PAN-E_Demo

PAN-E Team

20/06/2020

Introduction

This text provides worked examples of how we propose to analyse data sets depending on the type, duration and complexity of each source. For each dataset, we aim to quantify a percentage change from expected or typical values, as well as a standardised effect size in the form of a t-statistic. In order to calculate these metrics, we will ask you to provide a summary table based on analyses that will hopfeully be similar to one of the examples provided below. However, we are aware that each analysis will likely need some conversation, so please let us know if you have ANY questions.

Once you provide a summary table, please get in touch with us and we will send a google form to collect information on your project's methods and results, as well as details on co-authors, grants, acknowledgements, data sources to be included in the publication.

1 Regression on count data

Our first example could apply to changes in the abundance of a species observed over time. For instance, we might observe increased occurrance of urban species as a linear trend, defining the start of the lockdown as time 0 and going to time=50. The code block below simulates a a poisson-distributed response, where the observations start at the beginning of lockdown.

```
# Simulate data
n = 50
time = 1:(n)
a = 0.1
b = 0.025
Obs = rpois(n, exp(a + b * time))
y1 = data.frame(time, Obs)
```

Figure 1, shows counts increasing from time 0 to time 50. In our model we use a quasipoisson distribution to allow us to get the required t-statistic using the 'glm' function in the base R stats package. Table 1 gives us the t-statistic for the centered treatment, as well as the coefficients needed to calculate the percentage change.

```
m1 <- glm(Obs ~ time, family = stats::quasipoisson, data = y1)
```

In this example, we will use the intercept and slope terms calculate percentage change as the difference:

Table 1: table for the quasipoisson regression for the response over time

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.2051311	0.2479822	-0.8272006	0.4122169
time	0.0372045	0.0070008	5.3143538	0.0000027

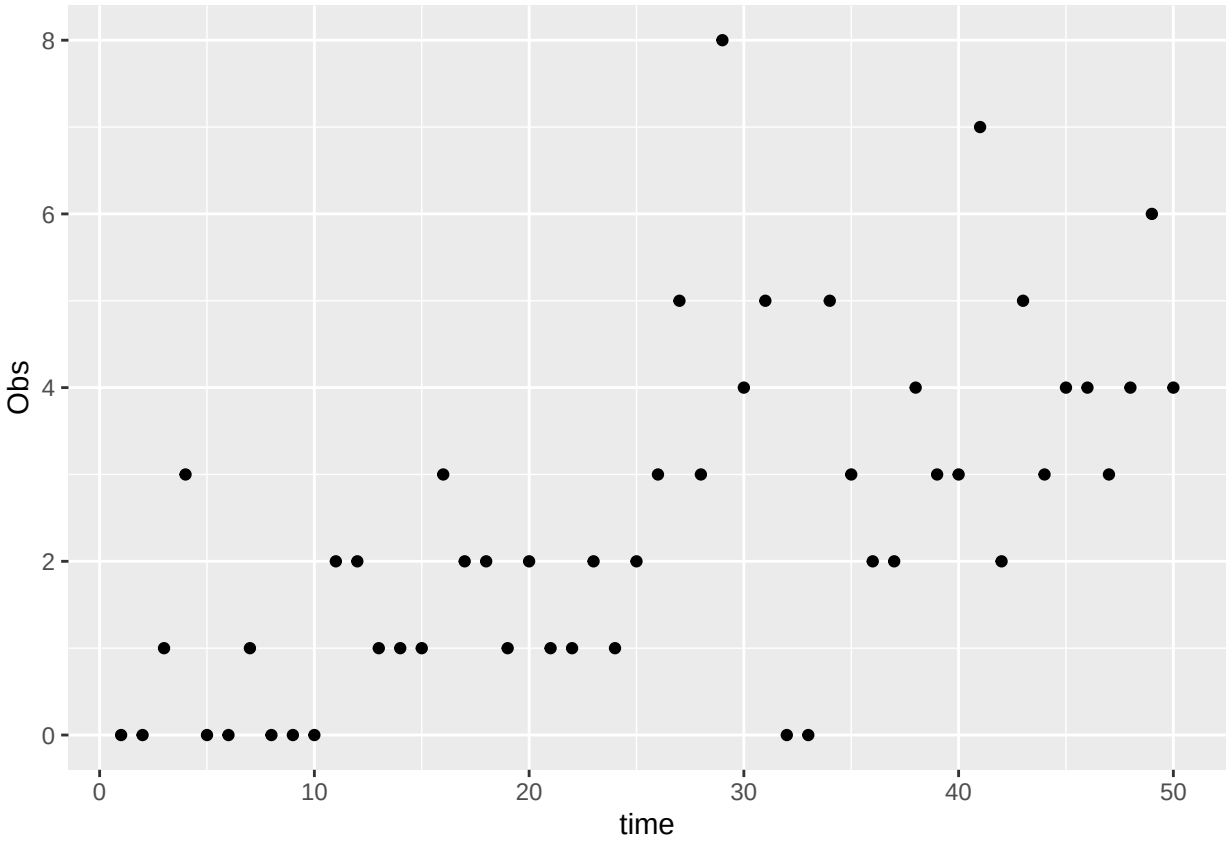

Figure 1: Linear trend over time.

$$\%Change = 100 \times \frac{\exp(Int + beta * 50) - exp(Int)}{\exp(Int)}$$

, Which gives us 535.98.

2 Before-during lockdown, gaussian data

Our second example describes normally-distributed response data in a typical pre-covid period versus a similar period during covid lockdown. For example, a study might have observations in May 2019 and then wish to contrast them against May of 2020 (Figure 2). For Normally distributed data, this analysis can use the standard 'lm' function which gives us the summary found in table 2. In this case, we want to set 'During' as the contrast in the linear model, whcih we achieve by releveling the treatment factor.

```
Treatment = rep(c("Before", "During"), each = n)
mn = rep(c(1, 2), each = n)
std = rep(c(1, 1), each = n)
Obs = rnorm((2 * n), mn, std)
y2 = data.frame(Treatment, Obs)
# We set the contrast so that 'before' is the reference.
y2$Treatment = relevel(y2$Treatment, "Before")
```

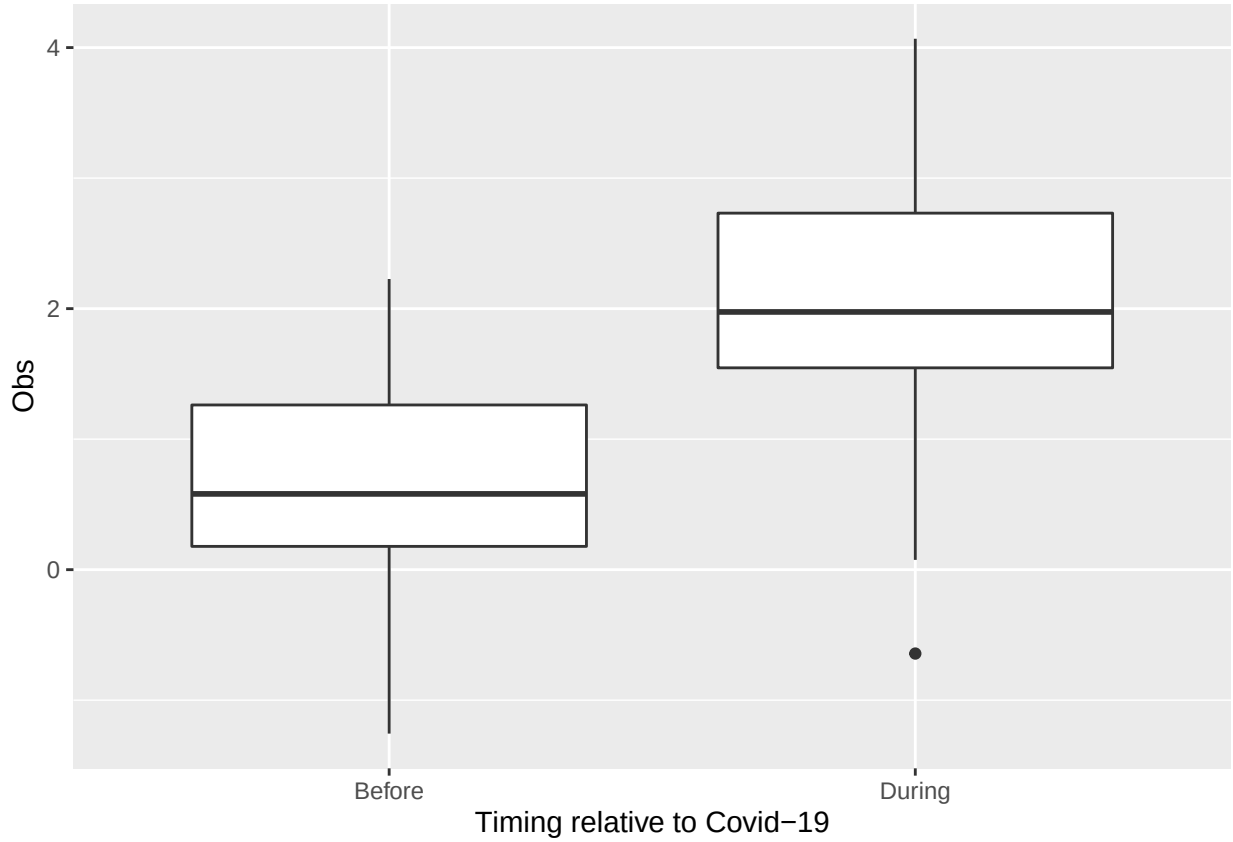

Figure 2: Example of normally distributed before vs during covid data.

```
m2 <- glm(Obs ~ Treatment, family = stats::gaussian, data = y2)
```

Table 2: table for the linear regression for a before-during response

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.6269517	0.1290894	4.856725	4.5e-06
TreatmentDuring	1.4313917	0.1825600	7.840666	0.0e+00

3 QuasiBinomial spatial block treatment

The comparison in this case is with two locations in which we would normally expect similar occurrence data, but where the lockdown has had an effect on one location but not the other. We treat this as a binomial distributed dataset with a spatial block/treatment during the lockdown, as might be found with occurrance data (Figure 3).

```
Block = rep(c("A", "B"), each = n)
p = rep((c(0.2, 0.6)), each = n)
Present = rbinom((2 * n), rep(1, 2 * n), p)
y3 = data.frame(Block, factor(Present))
```

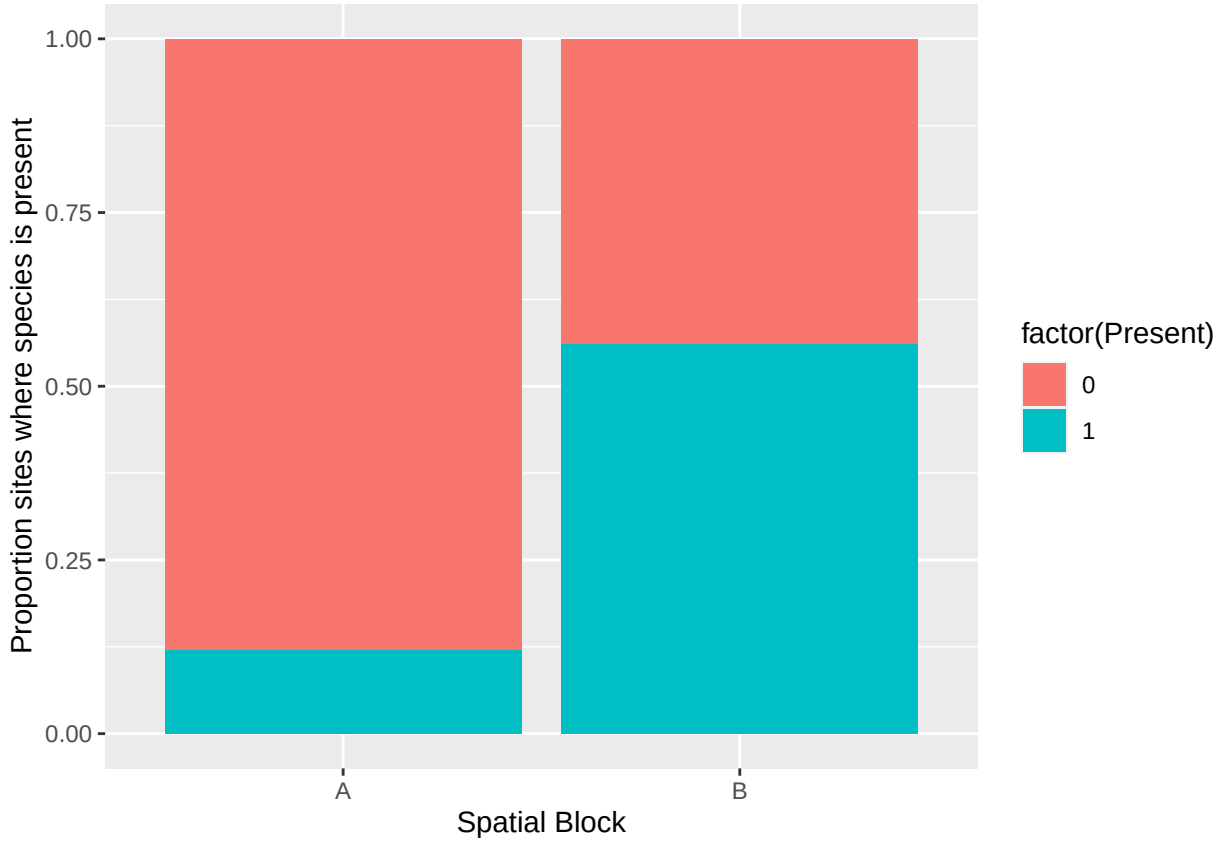

Figure 3: Binary data in two spatial blocks

```
m3 <- glm(Present ~ Block, family = stats::quasibinomial, data = y3)
```

Table 3: table for quasibinomial spatial block treatment

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-1.992430	0.4396107	-4.532260	1.65e-05
BlockB	2.233592	0.5254366	4.250927	4.86e-05

Table 4: Table for the fixed effects model for a pre-versus during lockdown response with random effects

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.5506473	0.2832026	66	1.944359	0.0561162
TimingDuring	1.2016180	0.0918731	66	13.079104	0.0000000

4 Random effects, pre versus during lockdown

We might also encounter data where the within-year data are in a blocked design - for instance spatial blocks "A", "B" and "C", but where the effort in each is unbalanced (Figure 4). To look for Pre (2019)- versus during-lockdown (2020) differences, we can accomodate this blocked design using random effects.

```
library(nlme)

block = rep(c(rep("A", 10), rep("B", 20), rep("C", 5)))
mn_before = c(0.5, 1, 0.2)
sd_before = c(0.3, 0.4, 0.5)
mn_during = c(1.5, 2.3, 1.2)
sd_during = c(0.3, 0.4, 0.5)
Before.df = data.frame(means = c(mn_before), sds = c(sd_before))
During.df = data.frame(means = c(mn_during), sds = c(sd_during))
rownames(Before.df) = rownames(During.df) = c("A", "B", "C")

Obs = c(rnorm(length(block), mean = Before.df[block, ]$means,
    sd = Before.df[block, ]$sds), rnorm(length(block), mean = During.df[block,
    ]$means, sd = During.df[block, ]$sds))

Timing = rep(c("Before", "During"), each = length(block))

y4 = data.frame(block, Obs, Timing)

# set 'Before' as reference so that 'During' is the contrast
y4$Timing = relevel(y4$Timing, "Before")
```

We accomodate the structure of these data using the 'lme' function from the package 'nlme', where the syntax 'random = ~1|block' indicates that the blocks are to be treated as the random effects.

```
m4 <- lme(data = y4, Obs ~ Timing, random = ~1 | block)
```

In this case, we are not interested in the differences between blocks, and are simply trying to find the glocal effect. The summary of the fixed effects (table 4) shows the reduction of 1.2 in the intercept for year 2020 relative to 2019.

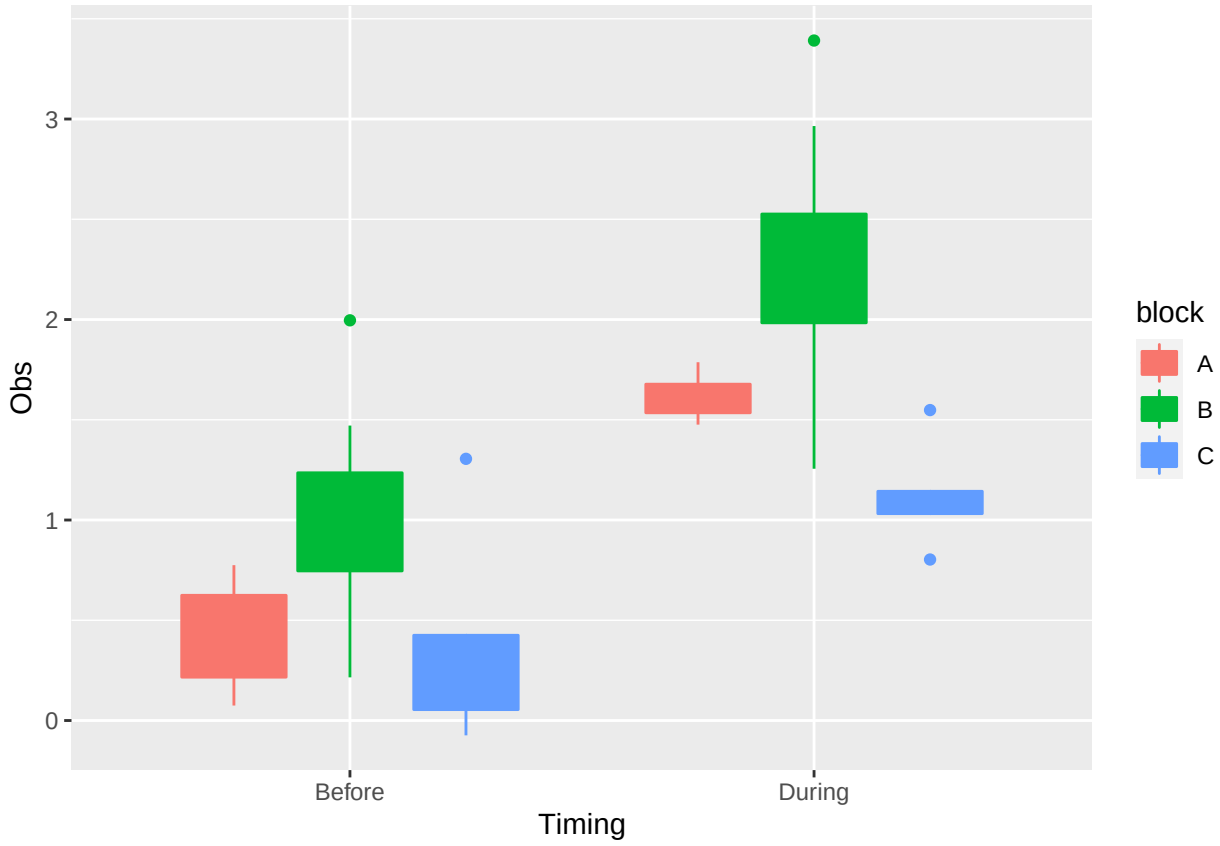

Figure 4: Before-covid data with three spatial blocks

5 Random effects, multiple linear trends

We might also encounter data where we observe linear trends within spatial blocks “A”, “B” and “C” (Figure 4). To look for Pre (2019)- versus during-lockdown (2020) differences, we can accomodate this blocked design using random effects.

```
library(nlme)

block = rep(c(rep("A", 10), rep("B", 20), rep("C", 5)))
time = c(sample(1:20, 10), 1:20, sample(1:20, 5))
means = c(0, -1, 1)
sds = c(0.8, 2, 1)
beta = c(0.35, 0.45, 0.55)
trend.df = data.frame(means = c(means), sds = c(sds), beta = beta)
rownames(trend.df) = c("A", "B", "C")

Obs = rnorm(length(block), mean = trend.df[block, ]$means + time *
    trend.df[block, ]$beta, sd = trend.df[block, ]$sds)

y5 = data.frame(block, Obs, time)
```

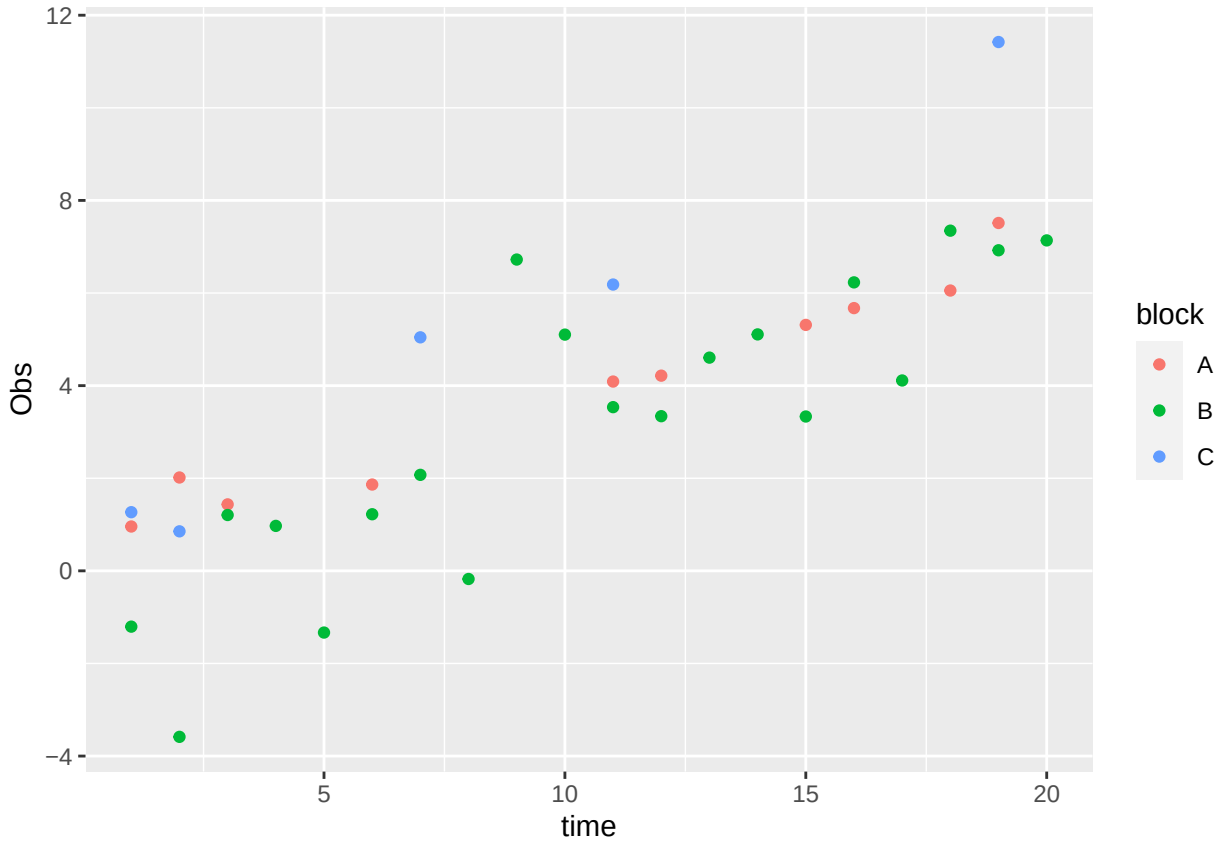

Figure 5: Before-during data with three spatial blocks

We accomodate the structure of these data using the ‘lme’ function from the package ‘nlme’, where the syntax ‘random = ~1|block’ indicates that the blocks are to be treated as the random effects.

Table 5: Table for the random effects model with multiple linear trends

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.2004561	0.9211903	31	-0.2176055	0.8291625
time	0.4298350	0.0401690	31	10.7006772	0.0000000

Table 6: t-Table from model 6 showing fixed effects coefficients

	Value	Std.Error	t-value	p-value
(Intercept)	0.9196639	0.2339495	3.931037	0.0003463
CovidDuring	3.8639753	0.6413713	6.024553	0.0000005

```
m5 <- lme(data = y5, Obs ~ time, random = ~1 | block)
```

The summary of the fixed effects (table 4) shows the reduction of 0.43 in the intercept for year 2020 relative to 2019. Here we can calculate the global change over time again using the slope and intercept terms from the model summary after accounting for the spatial blocks.

6 Temporal autocorrelation

Sometimes we may wish to look at responses before and during covid, while accounting for temporal autocorrelation in the data. Example 6 uses the same data as Example 6, (Figure 6). Here we use a temporal autocorrelation model to account for the time series. We accounted for temporal autocorrelation using the 'gls' function in the package 'nlme', and the results are shown in (table 6). Here, the autocorrelation term simply accounts for the temporal trend, but our interest is in the post covid increase.

```
years = 2015:2020
int = 0
nrep = sample(4:8, length(years), replace = T)
sampleyears = years %>% rep(nrep)
time = sampleyears - min(years)
Int = 0
Beta = 0.5
sds = 1
Obs = rnorm(length(sampleyears), mean = Int + Beta * (time) +
    (sampleyears == 2020) * 2, sd = sds)
y6 = data.frame(Obs = Obs, (sampleyears), time = time, Covid = ifelse(sampleyears ==
    2020, "During", "Before"))
# set the reference treatment
y6$Covid = relevel(y6$Covid, "Before")
```

```
m6 <- gls(data = y6, Obs ~ Covid, corr = corARMA(form = ~1 |
    time, q = 2))
```

7 Temporal autocorrelation with random effects (e.g., multiple locations or species)

Finally, we look at an example where we have multuple species or locations with similar temporal autocorrelation structure as model 6 7. In this case we want to estimate the global effect while accounting for the

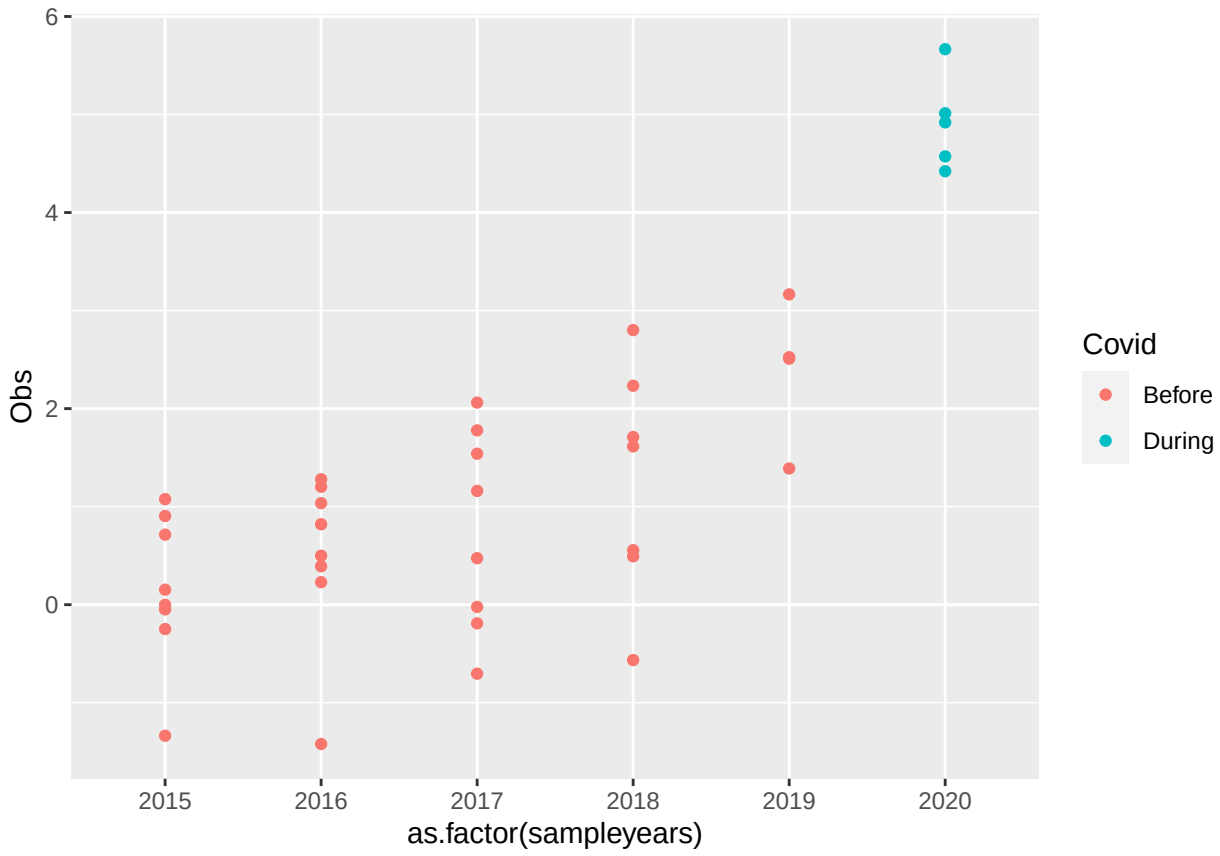

Figure 6: Before-during data from example 6 blocked into years.

within-location (or species) autocorrelation.

```
years = 2015:2020
int = 0
nrep = sample(4:8, length(years), replace = T)
locs = c("A", "B")
sampleyears = years %>% rep(nrep) %>% rep(each = length(locs))
location = rep(locs, length(sampleyears)/length(locs))
time = sampleyears - min(years)
covidresp = c(2, 0.5)
Int = 0
Beta = 0.5
sds = 1
Obs = rnorm(length(sampleyears), mean = Int + Beta * (time) +
    (sampleyears == 2020) * (location == "A") * 2 + (sampleyears ==
    2020) * (location == "B") * 1.5, sd = sds)
y7 = data.frame(Obs = Obs, (sampleyears), location = location,
    time = time, Covid = ifelse(sampleyears == 2020, "During",
        "Before"))
y7$Covid = relevel(y7$Covid, "Before")
```

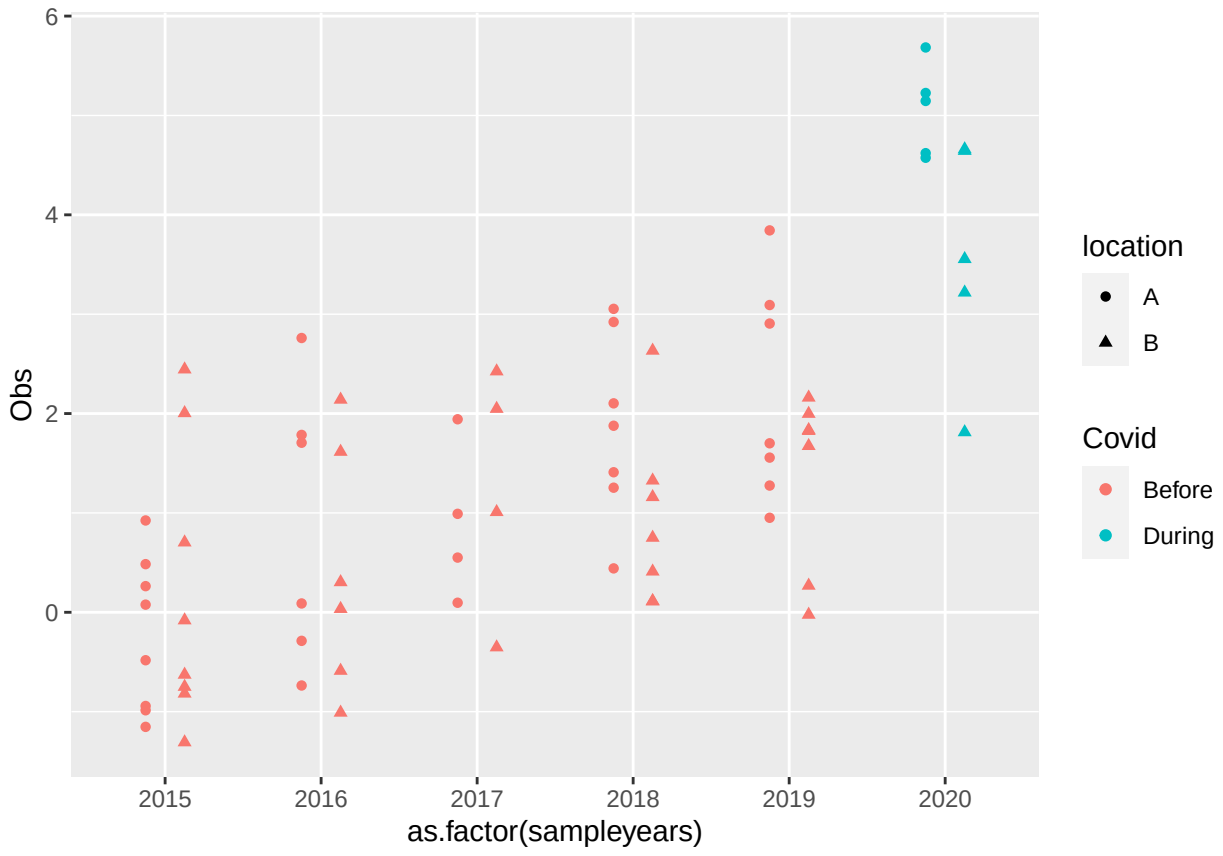

Figure 7: Before-during data from example 6 blocked into years.

```
m7 <- lme(data = y7, Obs ~ Covid, corr = corARMA(form = ~1 |
    location/time, q = 2), random = ~1 | location)
```

In the model summary table for the fixed effects, we see the slope

Table 7: t-Table from model 7 showing fixed effects coefficients

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.9300645	0.2418247	71	3.846028	0.0002593
CovidDuring	3.3411125	0.5025637	71	6.648137	0.0000000

Summary

These examples are a few of the ways we expect data to be analysed, however the best thing to do is talk with us if you have questions. And when you do have analyses ready to go, please get in touch and we will send through metadata forms for you to fill out with additional details on your study.

Many thanks for your efforts on this enormous collaboration!

The PAN-E team