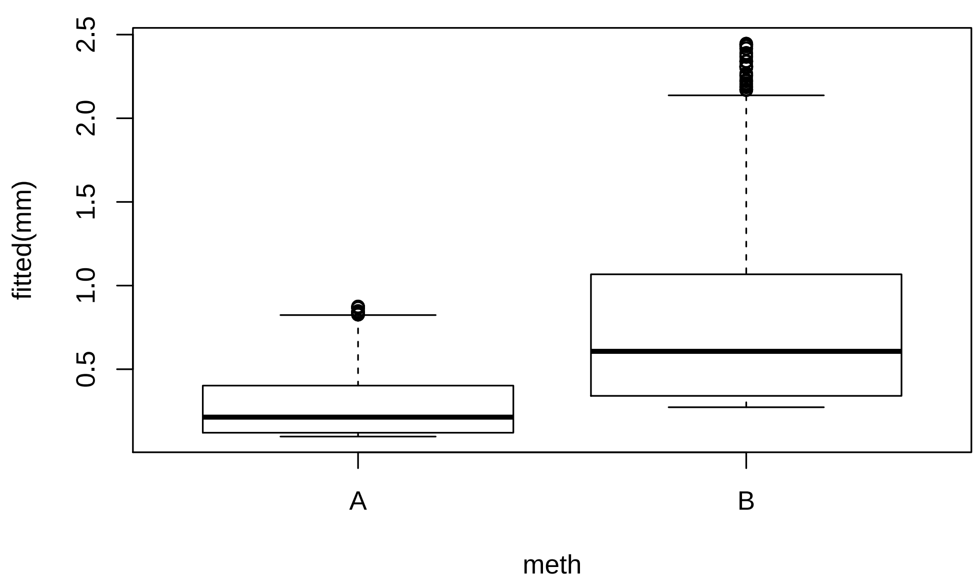
```
##  methA  methB
## 0.2881 0.7953
```

```
mm <- glm(y ~ Decid + meth, data=x, family=poisson)
summary(mm)
```

```
##
## Call:
## glm(formula = y ~ Decid + meth, family = poisson, data = x)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.209  -0.855  -0.589   0.244   3.577
##
## Coefficients:
##             Estimate Std. Error z value            Pr(>|z|)
## (Intercept)  -2.3275     0.0567   -41.0 <0.0000000000000002 ***
```

```
## Decid          2.1961     0.0689    31.9 <0.0000000000000002 ***
## methB          1.0284     0.0456    22.6 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 6247.1  on 4568  degrees of freedom
## Residual deviance: 4595.6  on 4566  degrees of freedom
## AIC: 8085
##
## Number of Fisher Scoring iterations: 6
```

```
boxplot(fitted(mm) ~ meth, x)
```




```
exp(coef(mm))
```

```
## (Intercept)       Decid       methB
##     0.09754     8.98996     2.79646
```

```r
cumsum(colMeans(y))[1:2]
```

```
##   0-50m 50-100m
##  0.2924  0.7846
```

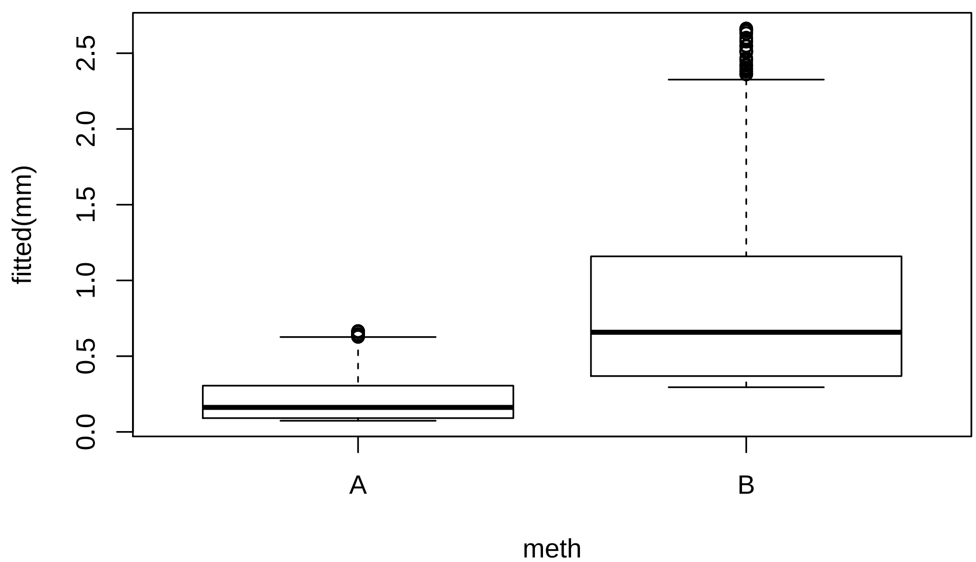
```r
mean(y[,1]) * c(1, exp(coef(mm))[3])
```

```
##         methB
## 0.2924 0.8177
```

```r
mm <- glm(y ~ Decid, data=x, family=poisson,
  offset=log(ifelse(x$meth == "A", 0.5, 1)^2*pi))
summary(mm)
```

```
##
## Call:
## glm(formula = y ~ Decid, family = poisson, data = x, offset = log(ifelse(x$meth ==
##     "A", 0.5, 1)^2 * pi))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.305  -0.842  -0.525   0.234   3.600
##
## Coefficients:
##             Estimate Std. Error z value            Pr(>|z|)
## (Intercept)  -2.3658     0.0453   -52.2 <0.0000000000000002 ***
## Decid         2.2031     0.0690    31.9 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 5746.0  on 4568  degrees of freedom
## Residual deviance: 4653.6  on 4567  degrees of freedom
## AIC: 8141
##
## Number of Fisher Scoring iterations: 6
```

```
boxplot(fitted(mm) ~ meth, x)
```




```
cumsum(colMeans(y))[1:2]
```

```
##   0-50m 50-100m
##  0.2924  0.7846
```

```
c(0.5, 1)^2*pi * mean(exp(model.matrix(mm) %*% coef(mm))) # /ha
```

```
## [1] 0.2173 0.8691
```

3.15 Definitions

Discuss definitions of:

- relative abundance,
- abundance,
- occupancy,
- density.

3.16 Binomial model and censoring

Try cloglog with a rare species, like BOCH

```
#spp <- "OVEN" # which species
spp <- "BOCH" # which species
#spp <- "CAWA" # which species

x <- data.frame(
  josm$surveys,
  y=as.numeric(ytot[rownames(x), spp]))
x$y01 <- ifelse(x$y > 0, 1, 0)

table(x$y)
```

```
##
##    0    1    2    3
## 4346  180   38    5
```

```
mP <- glm(y ~ Decid * ConifWet, x, family=poisson)
mBc <- glm(y01 ~ Decid * ConifWet, x, family=binomial("cloglog"))
mBl <- glm(y01 ~ Decid * ConifWet, x, family=binomial("logit"))

coef(mP)
```

```
##    (Intercept)          Decid       ConifWet Decid:ConifWet
##        -2.6836        -1.0105         0.2928         1.6797
```

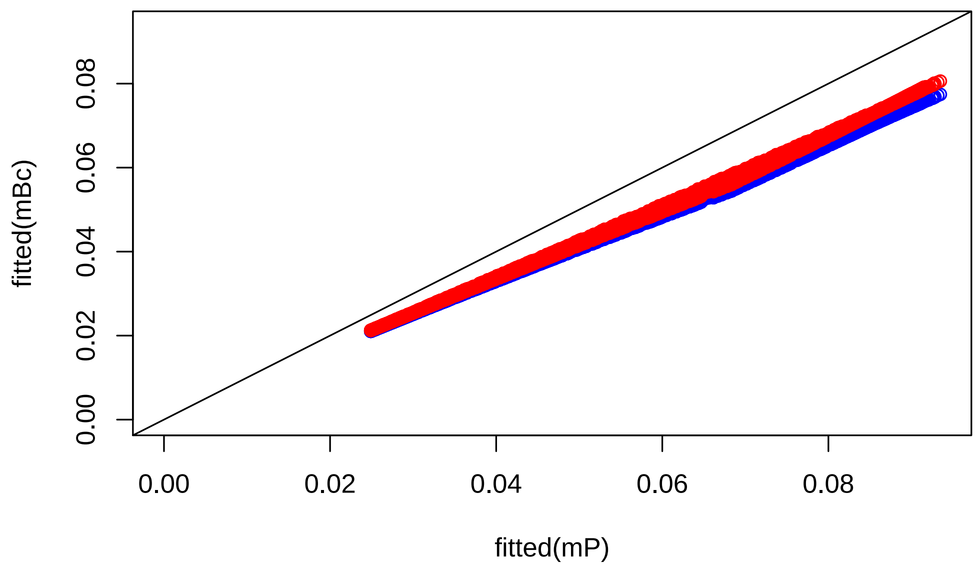
```
coef(mBc)
```

```
##    (Intercept)          Decid       ConifWet Decid:ConifWet
##        -2.8824        -0.9698         0.3459         1.6638
```

```
coef(mBl)
```

```
##    (Intercept)          Decid       ConifWet Decid:ConifWet
##        -2.8550        -0.9889         0.3592         1.6844
```

```
plot(fitted(mBc) ~ fitted(mP), col=4,
  ylim=c(0, max(fitted(mP))), xlim=c(0, max(fitted(mP))))
points(exp(model.matrix(mBc) %*% coef(mBc)) ~ fitted(mP), col=2)
abline(0,1)
```


0.00
0.02
0.04
0.06
0.08
fitted(mBc)
0.00
0.02
0.04
0.06
0.08
fitted(mP)

Chapter 4

Behavioral Complexities

Behaviour related stuff

constant p (time as covariate)

time varying p

finite mix

time varying p/c

rate, count, time-to-event

Chapter 5

The Detection Process

EDR, tau constant

truncated, unlimited

variable tau: habitat effect (continuous case?)

discrete: land cover, observer effects

contrast fixed effects with offsets – motivation for ARU

Chapter 6

Dealing with Recordings

integration challenges

calibration (exponential/cloglog approximation)

fixed effects

paired

sensor sensitivity - EDR

Chapter 7

A Closer Look at Assumptions

break thos assumptions

Chapter 8

Understanding Roadside Surveys

directional diff in signal transmission

Chapter 9

Miscellaneous Topics

model selection and conditional likelihood

variance/bias trade off

error propagation

MCMC?

N-mixture ideas

phylogenetic and life history/trait stuff

PIF methods

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2019). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.11.