

# Turf and macroalgae productivity on coral reefs: a modelling exercise in Moorea

2022-06-28

## Packages

```
library(tidyverse)
library(brms)
library(tidybayes)
library(patchwork)
```

**Loading data compiled and reworked by Tebbett and Bellwood 2021 Mar Env Res.**

Depth data were added manually by looking at each individual study

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

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

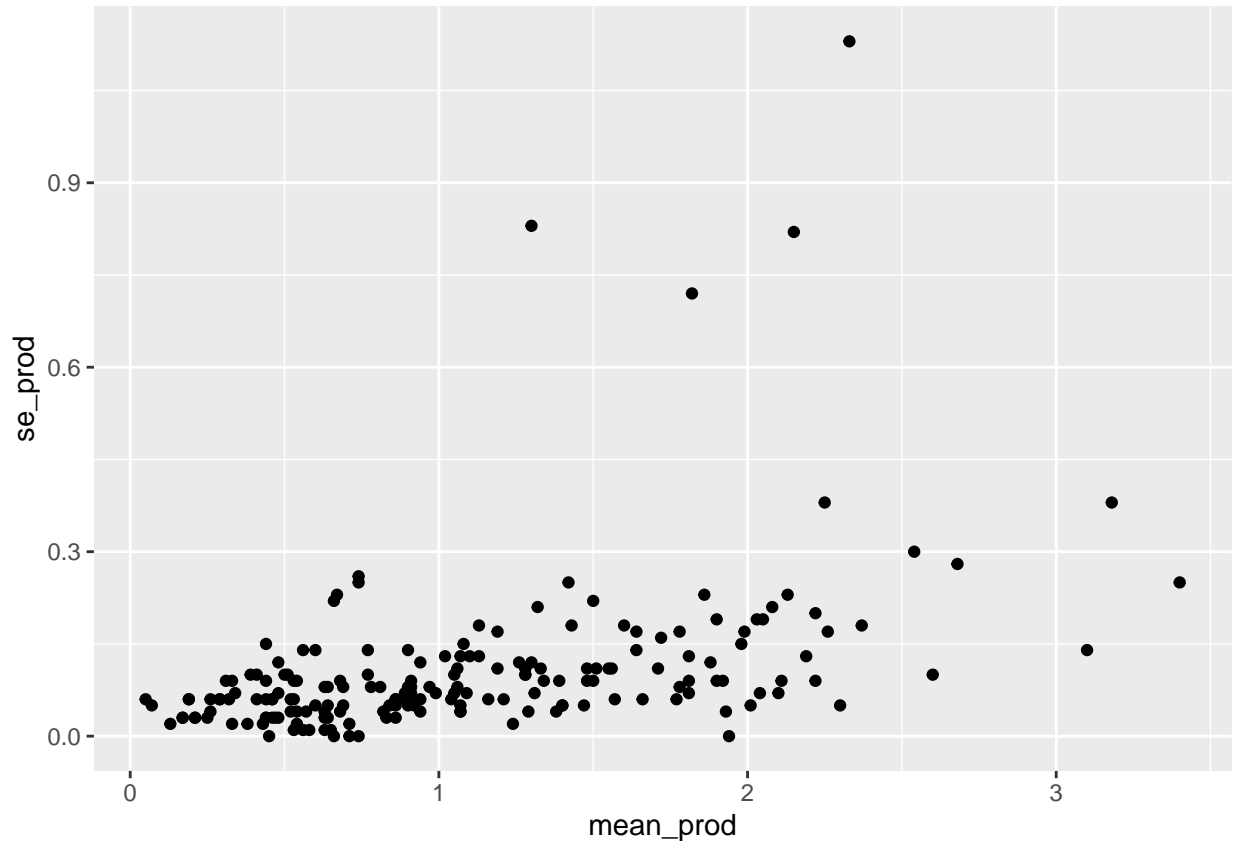
x <- str_split(data$prod, '\\xb1')
data$mean_prod <- as.numeric(substr(unlist(lapply(x, function(x)x[1])),1,4))
data$se_prod <- as.numeric(substr(unlist(lapply(x, function(x)x[2])),2,5))
```

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

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

Now, for the data points we do not have standard error values,  
determine them from the relationship between mean and se:

```
ggplot(data %>% filter(!is.na(se_prod))) +
  geom_point(aes(x=mean_prod,y=se_prod))
```



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)
## Model sucks, but better than to consider zero

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 algal turf productivity using a meta-analysis

Bayesian model with depth as the only predictor

```
pri <- get_prior(mean_prod | se(se_prod, sigma=TRUE) ~ 1 + log(depth),
  data = data,
  family=skew_normal())
```



```

## 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: 0.655896 seconds (Warm-up)
## Chain 1: 0.706782 seconds (Sampling)
## Chain 1: 1.36268 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '10f63a45fd17e5d9181b383b6c1bd659' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.33 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: 0.653188 seconds (Warm-up)
## Chain 2: 0.765872 seconds (Sampling)
## Chain 2: 1.41906 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '10f63a45fd17e5d9181b383b6c1bd659' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.34 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: 0.655523 seconds (Warm-up)
## Chain 3: 0.640786 seconds (Sampling)
## Chain 3: 1.29631 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '10f63a45fd17e5d9181b383b6c1bd659' NOW (CHAIN 4).
## 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: 0.656984 seconds (Warm-up)
## Chain 4: 0.621292 seconds (Sampling)
## Chain 4: 1.27828 seconds (Total)
## Chain 4:

```

```
saveRDS(brmod, 'turf_prod_brms.RDS')
```

## 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 categories of interest.

For Moorea, at the moment, these could be algal turfs, halimeda and macroalgae

```
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',
    Substrate == 'Halimeda' ~ 'halimeda',
    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 predicted

using the data compiled by Tebbett and Bellwood and turf cover?

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

tot_turf_prod <- ((ts_data$algal_turf * t(pred_val)) * 10000)/1000
## in kg C ha day-1

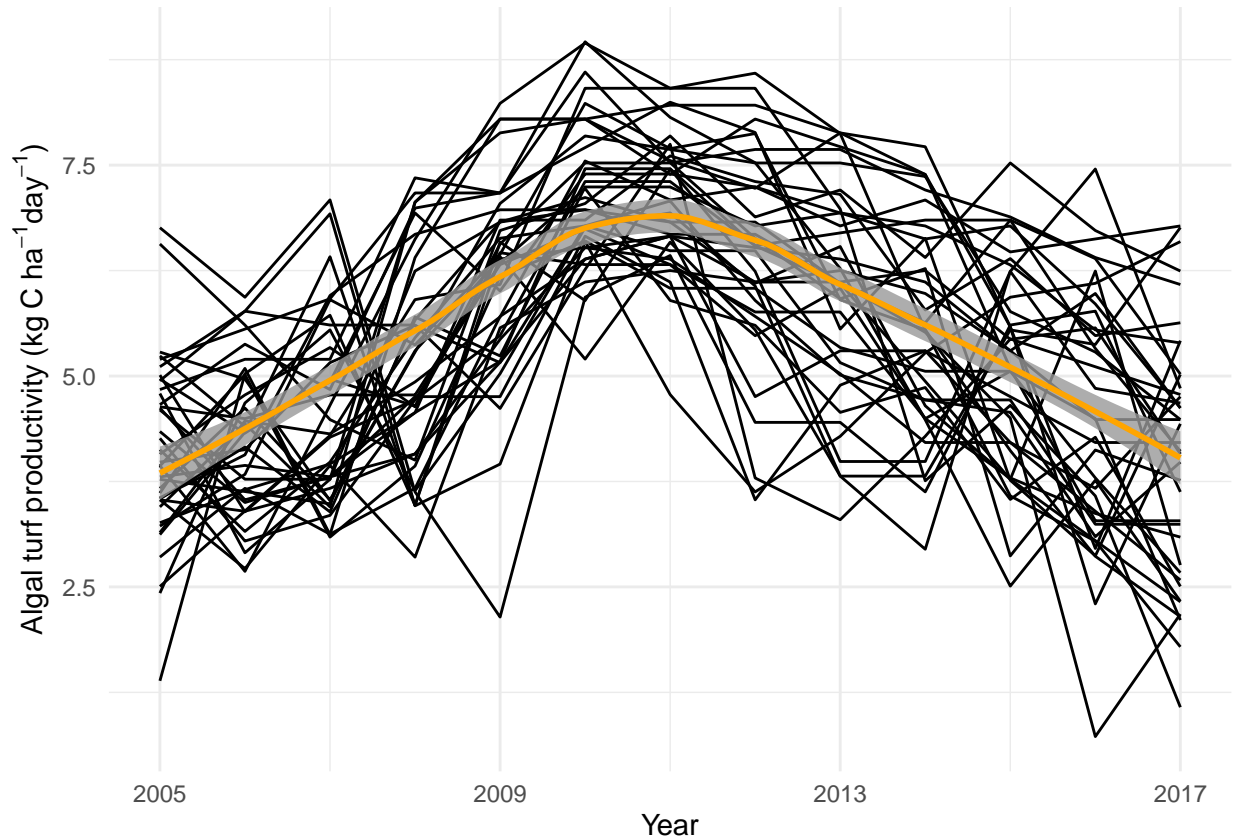
# aggregating

fts_data <- cbind(ts_data,
  turf_prod_kghaday_med=apply(tot_turf_prod, 1, median),
  turf_prod_kghaday_lhd=apply(tot_turf_prod, 1, function(x) median_hdci(x)$ymin),
  turf_prod_kghaday_uhd=apply(tot_turf_prod, 1, function(x) median_hdci(x)$ymax))
## in kg C ha-1 day-1
```

Now plotting

First the time series of turf productivity over time in Moorea

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



And using Duarte's et al 2022's data to explore a model for macroalgae

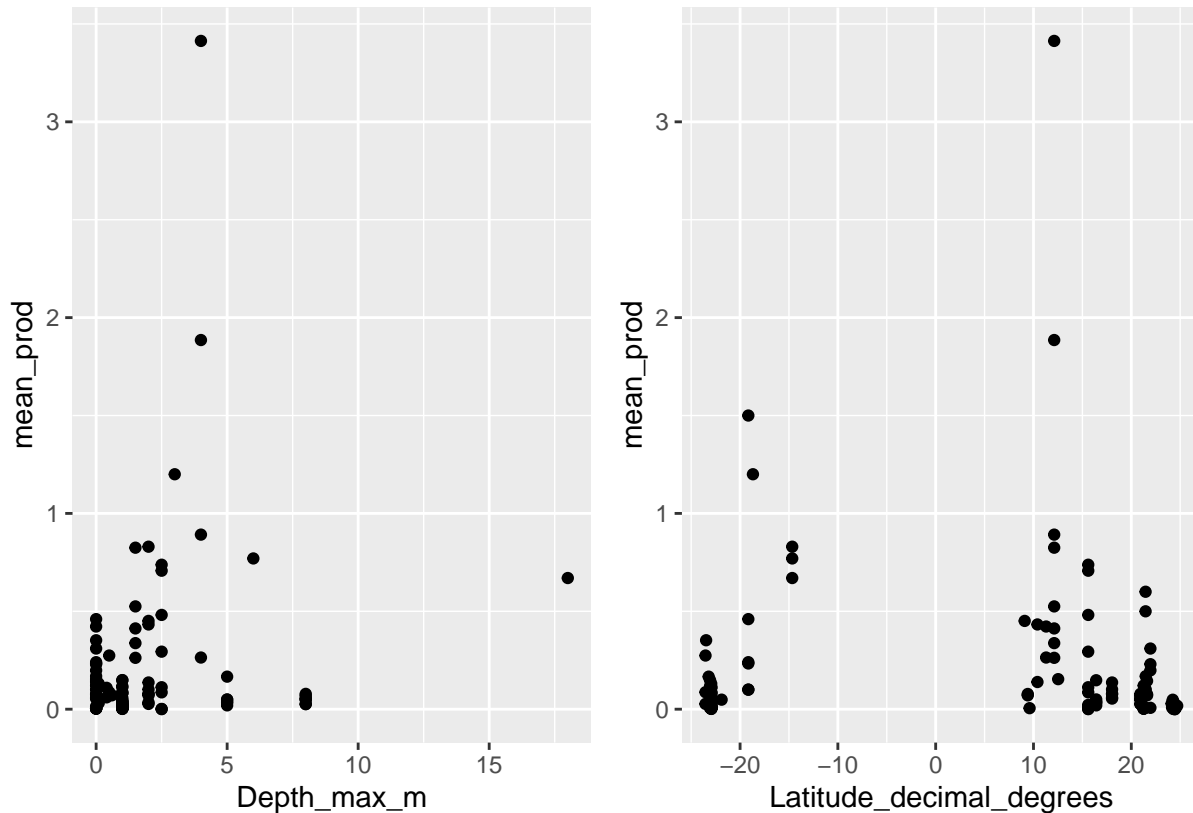
It could be possible to do the same for Halimeda, but there are less values

Maybe return to this possibility later?

```
dmachal <- read.csv('macroalgae_npp_dataset.csv') %>%
  filter(Habitat_category_Duarte_etal_2021 %in% c('Coral Reef Algae','Subtidal brown algae') & abs(Lati
  mutate(mean_prod = ((Avg_NPP_kg_C_m2_y) * 1000) / 365,
    se_prod = ((stdev_NPP_kg_C_m2_y) * 1000) / 365)
# in g C m-2 day-1
```

```
## Having a look at the data from a few perspectives
```

```
p1 <- ggplot(dmachal) +  
  geom_point(aes(x=Depth_max_m, y=mean_prod))  
  
p2 <- ggplot(dmachal) +  
  geom_point(aes(x=Latitude_decimal_degrees, y=mean_prod))  
  
p1 + p2
```

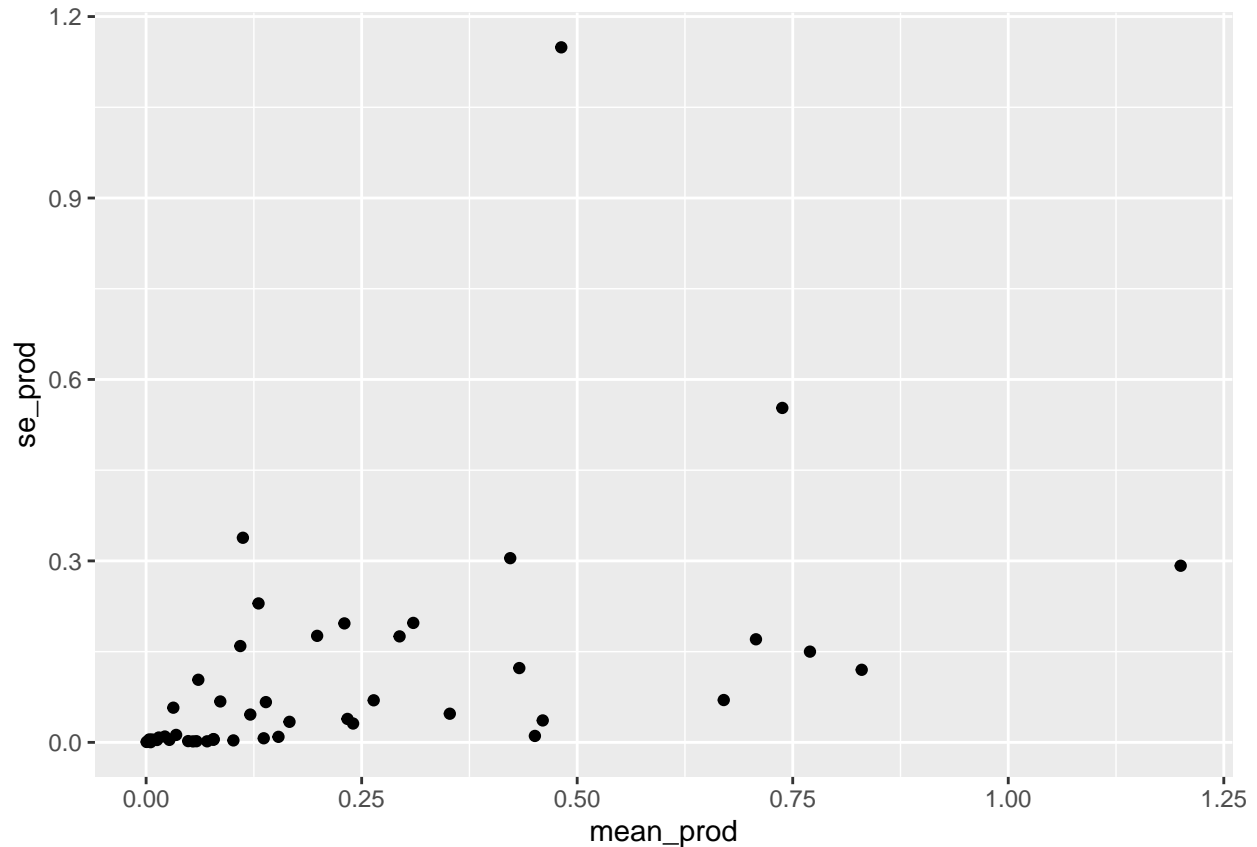


```
## Get outta here, nothing useful to predict so just including variability in a meta-analysis
```

Can we predict standard error values from a relationship between mean and se as we did for algal turfs?

```
ggplot(dmachal %>% filter(!is.na(se_prod))) +  
  geom_point(aes(x=mean_prod, y=se_prod))
```





```
mod_se2 <- lm(se_prod ~ mean_prod, data=dmachal)
## Again, Model sucks, but better than to consider zero

dmachal[is.na(dmachal$se_prod), 'se_prod'] <- round(predict(mod_se2, newdata=dmachal[is.na(dmachal$se_pr
```

And finally modelling macroalgae productivity using a meta-analysis

Bayesian model with depth as the only predictor

```
## 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/Resources/include"
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/abs.hpp:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/Core:1:
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/src/Core/Matrix.h:1:
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/src/Core/Matrix.h:1:
## namespace Eigen {
```

```

##          ^
##          ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/StanHeader:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen:
## /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 '65e53fdc92b4dc47b3a716bb70bf19ac' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.41 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: 0.805602 seconds (Warm-up)
## Chain 1:                0.624083 seconds (Sampling)
## Chain 1:                1.42969 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '65e53fdc92b4dc47b3a716bb70bf19ac' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.792401 seconds (Warm-up)
## Chain 2: 0.524499 seconds (Sampling)
## Chain 2: 1.3169 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '65e53fdc92b4dc47b3a716bb70bf19ac' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.9e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.88533 seconds (Warm-up)
## Chain 3: 0.504262 seconds (Sampling)
## Chain 3: 1.38959 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '65e53fdc92b4dc47b3a716bb70bf19ac' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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: 0.755897 seconds (Warm-up)
## Chain 4: 0.623869 seconds (Sampling)
## Chain 4: 1.37977 seconds (Total)
## Chain 4:
```

Now, how about trying to predict benthic macroalgae productivity by merging area specific turf productivity predicted using the data compiled by Duarte et al 2022 and macroalgae cover?

```
pred_val2 <- posterior_epred(brmod2,
  newdata=ts_data %>% mutate(se_prod=0.1), ndraws=1000)

tot_macr_prod <- ((ts_data$macroalgae * t(pred_val2)) * 10000)/1000
## in kg C ha day-1

# aggregating

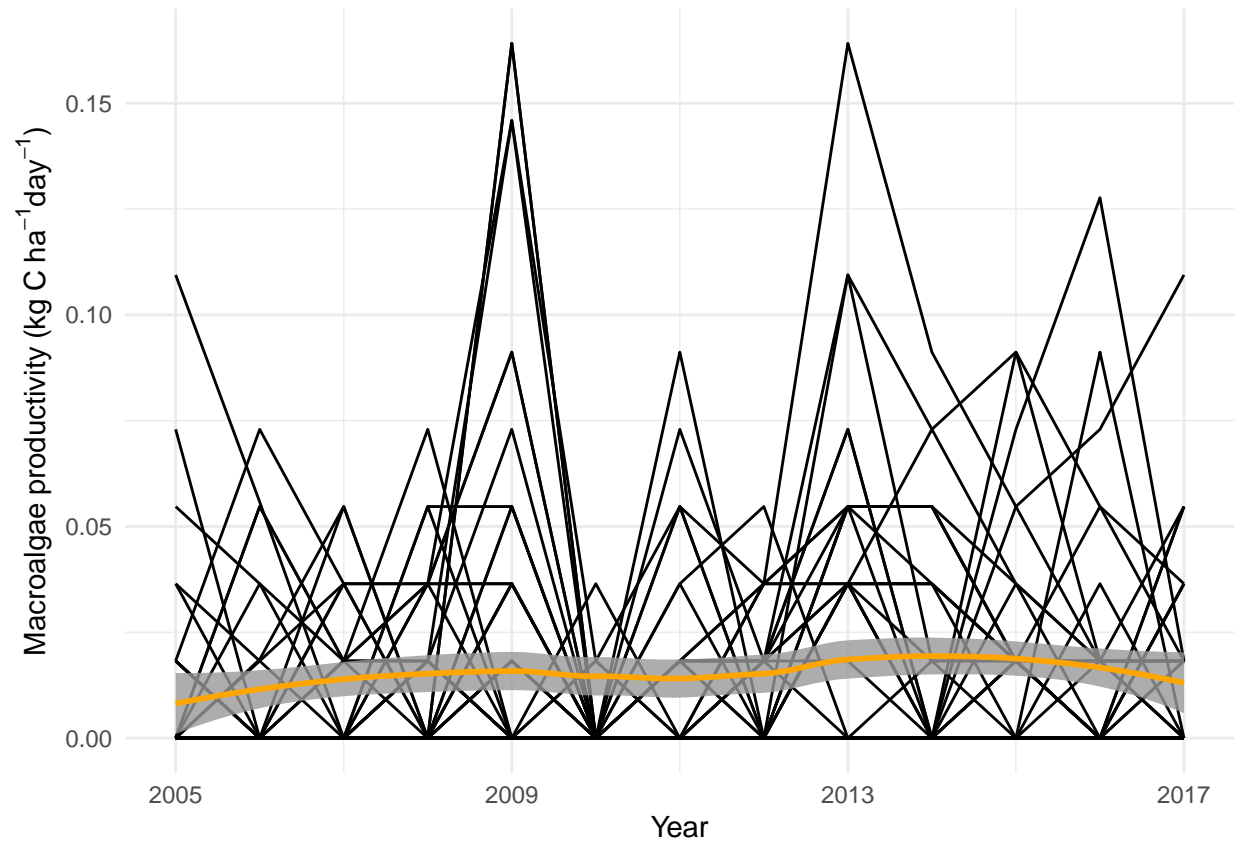
fts_data <- cbind(fts_data,
  macr_prod_kghaday_med=apply(tot_macr_prod,1,median),
  macr_prod_kghaday_lhd=apply(tot_macr_prod,1,function(x) median_hdci(x)$ymin),
  macr_prod_kghaday_uhd=apply(tot_macr_prod,1,function(x) median_hdci(x)$ymax))

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

Now plotting

First the time series of macroalgae productivity over time in Moorea

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



Saving the final estimates fro Moorea

```
write.csv(fts_data, 'Moorea_turf_macr_prod.csv', row.names=FALSE)
```