The intrinsic predictability of ecological time series and its potential to guide forecasting

Reproducible report of statistical analysis of empirical time series

Frank Pennekamp

22 June 2018

Contents

```r
rm(list=ls())

# uncomment to install
#devtools::install_github("ropensci/rgpdd")
#devtools::install_github("thomasp85/patchwork")
#devtools::install_github("ropenscilabs/skimr")

library(rgpdd)
library(patchwork)
library(ggplot2)
library(dplyr)
library(purrr)
library(tidyr)
library(rEDM)
library(forecast)
library(readxl)
library(psych)
library(lme4)
library(lmerTest)
library(cowplot)
library(r2glmm)
library(skimr)
library(padr)
library(corrr)
library(here)
library(lubridate)

source(here::here("Code", "pe_code.R"))
```

Motivation

Our conceptual framework as well as our simulation results suggest a relationship between time series complexity as measured by permutation entropy (PE) and forecasting error. However, the simulations have also shown that problems commonly encountered by ecologists such as short and noisy time series can considerably distort the relationship. Therefore our first goal was to test for such a relationship across a range of ecological time series, and our second goal was to test whether time series covariates such measurement error (i.e. lab versus field time series), the non-linearity of the system or the time series length can explain deviations from a general relationship. To do so, we relied on a freely available compilation of time series from the Global Population Dynamic Database and a second database of laboratory based time series for which time series complexity was estimated by PE. We also forecasted the time series using the Empirical Dynamic Modelling (EDM) approach developed by George Sugihara and co-workers and made available via the rEDM package. This notebook documents this analysis.

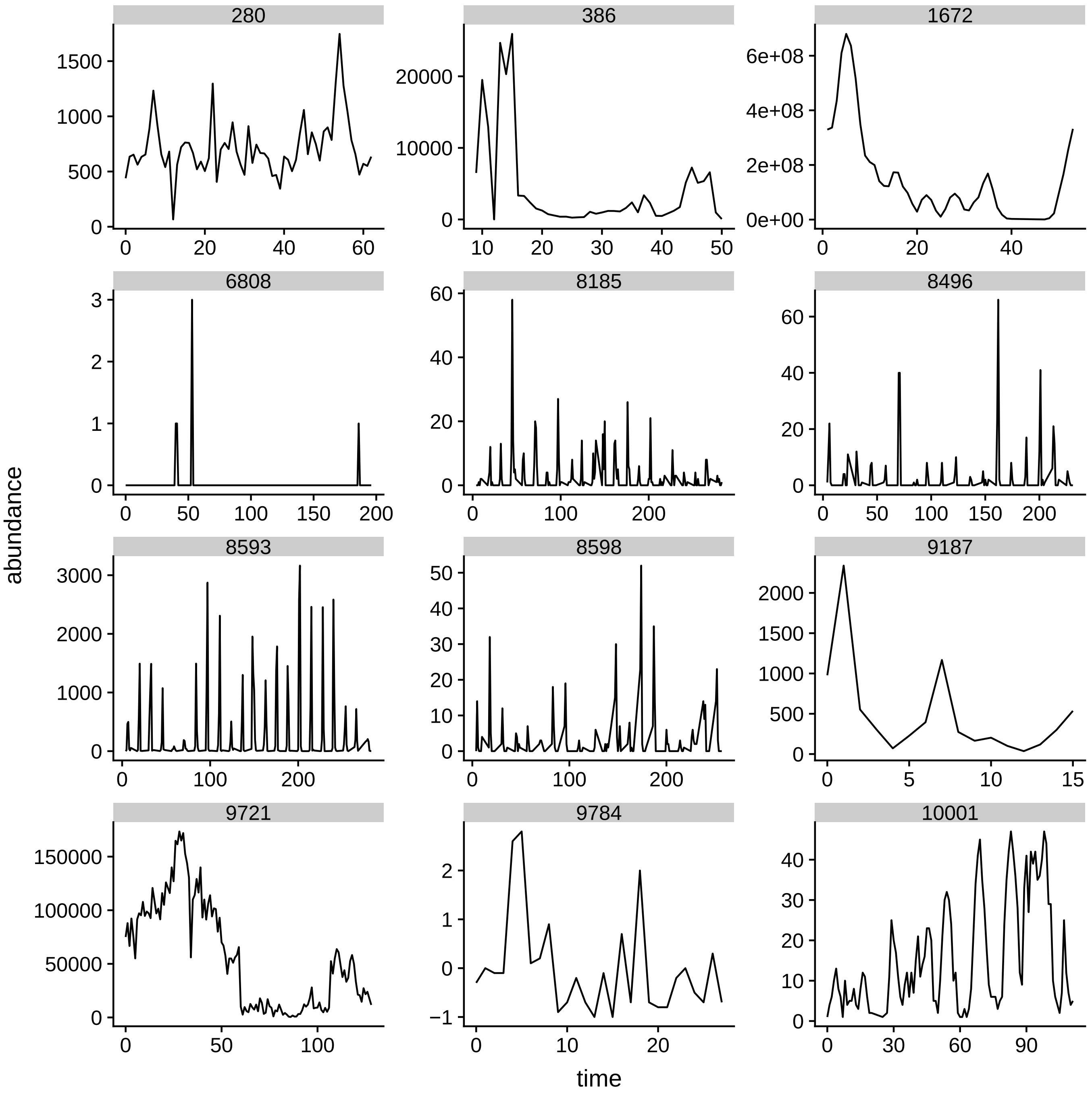
Load and prepare data for analysis

GPDD

The GPDD data was accessed via the rGDPP package. The database contains almost 5000 time series that vary widely in length and quality and is one of the go-to databases for ecologists to ask broad population dynamic questions. The rGPDD package contains tables with the time series information and additional information about the data sources, measurement units, and some fitted population dynamic parameters. For our purpose, we mainly use the "data" table, but added information about the source to control for potential non-independence in subsequent analyses. The plot shows a random sample of 12 time series from the GPDD.

```r
dd <- gpdd_data

# table with ts meta information
main <- gpdd_main
main <- select(main, MainID, DataSourceID, BiotopeID, TaxonID)

# table with source information
dd_source <- gpdd_datasource
dd_source <- select(dd_source, DatasourceID, Author, Year)
dd_source$source <- paste0(dd_source$Author," ", dd_source$Year )

# taxon info
gpdd_taxon <- gpdd_taxon
# unfortunately, no info on trophic level

metainfo <- merge(main, dd_source, by.x=c("DataSourceID"), by.y=c("DatasourceID"))
```

```r
# get actual time series
dd_GPDD <- rename(dd, abundance = PopulationUntransformed)
dd_GPDD$type <- "field"

# get time variable
dd_GPDD$time <- dd_GPDD$SeriesStep

# delete all time ids =  -9999
dd_GPDD <- dd_GPDD[dd_GPDD$time != -9999, ]

# nest data
dd_GPDD_nest <- dd_GPDD %>% group_by(MainID, type) %>% nest()

# add source info
dd_GPDD_nest <-merge(dd_GPDD_nest, select(metainfo, MainID, source),  by=c("MainID"))

# plot sample time series
ggplot(data=subset(dd_GPDD, MainID %in% sample(dd_GPDD$MainID, 12))) + geom_line(aes(x=time, y=abundanc
```



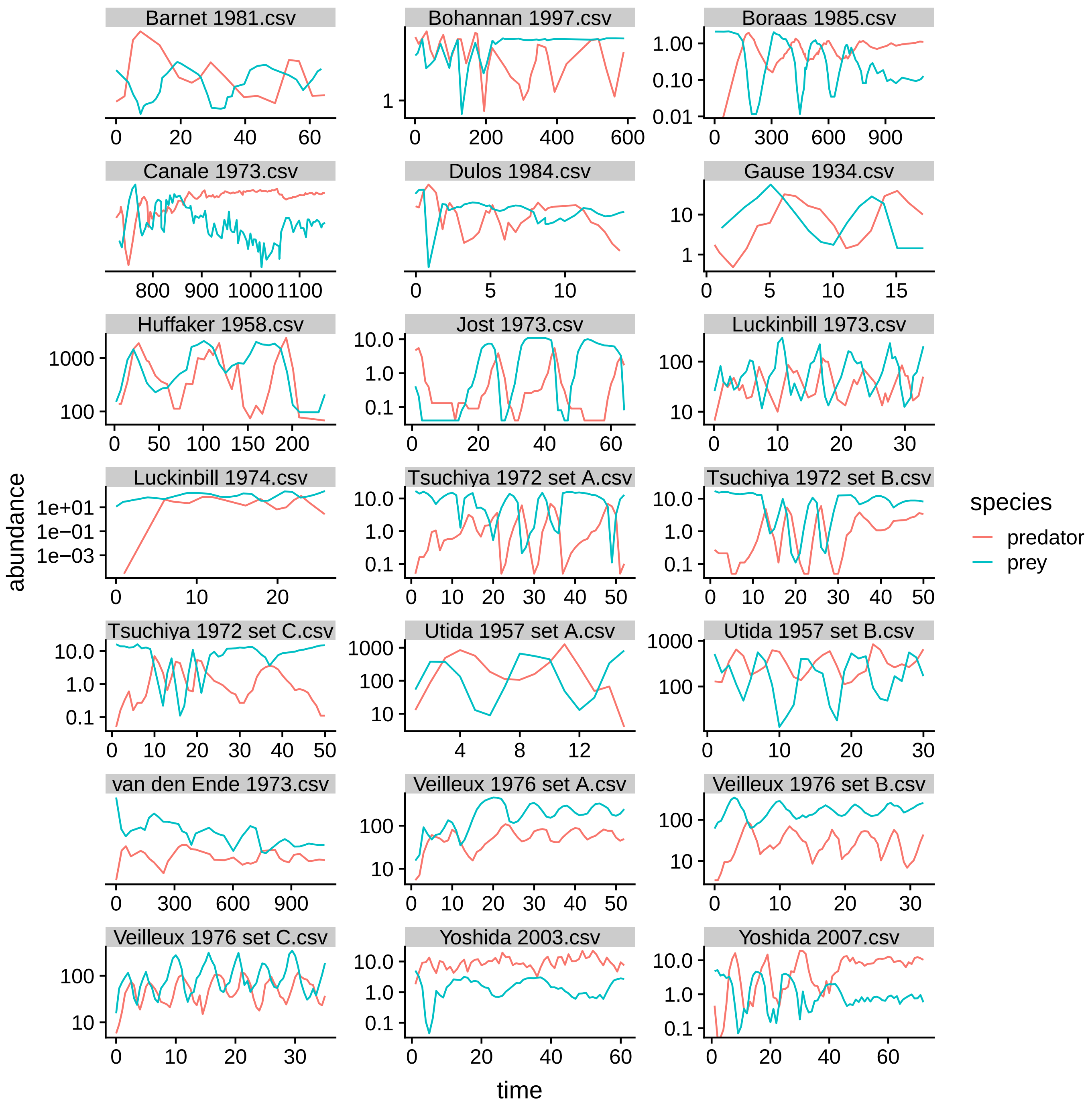
Laboratory data

For the database of laboratory time series we compiled data from Hiltunen et al. 2014, Fussmann et al. 2000 and Becks et al. 2005. Whereas the first publication is itself a compilation of laboratory time series, the second and third study used two systems (microbial predator-prey systems) across a range of chemostat dilution regimes. The two papers report how the different dilution regimes push the systems through a range of dynamical regimes spanning steady-state dynamics till chaotic dynamics. The following three plots show the dynamics across the systems. For the analysis we only used information from the univariate time series, although EDM would also allow to incorporate information the other species in the system to improve forecasts.

```r
# Hiltunen et al. dataset
dd1 <- read.csv(here::here("Data", "all_data_Hiltunen.csv"), stringsAsFactors = F)

dd1 <- subset(dd1, select = c(prey_time, prey_pop, pred_time, pred_pop, Source))
dd1 <- dplyr::rename(dd1, source = Source)

dd1a <- dd1[, c(1,2,5)]
dd1b <- dd1[, c(3,4,5)]
dd1a$species <- "prey"
dd1b$species <- "predator"

dd1a <- dd1a[, c(4, 1,2,3)]
dd1b <- dd1b[, c(4,1,2,3)]

names(dd1a) <- c("species", "time", "abundance", "source")
names(dd1b) <- c("species", "time", "abundance", "source")

dd1 <- bind_rows(dd1a, dd1b)

# delete NA rows
dd1 <- dd1[complete.cases(dd1),]
dd1$chemostat <- dd1$source
dd1$dilution_rate <- NA

# Fussmann et al. 2000 data
dd2 <- read.csv(here::here("Data", "analysisfigure3.csv"), header=T, skip = 1)

dd2 <- dd2[, c("Chlorella", "Brachionus", "meandelta", "day.", "chemostat")]
names(dd2) <- c("prey_pop", "pred_pop", "dilution_rate", "time", "chemostat")
dd2$source <- "Fussmann et al. 2000"
dd2 <- gather(dd2, "species", abundance, 1:2)

# next load data from microcosms
# Becks et al. dataset
dd3 <- read_excel(here::here("Data", "DATEN-Becksetal copy.xls"))
dd3 <- filter(dd3, !(predator_abundance == 0 & prey1_abundance == 0 & prey2_abundance == 0))

dd3 <- gather(dd3, "species", "abundance", 2:4)
dd3$species <- gsub("_abundance", "", paste0(dd3$species, "_", dd3$replicate))
dd3$source <- "Becks et al. 2005"

dd3 <- dplyr::rename(dd3, chemostat = dilution)

dd_lab <- bind_rows(dd1, dd2, dd3)
dd_lab$MainID <- as.numeric(as.factor(paste0(dd_lab$source, "_", dd_lab$chemostat, "_", dd_lab$species)
dd_lab$type <- "lab"

dd_lab_nest <- dd_lab %>% group_by(MainID, type, source) %>% nest()
```

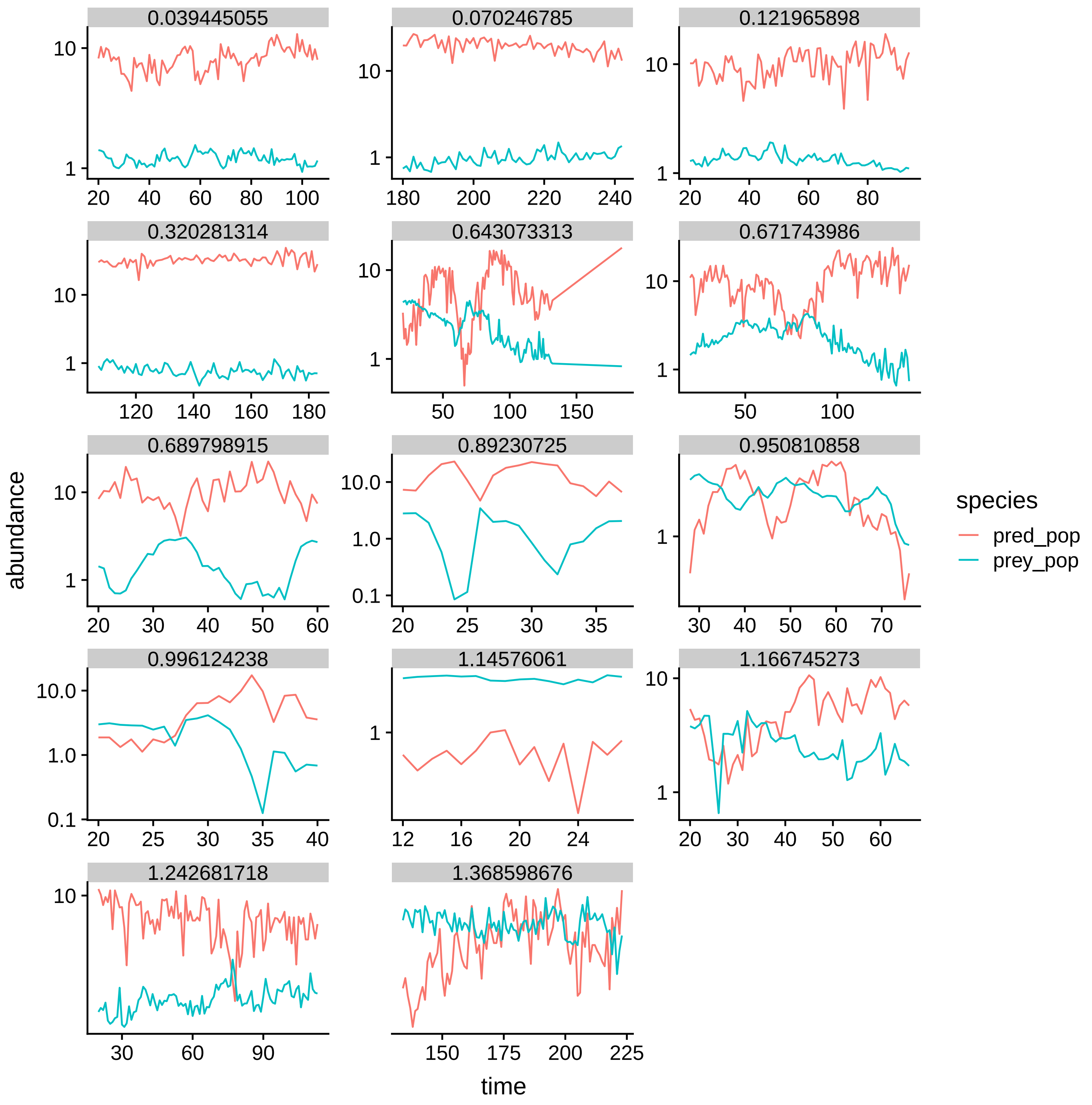
Hiltunen et al. 2014 data

```r
ggplot(data=dd1) + geom_line(aes(x=time, y=abundance, colour=species)) + facet_wrap(~source, scales = "
```



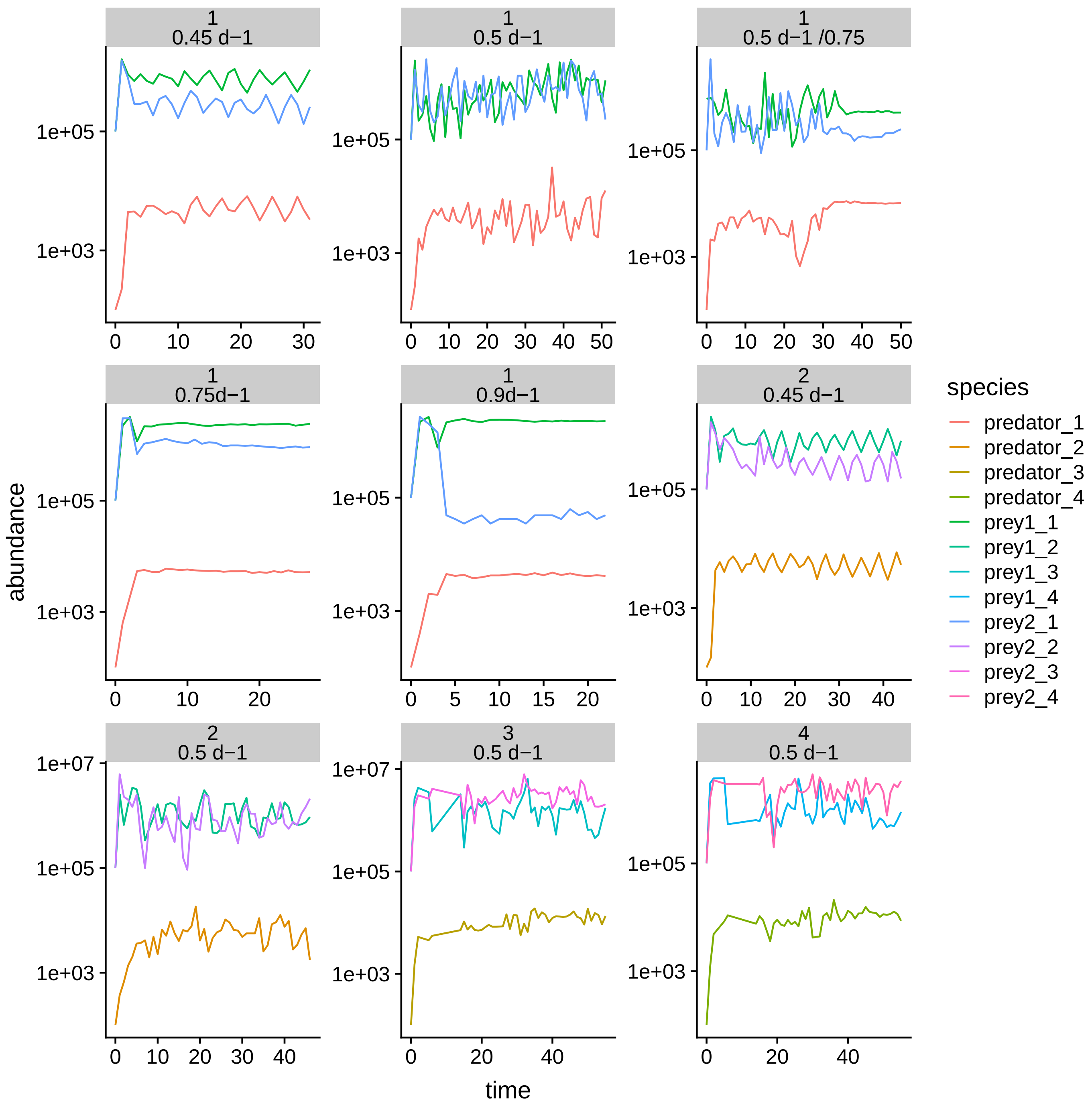
Fussmann et al. 2000 data

```
ggplot(data=dd2) + geom_line(aes(x=time, y=abundance, colour=species)) + facet_wrap(~dilution_rate, scal
```



Becks et al. 2005 data

```
ggplot(data=dd3) + geom_line(aes(x=time, y=abundance, colour=species)) + facet_wrap(replicate~chemostat
```



```
dd_ts <- bind_rows(dd_GPDD_nest, dd_lab_nest)
```

Data transformation

```
transform <- F
if(transform){
dd_ts$data <- lapply(1:nrow(dd_ts), function(x) mutate(dd_ts$data[[x]], abundance_sc = as.vector(scale(a
} else {
dd_ts$data <- lapply(1:nrow(dd_ts), function(x) mutate(dd_ts$data[[x]], abundance_sc = as.vector(scale(a
}
```

No data transformation applied. Centering of time series did not qualitative change the results.

Filtering datasets for analysis

We extracted some descriptive information about each of the time series. We calculate a number of time series properties for filtering: number of observations (N), number of zeros (N_zeros), proportion of zeros in the time series (zero_prop) and proportion of ties in the time series (ties_prop). Zero and tie proportions were calculated as they pose problems to the rank-based PE. However, different methods to deal with ties are available.

```
dd_ts$N <- unlist(lapply(dd_ts$data, nrow))
dd_ts$finite_prop <- unlist(lapply(1:length(dd_ts$N), function(x) sum(is.finite(dd_ts$data[[x]]$abundan
dd_ts$zeros <- unlist(lapply(1:length(dd_ts$N), function(x) sum(dd_ts$data[[x]]$abundance==0, na.rm = T
dd_ts$zero_prop <- dd_ts$zeros/dd_ts$N
dd_ts$ties_prop <- unlist(lapply(1:length(dd_ts$N), function(x) sum(diff(dd_ts$data[[x]]$abundance)==0,
dd_ts$number_of_gaps <- unlist(lapply(1:length(dd_ts$N), function(x) sum(abs(diff(dd_ts$data[[x]]$time)
```

We decided to only use time series with a minimum number of 30 observations and no gaps. This number is the minimum, as for very short time series PE may not be meaningful and also forecasting via EDM is limited. We also specified that the proportion of ties in the time series should be less than 15%.

```
dd_ts <- dd_ts %>% filter(N >= 30 & ties_prop < .15 & finite_prop > .9 & number_of_gaps == 0)
```

```
# Adding NAs for missing data

# create date variable to identify missing time points
dd_ts$data <- lapply(1:nrow(dd_ts), function(x) mutate(dd_ts$data[[x]], Date = ymd(paste0(as.character(

# identify where time is very unregularily sampled and exclude
dd_ts$var <- unlist(lapply(1:nrow(dd_ts), function(x) var(dd_ts$data[[x]]$Date)))
dd_ts <- dd_ts[!is.na(dd_ts$var), ]

# pad data frames with rows for missing data
dd_ts$data <- lapply(1:nrow(dd_ts), function(x) pad(dd_ts$data[[x]]))
dd_ts$data <- lapply(1:nrow(dd_ts), function(x) mutate(dd_ts$data[[x]], time = 1:n()))
```

Calculate permutation entropy across time series

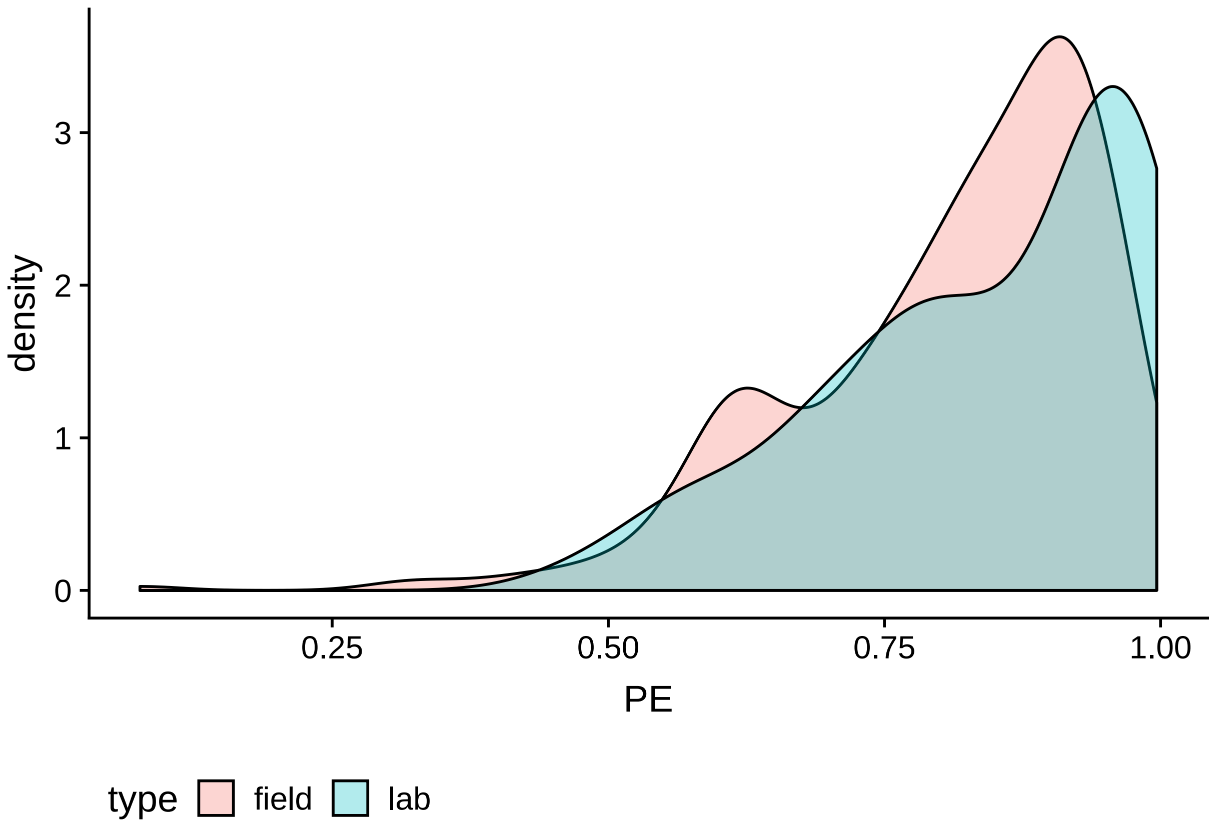
We calculated the weighted permutation entropy for all the time series using a word length of 3 and tau of 1. Other values for word length and tau did not change the results. Moreover, using other tie breaking methods (e.g. average or random) did not change the results.

```
dd_ts <- dd_ts %>% mutate(PE = map_dbl(data, ~ tryCatch(PE(.$abundance, weighted=T, tau=1, word_length 

# uncomment to use other tie breaking methods when calculating PE
#dd_ts <- dd_ts %>% mutate(PE_average = map_dbl(data, ~ PE(.$abundance, weighted=T, tau=1, word_length 
#dd_ts <- dd_ts %>% mutate(PE_random = map_dbl(data, ~ PE(.$abundance, weighted=T, tau=1, word_length =
```

Distribution of PE values over lab and field data

The following graph shows the distribution of permutation entropy values for field (GPDD) and lab studies.

```r
ggplot(data=dd_ts, aes(x=PE, group=type, fill=type)) + geom_density(alpha=.3) + theme(legend.position =
```



Forecasting via simplex and s-map projection (using a split dataset)

We forecasted each individual time series using the Empirical Dynamic Modelling (EDM) We divided the time series into a training and a test dataset. 2/3 of the time series was used for training, and the remaining 1/3 for testing.

The EDM approach consists of two steps to determine the embedding dimension E and the tuning parameter theta that determines the degree of non-linearity of the forecast. The embedding dimension determines the number of lags used for reconstructing the state-space and the tuning parameter theta determines how non-linear the system is (i.e. which of the neighboring points in the state-space are considered for forecasting).

We first fitted the simplex projection to identify the best embedding dimension E on the training data. Usually dimensions between one and ten are tested and the dimension with the highest forecast skill using a leave-one-out-cross validation is chosen. In the second step, we fitted the S-maps with the determined dimension E to the training data and tested the tuning parameter theta. Theta varied in 18 steps between 0 and 8 (with log scaled intervals). Finally, we tested the final EDM model on test part of each time series and recorded the forecasting skill.

```r
# define how much data you want to use
# by division -> 2 = 50% split, 4 = 25% etc.
split <- 3
s_maps <- list(NULL)

pdf(here::here("Figs", "EDM_fitting.pdf"))
```

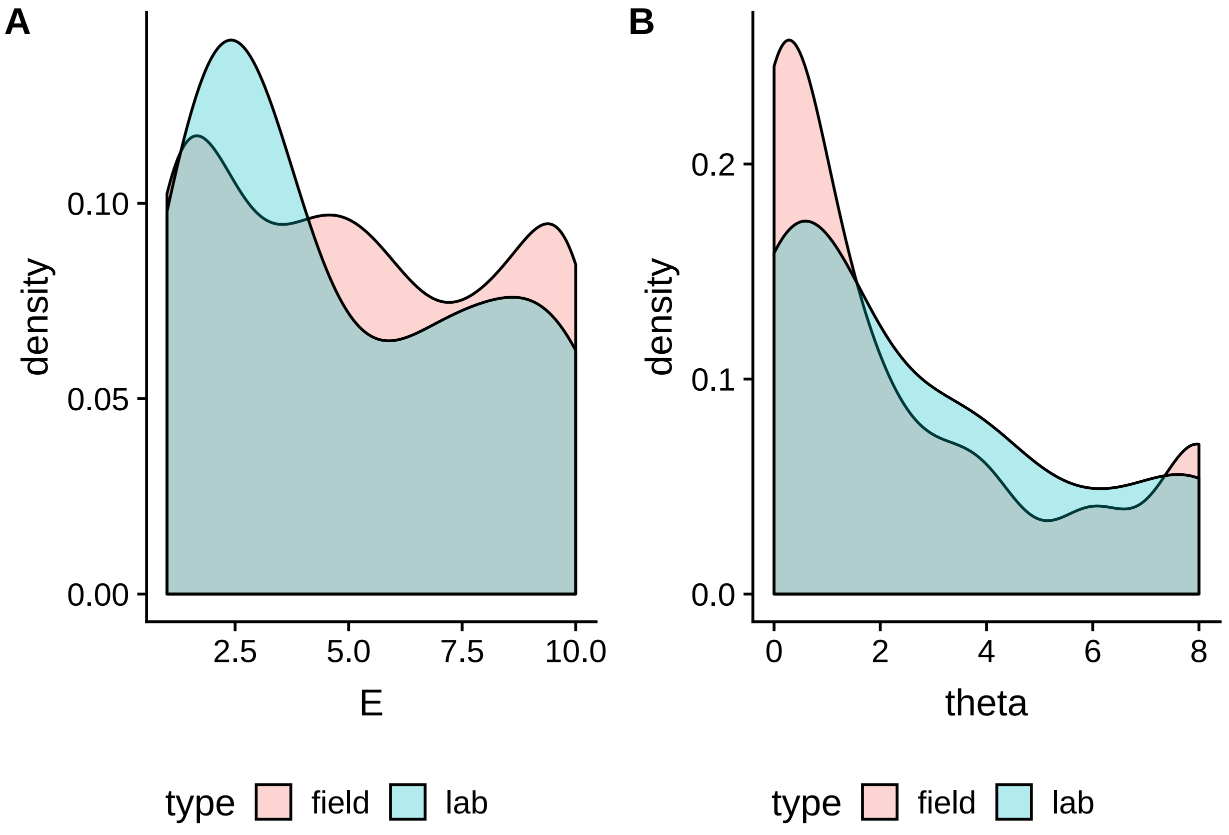
```
for (i in 1:nrow(dd_ts)){

  layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))

  ts_select <- as.data.frame(dd_ts$data[[i]])
  # define steps ahead forecast (based on split)
  step_ahead <- as.integer(nrow(ts_select)/split)

  simplex_fit <- simplex(ts_select$abundance_sc, lib = c(1, (length(ts_select$abundance_sc)-step_ahead)

  E_hat_rho <- simplex_fit[which(max(simplex_fit$rho, na.rm = T)==simplex_fit$rho)[1], ]$E

  plot(simplex_fit$E, simplex_fit$rho, type="l", xlab = "Embedding dimension (E)", ylab="Forecast skill
  abline(v=E_hat_rho, col="red", lty=2)

  smap_fit <- s_map(ts_select$abundance_sc, lib = c(1, (length(ts_select$abundance_sc)-step_ahead)),  s

  theta_hat_rho <- smap_fit[which(max(smap_fit$rho, na.rm = T)==smap_fit$rho), ]$theta

  plot(smap_fit$theta, smap_fit$rho, type="l", xlab = "Theta", ylab="Forecast skill (rho)")
  abline(v=theta_hat_rho, col="red", lty=2)


  smap_fit <- s_map(ts_select$abundance_sc, lib = c(1, (length(ts_select$abundance_sc)-step_ahead)),  s

  smap_pred  <- s_map(ts_select$abundance_sc, lib = c(1, (length(ts_select$abundance_sc)-step_ahead)),

  s_maps[[i]] <- smap_pred

    mtext(paste0("MainID = ", dd_ts$MainID[i]), side=3, line=-2, cex=1, outer=T)

   {plot(ts_select$time, ts_select$abundance_sc, type="l", xlab = "Time", ylab="Abundance"
         , ylim = range(min(smap_pred$model_output[[1]]$pred, ts_select$abundance_sc, na.rm=T), max(smap
     points(smap_pred$model_output[[1]]$time, smap_pred$model_output[[1]]$pred, col=2,ylim =range(min(sm
      lines(smap_pred$model_output[[1]]$time, smap_pred$model_output[[1]]$pred, col=2,ylim =range(min(sm

     }

  layout(matrix(c(1,1), 1, 1, byrow = TRUE))

}


dd_ts <- cbind(dd_ts,bind_rows(s_maps))

dev.off()
```

```
## pdf
##   2
```

Extract and plot information about the fitted EDM models (embedding dimension and theta)

Both embedding dimension and theta are additional features of the time series and may have explanatory value. We therefore added these metrics to our data.frame to control for the parameters in subsequent analyses. The two following plots show the distributions of D and theta across the two databases.

```
E_type <- ggplot(data=dd_ts, aes(x=E, group=type, fill=type)) + geom_density(position="identity", alpha
theta_type <- ggplot(data=dd_ts, aes(x=theta, group=type, fill=type)) + geom_density(position="identity
cowplot::plot_grid(E_type, theta_type, labels = c("A", "B"))
```



Calculate forecast error metrics

To evaluate the forecasting error, we used the normalised RMSE (nRMSE) as an appropriate measure to compare across time series.

```
dd_ts$nrmse <- dd_ts$rmse/unlist(lapply(1:nrow(dd_ts), function(x) {max(dd_ts$data[[x]]$abundance_sc, na
# center PE and add to analysis to compare types at mean PE range rather than at zero
dd_ts$PE_center <- scale(dd_ts$PE, center=T, scale=F)
```

Descriptive information about time series included

```
dd_ts %>% skim(PE, nrmse, N, ties_prop, zero_prop, E, theta) %>% skimr::kable()
```

Skim summary statistics
n obs: 461
n variables: 32

Variable type: integer

variable	missing	complete	n	mean	sd	p0	p25	p50	p75	p100	hist
E	0	461	461	5.18	3.1	1	2	5	8	10	
N	0	461	461	62.61	38.04	30	39	50	63	197	

Variable type: numeric

variable	missing	complete	n	mean	sd	p0	p25	p50	p75	p100	hist
nrmse	0	461	461	0.21	0.14	9.3e-06	0.12	0.19	0.27	1.37	
PE	0	461	461	0.81	0.14	0.076	0.73	0.84	0.92	1	
theta	0	461	461	2.39	2.78	0	0	1	4	8	
ties_prop	0	461	461	0.018	0.031	0	0	0	0.025	0.15	
zero_prop	0	461	461	0.014	0.044	0	0	0	0	0.3	

Is there a relationship between PE and FE? Which of the time series covariates explain deviations from the relationship?

For analysis and plotting we centered the PE values to test for differences in intercept in the mean range of PE values rather than at the theoretical limit of 0 (perfect intrinsic predictability).

We find a relationship between PE and nRSME. Forecast error increases with time series complexity (i.e. increasing PE). The analysis using a mixed model with PE and type as fixed effect and time series source as random effect shows that the PE slope is significantly different from zero. Neither type (i.e. field versus lab time series) nor the interaction with PE is significant. Additionally, we were interested in whether time series properties such as length or non-linearity can explain deviations from the general relationship. We found that in addition to PE, time series length (N), the proportion of zeros (zero_prop), time series nonlinearity (theta) and dimensionality (E) had significant effects on nRMSE.

```
dd_ts$nrmse_sqrt <- sqrt(dd_ts$nrmse)
dd_ts$rmse_sqrt <- sqrt(dd_ts$rmse)
dd_ts$PE_sqrt <- sqrt(dd_ts$PE)
# simplify data frame before model fitting
dd_ts_simple <- select(dd_ts, c(nrmse_sqrt, rmse_sqrt, PE, type, N, zero_prop, ties_prop, number_of_gaps

lmer_nrmse_all <- lmer(nrmse_sqrt ~ PE + type +  PE:type + N + zero_prop + ties_prop + E + theta + (1|so
knitr::kable(papeR::prettify(summary(lmer_nrmse_all)))
```

	Estimate	CI (lower)	CI (upper)	Std. Error	df	t value	Pr(>|t|)	
(Intercept)	0.0893415	0.0089560	0.1655384	0.0401782	365.56573	2.2236308	0.027	*
PE	0.4795813	0.3944650	0.5634972	0.0434555	451.66668	11.0361599	<0.001	***
type: lab	-0.0750952	-0.2920594	0.1486706	0.1141145	287.69581	-0.6580690	0.511	
N	-0.0016962	-0.0021239	-0.0012559	0.0002228	50.67777	-7.6139591	<0.001	***
zero_prop	0.4062110	0.0825512	0.7465093	0.1705835	392.89017	2.3813026	0.018	*
ties_prop	-0.3344360	-0.7657524	0.0930886	0.2221220	380.67666	-1.5056412	0.133	
E	0.0087700	0.0052181	0.0124675	0.0018536	447.91317	4.7312298	<0.001	***
theta	0.0113175	0.0073317	0.0154599	0.0020807	449.50360	5.4393905	<0.001	***
PE:typelab	0.1006060	-0.1729678	0.3632677	0.1387655	210.61495	0.7250073	0.469	

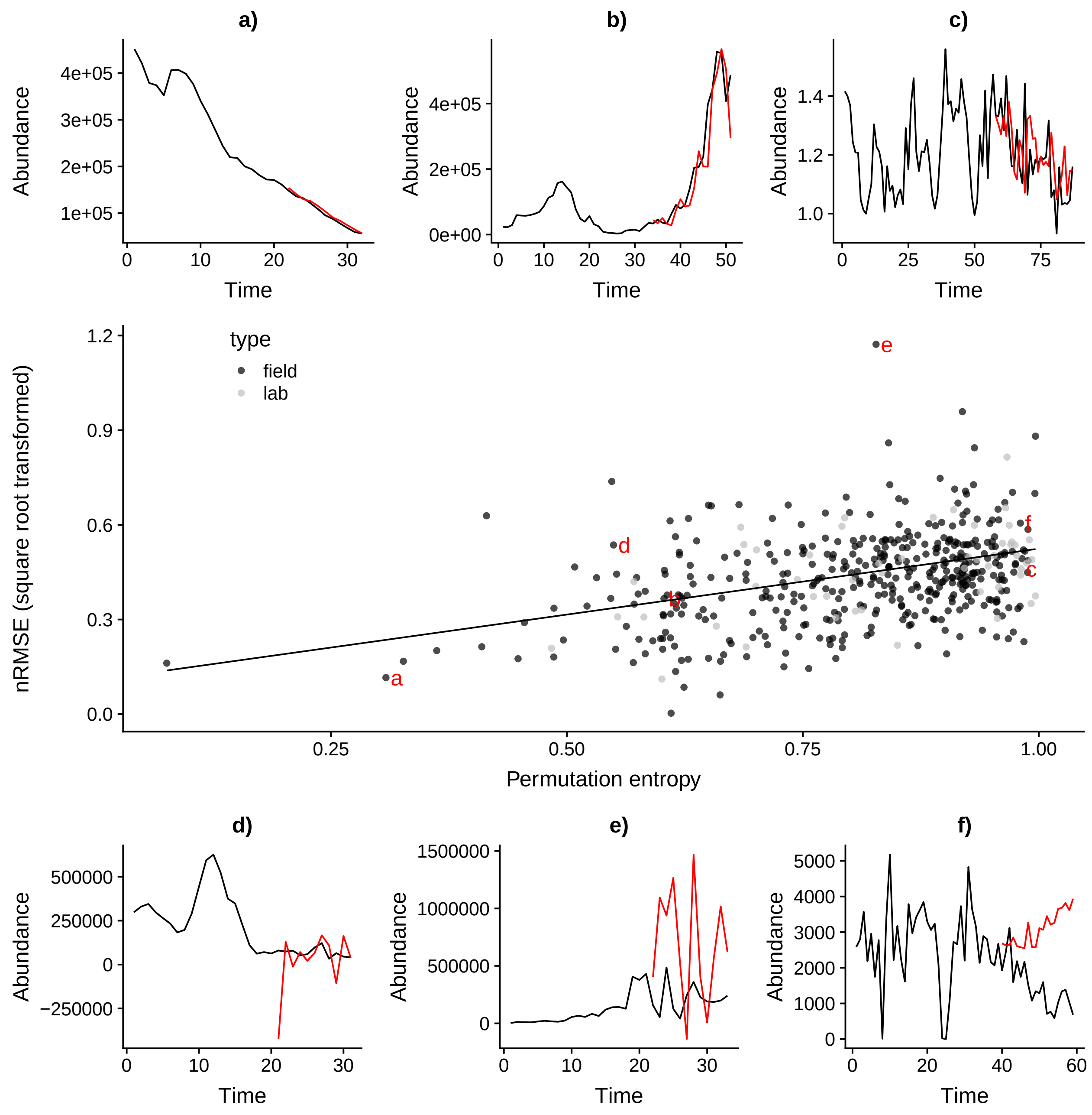
Calculate partial R squared to assess explanatory power of individual predictors

```
partial_df <- r2glmm::r2beta(lmer_nrmse_all, partial = T, method = "nsj", data = NULL)
partial_df %>% select(Effect, Rsq, lower.CL, upper.CL) %>% mutate_if(is.numeric, round, digits=3) %>% kn
```

Effect	Rsq	lower.CL	upper.CL
Model	0.347	0.290	0.417
PE	0.208	0.148	0.273
N	0.181	0.124	0.245
theta	0.057	0.023	0.104
E	0.043	0.014	0.086
zero_prop	0.013	0.001	0.040
ties_prop	0.005	0.000	0.026
PE:typelab	0.001	0.000	0.016
typelab	0.001	0.000	0.015

Plot PE vs FE relationship and highlight its potential to diagnose specific time series

```
dd_ts$label <- ifelse(dd_ts$MainID == 1950, "a",
                                     ifelse(dd_ts$MainID == 1986, "e",
                                           ifelse(dd_ts$MainID == 1936, "d",
                                                         ifelse(dd_ts$MainID == 278, "b",
                                                                       ifelse(dd_ts$MainID == 4528, "c"
                                                                                     ifelse(dd_ts$MainI

gg_panels <- function(x){
label_id <- which(dd_ts$label==x)
plt <- ggplot() + geom_line(aes(y=dd_ts$data[[label_id]]$abundance, x=dd_ts$data[[label_id]]$time)) +
  geom_line(aes(y=dd_ts$model_output[[label_id]]$pred, x=dd_ts$model_output[[label_id]]$time), colour="
}


p1 <-  gg_panels("a")
p2 <- gg_panels("b")
p3 <-  gg_panels("c")
p5 <- gg_panels("d")
p6 <-  gg_panels("e")
p7 <-  gg_panels("f")

# create prediction for PE
dd_ts2 <- dd_ts[,c("nrmse", "PE", "source", "type", "label", "N", "zero_prop", "ties_prop", "E", "theta
dd_ts2$nrmse_sqrt <- sqrt(dd_ts2$nrmse)
dd_ts2$type <- as.factor(dd_ts2$type)
lme_fit <- nlme::lme(nrmse_sqrt ~ PE * type + + N +  zero_prop + ties_prop + E + theta, random = ~ 1 | 
dat <-ggeffects::ggaverage(lme_fit, terms = c("PE"), type = c("fe"))
dat$label <- NA

label <- dd_ts$label

p4 <- ggplot(data=dd_ts, aes(x=PE, y=nrmse_sqrt, label=label)) +
  geom_point(aes(colour=type), alpha=.7) +
  xlab("Permutation entropy") + ylab("nRMSE (square root transformed)") +
  scale_colour_manual(values = c("black", "grey")) +
  theme(legend.position = c(.1, .9)) +
  geom_text(colour="red", hjust = 0, nudge_x = 0.005, size=5) +
  geom_line(data=dat, aes(x=x, y=(predicted), label=NULL))


fig4 <- (p1 | p2 | p3 ) / {p4} / (p5 | p6 | p7)  + plot_layout(ncol = 1, heights = c(1, 2, 1), widths =
fig4
```



```r
png(here::here("Figs", "Figure_4.png"), width=2400, height=2400, res=300)
print(fig4)
dev.off()
```

```
## pdf
##   2
```