

# Case Study 1

E.Wolkovich, D.Loughnan and X.Wang

This code for the case study includes two sections. The first section explains how to simulate data that fits our needs. We aim to simulate population data where the population is declining, but the p-value remains greater than 0.05. The second section is on data visualization, comparing how the results differ when using Fisherian approaches and Bayesian approaches.

## Data simulation

To simulate data that fits our needs, we use a simple linear function and add noise. We want the population data to exhibit either an increasing or decreasing trend, but some trends are not statistically significant (with a p-value greater than 0.05).

## Function

Function to simulate a population with linear growth and noise:

```
simulate_population <- function(a, b, t, noise_sd) {  
  y <- numeric(t)  
  time <- 1:t  
  y <- a*time + b + rnorm(t, 0, noise_sd)  
  # y is the population of certain time  
  # a is the increasing/decreasing rate  
  # b is the starting population size  
  # time is the x variable  
  # noise_sd  
  model <- lm(y ~ time)  
  # Extract the estimated slope (coefficient of time)  
  estimated_slope <- coef(model)["time"]  
  pval <- coef(summary(lm(y ~ time)))[["time", "Pr(>|t|)"]]  
  return(c(estimated_slope, pval))  
}
```

## Simulate population data

Simulate population data with different increasing and decreasing rates as well as different noises

```
t <- 10
a <- seq(from=-2000, to=2000, by=200)
b <- 100000
noise_sd <- seq(from=1000, to=10000, by=400)

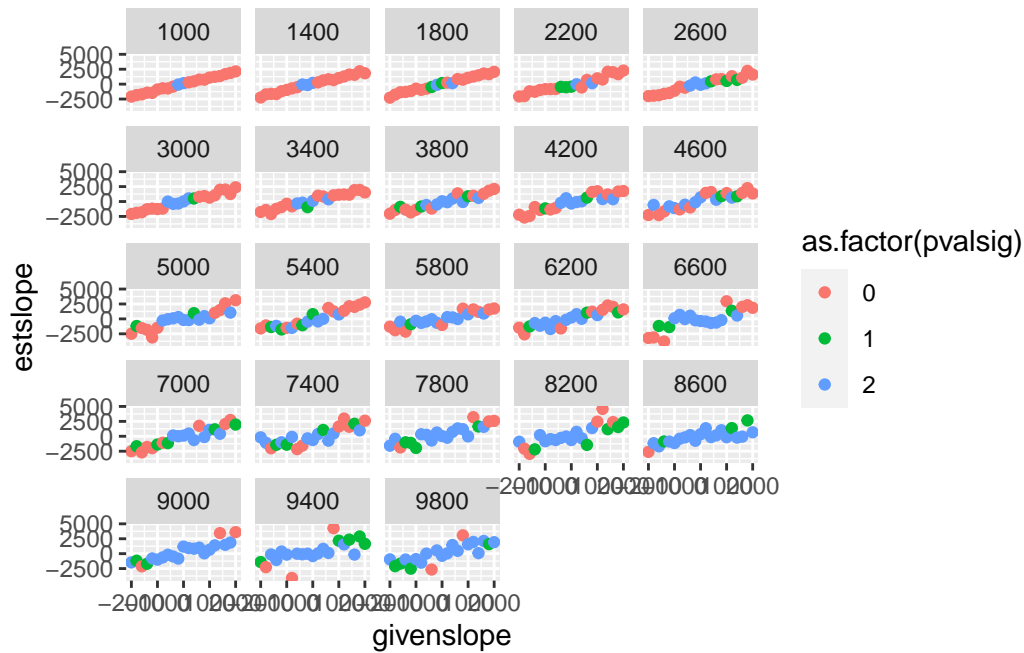
dfout <- data.frame(givenslope=numeric(), noise=numeric(), estslope=numeric(),
                    pval=numeric())

for (i in 1:length(a)) {
  for (j in 1:length(noise_sd)){
    simpopout <- simulate_population(a[i], b, t, noise_sd[j])
    dfadd <- data.frame(givenslope=a[i], noise=noise_sd[j],
                       estslope=simpopout[1], pval=simpopout[2])
    dfout <- rbind(dfout, dfadd)
  }
}
```

## Select populations

### p-values

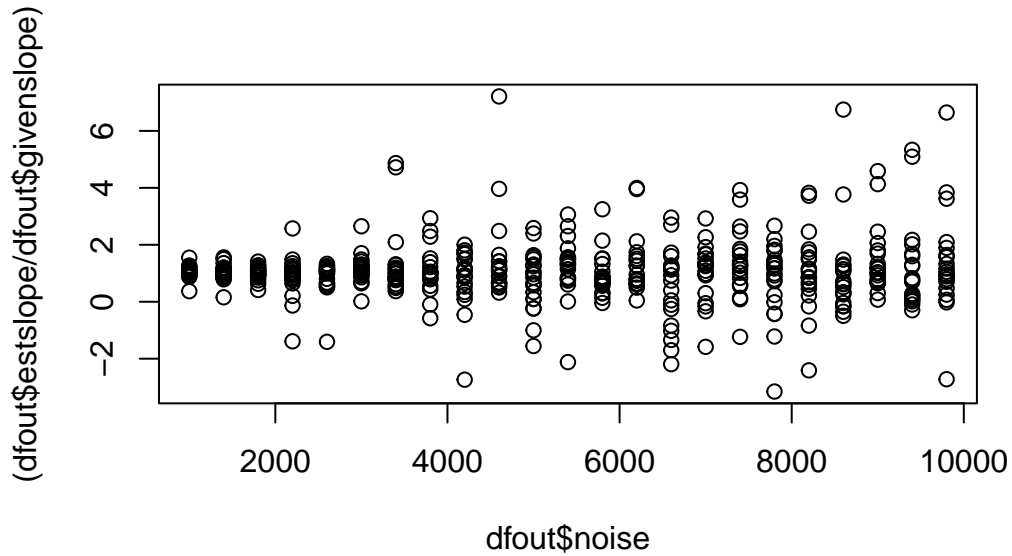
Set p-value thresholds to 0, 1, 2 where 0 is  $<0.05$ , 1 is 0.05 to 0.15 and 2 is  $> 0.15$ , and we are looking for 1 appears for both positive slope and negative slope, which means not statistically significant



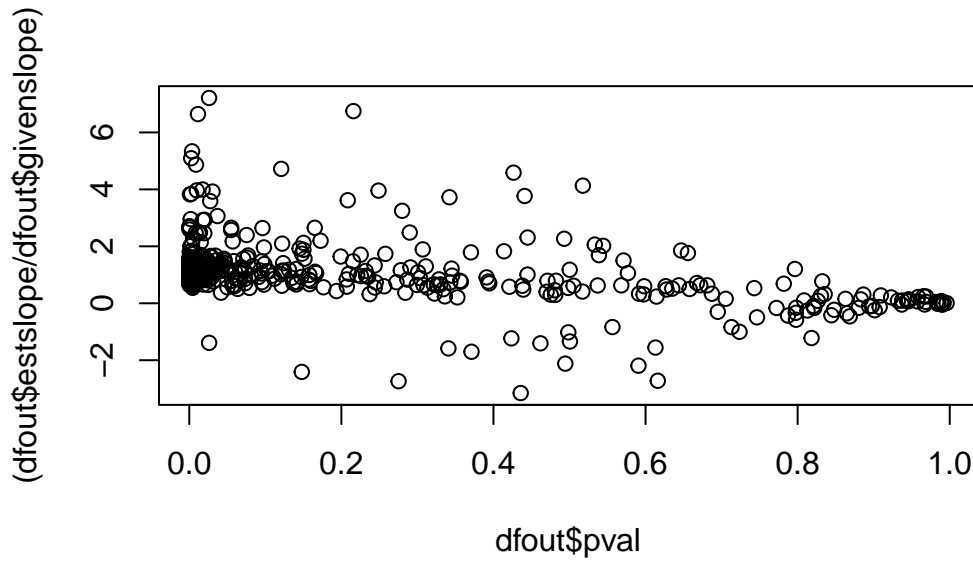
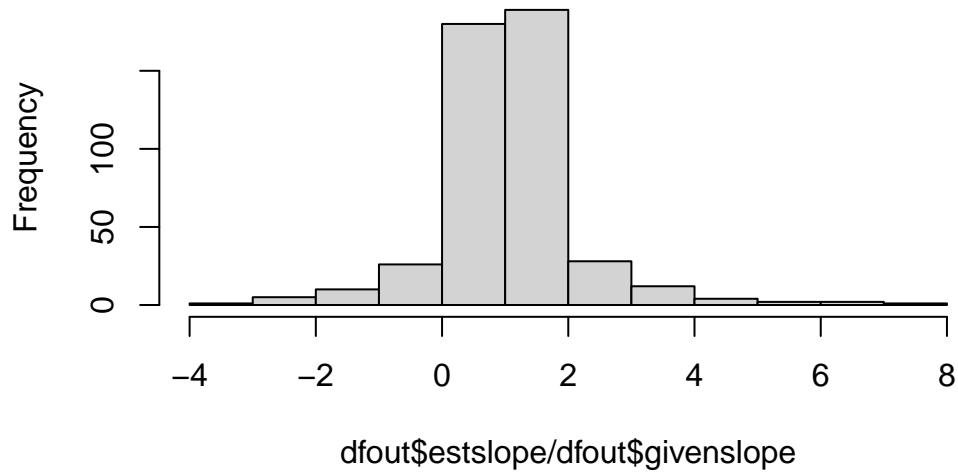
Visualization suggests that MAYBE noise of 7000-9000 could work

### Estimated slopes

Which noise range can still give us similar estimated slopes close to given slopes



## Histogram of $\text{dfout\$estslope}/\text{dfout\$givenslope}$



Visualization suggests that noise of 7000-9000 also fits

### Search for populations:

Semi-brute force search for cases were:

- (1) All the estimated slopes are pretty close to given slopes and
- (2) The bottom populations has a p-value between 0.05-0.15

```

a <- seq(from=-2000, to=2000, by=200)
a <- a[-c(5:17)]
# remove middle close-to-zero slopes, as though are hard to get pval low
noise_sd <- seq(from=7000, to=9000, by=400)
srhere <- 0.3

for(seedz in c(333:3999)){
  set.seed(seedz)
  for (j in 1:length(noise_sd)){
    dfout <- data.frame(noise=numeric(), sloperatio=numeric(), pval=numeric(),
                        smratio=numeric())
    for (i in 1:length(a)) {
      simpopout <- simulate_population(a[i], b, t, noise_sd[j])
      smratiohere <- ifelse((abs(simpopout[1]/a[i])<(1+srhere)
                           & abs(simpopout[1]/a[i])>(1-srhere)), 0, 1)
      dfadd <- data.frame(noise=noise_sd[j], sloperatio=simpopout[1]/a[i],
                          pval=simpopout[2], smratio=smratiohere)
      dfout <- rbind(dfout, dfadd)
    }
    if(sum(dfout$smratio)==0 & dfout$pval[1]>0.05 & dfout$pval[1]<0.15){
      print(paste("try seed: ", seedz, "with noise of ", noise_sd[j]))
    }
  }
}

```

```

[1] "try seed: 423 with noise of 7000"
[1] "try seed: 1546 with noise of 7000"
[1] "try seed: 1869 with noise of 7800"
[1] "try seed: 2154 with noise of 7800"
[1] "try seed: 2954 with noise of 7800"
[1] "try seed: 3069 with noise of 7400"
[1] "try seed: 3201 with noise of 7400"
[1] "try seed: 3488 with noise of 8600"
[1] "try seed: 3545 with noise of 8600"
[1] "try seed: 3786 with noise of 8600"

```

We got seeds: 423, 1546, 1869, 2154, 2954, 3069, 3201, 3488, 3545 and 3786.

### Try different seeds

After checking different seeds, we finally decided on seed 1546

	givenslope	noise	estslope	pval
time	-2000	8600	-1708.306	0.146969632
time1	-1800	8200	-2121.081	0.031744531
time2	-1600	7800	-1494.864	0.190847601
time3	-1400	7400	-1330.010	0.164436645
time4	1400	7000	1499.433	0.035391410
time5	1600	6600	1260.549	0.116379852
time6	1800	6200	1600.972	0.005141529
time7	2000	5800	2212.688	0.004420759

We dropped the 2nd population with p-value smaller than 0.05.

## Data analysis

The following code performs data analysis and visualization using both the traditional Fisherian approach and the Bayesian approach. For the Fisherian approach, we use an `lm` model and plot the p-values for different populations, indicating whether the increase or decrease is statistically significant. For the Bayesian approach, we use Stan code with priors and plot the posterior distribution.

```
# Decided to run with 1546
set.seed(1546)
# Dropping -1600, 1600
a <- c(-2000, -1600, -1400, 600, 1400, 1600, 2000)
t <- 10
time <- 1:t
b <- seq(40000, 101000, by=10000)
noise_sd <- c(7000, 6600, 6200, 5800, 5400, 5000, 4600)
```

### Traditional Fisherian approach

With traditional Fisherian approach, we use `lm` model to plot

```
output <- data.frame(iter = seq(1:(length(a)*length(time))),
                     pop = rep(1:length(a), each = length(time)),
                     year = rep(1:t, times = length(a)))
output$pred <- NA

abund <- vector()
for(i in 1:length(a)){
```

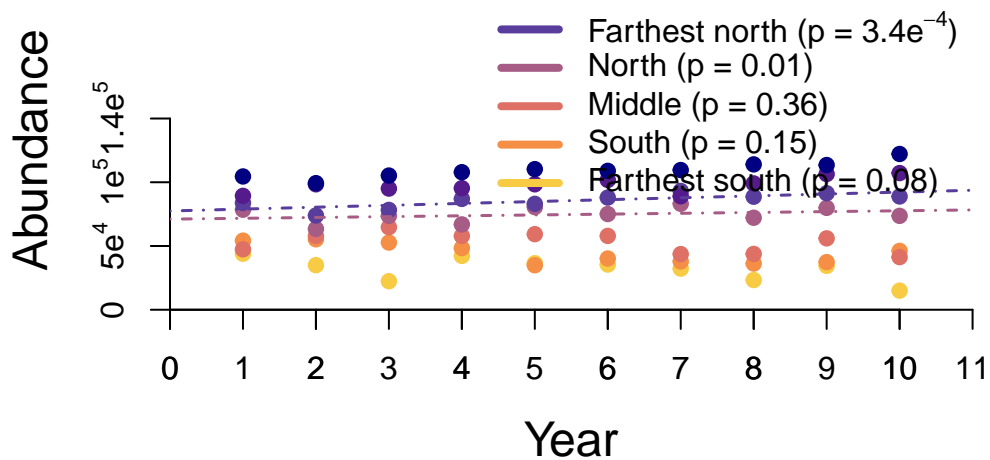
```

y <- numeric(t)
time <- 1:t
y <- a[i]*time + b[i] + rnorm(t, 0, noise_sd[i])
abund <- rbind(abund, y)
}

abund <- data.frame(reshape2::melt(t(abund)))
output$pred <- abund$value

```

## Visualization



## Bayesian approach

With Bayesian approach, we run the stan code:

```

//
data {
  int<lower=0> N; //No. obs
  int<lower=0> Ngrp; //No. in group---population or species
  int group[N]; // Group type
  vector[N] year;
  real ypred[N]; //response
}

parameters {
  real a[Ngrp] ;
  real b[Ngrp];

```

```

    real mu_a;
    real<lower=0> sigma_a;
    real mu_b;
    real<lower=0> sigma_b;
    real<lower=0> sigma_y;

}

model {

    real mu_y[N];

    for(i in 1:N){
        mu_y[i]=a[group[i]]+b[group[i]]*year[i];
    }

    a ~ normal(mu_a, sigma_a);
    b ~ normal(mu_b, sigma_b);

    //Priors
    mu_a ~normal(188, 50);
    sigma_a ~normal(0,50);
    mu_b ~normal(0,10); //could also be centred at zero, 10
    sigma_b ~normal(0,10); //sigma_b 0,10
    sigma_y ~normal(0,10);

    ypred ~ normal(mu_y, sigma_y);
}

```

```

'data.frame':  50 obs. of  4 variables:
 $ iter: int  1 2 3 4 5 6 7 8 9 10 ...
 $ pop : int  1 1 1 1 1 1 1 1 1 1 ...
 $ year: int  1 2 3 4 5 6 7 8 9 10 ...
 $ pred: num  43951 34961 22390 42250 36469 ...

```

```

mdlPop <- stan("partialPoolSimMdl.stan",
              data = datalistGrp)

sum <- summary(mdlPop)$summary

intercept <- sum[grep("a\\[", rownames(sum)), "mean"]
slopes <- sum[grep("b\\[", rownames(sum)), "mean"]

```



```
# Posterior distribution  
post <- rstan::extract mdlPop)
```

### Visualization

