Appendix IV: Power analysis simulations for correlated variables

```
library(broom)
library(plyr)
library(tidyverse)

theme_set(theme_classic())

# function to generate pairs of uncorrelated variables
sim_z <- function (n) matrix(c(rnorm(n, 0, 1), rnorm(n, 0, 1)), nrow = n)

# function to convert an uncorrelated pair of variables z to a correlated pair with
# coefficient r
corr_pair <- function(z, r) {
  # create the covariance matrix
  cormat <- matrix(1, nrow = 2, ncol = 2)
  cormat[upper.tri(cormat)] <- cormat[lower.tri(cormat)] <- r
  covmat <- cormat * tcrossprod(rep(1, 2))

  # calculate the Cholesky decomposition of the covariance matrix
  ch <- chol(covmat)

  # we can then multiply ch by a set %>% of uncorrelated variables z to get the
  # appropriately correlated variables.
  y <- z %*% ch
}
```

Background

The purpose of these simulations is to determine the conditions under which we are likely to find a difference between two correlated variables. The reasoning for this is that is we want to include the window-based upscaling measure and a mean value for the same variable (e.g. MW elevation and mean elevation) in the same analysis, they are often correlated. Additionally, in our paper the MW and LS versions are highly correlated (in our example, the two measures have a correlation coefficient of 0.97) and we need to know that we have the ability to detect the right effect. We are still able to detect the difference between the two approaches because of a large sample size characteristic of macroecological approaches. However, in order to understand wider applicability - and under which circumstances the method is applicable - we will do a simulation to determine, for a given correlation coefficient, at what sample size do you no longer have the ability to detect an effect.

In each of our examples, we will simulate three variables: 2 correlated (x1, x2) and one independent (x3). We Will need to consider a range of R, a range of n, and three classes of effects (1) y ~ x1 + x3, (2) y ~ x1 + x2 + x3, (3) y ~ x1 - x2 + x3.

Predictors

In order to do this, we simulated several sets of predictors of varying sample size and correlation. For each pair of correlated predictors (x1, x2), we also generated a third independent predictor (x3). We therefore have a set of predictors $\boldsymbol{X}$ where x_1 and x_2 are correlated, and x_3 is independent. The changing parameters in the dataset simulations were:

- sample size: 30, 50, 100, 500, 1000 (for now, we may wish to extend)
- correlation: 0.7, 0.75, 0.8, 0.85, 0.9, 0.95

For each sample size, we generated 100 replicates.

```r
n <- c(30, 50, 100, 250, 500, 750, 1000, 2000, 5000)
r <- c(0.7, 0.75, 0.8, 0.85, 0.9, 0.95)
n_reps <- 100
rep <- 1:n_reps

param_table <- expand.grid(n = n, r = r, rep = rep)

res <- apply(param_table, 1, function(p) {
  n = p['n']
  r = p['r']
  rep = p['rep']
  z <- mapply(rnorm, n, rep(0, 3), rep(1, 3))
  x <- corr_pair(z[,1:2], r)
  out <- data.frame(n, r, rep, x, z[,3], row.names = NULL)
  names(out) <- c("n", "r", "rep","x1", "x2", "x3")
  return(out)
})

pred_df <- ldply(res)
```

Responses

In order to understand how the ability to detect an effect might differ between models, we will generate response variables for each dataset for the following models:

- $y = 0.5x_1 + 0.5x_3 + \epsilon$ - two uncorrelated variables (similar to jay case study)
- $y = 0.5x_1 + 0.5x_2 + 0.5x_3 + \epsilon$ - two correlated variables, same effect, plus third independent variable
- $y = 0.5x_1 - 0.5x_2 + 0.5x_3 + \epsilon$ - two correlated variables, opposite effect, plus third independent variable (similar to forests case study)

In each case here ϵ has a normal error structure with a reasonable amount of noise in the data $\epsilon \sim N(0, 1)$. We do not consider interaction terms in these simulations.

```r
df <- pred_df %>%
  mutate(y1 = 0.5*x1 + 0.5*x3 + rnorm(n, 0, 1),
         y2 = 0.5*x1 + 0.5*x2 + 0.5*x3 + rnorm(n, 0, 1),
         y3 = 0.5*x1 - 0.5*x2 + 0.5*x3 + rnorm(n, 0, 1))
```

Models

For each response, we fit a linear model of the form:

$y_i = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$

We should get the following results:

- For y_1, $\beta_1 = 0.5$, $\beta_2 = 0$, $\beta_3 = 0.5$
- For y_2, $\beta_1 = 0.5$, $\beta_2 = 0.5$, $\beta_3 = 0.5$

- For y_3, $\beta_1 = 0.5$, $\beta_2 = -0.5$, $\beta_3 = 0.5$

We calculate the mean of the estimates and the standard errors across the 100 replicates for final analysis.

```r
df_mod <- df %>% group_by(n, r, rep) %>%
  nest() %>%
  mutate(fit1 = map(data, ~ lm(y1 ~ x1 + x2 + x3, data = .x)),
         fit2 = map(data, ~ lm(y2 ~ x1 + x2 + x3, data = .x)),
         fit3 = map(data, ~ lm(y3 ~ x1 + x2 + x3, data = .x)),
         mod1 = map(fit1, tidy),
         mod2 = map(fit2, tidy),
         mod3 = map(fit3, tidy)
  ) %>%
  select(n, r, rep, mod1, mod2, mod3) %>%
  gather(key = model, value = result, -n, -r, -rep) %>%
  unnest()
```

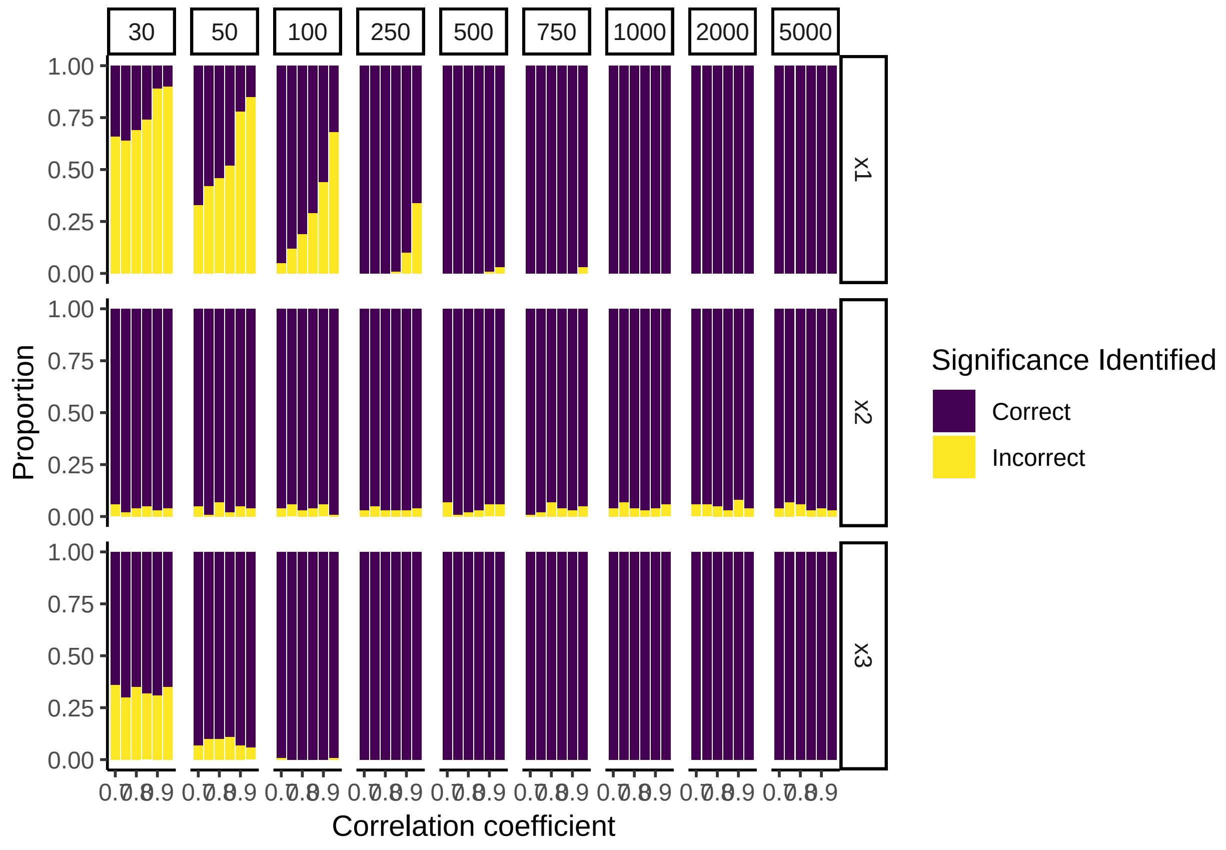
Results

We display two plots for each model: 1) the proportion of replicates where the significance of a coefficient was correctly identified (note, for coefficients where the true value is not equal to zero, this is the number of replicates which identify the coefficient as significant, where the true value is zero, this is the number of replicates where the coefficient was correctly estimated to be non-significant); and 2) the distribution of the coefficient estimate for those replicates (out of 100) for which the significance of the estimate is correctly identified (as determined in plot 1). In all cases, the dashed line in plot 2 is at the true value for this coefficient (used to simulate the y value).

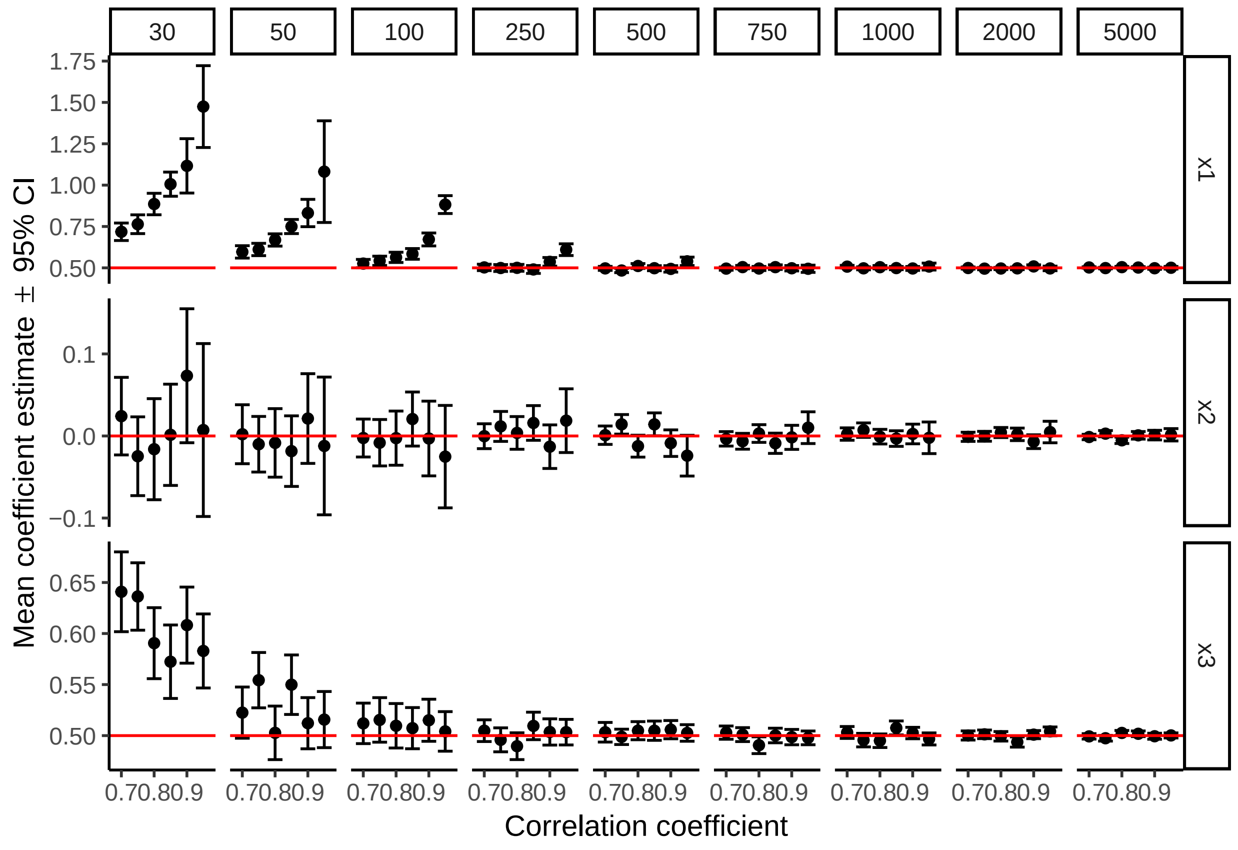
No effect of second correlated variable

$y = 0.5x_1 + 0.5x_3 + \epsilon$

```r
df <- filter(df_mod, model == "mod1", term != "(Intercept)") %>%
  inner_join(tibble(truth = c(0.5, 0, 0.5), term = c("x1", "x2", "x3"))) %>%
  mutate(signif = case_when(truth == 0 & p.value < 0.05 ~ "Incorrect",
                            truth == 0 & p.value >= 0.05 ~ "Correct",
                            truth != 0 & p.value < 0.05 ~ "Correct",
                            truth != 0 & p.value >= 0.05 ~ "Incorrect"))

mod_vals <- filter(df, signif == "Correct") %>%
  group_by(n, r, model, term, truth) %>%
  summarise(mean_estimate = mean(estimate),
            se_estimate = sd(estimate)/sqrt(n()))

mod_identified <- group_by(df, n, r, term, signif) %>%
  summarise(prop = n()/n_reps)

ggplot(mod_identified, aes(x = r, y = prop, fill = signif)) +
  geom_bar(stat = "identity") +
  scale_fill_viridis_d(name = "Significance Identified") +
  facet_grid(term ~ n) +
  labs(x = "Correlation coefficient", y = "Proportion")
```



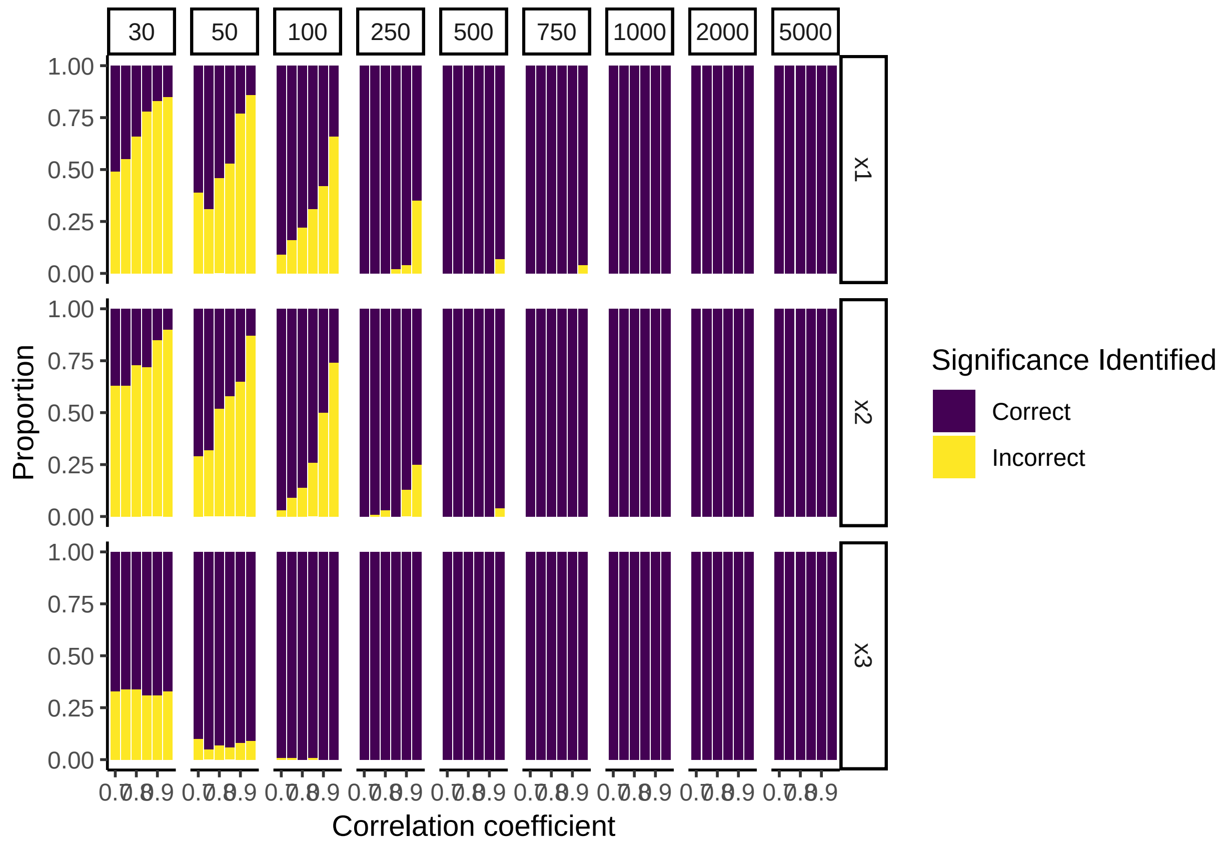
```r
ggplot(mod_vals, aes(x = r, y = mean_estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = mean_estimate - 1.96*se_estimate,
                    ymax = mean_estimate + 1.96*se_estimate)) +
  facet_grid(term ~ n, scales = "free") +
  geom_hline(aes(yintercept = truth), colour = "red") +
  labs(x = "Correlation coefficient",
       y = expression(paste("Mean coefficient estimate " %+-% " 95% CI")))
```



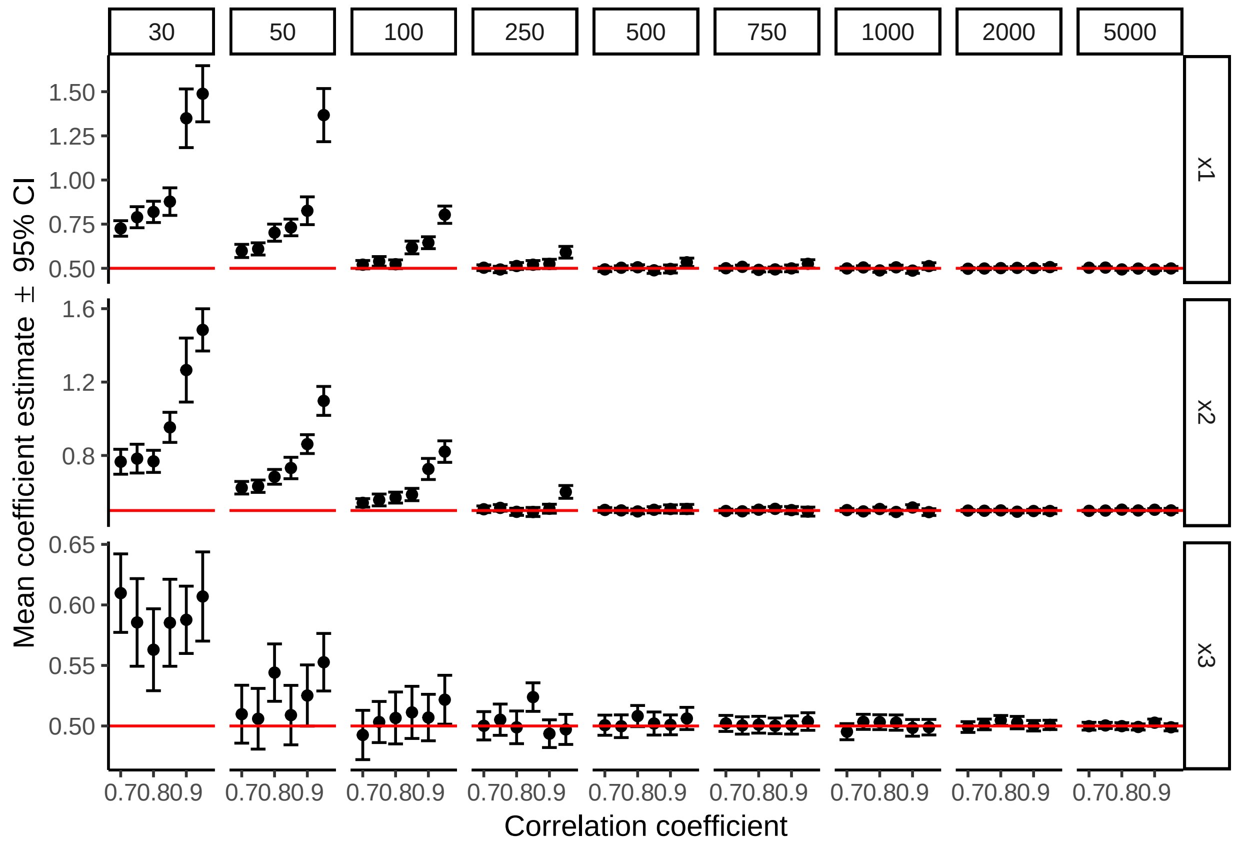
Positive effect of second correlated variable

$y = 0.5x_1 + 0.5x_2 + 0.5x_3 + \epsilon$

```
df <- filter(df_mod, model == "mod2", term != "(Intercept)") %>%
  inner_join(tibble(truth = c(0.5, 0.5, 0.5), term = c("x1", "x2", "x3"))) %>%
  mutate(signif = case_when(truth == 0 & p.value < 0.05 ~ "Incorrect",
                            truth == 0 & p.value >= 0.05 ~ "Correct",
                            truth != 0 & p.value < 0.05 ~ "Correct",
                            truth != 0 & p.value >= 0.05 ~ "Incorrect"))

mod_vals <- filter(df, signif == "Correct") %>%
  group_by(n, r, model, term, truth) %>%
  summarise(mean_estimate = mean(estimate),
            se_estimate = sd(estimate)/sqrt(n()))

mod_identified <- group_by(df, n, r, term, signif) %>%
  summarise(prop = n()/n_reps)

ggplot(mod_identified, aes(x = r, y = prop, fill = signif)) +
  geom_bar(stat = "identity") +
  scale_fill_viridis_d(name = "Significance Identified") +
  facet_grid(term ~ n) +
  labs(x = "Correlation coefficient", y = "Proportion")
```



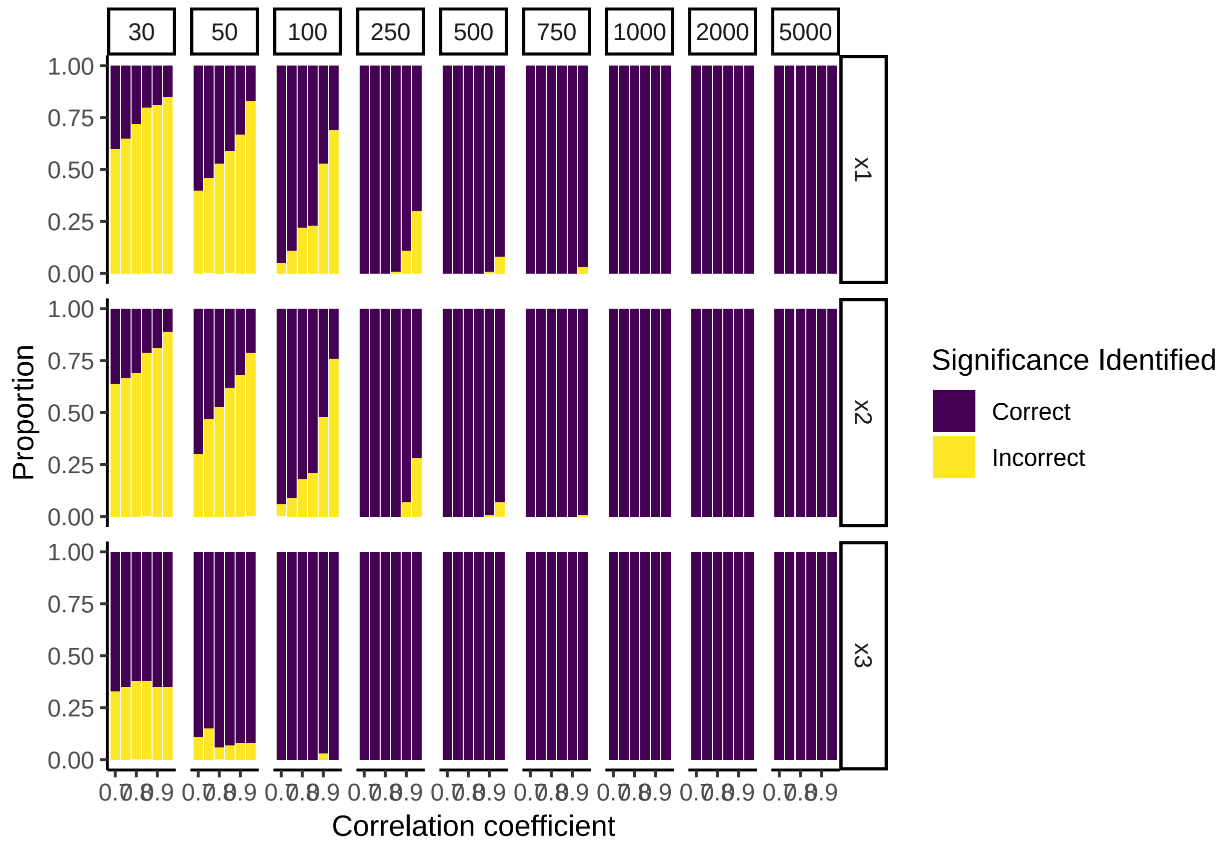
```r
ggplot(mod_vals, aes(x = r, y = mean_estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = mean_estimate - 1.96*se_estimate,
                    ymax = mean_estimate + 1.96*se_estimate)) +
  facet_grid(term ~ n, scales = "free") +
  geom_hline(aes(yintercept = truth), colour = "red") +
  labs(x = "Correlation coefficient",
       y = expression(paste("Mean coefficient estimate " %+-% " 95% CI")))
```



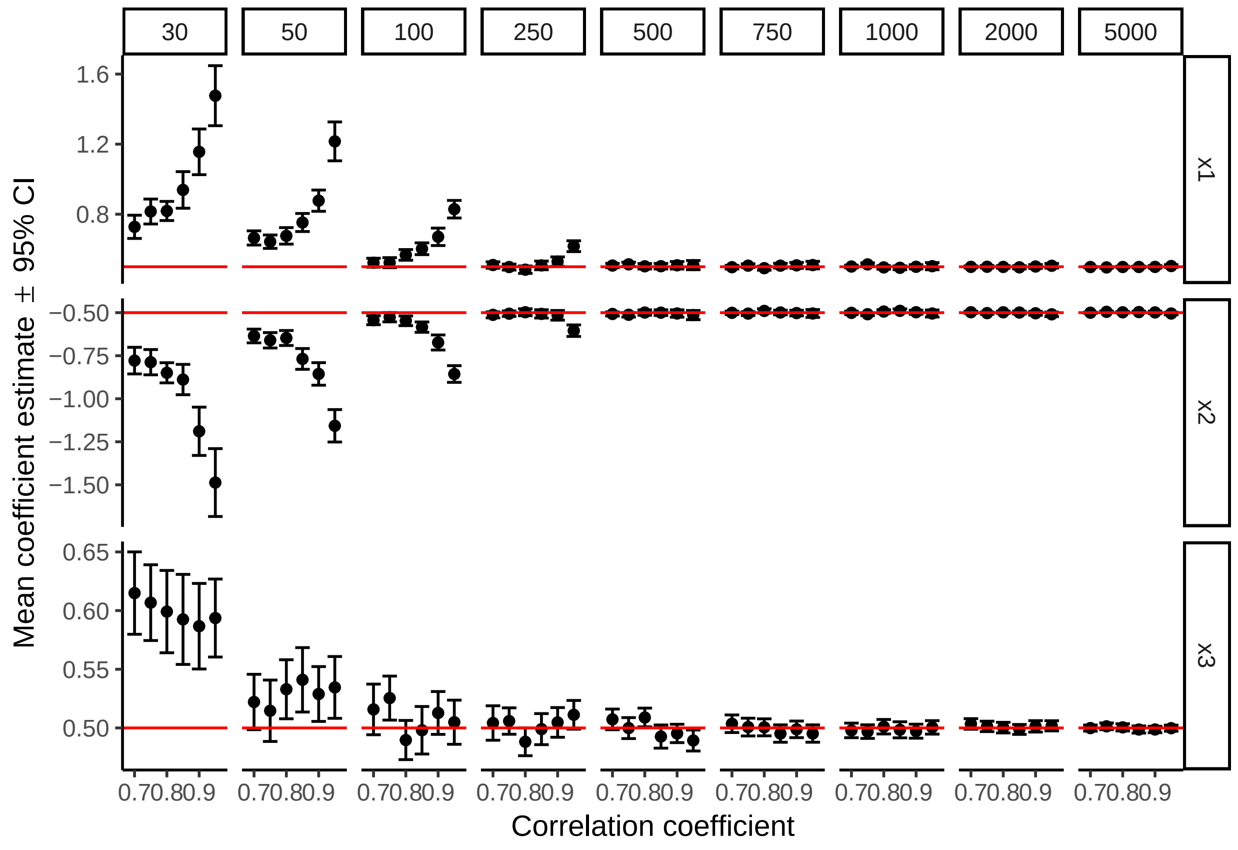
Negative effect of second correlated variable

$y = 0.5x_1 - 0.5x_2 + 0.5x_3 + \epsilon$

```
df <- filter(df_mod, model == "mod3", term != "(Intercept)") %>%
  inner_join(tibble(truth = c(0.5, -0.5, 0.5), term = c("x1", "x2", "x3"))) %>%
  mutate(signif = case_when(truth == 0 & p.value < 0.05 ~ "Incorrect",
                            truth == 0 & p.value >= 0.05 ~ "Correct",
                            truth != 0 & p.value < 0.05 ~ "Correct",
                            truth != 0 & p.value >= 0.05 ~ "Incorrect"))

mod_vals <- filter(df, signif == "Correct") %>%
  group_by(n, r, model, term, truth) %>%
  summarise(mean_estimate = mean(estimate),
            se_estimate = sd(estimate)/sqrt(n()))

mod_identified <- group_by(df, n, r, term, signif) %>%
  summarise(prop = n()/n_reps)

ggplot(mod_identified, aes(x = r, y = prop, fill = signif)) +
  geom_bar(stat = "identity") +
  scale_fill_viridis_d(name = "Significance Identified") +
  facet_grid(term ~ n) +
  labs(x = "Correlation coefficient", y = "Proportion")
```



```
ggplot(mod_vals, aes(x = r, y = mean_estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = mean_estimate - 1.96*se_estimate,
                    ymax = mean_estimate + 1.96*se_estimate)) +
  facet_grid(term ~ n, scales = "free") +
  geom_hline(aes(yintercept = truth), colour = "red") +
  labs(x = "Correlation coefficient",
       y = expression(paste("Mean coefficient estimate " %+-% " 95% CI")))
```



Conclusion

In all cases, a sample size ≥ 500 is required to correctly identify the significance and magnitude of an effect at all correlation levels. For correlation $\leq r = 0.9$, this drops to $n = 250$. The majority of macroecological analyses will have sample sizes of this magnitude.

Session Info

```
devtools::session_info()
```

```
##  setting  value
##  version  R version 3.5.0 (2018-04-23)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United Kingdom.1252
##  tz       Europe/London
##  date     2018-06-22
##
##  package     * version    date       source
##  assertthat    0.2.0      2017-04-11 CRAN (R 3.5.0)
##  backports     1.1.2      2017-12-13 CRAN (R 3.5.0)
##  base        * 3.5.0      2018-04-23 local
##  bindr         0.1.1      2018-03-13 CRAN (R 3.5.0)
##  bindrcpp    * 0.2.2      2018-03-29 CRAN (R 3.5.0)
##  broom       * 0.4.4      2018-03-29 CRAN (R 3.5.0)
##  cellranger    1.1.0      2016-07-27 CRAN (R 3.5.0)
##  cli           1.0.0      2017-11-05 CRAN (R 3.5.0)
##  colorspace    1.3-2      2016-12-14 CRAN (R 3.5.0)
##  compiler      3.5.0      2018-04-23 local
##  crayon        1.3.4      2017-09-16 CRAN (R 3.5.0)
##  datasets    * 3.5.0      2018-04-23 local
##  devtools      1.13.5     2018-02-18 CRAN (R 3.5.0)
##  digest        0.6.15     2018-01-28 CRAN (R 3.5.0)
##  dplyr       * 0.7.4      2017-09-28 CRAN (R 3.5.0)
##  evaluate      0.10.1     2017-06-24 CRAN (R 3.5.0)
##  forcats     * 0.3.0      2018-02-19 CRAN (R 3.5.0)
##  foreign       0.8-70     2017-11-28 CRAN (R 3.5.0)
##  ggplot2     * 2.2.1.9000 2018-05-23 Github (tidyverse/ggplot2@eecc450)
##  glue          1.2.0      2017-10-29 CRAN (R 3.5.0)
##  graphics    * 3.5.0      2018-04-23 local
##  grDevices   * 3.5.0      2018-04-23 local
##  grid          3.5.0      2018-04-23 local
##  gtable        0.2.0      2016-02-26 CRAN (R 3.5.0)
##  haven         1.1.1      2018-01-18 CRAN (R 3.5.0)
##  hms           0.4.2      2018-03-10 CRAN (R 3.5.0)
##  htmltools     0.3.6      2017-04-28 CRAN (R 3.5.0)
##  httr          1.3.1      2017-08-20 CRAN (R 3.5.0)
##  jsonlite      1.5        2017-06-01 CRAN (R 3.5.0)
##  knitr         1.20       2018-02-20 CRAN (R 3.5.0)
##  labeling      0.3        2014-08-23 CRAN (R 3.5.0)
##  lattice       0.20-35    2017-03-25 CRAN (R 3.5.0)
##  lazyeval      0.2.1      2017-10-29 CRAN (R 3.5.0)
##  lubridate     1.7.4      2018-04-11 CRAN (R 3.5.0)
##  magrittr      1.5        2014-11-22 CRAN (R 3.5.0)
##  memoise       1.1.0      2017-04-21 CRAN (R 3.5.0)
##  methods     * 3.5.0      2018-04-23 local
##  mnormt        1.5-5      2016-10-15 CRAN (R 3.5.0)
##  modelr        0.1.2      2018-05-11 CRAN (R 3.5.0)
##  munsell       0.4.3      2016-02-13 CRAN (R 3.5.0)
##  nlme          3.1-137    2018-04-07 CRAN (R 3.5.0)
##  parallel      3.5.0      2018-04-23 local
##  pillar        1.2.2      2018-04-26 CRAN (R 3.5.0)
##  pkgconfig     2.0.1      2017-03-21 CRAN (R 3.5.0)
##  plyr        * 1.8.4      2016-06-08 CRAN (R 3.5.0)
##  psych         1.8.4      2018-05-06 CRAN (R 3.5.0)
##  purrr       * 0.2.4      2017-10-18 CRAN (R 3.5.0)
##  R6            2.2.2      2017-06-17 CRAN (R 3.5.0)
##  Rcpp          0.12.16    2018-03-13 CRAN (R 3.5.0)
##  readr       * 1.1.1      2017-05-16 CRAN (R 3.5.0)
##  readxl        1.1.0      2018-04-20 CRAN (R 3.5.0)
##  reshape2      1.4.3      2017-12-11 CRAN (R 3.5.0)
##  rlang         0.2.0      2018-02-20 CRAN (R 3.5.0)
##  rmarkdown     1.9        2018-03-01 CRAN (R 3.5.0)
##  rprojroot     1.3-2      2018-01-03 CRAN (R 3.5.0)
##  rstudioapi    0.7        2017-09-07 CRAN (R 3.5.0)
##  rvest         0.3.2      2016-06-17 CRAN (R 3.5.0)
##  scales        0.5.0      2017-08-24 CRAN (R 3.5.0)
##  stats       * 3.5.0      2018-04-23 local
##  stringi       1.1.7      2018-03-12 CRAN (R 3.5.0)
##  stringr     * 1.3.1      2018-05-10 CRAN (R 3.5.0)
##  tibble      * 1.4.2      2018-01-22 CRAN (R 3.5.0)
##  tidyr       * 0.8.0      2018-01-29 CRAN (R 3.5.0)
##  tidyselect    0.2.4      2018-02-26 CRAN (R 3.5.0)
##  tidyverse   * 1.2.1      2017-11-14 CRAN (R 3.5.0)
##  tools         3.5.0      2018-04-23 local
##  utils       * 3.5.0      2018-04-23 local
##  viridisLite   0.3.0      2018-02-01 CRAN (R 3.5.0)
##  withr         2.1.2      2018-05-15 Github (jimhester/withr@79d7b0d)
##  xml2          1.2.0      2018-01-24 CRAN (R 3.5.0)
##  yaml          2.1.19     2018-05-01 CRAN (R 3.5.0)
```