

Data Exploration

2022-06-28

Packages

```
## -- Attaching packages ----- tidyverse 1.3.1 --

## v ggplot2 3.3.6      v purrr  0.3.4
## v tibble  3.1.7      v dplyr  1.0.9
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: Rcpp

## Loading 'brms' package (version 2.17.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').

##
## Attaching package: 'brms'

## The following object is masked from 'package:stats':
##
##   ar

##
## Attaching package: 'tidybayes'

## The following objects are masked from 'package:brms':
##
##   dstudent_t, pstudent_t, qstudent_t, rstudent_t
```

Loading data

```
data <- read_csv('turf_prod_val.csv') |>
  filter(!is.na(depth))
```

```
## Rows: 214 Columns: 4
## -- Column specification -----
## Delimiter: ","
## chr (3): mean_prod (g C m-2 day-1), obs, Ref
## dbl (1): depth
##
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.
```

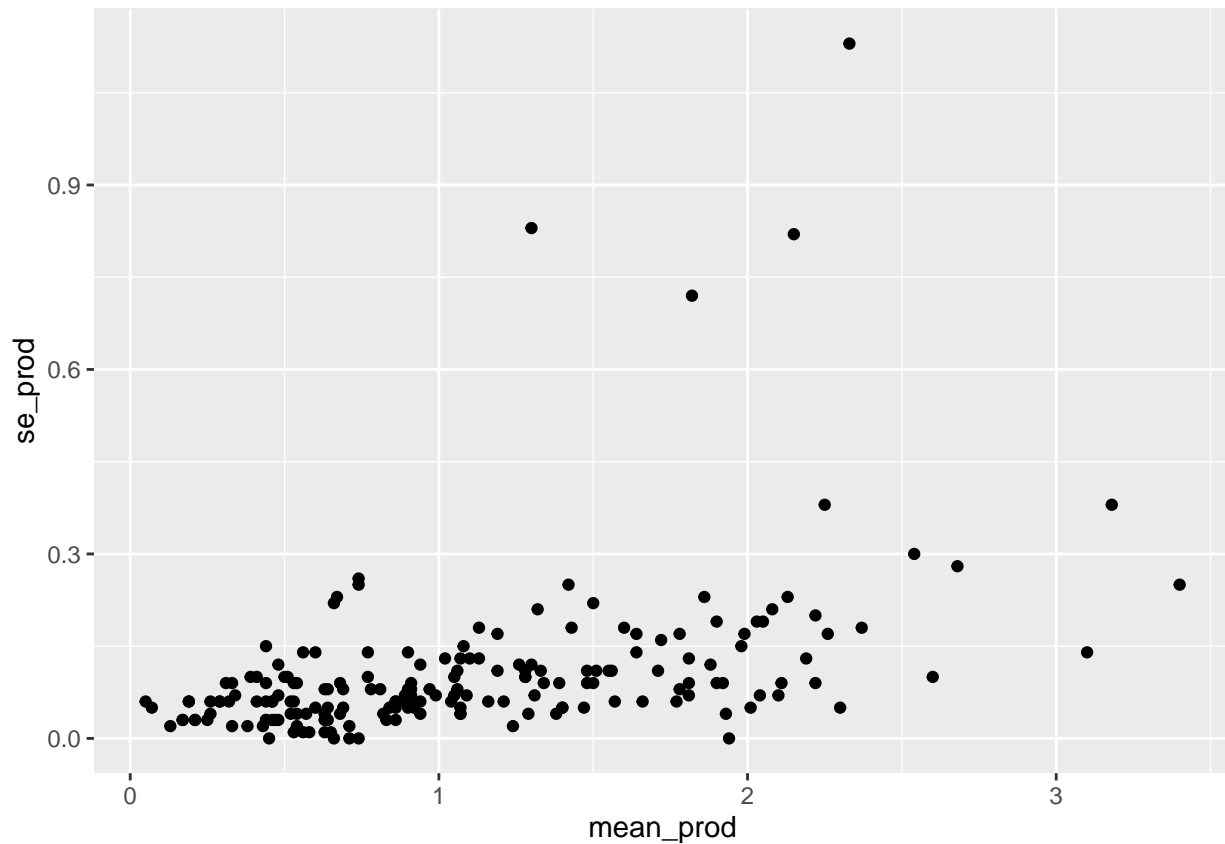
```
unit <- names(data)[1]
names(data)[1] <- 'prod'

x <- str_split(data$prod, '\\xb1')
data$mean_prod <- as.numeric(substr(unlist(lapply(x, function(x)x[1])),1,4))
data$se_prod <- as.numeric(substr(unlist(lapply(x, function(x)x[2])),2,5))
```

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

```
data <- data %>% filter(mean_prod != 0)
```

Now, for the data points we do not have se, determine one from the relationship between mean and se:



```
## Predicting variability using the mean for 14 points and also adjusting McClure 2019, ## which is a ci
and not se, and also adding a small non zero value to all zero se
```

```
mod_se <- lm(se_prod ~ mean_prod, data=data)

data[is.na(data$se_prod), 'se_prod'] <- round(predict(mod_se, newdata=data[is.na(data$se_prod),]), 2)

data[data$Ref == 'McClure 2019', 'se_prod'] <- data[data$Ref == 'McClure 2019', 'se_prod'] / 1.96

nzmin <- function(x) min(x[x>0])

data[data$se_prod == 0, 'se_prod'] <- nzmin(data$se_prod)
```

And finally modelling

```
## Compiling Stan program...
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG -I"/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/StanHeaders/include"
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/StanHeaders/include/StanHeaders/math_functions.hpp:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/Core:96:
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/src/Core/Matrix.h:13:1: error: namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/src/Core/Matrix.h:13:1: error: namespace Eigen {
## namespace Eigen {
## ^
## ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/StanHeaders/include/StanHeaders/math_functions.hpp:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/Core:96:
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/Core:96:
## #include <complex>
## ^~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL '10f63a45fd17e5d9181b383b6c1bd659' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 7.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.74 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [ 0%] (Warmup)
## Chain 1: Iteration:   500 / 5000 [ 10%] (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%] (Warmup)
```

```

## Chain 1: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%] (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%] (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 1:
## Chain 1: Elapsed Time: 0.666491 seconds (Warm-up)
## Chain 1: 0.739549 seconds (Sampling)
## Chain 1: 1.40604 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '10f63a45fd17e5d9181b383b6c1bd659' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.31 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration: 1 / 5000 [ 0%] (Warmup)
## Chain 2: Iteration: 500 / 5000 [ 10%] (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%] (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%] (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%] (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 2:
## Chain 2: Elapsed Time: 0.683402 seconds (Warm-up)
## Chain 2: 0.654462 seconds (Sampling)
## Chain 2: 1.33786 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '10f63a45fd17e5d9181b383b6c1bd659' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration: 1 / 5000 [ 0%] (Warmup)
## Chain 3: Iteration: 500 / 5000 [ 10%] (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%] (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%] (Sampling)

```

```

## Chain 3: Iteration: 3000 / 5000 [ 60%] (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 3:
## Chain 3: Elapsed Time: 0.695507 seconds (Warm-up)
## Chain 3: 0.712017 seconds (Sampling)
## Chain 3: 1.40752 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '10f63a45fd17e5d9181b383b6c1bd659' NOW (CHAIN 4).
## Chain 4: Rejecting initial value:
## Chain 4: Gradient evaluated at the initial value is not finite.
## Chain 4: Stan can't start sampling from this initial value.
## Chain 4:
## Chain 4: Gradient evaluation took 3.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.31 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration: 1 / 5000 [ 0%] (Warmup)
## Chain 4: Iteration: 500 / 5000 [ 10%] (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%] (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%] (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%] (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%] (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%] (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%] (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%] (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%] (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%] (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%] (Sampling)
## Chain 4:
## Chain 4: Elapsed Time: 0.716181 seconds (Warm-up)
## Chain 4: 0.637829 seconds (Sampling)
## Chain 4: 1.35401 seconds (Total)
## Chain 4:

```

Loading and tidying data to predict for

```

## Constrained max depth of site to 15m

pred_depth <- read.csv('moorea_depth.csv') %>%
  mutate(site=tolower(site)) %>%
  group_by(site) %>%
  mutate(depth=if_else(depth < -15,-15, depth)*-1) %>%
  slice_max(depth)

pred_data <- read.csv('moorea_benthos.csv') %>%
  mutate(site=gsub('\\s','_',tolower(site)))

```

Filtering and manipulating the time series for the correct categories

```
ts_data <- left_join(pred_data, pred_depth, by='site') %>%
  filter(Habitat=='Outer slope' & Season=='Mar') %>%
  mutate(subs_group=case_when(
    Substrate == 'Dead coral' ~ 'algal_turf',
    Substrate == 'Stegastes Turf' ~ 'algal_turf',
    Substrate == 'Rubble' ~ 'algal_turf',
    Substrate == 'Pavement' ~ 'algal_turf',
    Substrate == 'Macroalgae' ~ 'macroalgae',
    Substrate == 'Turbinaria' ~ 'macroalgae',
    TRUE ~ Substrate,
  )) %>%
  filter(subs_group %in% c('algal_turf', 'macroalgae')) %>%
  group_by(Year, site, Transect, lat, long, depth, subs_group) %>%
  summarise(prop=sum(proportion), .groups='drop_last') %>%
  pivot_wider(names_from=subs_group, values_from=prop, values_fill=0)
```

Now, how about trying to predict benthic reef productivity
by merging area specific turf productivity with turf cover

```
pred_val <- posterior_epred(brmod,
  newdata=ts_data %>% mutate(se_prod=0.1), ndraws=1000)

dim(ts_data)

## [1] 507 8

dim(pred_val)

## [1] 1000 507

tot_turf_prod <- (ts_data$algal_turf * t(pred_val)) * 10000
## in g C ha day-1

# aggregating

fts_data <- cbind(ts_data,
  turf_prod_mean=apply(tot_turf_prod, 1, median),
  turf_prod_lhdi=apply(tot_turf_prod, 1, function(x) median_hdci(x)$ymin),
  turf_prod_uhdi=apply(tot_turf_prod, 1, function(x) median_hdci(x)$ymax))

## in g C ha-1 day-1
```

Now doing some plots

```
ggplot(data=fts_data) +  
  geom_line(aes(x=Year,y=turf_prod_mean,group=interaction(Transect,site))) +  
  geom_smooth(aes(x=Year,y=turf_prod_mean), colour='orange', alpha=0.8) +  
  scale_x_continuous(breaks=c(2005,2009,2013,2017)) +  
  ylab(expression(Algal~turf~productivity~'('g~C~ha^-1*day^-1*')')) +  
  theme_minimal()
```

```
## 'geom_smooth()' using method = 'loess' and formula 'y ~ x'
```

