

Appendix 4: Results

Inferring species interactions from co-occurrence data with Markov networks

David J. Harris

Contents

A: Import packages and data	1
Import the results from Appendix 3:	1
Import the results from the <i>Pairs</i> software:	2
B: Compare model estimates	4
Calculate model performance (R-squared):	4
Save the finer-grained R-squared results as Figure 3	9
Estimate uncertainty in mean R-squared:	9
C: Inferential statistics	10
Identify statistical significance for the Markov network and Pairs	11
Create Figure 4	11
Summarize inferential statistics:	13
Type I error rates:	14
Plot Markov network confidence interval coverage versus true β value	15
Plot power versus true β value	16

A: Import packages and data

```
library(dplyr)
library(magrittr)
library(mgcv)
library(ggplot2)
library(tidyr)
library(knitr)
library(lme4)
```

Import the results from Appendix 3:

```
x = read.csv("estimates.csv", stringsAsFactors = FALSE)
x$simulation_type = gsub("[0-9]", "", x$rep_name)
```

Import the results from the *Pairs* software:

```
pairs_txt = readLines("fakedata/matrices/Pairs.txt")
library(stringr)

# Find areas of the data file that correspond
# to species pairs' results
beginnings = grep("Sp1", pairs_txt) + 1
ends = c(
  grep("[^ ]", pairs_txt)[-1],
  length(pairs_txt) + 1
) - 1

partial_names = sapply(
  strsplit(grep(">", pairs_txt, value = TRUE), " +"),
  function(x) x[[3]]
)
filename_lines = grep(">", pairs_txt)

# Sort a vector of alphanumeric strings by the numeric component
# as if they were integers. For example, V20 should be larger
# than V12, even though V12 comes first alphabetically
alnum_sort = function(x){
  raw = as.integer(gsub("[[:alpha:]]", "", x))
  x[order(raw)]
}

pairs_results = lapply(
  1:length(filename_lines),
  function(i){

    n_sites = as.integer(strsplit(partial_names[[i]], "-")[1][[1]])
    rep_name = strsplit(partial_names[[i]], "-")[1][[2]]

    # Find the line where the current data set is mentioned in
    # pairs.txt
    filename_line = filename_lines[i]

    # Which chunk of the data file corresponds to this file?
    chunk = min(which(beginnings > filename_line))
```

```

# Split the chunk on whitespace.
splitted = strsplit(pairs_txt[beginnings[chunk]:ends[chunk]], " ")

# Pull out the corresponding chunk of the "x" data frame, based on n_sites
# and rep_name
is_correct_sim = x$n_sites == n_sites & x$rep_name == rep_name
x_subset = x[is_correct_sim & x$method == "correlation", ]

# Pull out the species numbers and their Z-scores, then join to x_subset
pairs_results = lapply(
  splitted,
  function(x){
    # in the x data frame, species 1 is always a lower number than species 2
    spp = alnum_sort(x[3:4])
    data.frame(
      sp1 = spp[1],
      sp2 = spp[2],
      z = x[14],
      stringsAsFactors = FALSE
    )
  }
) %>%
  bind_rows %>%
  mutate(spp = paste(sp1, sp2, sep = "-"))

n_spp = 20

pairs_results$z = as.numeric(pairs_results$z)

# Re-order the pairs_results to match the other methods
m = matrix(NA, n_spp, n_spp)
new_order = match(
  paste0("V", row(m)[upper.tri(m)], "-V", col(m)[upper.tri(m)]),
  pairs_results$spp
)
ordered_pairs_results = pairs_results[na.omit(new_order), ]

ordered_pairs_results = ordered_pairs_results %>%
  filter(sp1 %in% c(x_subset$sp1) & sp2 %in% x_subset$sp2)

x_subset$estimate = ordered_pairs_results$z
x_subset$method = "null"

x_subset
}
) %>% bind_rows()

```

```

# Manually adjust the Z values less than -1000 so that these outliers
# won't completely dominate the analyses below
pairs_results$estimate[pairs_results$estimate < -1000] = -50

x = rbind(x, pairs_results)

```

B: Compare model estimates

Calculate model performance (R-squared):

For each combination of `method` and `simulation_type`, fit a linear model. Then, for each combination of `method`, `simulation_type`, and `n_sites`, report the proportion of variance explained by the corresponding linear model.

```

resids = function(data){
  resid(lm(truth ~ estimate + 0, data = data))
}

result_summary = x %>%
  group_by(method, simulation_type) %>%
  do(data.frame(., resids = resids(.))) %>%
  ungroup %>%
  group_by(method, simulation_type, n_sites) %>%
  summarise(r2 = 1 - sum(resids^2) / sum(truth^2))

# Set the ordering of the data frame based on R-squared
result_summary$method = reorder(result_summary$method, -result_summary$r2)
result_summary$simulation_type = reorder(
  result_summary$simulation_type,
  -result_summary$r2
)

# Add a column that includes method name and its mean R-squared,
# for plotting purposes below
result_summary = result_summary %>%
  group_by(method) %>%
  summarise(mean_r2 = round(100 * mean(r2))) %>%
  mutate(method_r2 = paste0(method, " (0.", mean_r2, ")")) %>%
  select(method, method_r2) %>%
  inner_join(result_summary, "method")

# Rename the simulation types for clearer graph labels
result_summary$simulation_type_long = plyr::revalue(
  result_summary$simulation_type,
  c(no_env = "constant environment",
    env = "heterogeneous environment",

```

```

    abund = "abundance")
)

result_summary$method_r2 = reorder(result_summary$method_r2, -result_summary$r2)

```

Plot the regressions used above to calculate R^2

The plots below each correspond to one of the regressions performed above to calculate R^2 for a combination of simulation method and estimation method. The null model (“Pairs”) was not included in these plots because its Z-scores were frequently too large to fit on the same scale as the other methods’ estimates. Below, the thinner dashed lines represent the 1:1 line, where the expected estimate matches the “true” value, and the thicker solid lines represent a linear regression through the origin, as calculated above. In general, the four correlation- or partial correlation-based methods had slopes that were steeper than the 1:1 line, because their estimates were constrained between -1 and 1. The GLM and Markov network usually had slopes that were closer to 1.

(Warnings of the form “Removed N rows containing missing values (geom_smooth)” can be ignored: they are simply noting that the regression lines were cut off in some panels of the figure).

```

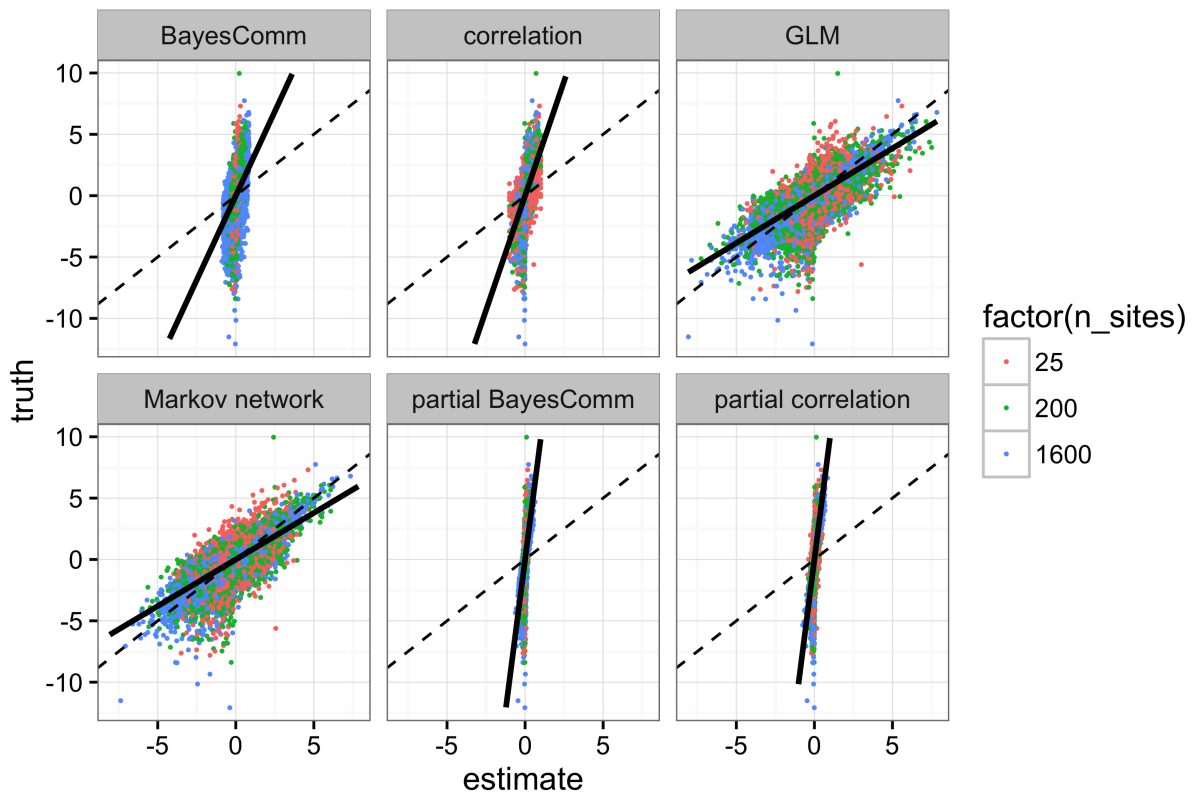
plot_regressions = function(type, title){
  x[sample.int(nrow(x)) , ] %>%
  subset(simulation_type == type & method != "null") %>%
  ggplot(aes(x = estimate, y = truth)) +
    facet_wrap(~method) +
    ylim(min(x$truth), max(x$truth)) +
    geom_point(aes(color = factor(n_sites)), size = 0.2) +
    geom_smooth(method = "lm", color = "black", formula = y ~ 0 + x,
               fullrange = TRUE) +
    geom_abline(intercept = 0, slope = 1, linetype = 2) +
    theme_bw() +
    ggtitle(title)
}

plot_regressions("no_env", "'Baseline' regressions")

```

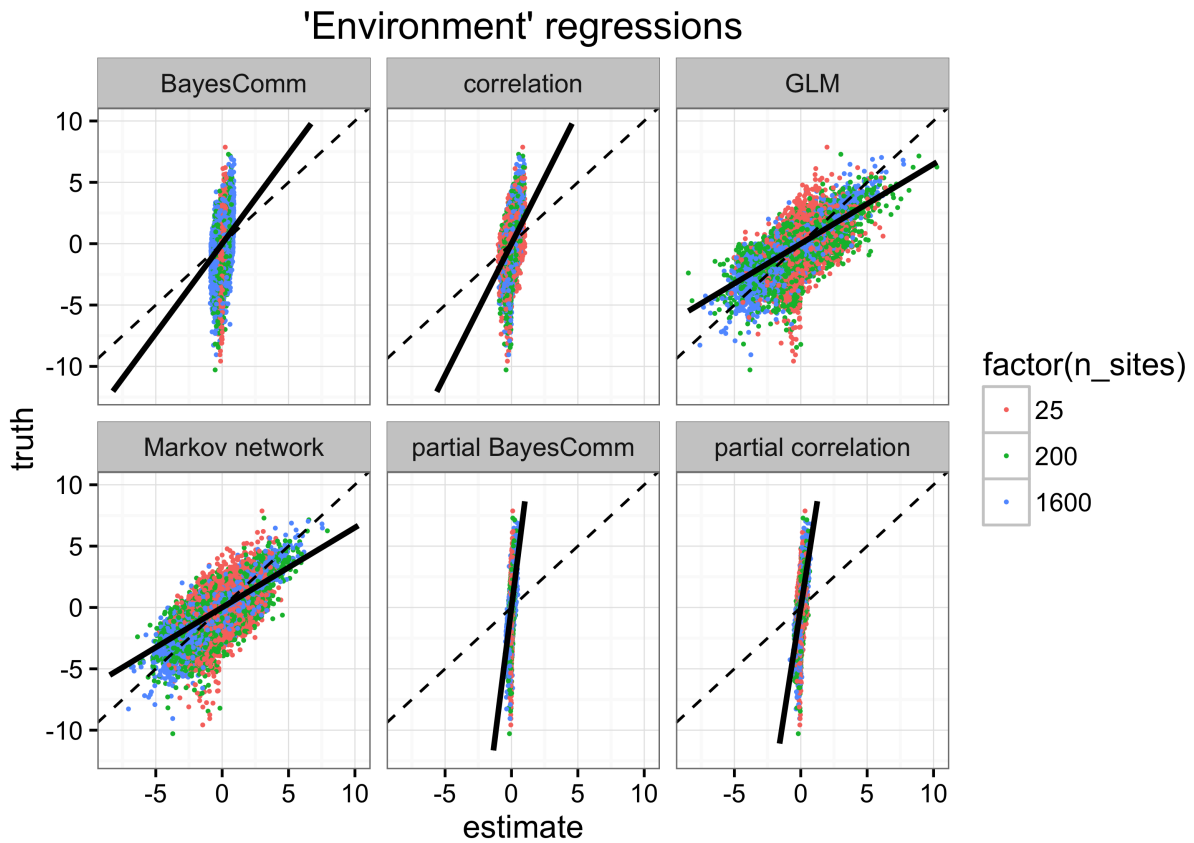
```
## Warning: Removed 227 rows containing missing values (geom_smooth).
```

'Baseline' regressions



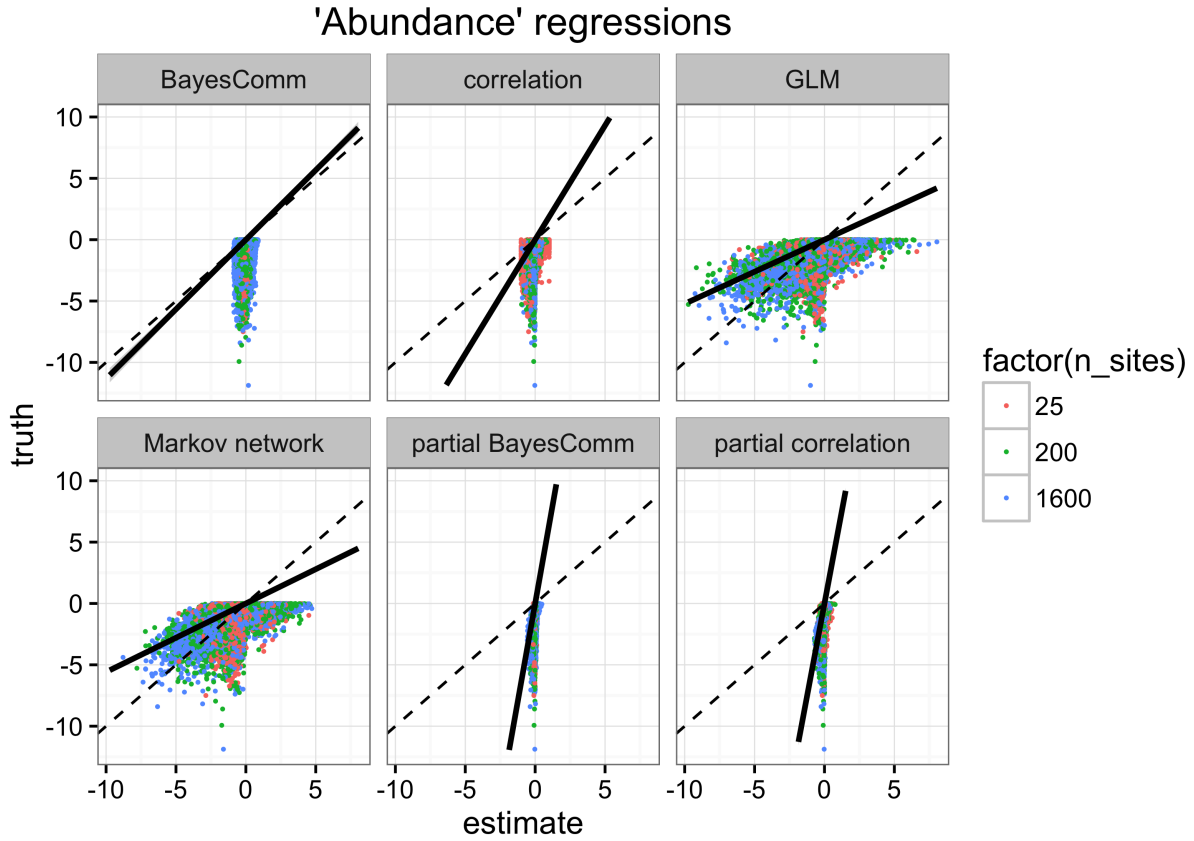
```
plot_regressions("env", "'Environment' regressions")
```

```
## Warning: Removed 188 rows containing missing values (geom_smooth).
```



```
plot_regressions("abund", "'Abundance' regressions")
```

```
## Warning: Removed 155 rows containing missing values (geom_smooth).
```



Average the R-squared values across methods and simulation types

```
result_summary %>%
  group_by(method, simulation_type) %>%
  summarise(mean(r2)) %>%
  spread(simulation_type, `mean(r2)` ) %>%
  kable(digits = 3)
```

method	no_env	env	abund
Markov network	0.525	0.451	0.384
GLM	0.472	0.405	0.283
partial correlation	0.403	0.322	0.200
partial BayesComm	0.394	0.302	0.166
correlation	0.291	0.183	0.117
null	0.227	0.125	0.075
BayesComm	0.206	0.110	0.060

Save the finer-grained R-squared results as Figure 3

```
legend_name = expression(Method~(mean~R2))

pdf("manuscript-materials/figures/performance.pdf", width = 8, height = 2.5)
ggplot(result_summary, aes(x = n_sites, y = r2, col = method_r2, shape = method_r2)) +
  facet_grid(~simulation_type_long) +
  geom_line(size = .5) +
  geom_point(size = 2.5, fill = "white") +
  scale_shape_manual(values = c(16, 22, 17, 23, 18, 24, 15), name = legend_name) +
  geom_hline(yintercept = 0, size = 1/2) +
  geom_vline(xintercept = 0, size = 1) +
  coord_cartesian(ylim = c(-.01, 0.76)) +
  ylab(expression(R2)) +
  xlab("Number of sites (log scale)") +
  scale_x_log10(breaks = unique(x$n_sites), limits = range(x$n_sites)) +
  theme_bw(base_size = 11) +
  theme(panel.margin = grid::unit(1.25, "lines")) +
  theme(panel.border = element_blank(), axis.line = element_blank()) +
  theme(
    panel.grid.minor = element_blank(),
    panel.grid.major.y = element_line(color = "lightgray", size = 1/4),
    panel.grid.major.x = element_blank()
  ) +
  theme(strip.background = element_blank(), legend.key = element_blank()) +
  theme(plot.margin = grid::unit(c(.01, .01, .75, .1), "lines")) +
  theme(
    axis.title.x = element_text(vjust = -0.2, size = 12),
    axis.title.y = element_text(angle = 0, hjust = -.1, size = 12)
  ) +
  scale_color_brewer(palette = "Dark2", name = legend_name)
dev.off()
```

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

Estimate uncertainty in mean R-squared:

Fit a linear mixed model describing R-squared as a function of method, landscape size, and simulation type. Note the small standard errors associated with the effect of estimation method.

```
landscape_estimates = x %>%
  group_by(method, simulation_type) %>%
  do(data.frame(., resids = resids(.))) %>%
  ungroup %>%
  group_by(method, simulation_type, rep_name, n_sites) %>%
```

```

summarise(r2 = 1 - sum(resids^2) / sum(truth^2))

summary(
  lmer(
    r2 ~ method + n_sites + simulation_type + (1|rep_name),
    data = landscape_estimates
  ),
  correlation = FALSE
)

## Linear mixed model fit by REML ['lmerMod']
## Formula: r2 ~ method + n_sites + simulation_type + (1 | rep_name)
## Data: landscape_estimates
##
## REML criterion at convergence: -4945.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -7.3997 -0.6350  0.0758  0.6883  2.7104
##
## Random effects:
## Groups Name Variance Std.Dev.
## rep_name (Intercept) 0.0008304 0.02882
## Residual 0.0113238 0.10641
## Number of obs: 3150, groups: rep_name, 150
##
## Fixed effects:
##              Estimate Std. Error t value
## (Intercept) -2.481e-02  7.186e-03  -3.45
## methodcorrelation 6.965e-02  7.094e-03   9.82
## methodGLM 2.544e-01  7.094e-03  35.85
## methodMarkov network 3.217e-01  7.094e-03  45.35
## methodnull 1.504e-02  7.094e-03   2.12
## methodpartial BayesComm 1.616e-01  7.094e-03  22.78
## methodpartial correlation 1.796e-01  7.094e-03  25.31
## n_sites 1.050e-04  2.690e-06  39.04
## simulation_typeenv 8.853e-02  7.402e-03  11.96
## simulation_typeno_env 1.795e-01  7.402e-03  24.25

```

C: Inferential statistics

Note that, as mentioned in the main text, inferential statistics are only calculated for the `env` and `no_env` simulations (and not for the `abund` simulations).

Identify statistical significance for the Markov network and Pairs

For Pairs, compare the Z-scores with Gaussian quantiles for 95% coverage; for the Markov network, use the approximate confidence intervals estimated in Appendix 3.

```
pairs_summary = x[x$method == "null" & x$simulation_type != "abund", ]
markov_summary = x[x$method == "Markov network" & x$simulation_type != "abund", ]

# Pairs's Z score is based on C-scores, which are positive when species are
# disaggregated. So significantly negative scores imply positive interactions
# and vice versa
pairs_summary$sig_pos = pairs_summary$estimate < qnorm(.025)
pairs_summary$sig_neg = pairs_summary$estimate > qnorm(.975)

# Markov network is significant when lower bound is above zero or lower bound is
# below zero
markov_summary$sig_pos = markov_summary$lower > 0
markov_summary$sig_neg = markov_summary$upper < 0
```

The `error_smoother` is a function that fits a kernel smoother model to estimate the probability that a model estimate will match some criterion (e.g. statistical significance) for some parameter as a function of that parameter's "true" value. This function is used in Figure 4C and in calculating Type I error rates below.

```
truth_seq = seq(min(markov_summary$truth), max(markov_summary$truth), length = 1000)

error_smoother = function(data, f, values = truth_seq){
  out = ksmooth(
    x = data$truth,
    y = as.numeric(f(data)),
    x.points = values,
    kernel = "normal",
    bandwidth = 0.25
  )

  out$y[is.na(out$y)] = 0

  out
}
```

Create Figure 4

Panels A and B plot the point estimates or test statistics for different models against one another.

Panel C shows the probability of rejecting the null hypothesis (of no interaction between two species) in the opposite direction of the true value (see main text).

```
pdf("manuscript-materials/figures/error_rates.pdf", height = 8.5, width = 8.5/3)
par(mfrow = c(3, 1))

# Calculate p(confidently wrong) for each model as a function of
# the true values
confidently_wrong = function(data){
  (data$sig_pos & data$truth < 0) | (data$sig_neg & data$truth > 0)
}
smoothed_markov = error_smoother(markov_summary, confidently_wrong)
smoothed_pairs = error_smoother(pairs_summary, confidently_wrong)

# Compare estimates
spread_estimates = x %>%
  dplyr::select(-lower, -upper, -X) %>%
  spread(method, estimate) %>%
  na.omit()

# R-squared for null versus correlation
round(
  summary(lm(null ~ I(correlation*sqrt(n_sites)), data = spread_estimates))$r.squared,
  2
)
```

```
## [1] 0.95
```

```
# R-squared for glm markov network versus glm
round(summary(lm(`Markov network` ~ GLM, data = spread_estimates))$r.squared, 2)
```

```
## [1] 0.94
```

```
with(
  spread_estimates,
  plot(
    `Markov network`,
    GLM,
    pch = ".",
    col = "#00000020",
    ylab = "Markov network estimate",
    xlab = "GLM estimate",
    bty = "l"
  )
)
mtext("A. Markov network estimates\nvs. GLM estimates", adj = 0, side = 3,
      font = 2, line = 1.2, cex = .9)
abline(lm(`Markov network` ~ GLM, data = spread_estimates))
text(0, 5, expression(R^2==0.94))
```

```

with(
  spread_estimates,
  plot(
    correlation * sqrt(n_sites),
    null,
    pch = ".",
    col = "#00000020",
    ylab = "Z-score",
    xlab = expression("correlation" %*% sqrt(number~~of~~sites)),
    bty = "l"
  )
)
mtext("B. Null model estimates vs.\nscaled correlation coefficients",
      adj = 0, side = 3, font = 2, line = 1.2, cex = .9)
abline(lm(null ~ I(correlation*sqrt(n_sites)), data = spread_estimates))
text(10, 12, expression(R^2==0.95))

plot(
  smoothed_pairs$x,
  smoothed_pairs$y,
  type = "l",
  xlab = "\"True\" interaction strength",
  ylab = "P(confidently predict wrong sign)",
  bty = "l",
  yaxs = "i",
  col = 2,
  ylim = c(0, .3),
  lwd = 2
)
mtext("C. Error rate vs.\ninteraction strength", side = 3, adj = 0,
      font = 2, line = 1.2, cex = .9)
lines(smoothed_markov$x, smoothed_markov$y, lwd = 2)
legend("topleft", lwd = 2, legend = c("Null model", "Markov network"),
      col = c(2, 1), bty = "n")
dev.off()

## pdf
## 2

```

Summarize inferential statistics:

P(Pairs confidently wrong):

```

with(pairs_summary, mean((sig_neg & truth > 0) | (sig_pos & truth < 0)))

```

```
## [1] 0.1234429
```

P(Markov network confidently wrong)

```
with(markov_summary, mean((sig_neg & truth > 0) | (sig_pos & truth < 0)))
```

```
## [1] 0.01529933
```

P(Pairs rejects null)

```
with(pairs_summary, mean(sig_neg | sig_pos))
```

```
## [1] 0.4640395
```

P(Markov network rejects null)

```
with(markov_summary, mean(sig_neg | sig_pos))
```

```
## [1] 0.22179
```

Type I error rates:

Type I error rates (probability of rejecting the null hypothesis given a “true” interaction strength of zero, according to the kernel smoother defined by `error_smoother` above).

Markov network Type I error rates

```
markov_summary %>%
  filter(abs(truth) < 1) %>%
  group_by(simulation_type) %>%
  do(data.frame(
    `Type I error rate` =
      error_smoother(., function(x){x$`sig_pos` | x$`sig_neg`}, values = 0)$y
  )) %>%
  kable(digits = 3)
```

simulation_type	Type.I.error.rate
env	0.138
no_env	0.024

Null model Type I error rates

```

pairs_summary %>%
  filter(abs(truth) < 1) %>%
  group_by(simulation_type) %>%
  do(data.frame(
    `Type I error rate` =
      error_smoother(., function(x){x$sig_pos | x$sig_neg}, values = 0)$y
  )) %>%
  kable(digits = 3)

```

simulation_type	Type.I.error.rate
env	0.510
no_env	0.298

Plot Markov network confidence interval coverage versus true β value

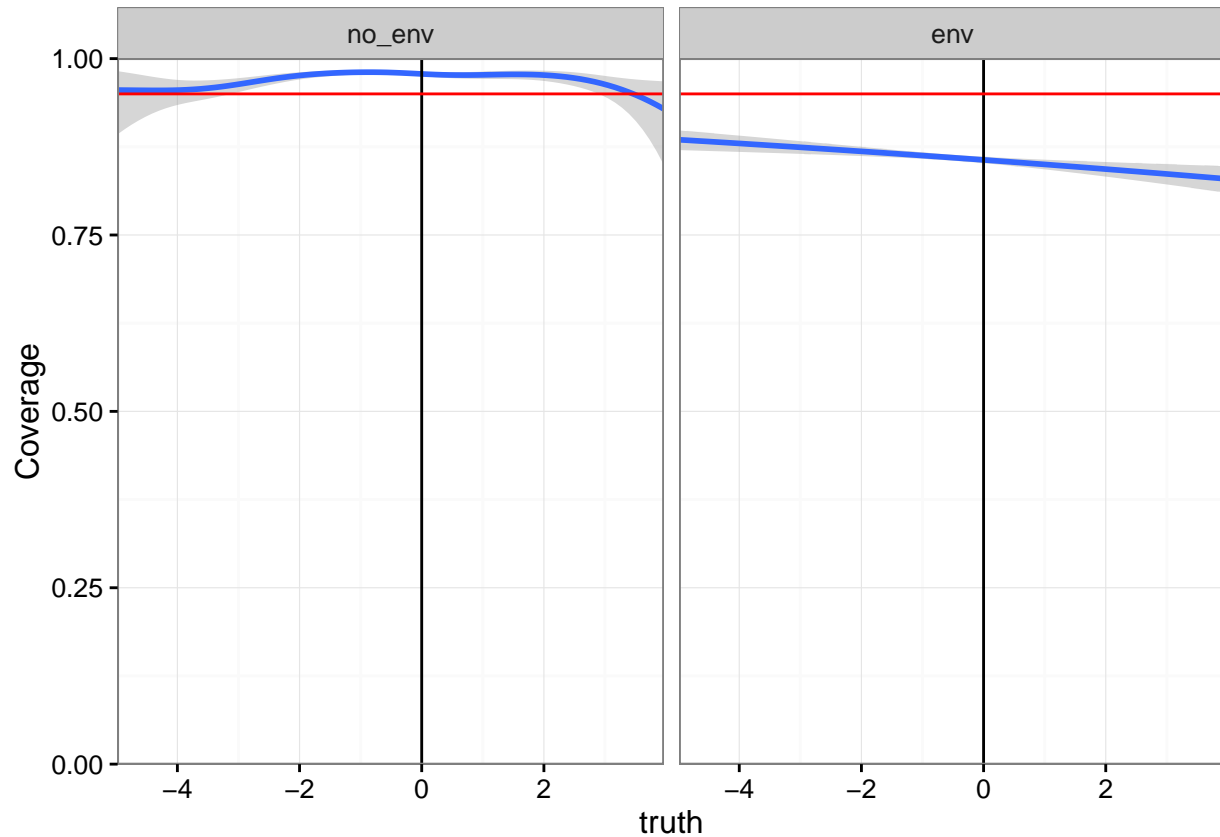
The top and bottom 0.5% of the distribution have been omitted to prevent bad behavior by the smoother (a generalized additive model) in the tails. The horizontal red line shows the nominal 95% coverage rate and the black vertical line marks where the “true” value of β was zero. The Type I error rates calculated above for the Markov network closely match (one minus) the values of these curves at zero.

```

coverage_data = markov_summary %>%
  filter(percent_rank(truth) > .005 & percent_rank(truth) < .995) %>%
  mutate(covered = truth > lower & truth < upper) %>%
  mutate(simulation_type = factor(simulation_type, c("no_env", "env", "abund")))

ggplot(coverage_data, aes(x = truth, y = as.integer(covered))) +
  facet_grid(~simulation_type) +
  geom_smooth(method = gam, formula = y ~ s(x), method.args = list(family = binomial)) +
  theme_bw() +
  coord_cartesian(ylim = c(0, 1), expand = FALSE) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0.95, color = "red") +
  ylab("Coverage")

```



Plot power versus true β value

Calculate $p(\text{confidently detect sign of } \beta)$ as a function of the “true” values. Pairs has a somewhat higher rate of rejecting the null hypothesis in the correct direction than the Markov network, but the benefit of this increase in power is more than offset by the large increases in Type I errors and sign errors demonstrated above and in Figure 4 of the main text.

```
confidently_right = function(data){
  (data$sign_pos & data$truth > 0) | (data$sign_neg & data$truth < 0)
}

plot(error_smoother(markov_summary, confidently_right),
     type = "l",
     xlim = c(-5, 5),
     xlab = "'True' coefficient value",
     ylab = "Power",
     ylim = c(0, 1),
     yaxp = "i"
)
lines(error_smoother(pairs_summary, confidently_right), col = 2)
legend("topleft", legend = c("Markov network", "Pairs"), lty = 1, col = 1:2)
```