Appendix 4

April 25, 2014

1 Introduction

This document is designed to help you replicate the analysis presented in Isaac et al., 2014. If you are only interested in replicating the analysis as presented in our paper then sections 2 and 3 should satisfy your needs, however if you are interested in learning more about the code we used to produce our results, sections 4 onward go into a little more detail.

2 Setup

Getting setup is easy if you are already a regular R user but a bit more complicated if you are not. Not to worry, we are here to hold your hand.

1. You will need to install two pieces of software to run the full analysis. The first is R, you will need to download this from `http://cran.r-project.org/`.

2. The second is JAGS, which is used to run occupancy models. You can download this from `http://sourceforge.net/projects/mcmc-jags/files/JAGS/3.x/`

3. To download all the code for our analyses, and to have a go yourself use this link: `https://github.com/BiologicalRecordsCentre/RangeChangeSims/archive/master.zip`. This will download a .zip file (10.9MB) containing everything you need.

4. Extract the .zip file to your machine. Now we are ready to go!

3 Replicating the paper

To replicate the analysis as in Isaac et al. 2014, find the script 'Run_full_analysis.r' from the .zip file and run it in R. The script will install required packages, check that JAGS is working, run the analyses (you can specify the number of runs to do) and plot all figures and tables from the results. The results will be saved in the working directory. However, this is a computationally intensive task and to replicate the full 1000 runs presented in the paper on a regular desktop PC will take approximately 170 days! We addressed this problem by using a computer cluster, achieving a run time of under a week. The only function that cannot be run in parallel on a cluster is frescalo as this uses a windows complied .exe.

4 Analysis: Step by Step

Here we demonstrate a few of the functions we have written that will give you a deeper understanding of our methods and provide the potential to test out your own methods using our framework.

To access all the functions we simply source in the function script from the R console by typing the following:

```
source("Sim_functions.r")

## Loading required package:  reshape2
## Loading required package:  lme4
## Loading required package:  Matrix
## Loading required package:  Rcpp
```

You will need to ensure that 'Sim_functions.r' reflects the file path from your current working directory to the functions script. You can change your working directory using the `setwd` function

We will look at the following functions:

`create_data`
: Randomly generates 'true' data. This is a binary table indicating the presence/absence of each species at each site.

`recording_visit`
: Send out our virtual recorders on a visit and return what they find

`recording_cycle`
: Send out our virtual recorders on a year's worth of sampling to all sites

`generate_records`
: Undertake virtual recording across all years in our simulated study

`generate_all_scenarios`
: Undertake virtual recording, producing results for each recording scenario

`run_all_methods`
: Take a dataset of records and apply to this a suite of modelling methods

`iterate_all_scenarios`
: Combine the previous two functions and allow multiple runs, each run new data generated by `generate_all_scenarios`

`get_all_stats`
: Produce summary information from the results of `iterate_all_scenarios`

4.1 Generating simulated data

Simulated data can be created using the 'true_data' function. This function generates 'true' data telling us whether each species is present or absent from each location which we can later use to build our observation data (see section 'Species occurrence matrices' in the methods section of the manuscript).

```
# Create our 'true' data
true_data <- create_data(nSites = 1000, nSpecies = 25, pFocal = list(Occ = 0.5,
    DetP = 0.5))

head(true_data)

##       focal spp1 spp2 spp3 spp4 spp5 spp6 spp7 spp8 spp9 spp10 spp11 spp12
## site1     1    0    0    1    1    0    0    0    0    1     0     0     0
## site2     1    1    1    1    1    1    1    1    0    1     0     0     0
## site3     1    0    1    1    1    1    1    1    1    1     1     0     0
## site4     1    0    0    1    1    1    1    0    0    1     0     1     1
## site5     0    1    1    1    1    1    1    0    1    1     1     1     1
## site6      1    1    0    1    1    0    1    0    0    1     1     0     0
##        spp13 spp14 spp15 spp16 spp17 spp18 spp19 spp20 spp21 spp22 spp23
## site1     0     0     1     1     1     1     0     0     0     0     0
## site2     1     0     1     1     1     0     1     1     0     0     0
## site3     1     1     1     0     0     1     1     0     0     1     1
## site4     1     0     1     1     1     1     0     0     0     1     1
## site5     1     1     1     1     0     1     0     1     0     1     0
## site6     1     0     0     1     0     1     1     0     0     0     1
##        spp24 spp25
## site1     0     0
## site2     0     0
## site3     1     0
## site4     0     0
## site5     1     0
## site6     1     0

# true_data has some useful attributes
str(attributes(true_data))

## List of 4
##  $ dim     : int [1:2] 1000 26
##  $ dimnames:List of 2
##   ..$ : chr [1:1000] "site1" "site2" "site3" "site4" ...
##   ..$ : chr [1:26] "focal" "spp1" "spp2" "spp3" ...
##  $ richness: num [1:1000] 8 15 18 15 19 13 19 12 17 12 ...
##  $ p_detect: num [1:26] 0.5 0.866 0.838 0.806 0.773 ...
```

`nSites` specifies the number of sites we want, `nSpecies` specifies the number of species, and `pFocal` specifies the occurrence probability of the species (`Occ`), and the probability of detection of the focal species on a given visit (`DetP`). There are some attributes assigned to `true_data` that can be quite informative. The two elements of `dimnames` give the names of sites and species respectively, `richness` gives the species richness of each site as listed in `dimnames`), and `p_detect` gives the probability of detection for each species as listed in `dimnames`.

4.2 Nested functions: Visits within years within year ranges

The following examples detail the generation of the control scenario data described in the 'Control scenario' section in the methods section of the manuscript. At the lowest level of our models is the concept of a visit. Here we take the 'true' data that we just generated and send out our virtual recorders to a site. The function takes the true occurrence of the species at the site as well as the sampling intensity for each species and returns what was recorded by our virtual recorders.

```
# Get the true data for one site
site1 <- true_data["site1", ]

# Send out our virtual recorders
observations <- recording_visit(spp_vector = site1, p_obs = attr(true_data,
    "p_detect"))

# View the species actually present
names(site1[site1 == 1])

## [1] "focal" "spp3"  "spp4"  "spp9"  "spp15" "spp16" "spp17" "spp18"
```

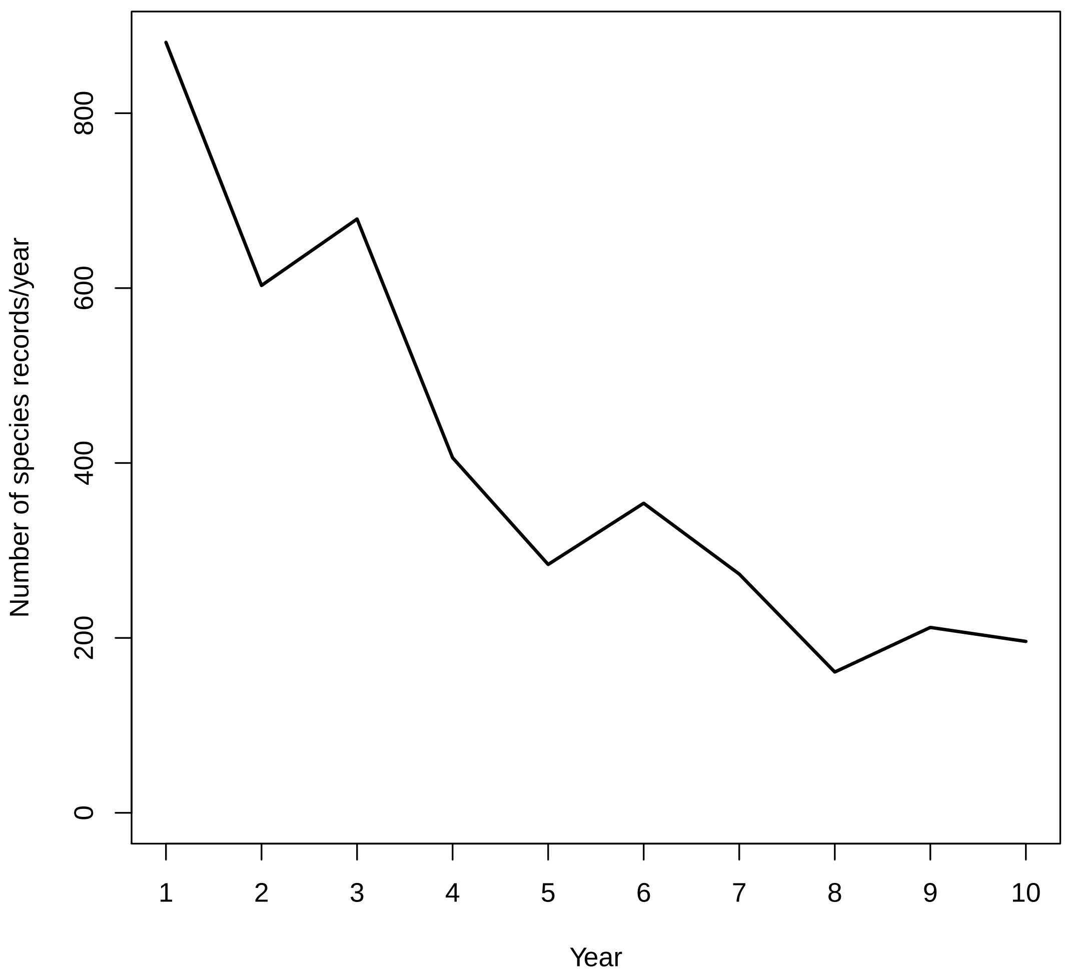
```
# View the species recorded by our observers
names(observations[observations == 1])

## [1] "focal" "spp3"  "spp4"  "spp16" "spp18"
```

`recording_visit` is wrapped up in the function `recording_cycle` which allows us to undertake a years sampling in one go. The resulting data.frame gives us a row for each observation with columns for species, visit and site.

```
# Undertake a year's sampling setting the recording intensity to 7% (medium)
year1 <- recording_cycle(pSVS = 0.07, true_data = true_data)

# Preview our results
head(year1)

##    Species Visit Site
## 1    focal     1    1
## 4     spp3     1    1
## 5     spp4     1    1
## 17   spp16     1    1
## 28    spp1    11   10
## 30    spp3    11   10
```

In `recording_cycle`, `pSVS` specifies the overall recording intensity as described in the 'control scenario' section of the methods section of the manuscript. We further wrap `recording_cycle` within the function `generate_records` which allows us to create our baseline unbiased records for all years of the study

```
# Set up our parameters
nYrs = 10  # The number of years we wish to run
which.decline = 1:20  # Specify which species we wish to be in decline
decline = rep(0.9, 20)  # The rate of decline of each speces (here all 90%)
pSVS = 0.05  # Set our recording effort to 7% (medium)
mv = 10  # Set the maximum number of visits to a site in a year to 10

# Generate our records
years10 <- generate_records(nYrs = nYrs, true_data = true_data,
    decline = decline, pSVS = pSVS, mv = mv, which.decline = which.decline)

# Preview our results
head(years10)

##   Species Visit Site Year
## 1    spp1    14  102    1
## 2    spp4    14  102    1
## 3    spp5    14  102    1
## 4    spp6    14  102    1
## 5    spp8    14  102    1
## 6    spp9    14  102    1

# Look for our simulated decline
plot(table(years10$Year), xlab = "Year", ylab = "Number of species records/year",
    type = "l")
```



The data is returned in the same format as in `recording_cycle` but now contains data across all the years of our virtual study

4.3 Applying sampling scenarios

To create our simulated recording data we use the true data and a number of sampling protocols described in our paper. Each scenario uses the `generate_records` function (described above) to generate data and then subsamples this data according to the specifics of the scenario (see 'Biased recording scenarios' in the methods section of the manuscript).

For ease of coding each scenario was given a letter and the look up table is presented in Table 1

Table 1: Scenario codes

Code	Short name	Long name
A	A_EvenRcrdng	Control
B	B2_IncrnVisit	MoreVisits
B	B3f_IncrnVBiasFc	MoreVisits+Bias
C	C1_pShortLEven	C1_pShortLEven
C	C2_pShortLIncr	LessEffortPerVisit
D	D2_SelectvIncr	MoreDetectable
F	F_NfDecline	NonFocalDeclines

The task of generating the true data and from this generating records for each simulated recording scenario is handle by `generate_all_scenarios`.

```
# Produce the records for each recording scenario
recs <- generate_all_scenarios(nSites = 1000, nSpecies = 25,
    pFocal = list(Occ = 0.5, DetP = 0.5), nYrs = 10, pSVS = 0.07,
    mv = 10, Scenarios = "BCDF", p_short = list(init = 0.6, final = 0.9),
    pDetMod = 0.2, decline = 0)

# The output is a list of data.frames, one for each Scenario
length(recs)

## [1] 7

# A preview of the results from the MoreVisits+Bias scenario
head(recs$B3f_IncrnVBiasF, 10)

##    Species Visit Site Year
## 1     spp3     1    1   10
## 2     spp4     1    1   10
## 3     spp9     1    1   10
## 4    spp20     1    1   10
## 5    spp23     1    1   10
## 6     spp3     1    1    9
## 7    spp20     1    1    9
## 8    spp25     1    1    9
## 14    spp2    32  100    4
## 15    spp3    32  100    4
```

We have covered some of these parameters in previous sections (`nSites`, `nSpecies`, `pFocal`, `nYrs`, `pSVS`, `mv`), here is a brief description of those that are new. `Scenarios` is a string that specifies the scenarios that are to be generated as given in the table above (scenario A is run by default). `p_short` is used for both 'C' scenarios and gives the proportion of lists that should be 'short' on the first (`init`) and last (`final`) year of the simulation. `pDetMod` defines the proportion of lists where the focal species should be 'unrecorded' and is used to simulate reduced detectability in scenario 'D'. `decline` gives the proportional decline of the focal species across the duration of the study period (here 10 years). We set this to 0.3 when testing the power of statistical methods.

`generate_all_scenarios` returns a list of data.frames. Each data.frame is the result of a recording scenario, giving a list of observations. This details the species recorded, the year, visit and site on which it was recorded. Note that multiple visits can be made to a site in a given year.

Table 2: Model codes

Code	Name
nSites	Naive
Telfer	Telfer
Frescalo1tp	Frescalo_Y
Frescalo2tp	Frescalo_P
VRsimple	ReportingRate
LLsimple	RR+LL
RR_SS	RR+SF
MMbin0	RR+Site
LLmm	RR+LL+Site
mmSS	RR+SF+Site
LLSS	RR+SF+LL
LLmmSS	RR+SF+LL+Site
Occ+Simple	OccDetSimple
Occ+LL+Site	Occ+LL+Site
Occ+SS+Site	Occ+SF+Site
Occ+SS+LL	OD+SF+LL
RR_SS3	RR+SF3
LLmmSS3	RR+LL+SF3+Site
Occ+SS3+Site	Occ+SF3+Site
Occ+SS3+LL	Occ+LL+SF3
Occ+Full	OD+SF+LL+Site
MMbin2sp	WSS_2
MMbin4sp	WSS_4
Maes	RDC

4.4 Running analyses

Once we have generated our records we can them test for trends using the various methods explored in the paper. When undertaking the analyses we explored a range of methods but have chosen not to describe all of them in the manuscript. Table 2 gives the codes used for each of the methods discussed in the manuscript and some extras. The details of the methods included in the manuscript can be found in Appendix S2.

Here we use the function `run_all_methods` to run a range of analytical methods on one of the recording scenarios we have simulated. While this function runs most of the methods by default a couple of methods require us to provide extra details.

```
# Lets analyse the control scenario
scenario1 <- recs$A_EvenRcrdng

# Specify the occupancy models we want to run These model
# specify OccDetSimple and OD+SF+LL+Site respectivly
Occ <- c("Simple", "Full")

# Run the analyses
results <- run_all_methods(records = scenario1, nyr = 2, OccMods = Occ,
    Frescalo = c(1, 2))

##   Site Species Year
## 1  101    spp5    1
## 2  101   spp11    1
## 3  101   spp12    1
## 4  104    spp8    1
## 5  104   spp14    1
## 6  114   focal    1
##   Site Species Year
## 1  101    spp5    1
## 2  101   spp11    1
## 3  101   spp12    1
## 4  104    spp8    1
## 5  104   spp14    1
## 6  114   focal    1
```

```
## Loading required package:  R2jags
## Loading required package:  rjags
## Loading required package:  coda
## Loading required package:  lattice
## Linked to JAGS 3.4.0
## Loaded modules:  basemod,bugs
##
## Attaching package:  'R2jags'
##
## The following object is masked from 'package:coda':
##
##    traceplot
##
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 21041
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 20751
##
## Initializing model
```

```
# Preview the results
head(results)
```

```
##  prop.diff        plr     Telfer      plr_p   Telfer_p Maes_trend
##    0.07285    1.09379    1.25488    0.27405    0.20952   -0.06679
```

```
# List the names of all the parameters returned
names(results)
```

```
##  [1] "prop.diff"            "plr"                  "Telfer"
##  [4] "plr_p"                "Telfer_p"             "Maes_trend"
##  [7] "Maes_p"               "Maes5_trend"          "Maes5_p"
## [10] "LLsimple_trend"       "LLsimple_p"           "LLfs_trend"
## [13] "LLfs_p"               "LLmmSS_trend"         "LLmmSS_p"
## [16] "LLSS_trend"           "LLSS_p"               "mmSS_trend"
## [19] "mmSS_p"               "VRsimple_trend"       "VRsimple_p"
## [22] "VRfs_trend"           "VRfs_p"               "RR_SS_trend"
## [25] "RR_SS_p"              "MMbin0_trend"         "MMbin0_p"
## [28] "MMbin0_pCombosUsed"   "MMber2sp_trend"       "MMber2sp_p"
## [31] "MMber2sp_pCombosUsed" "MMbin2sp_trend"       "MMbin2sp_p"
## [34] "MMbin2sp_pCombosUsed" "Frescalo1tp_trend"    "Frescalo1tp_p"
## [37] "Frescalo2tp_trend"    "Frescalo2tp_p"        "Occ+Simple_trend"
## [40] "Occ+Full_trend"       "Occ+Simple_p"         "Occ+Full_p"
## [43] "Occ+Simple_Rhat"      "Occ+Full_Rhat"        "nRecords_trend"
## [46] "nRecords_p"           "nSites_trend"         "nSites_p"
## [49] "Fr_Phi"               "Fr_MedianAlpha"       "Recs_pBnchmk"
## [52] "Recs_qFocal"          "Recs_tot_visits"      "Recs_Sp_Yr"
## [55] "Recs_total"           "Recs_focal"           "VisPerYr_t1"
## [58] "VisPerYr_t2"          "focal_t1"             "focal_t2"
## [61] "nonfocal_t1"          "nonfocal_t2"          "meanL_t1"
## [64] "meanL_t2"             "p_shortLists_t1"      "p_shortLists_t2"
## [67] "MeanSitesVisited_t1"  "MeanSitesVisited_t2"  "PrSitesMultiVisit_t1"
## [70] "PrSitesMultiVisit_t2"
```

`nyr` dictates that only sites which received visits in at least `nyr` of the ten years in the simulation are included in site-filtered models (+SF). `OccMods` names the occupancy models that we would like to use. `Frescalo` specifies the versions of frescalo we want to run, 1 meaning Frescalo_Y and 2 meaning Frescalo_P. The returned object 'results' is a vector of attributes. There are numerous parameters returned and Table 3 gives details for those of note.

Table 3: run_all_methods output parameters

Parameter name	**Explanation**
*_trend	Specifies the trend estimated for the focal species
*_p	Indicates the significance of the estimated trend
*_pCombosUsed	Specifies, for models that used only data from 'well sampled sites', the proportion of visits that were used in the model
*_Rhat	A test of convergence in MCMC chains in occupancy models, values around 1 indicate convergence
*_t1	Time period 1: years 1-5
*_t2	Time period 2: years 6-10
Fr_Phi	In Frescalo, the target frequency of frequency-weighted mean frequency (see Hill, 2011)
Fr_MedianAlpha	The median value of alpha, an estimate of recording intensity, across all sites (see Hill, 2011)
Recs_pBnchmk	The proportion of records that are of 'benchmark' species (see Hill, 2011)
Recs_qFocal	The quantile of the focal species when ranked against all other species be richness
Recs_tot_visits	The total number of visits in the simulation
Recs_Sp_Yr	Average number of records per species year
Recs_total	The total number of records in the simulation
Recs_focal	The the number of records of the focal species in the simulation
VisPerYr	The average number of visits per year
focal	The proportion of visits on which the focal species was recorded
nonfocal	The proportion for visits on which a species was recorded, averaged across all non-focal species
meanL	The mean list length across visits
p_shortLists	The proportion of visits that returned short lists (3 species or less)
MeanSitesVisited	The mean number of sites visited per year
PrSitesMultiVisit	The proportion of sites that once visited once receive another visit in the same year

When running our full analysis we wanted to run numerous simulations for testing. We therefore wanted to run `generate_all_scenarios` many times and use `run_all_methods` on each scenario of each run. We wrote a function called `iterate_all_scenarios` that combines the former methods and allows us to do multiple runs. In the example below I turn off occupancy models (`Occ = NULL`) and frescalo (`Frescalo = FALSE`), and consider fewer scenarios to reduce the amount of time it takes to run.

```
# Run multiple iterations, here set to 2
output <- iterate_all_scenarios(nreps = 2, nSpecies = 25, nSites = 500,
    nYrs = 10, pSVS = 0.07, Scenarios = "B", pFocal = list(Occ = 0.5,
        DetP = 0.5), decline = 0, Frescalo = FALSE, Occ = NULL,
    nyr = 2, writePath = "Output")

# Examine dimensions
dim(output)

## [1] 52  3  2

dimnames(output)

## [[1]]
##  [1] "prop.diff"            "plr"                  "Telfer"
##  [4] "plr_p"                "Telfer_p"             "Maes_trend"
##  [7] "Maes_p"               "Maes5_trend"          "Maes5_p"
## [10] "LLsimple_trend"       "LLsimple_p"           "LLfs_trend"
## [13] "LLfs_p"               "LLmmSS_trend"         "LLmmSS_p"
## [16] "LLSS_trend"           "LLSS_p"               "mmSS_trend"
## [19] "mmSS_p"               "VRsimple_trend"       "VRsimple_p"
## [22] "VRfs_trend"           "VRfs_p"               "RR_SS_trend"
## [25] "RR_SS_p"              "MMbin0_trend"         "MMbin0_p"
## [28] "MMbin0_pCombosUsed"   "nRecords_trend"       "nRecords_p"
## [31] "nSites_trend"         "nSites_p"             "Recs_pBnchmk"
## [34] "Recs_qFocal"          "Recs_tot_visits"      "Recs_Sp_Yr"
## [37] "Recs_total"           "Recs_focal"           "VisPerYr_t1"
## [40] "VisPerYr_t2"          "focal_t1"             "focal_t2"
## [43] "nonfocal_t1"          "nonfocal_t2"          "meanL_t1"
## [46] "meanL_t2"             "p_shortLists_t1"      "p_shortLists_t2"
## [49] "MeanSitesVisited_t1"  "MeanSitesVisited_t2"  "PrSitesMultiVisit_t1"
## [52] "PrSitesMultiVisit_t2"
##
## [[2]]
## [1] "A_EvenRcrdng"     "B2_IncrnVisit"    "B3f_IncrnVBiasFc"
##
## [[3]]
## NULL
```

`iterate_all_scenarios` returns a three dimensional object. The first dimension gives the parameters as returned by `run_all_methods` and detailed in the table above. The second dimension represents each of the recording scenarios selected and the third represents each of the repeats as specified by `nreps`. This function also takes an argument `writePath` which is were the data generated will be saved.

4.5 Error rates

To examine changes in error rate and power as we have done in our manuscript we must expand on the code in the previous section. Here we analyse two levels of recording effort, and with the focal species both stable and in decline. The former allows us to test the effects of recording effort and the latter allows us to explore model validity and power. In addition we use the function `get_all_stats` to get meaningful results from the data generated.

```
# Create a parameter to hold our results
SimOut <- NULL

# Loop through our parameter settings
for (decline in c(0, 0.3)) {
    for (pSVS in c(0.05, 0.07, 0.1)) {

        # Create an id which captures information about the run V for
        # Validation, P for power (depends if the focal species is in
        # decline or not)
        ch <- ifelse(decline == 0, "V", "P")
        # Add to this details of the pSVS setting and the date
        id = paste(ch, "_", pSVS * 100, "SVS_", format(Sys.Date(),
            "%y%m%d"), sep = "")

        # Run multiple iterations 10 takes 1 minute
        output <- iterate_all_scenarios(nreps = 10, nSpecies = 25,
            nSites = 500, nYrs = 10, pSVS = pSVS, Scenarios = "B",
            pFocal = list(Occ = 0.5, DetP = 0.5), decline = decline,
            Frescalo = FALSE, Occ = NULL, nyr = 2, id = id, writePath = "Output")

        # Create meaningful results from the data
        Out <- get_all_stats(output, save_to_txt = TRUE, writePath = "Output")

        # Create attributes to capture the parameters used
        attr(Out, "decline") <- decline
        attr(Out, "pSVS") <- pSVS

        # Add these results to a list
        SimOut <- c(SimOut, list(Out))
    }
}

## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
## [1] "significant correlation opposite to true!"
```

`get_all_stats` creates a series of summary statistics for each scenario, across repeats. During its run it prints to screen on the occassions when, for a declining focal species, the model suggests it is in fact increasing. At the top of our results, against the name of each method we have the proportion of repeats that gave a significant decline for the focal species:

```
# Lets look at one set of results
res <- SimOut[[1]]

# This set of results is for pSVS = 0.07, decline = 0
attr(res, "pSVS")

## [1] 0.05

attr(res, "decline")

## [1] 0

# The first rows give proportion of repeats that gave a
# significant decline for the focal species Each column
# represents a recording scenario
head(res)

##          A_EvenRcrdng B2_IncrnVisit B3f_IncrnVBiasFc
## plr               0.0           0.0              0.0
## Telfer            0.0           0.0              0.1
## Maes              0.0           0.0              0.0
## Maes5             0.0           0.0              0.0
## LLsimple          0.5           0.3              0.7
## LLfs              0.2           0.1              0.2
```

For those parameters discussed in Table 3 the figure in this summary table represents the average:

```
# Summariesd parameters are given.  These include records
# parameters...
res[17:24, ]

##             A_EvenRcrdng B2_IncrnVisit B3f_IncrnVBiasFc
## VisPerYr_t1       86.800        55.700           55.800
## VisPerYr_t2       97.000        86.000           86.200
## focal_t1           0.264         0.240            0.315
## focal_t2           0.253         0.250            0.269
## nonfocal_t1        0.244         0.246            0.245
## nonfocal_t2        0.247         0.247            0.248
## meanL_t1           6.387         6.364            6.484
## meanL_t2           6.414         6.401            6.442
```

```
# ...and trend estimates
res[37:44, ]

##                  A_EvenRcrdng B2_IncrnVisit B3f_IncrnVBiasFc
## Maes_trend             -0.020        -0.007           -0.111
## Maes5_trend             0.014         0.000           -0.149
## LLsimple_trend         -0.044        -0.029           -0.082
## LLfs_trend             -0.015         0.013           -0.023
## LLmmSS_trend           -0.041        -0.024           -0.046
## LLSS_trend             -0.047        -0.026           -0.064
## mmSS_trend             -0.029        -0.038           -0.046
## VRsimple_trend         -0.031        -0.026           -0.066
```

At the end of this table we have information on the number of valid repeats that were done:

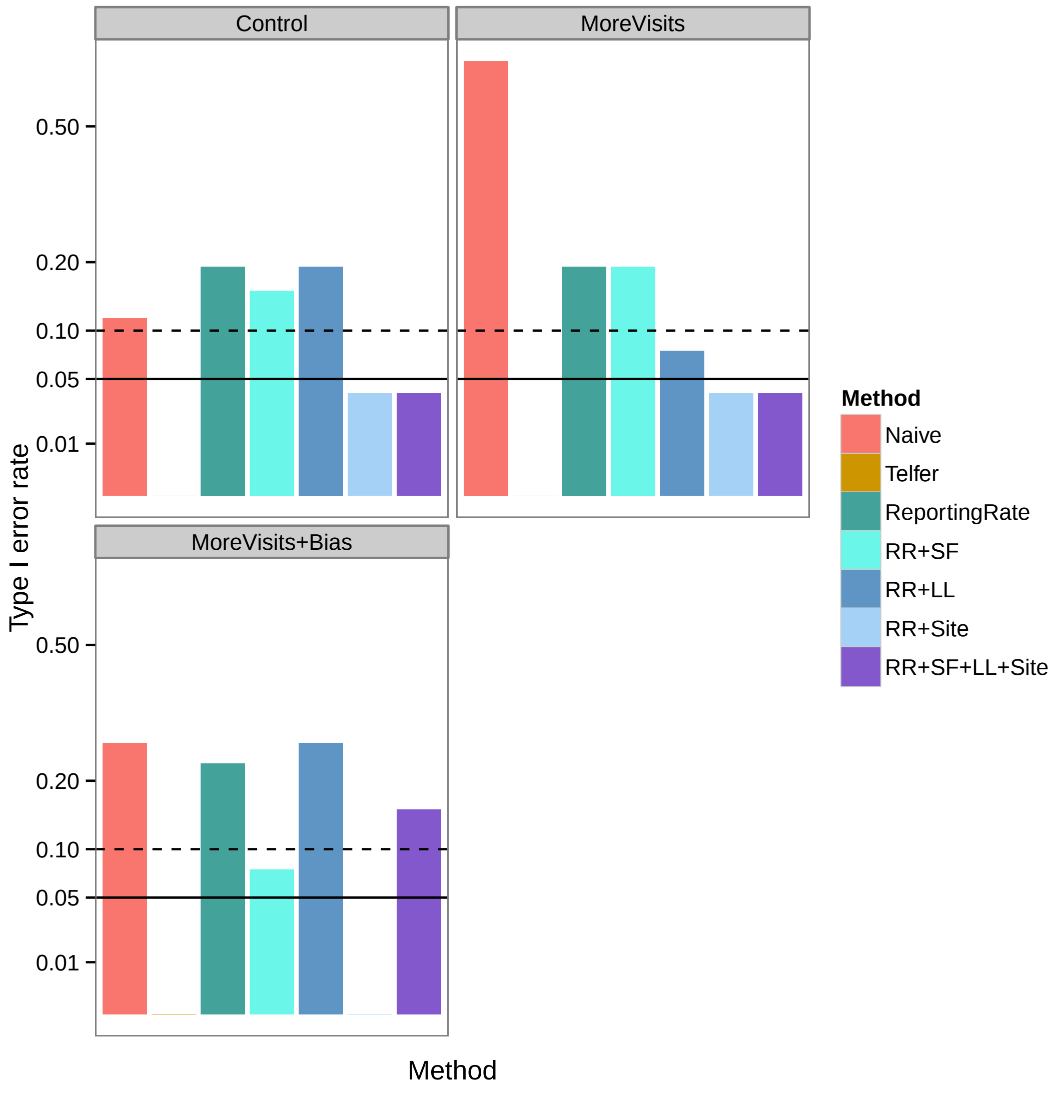
```
# The number of repeats are given for each model
res[52:57, ]

##                  A_EvenRcrdng B2_IncrnVisit B3f_IncrnVBiasFc
## Maes_nReps                 10            10               10
## Maes5_nReps                10            10               10
## LLsimple_nReps             10            10               10
## LLfs_nReps                 10            10               10
## LLmmSS_nReps               10            10               10
## LLSS_nReps                 10            10               10
```

These summary statistics are the basis of the graphs presented in our manuscript

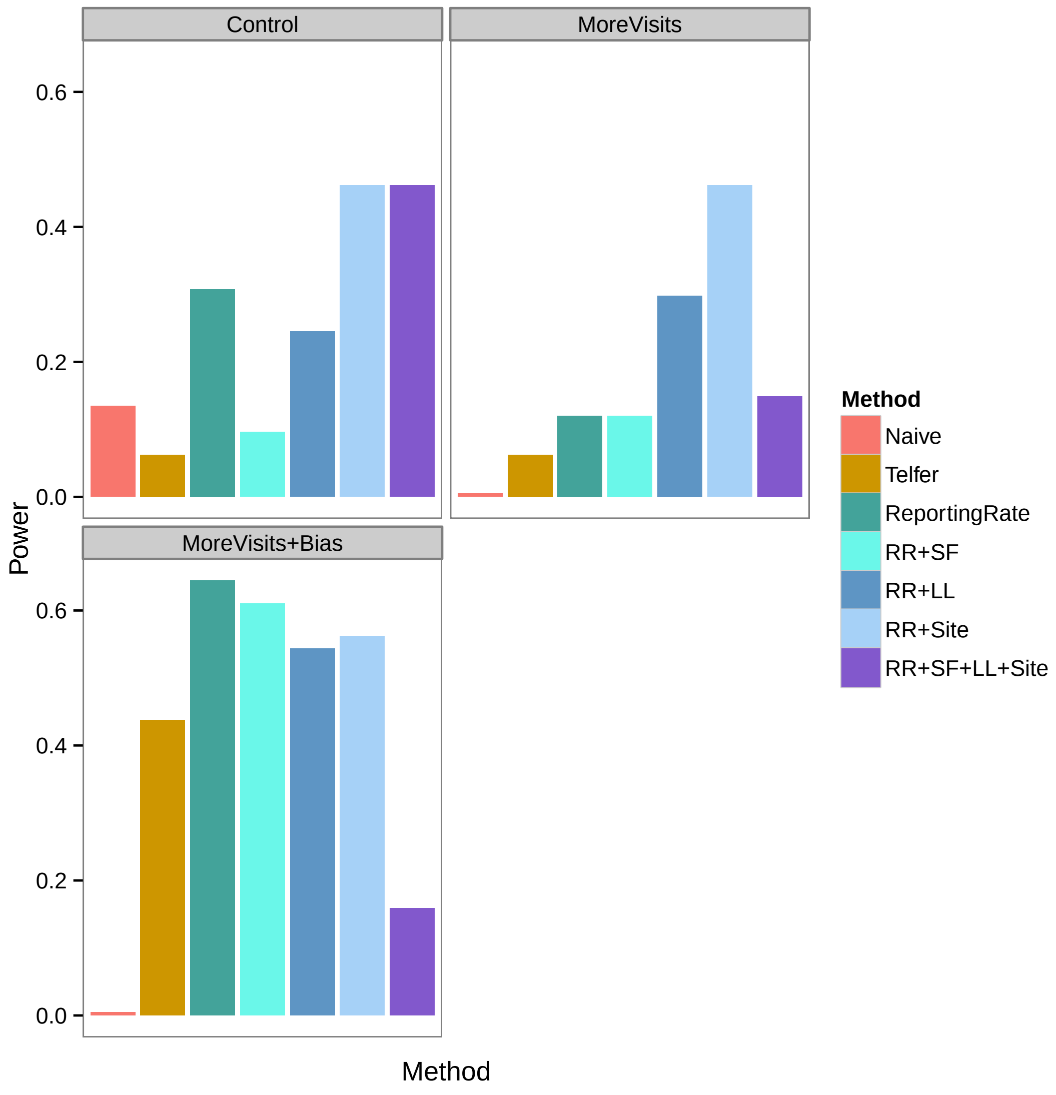
5 Creating graphs

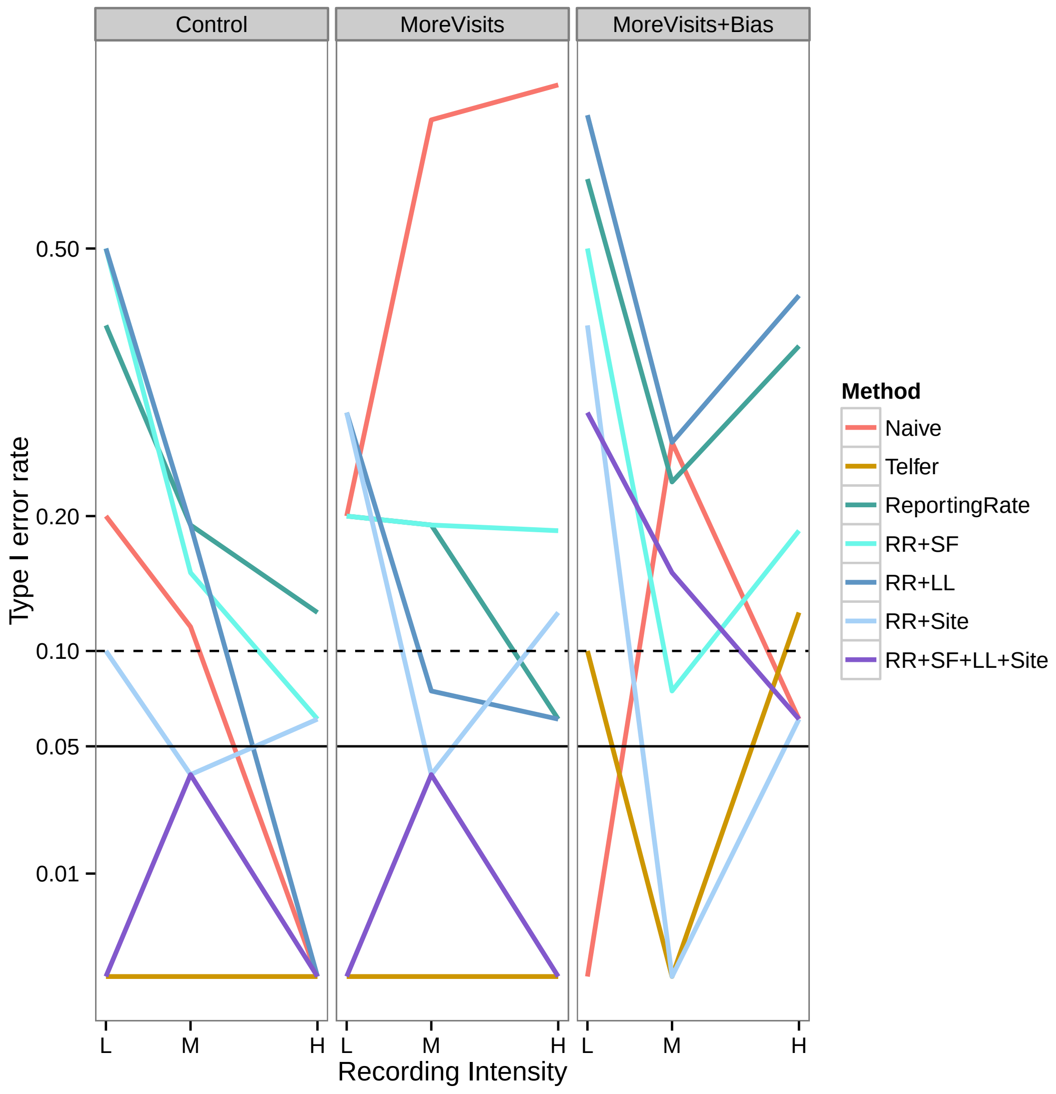
In the manuscript we present a number of figures that summarize our results. These can be replicated using the function `Explore_results`. This function takes two parameters: `writePath` and `readPath`. These specify the file path where you want data to be written to and the path where you are reading your data files in from. In this case we saved our results from the previous section to a folder called 'Output' so we will use that as our `readPath`. By setting `plot = TRUE` we can get the the figures to print to screen as well as being saved to file.

```
# Createmanuscript tables and figures from our results
Explore_results(readPath = "Output", writePath = "Results", plot = TRUE)
```

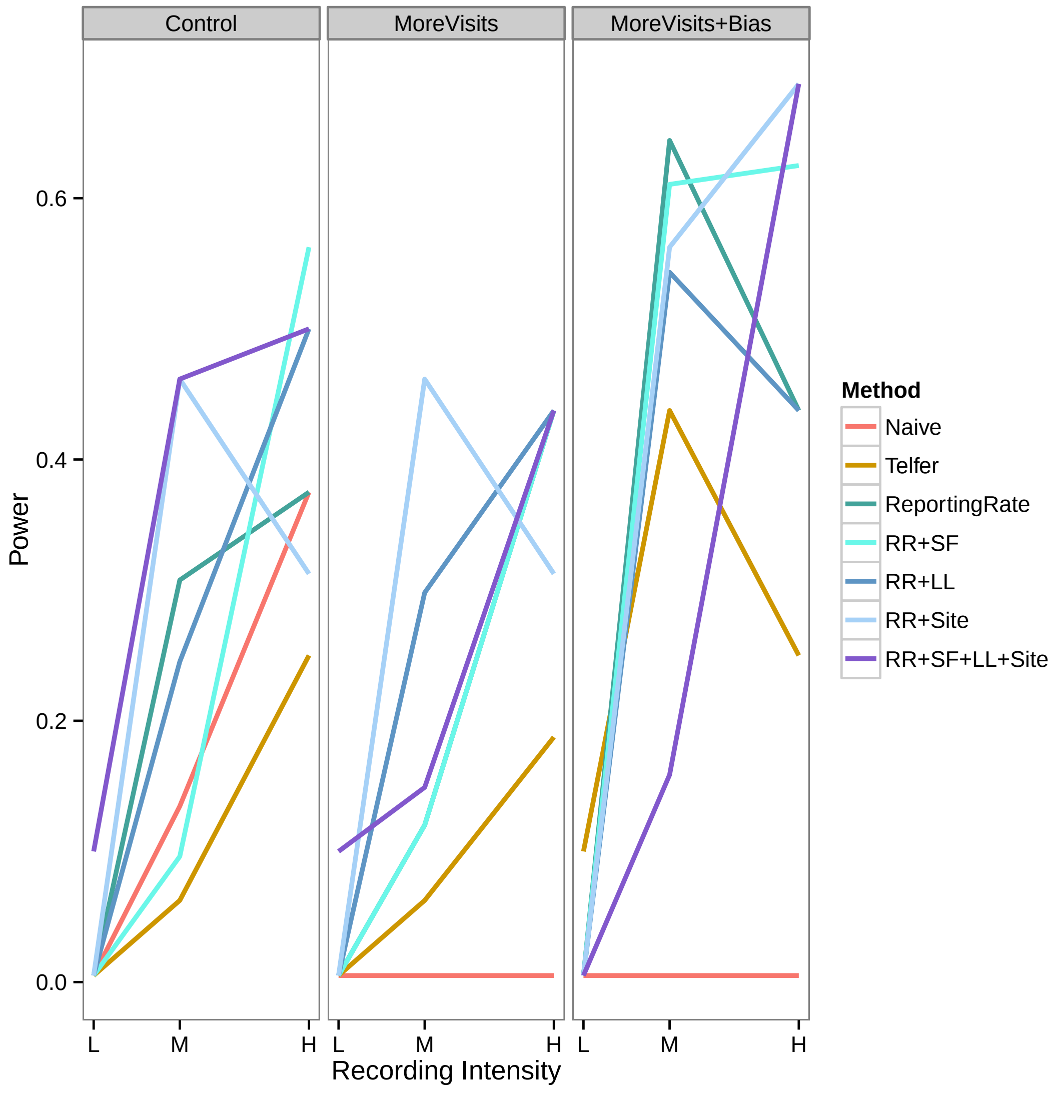


Control
MoreVisits
MoreVisits+Bias
Type I error rate
0.50
0.20
0.10
0.05
0.01
Method
Method
Naive
Telfer
ReportingRate
RR+SF
RR+LL
RR+Site
RR+SF+LL+Site

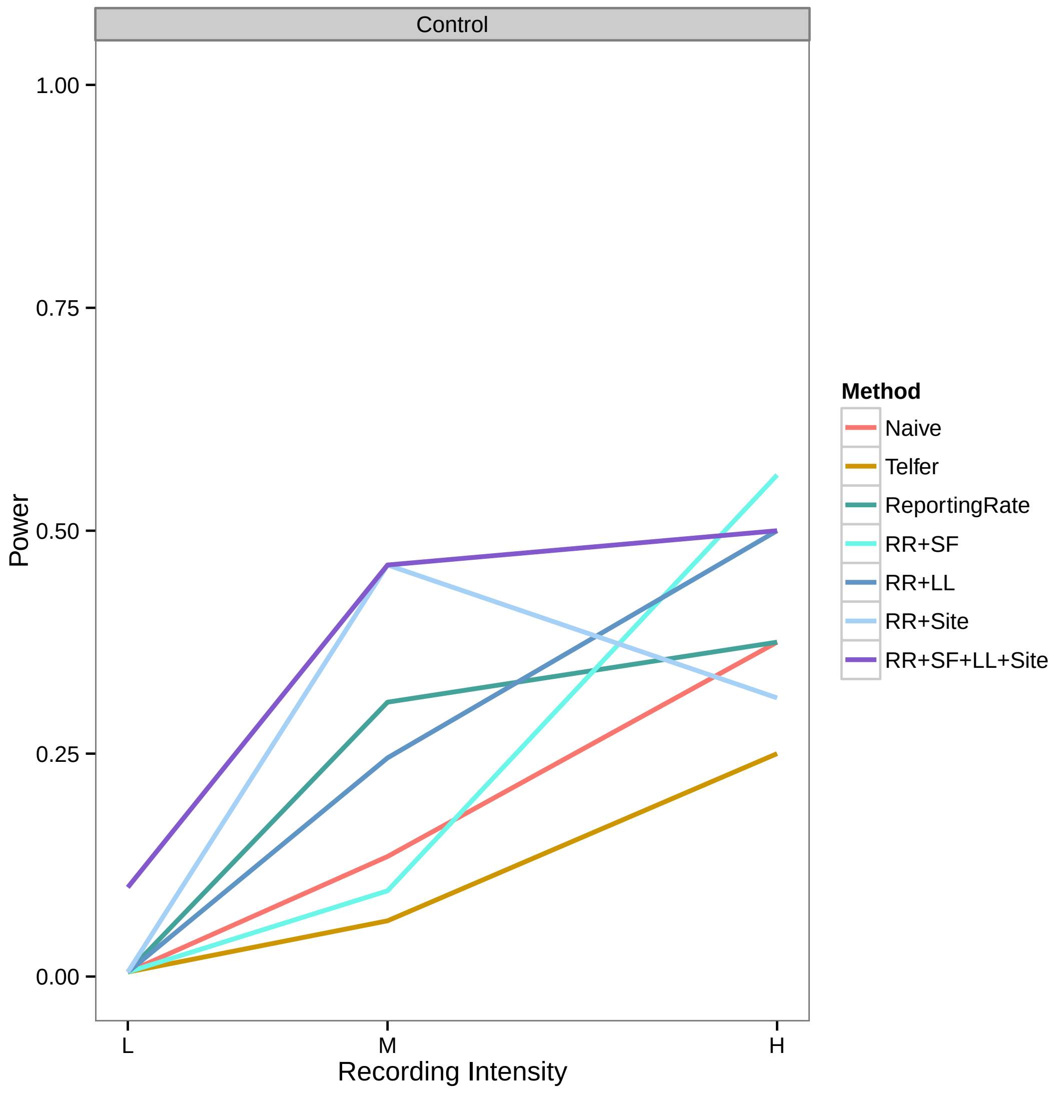

Control
MoreVisits
MoreVisits+Bias
Power
0.6
0.4
0.2
0.0
Method
Method
Naive
Telfer
ReportingRate
RR+SF
RR+LL
RR+Site
RR+SF+LL+Site



Control
MoreVisits
MoreVisits+Bias
Type I error rate
0.50
0.20
0.10
0.05
0.01
L
M
H
Recording Intensity
Method
Naive
Telfer
ReportingRate
RR+SF
RR+LL
RR+Site
RR+SF+LL+Site



Control
MoreVisits
MoreVisits+Bias
Power
0.0
0.2
0.4
0.6
L
M
H
Recording Intensity
Method
Naive
Telfer
ReportingRate
RR+SF
RR+LL
RR+Site
RR+SF+LL+Site


Control
1.00
0.75
0.50
0.25
0.00
Power
L
M
H
Recording Intensity
Method
Naive
Telfer
ReportingRate
RR+SF
RR+LL
RR+Site
RR+SF+LL+Site