Estimate Robustness of Real Networks

Andrew L Jackson, Jean-Francois Arnoldi, Sam P. Ross & Ian Donohue

30 March 2020

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(furrr) # for parallel implmenetation of purrr
```

```
## Loading required package: future
```

```
library(latex2exp) # for latex style equations in figures (currently not used)
```

Import the Web of Life data

Loop over the datasets and import them into a list.

```
# get all file names in the folder data/
all_files <- dir("data/")

# find the indices that are README or reference files
remove_these <- c(grep("README", all_files), grep("references", all_files))

# remove them from the vector of names and prepend the data/ folder address
all_web_files <- paste0("data/", all_files[-remove_these])

# import all the webs and covert then to binary association form
all_webs <- map(all_web_files, ~sign(read.csv(.x)))

# add names to each of the list entries. Required by map()
names(all_webs) <- all_web_files

# a table of all the webs by type
types <- unlist(map(names(all_webs), ~substr(.x, 8, 9)))
n_types <- table(types)
```

We used all the available bipartite webs from http://www.web-of-life.es as downloaded on 18 Febrary 2020. The original matrices included measures of times recorded, which we converted to binary association matrices.

Each interaction matrix types described below comprise frist the species in rows (S) and second the traits (N) by columns for example for species of Anemone (rows) provide habitat for Fish (columns):

- AF = Anemone - Fish (n = 17)
- HP = Host - Parasite (n = 51)
- PA = Plant - Ant (n = 4)
- PH = Plant - Herbivore (n = 4)
- PL = Plant - Pollinator (n = 148)
- SD = Seed - dispersers (n = 34)

Summary statistics

Define some functions that we will use to generate summary statics

A function to calculate our measure of complexity for a given web.

```
complexityEstimate <- function(A) {

  # rows are traits
  N <- nrow(A)

  # columns are species
  S <- ncol(A)

  # original complexity estimate
  c <- (N / sum(A)) * log(S)

  # new complexity estimate 3/Feb/2020
  # c <- N * log(N) / sum(A)

  # 6/feb/2020
  log_q <- -log ( (S * N - sum(A)) / (S * N))
  c <- (1/S) * (log(N) / log_q)

  return(c)

}
```

A function to calculate our measure of fragility f,

$$f = \frac{1}{S} \frac{\log(N)}{|\log(q)|}$$

where q=1-p and p is the connectance estimated as p~#links/(SxN). I wrote this more cleanly in overleaf.

```
fragilityEstimate <- function(A) {

  # rows are traits
  N <- nrow(A)

  # columns are species
  S <- ncol(A)

  # connectance measures
  p <- sum(A) / (S * N)
  q <-  1 - p
```

```
  # fragility
  # lines below are equivalent
  # f <- (1/S) * (log(N) / abs(log(q)))
  f <- -log(N) / log(q) / S


  return(f)

}
```

A function to calculate the robustness of a given web.

```
robustness <- function(Aperm, S = ncol(Aperm)){

   # number of Species
  # S <- ncol(Aperm)

  # number of traits
  # N <- nrow(Aperm)

  nn <- 0
  mm <- 0

  while (mm==0 & nn <= S) {
    nn <- nn + 1
    # mm <- min(rowSums( matrix(Aperm[, 1:nn], nrow = N, ncol = nn) ))
    mm <- min(rowSums( as.matrix(Aperm[, 1:nn])))
  }

  return( (S-nn+1) / (S * 1.0))
}
```

A function to take a given web, repeatedly resample it and remove species until it breaks. This lets us create an estimate of the mean robustness of a given web given random extinctions.

```
shuffleExinctionSingle <- function(A, S) {

  # generate a permuation for species in A
  species <- sample(S)

  # premute within rows
  Aperm <- A[,species]

  # calculate robustness
  rob <- robustness(Aperm)

  return(rob)

}

sampleRobustness <- function(A, S = ncol(A), nb = 10){

  # loop over replicates using parallel furrr map function.
  res <- future_map_dbl(1:nb, ~shuffleExinctionSingle( A, S))
```

```
  # return the mean of the results vector which is mean of
  return(mean(res))

}
```

A gaussian-exponential function to estimate robustness from fragility

```
predictRobustness <- function(x) {exp(-x - (2/3)* x ^2 )}
```

Apply our functions to each web

Calculate some summary statistics on each web and conver to data.frame format.

```
df_webs <-  all_webs %>%
  map_dfr(~data.frame(S = as.numeric(ncol(.x)),
                      N = as.numeric(nrow(.x)),
                      sum_A = as.numeric(sum(.x)),
                      min_S_per_N = min(rowSums(.x)),
                      var_S_per_N = var(rowSums(.x)),
                      mean_S_per_N = mean(rowSums(.x)),
                      est_p = sum(.x) / (ncol(.x) * nrow(.x)),
                      complexity = complexityEstimate(.x),
                      fragility = fragilityEstimate(.x), .id = "file"
          )) %>%
  mutate(web_type = types)

# calculate dispersion, where the expected value according to the binomial
# distributio is 1.
df_webs <- df_webs %>%
  mutate(dispersion = var_S_per_N / mean_S_per_N * (1 / 1 - est_p))
```

Simulate Extinctions to calculate robustness

```
do_robustness <- FALSE

if (do_robustness == TRUE ) {

  # set up the multicore for the function sampleRobustness which calls them.
  # im not sure if i should do this within sampleRobustness or if its ok to
  # do it once here outside.
  plan(multisession(workers = 3))

  robustness_results <- all_webs %>%
    map_dbl(~sampleRobustness(.x, nb = 500))

  save(robustness_results, file = "robustness_run.rda", compress = "xz")


}

if (do_robustness == FALSE){
  print("NB sampling for robustness not run. Loaded instead from previous run.")
  load("robustness_run.rda")
}
```

```
## [1] "NB sampling for robustness not run. Loaded instead from previous run."
```

```
# Add the robustness estimate to the data.frame and
# calcuate the residual to Jeff's model
df_webs <- df_webs %>%
  mutate(robustness = robustness_results)

df_webs <- df_webs %>%
  mutate(residual = robustness - predictRobustness(complexity))
```

Filter webs

We remove webs that have only 1 trait (n = 3) and webs that show no variation in the number of species S per trait N (n = 4, and all of which are Anemone-Fish type webs, all of which had anemones being associated with only 1 or 0 species of fish).

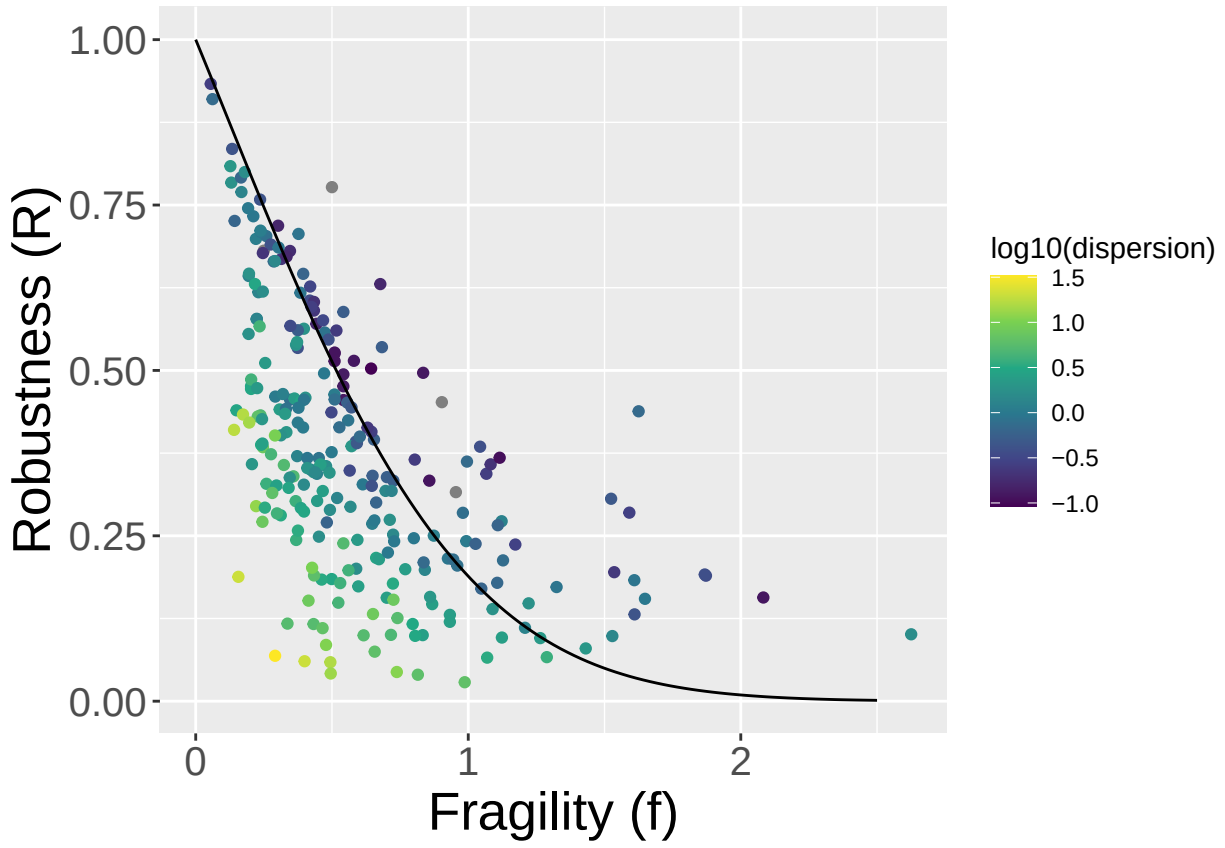
```
df_webs <- df_webs %>% filter( (var_S_per_N != 0) | !is.na(var_S_per_N) )
```

Plot results

In order to plot the analytical model, we create a dataframe object of a sequence of fragility values spanning out observed range and the corresponding predicted robustness ($\hat{R}$).

```
new_f <- seq(from = 0, to = 2.5, length.out = 100)

# original approximation
# est_r <- 0.7*exp(-3.5*new_c**2)+.3*exp(-0.8*new_c)

# new approximation 3/Feb/2020
N_vec <- 30
beta <- 1.3

# the estimated Robusteness (Rhat)
est_r <- predictRobustness(new_f)

fit_df <- data.frame(new_f = new_f,
                     est_r = est_r)
```

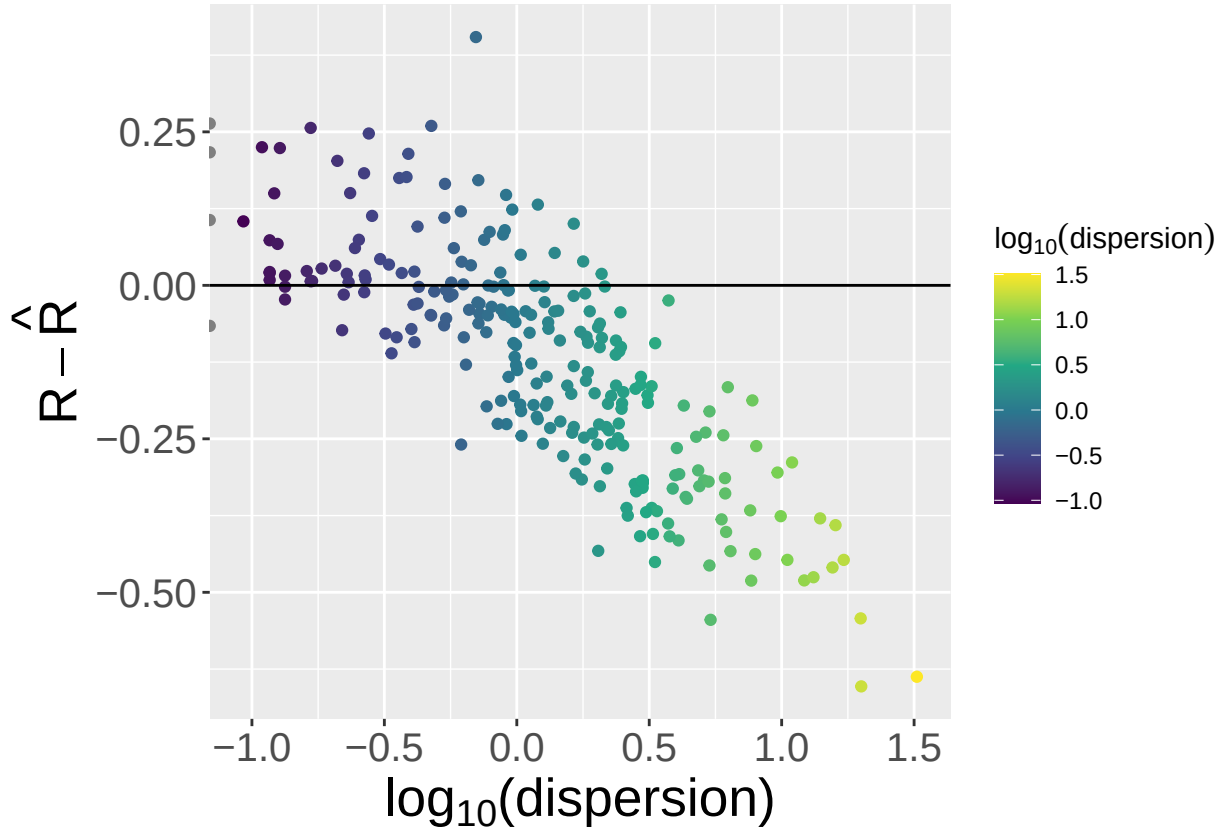
```
g1 <- ggplot(df_webs %>% filter(N <= Inf), aes(x = fragility,
                     y = robustness,
                     color = log10(dispersion))) +
  geom_point() +
  scale_color_viridis_c() +
  geom_line(data = fit_df, mapping = aes(x = new_f, y = est_r, color = NULL)) +
  ylab("Robustness (R)") +
  xlab("Fragility (f)") +
  theme(axis.text = element_text(size = 15),
        axis.title = element_text(size = 20))

print(g1)
```




```
ggsave(g1,filename = "robust-fragility-empirical.pdf",
       width = 15, height = 9, units = "cm",
       scale = 1)
```

Figure S2.1. The relationship between simulated robustness and analytically esimated fragility as calculated on the $n = $ 1 empirical networks matches close the analytical approximation (black line). Each network is coloured by the degree of disperion (log10 scale) according to a binomial distribution, where values below 0 represent under-dispersion, 0 is ideally dispersed and values greater than 0 indciate over-dispersion.

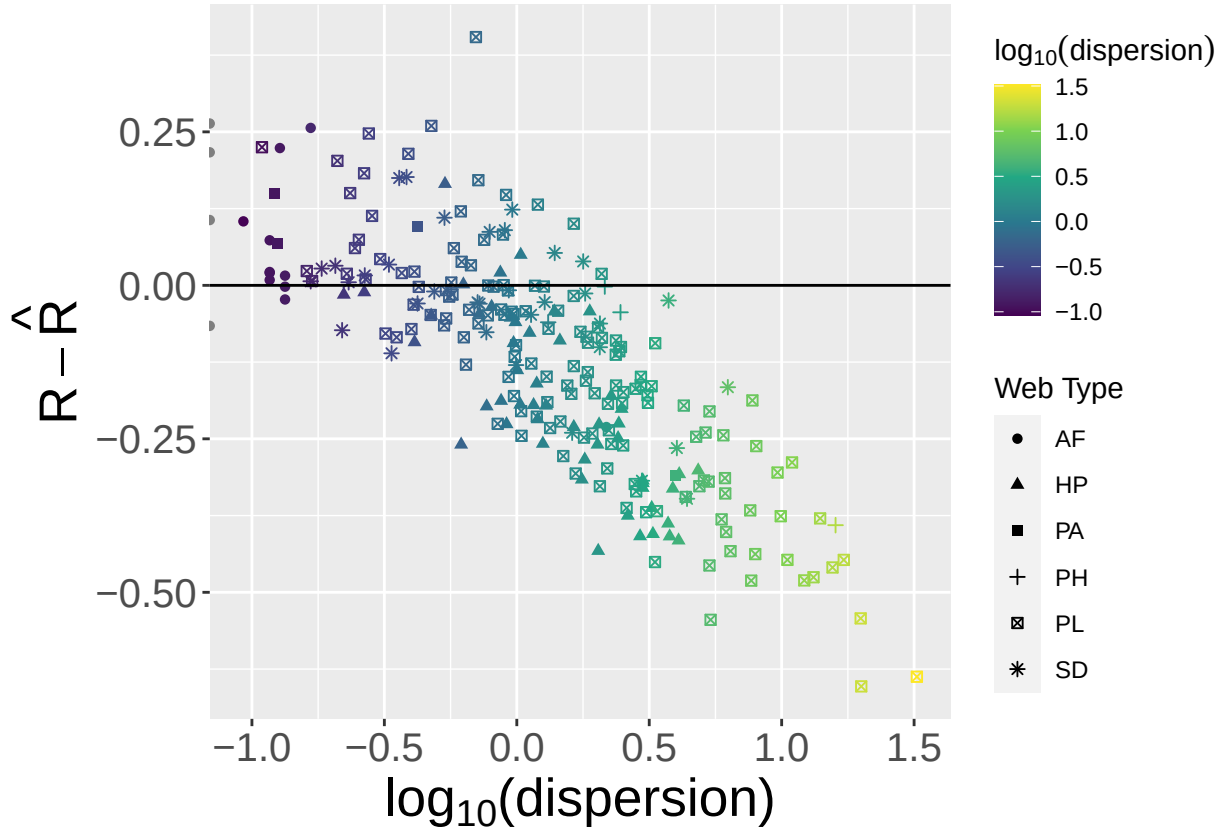
```
g3_simplified <- ggplot(df_webs, aes(x = log10(dispersion),
                     y = residual,
                     color = log10(dispersion))) +
  geom_point() +
  scale_color_viridis_c() +
  geom_hline(yintercept = 0) +
  labs(color = expression(log[10](dispersion)),
       shape = "Web Type") +
  ylab(expression(R - hat(R))) +
  xlab(TeX("$\\log_{10}(\\dispersion)$")) +
  theme(axis.text = element_text(size = 15),
        axis.title = element_text(size = 20))


print(g3_simplified)
```




```
ggsave(g3_simplified,filename = "resid-dispersion.pdf", width = 10, height = 7)
```

Figure S2.2. The residuals of the observed robustness to the analytical approximation in Figure S2.1 show a strong relationship with $\log_{10}(dispersion)$. Colours show the same $\log_{10}(dispersion)$ for direct comparison with points in Figure S2.1.

```
g3 <- ggplot(df_webs, aes(x = log10(dispersion),
                          y = residual,
                          color = log10(dispersion))) +
  geom_point(aes(shape = web_type)) +
  scale_color_viridis_c() +
  geom_hline(yintercept = 0) +
  labs(color = expression(log[10](dispersion)),
       shape = "Web Type") +
  ylab(expression(R - hat(R))) +
  xlab(TeX("$\\log_{10}(\\dispersion)$")) +
  theme(axis.text = element_text(size = 15),
        axis.title = element_text(size = 20))

print(g3)
```

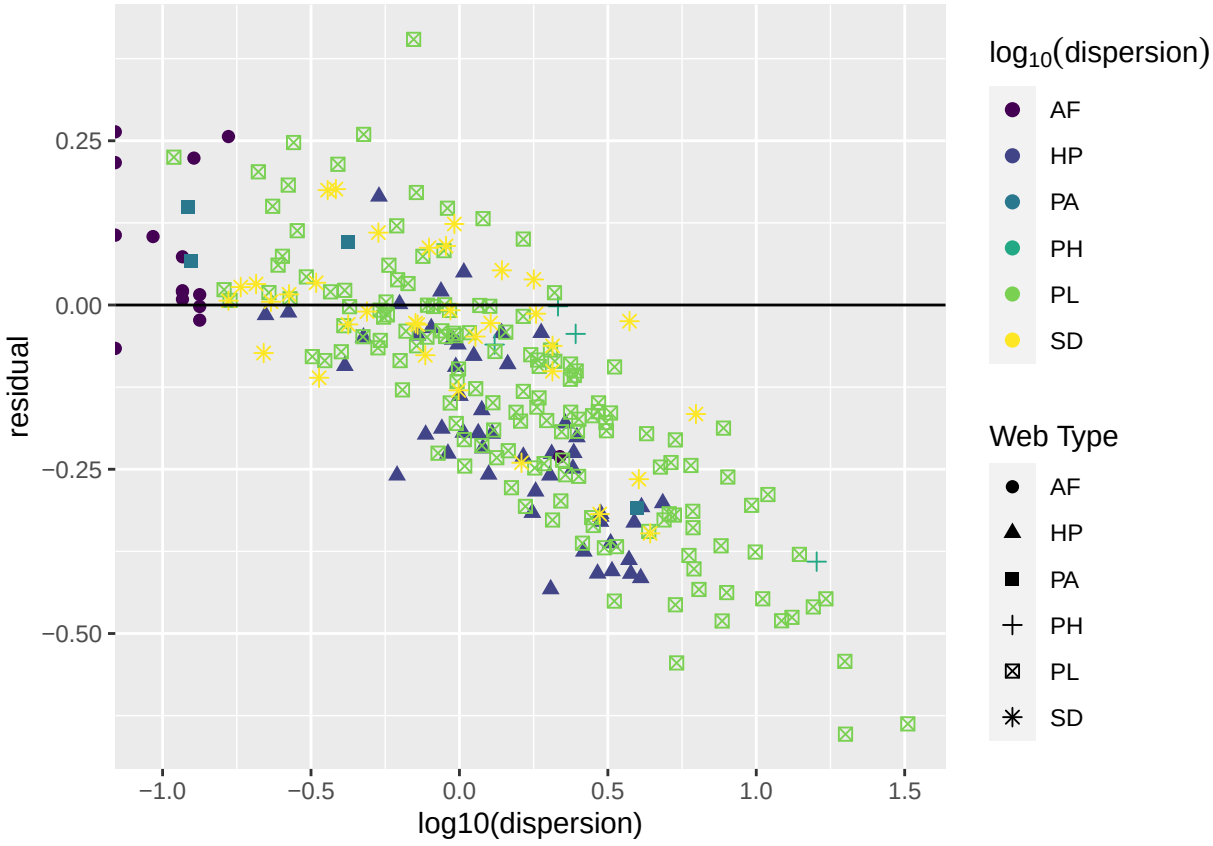



```
ggsave(g3,filename = "resid-dispersion-web-type.pdf", width = 10, height = 7)
```

Figure S2.3. Same as Figure S2.2 but with additional information of web type as indicated by shape of points for: AF = Anemone-Fish; HP = Host-Parasite; PA = Plant-Ant; PH = Plant-Herbivore; PL = Plant-Pollinator; SD = Seed-Disperser.

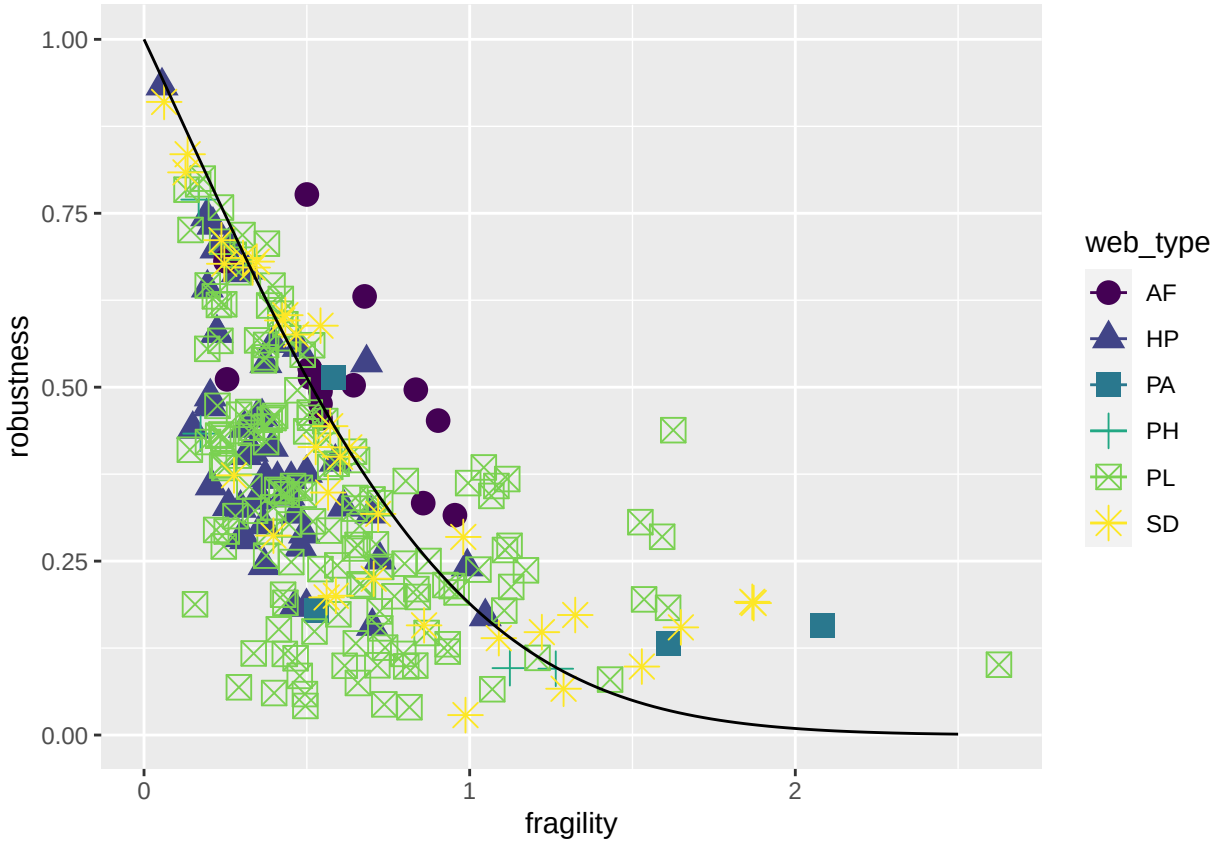
- AF = Anemone - Fish (n = 17)
- HP = Host - Parasite (n = 51)
- PA = Plant - Ant (n = 4)
- PH = Plant - Herbivore (n = 4)
- PL = Plant - Pollinator (n = 148)
- SD = Seed - dispersers (n = 34)

```
g4 <- ggplot(df_webs, aes(x = log10(dispersion),
                          y = residual,
                          color = web_type)) +
  geom_point(aes(shape = web_type), size = 2) +
  scale_color_viridis_d() +
  geom_hline(yintercept = 0) +
  labs(color = expression(log[10](dispersion)),
       shape = "Web Type")

print(g4)
```




```
# ggsave(g4,filename = "resid-dispersion.pdf", width = 10, height = 7)
```

```
g5 <- ggplot(df_webs %>% filter(N <= Inf), aes(x = fragility,
                      y = robustness,
                      color = web_type)) +
  geom_point(aes(shape = web_type), size = 4) +
  scale_color_viridis_d() +
  geom_line(data = fit_df, mapping = aes(x = new_f, y = est_r, color = NULL))

print(g5)
```




```
# ggsave(g5,filename = "robust-complexity.pdf", width = 10, height = 7)
```

Fit linear models

We can fit some linear models to calculate the R^2 values of the various explanatory variables of robustness.

```
N_vec <- 30
beta <- 1.3

df_webs <- df_webs %>% mutate(robustness_hat = predictRobustness(fragility))

# analytical_model <-  with( df_webs,
#                            {(N_vec *
#                                (exp(-complexity - beta * complexity ^ 2 )) )  /
#                                (N_vec + 1)}
#                            )


# null model
m0 <- lm(robustness ~1,
         data = df_webs %>% filter(!is.na(dispersion) &
                                             (dispersion > 0)))

# model as above with analytical model as offset
m1 <- lm(robustness ~ -1,
         offset =  robustness_hat,
         data = df_webs %>% filter(!is.na(dispersion) &
```

```
                                          (dispersion > 0)))

# model as above including log10(dispersion) as covariate
m2 <- lm(robustness ~ I(log10(dispersion)),
         offset =  robustness_hat,
         data = df_webs %>% filter(!is.na(dispersion) &
                                          (dispersion > 0)))

# model as above including web type as covariate
m3 <- lm(robustness ~ I(log10(dispersion)) + factor(web_type),
         offset =  robustness_hat,
         data = df_webs %>% filter(!is.na(dispersion) &
                                          (dispersion > 0)))

# model as above including web type, log(dispersion) and web type as covariates
m4 <- lm(robustness ~ I(log10(dispersion)) + factor(web_type),
         offset =  robustness_hat,
         data = df_webs %>% filter(!is.na(dispersion) &
                                          (dispersion > 0)))
```