Simulation: Sample DAGs with local structure

Vera Kvisgaard

2024-05-24

Prep

```
library(doSNOW)
```

```
## Loading required package: foreach
```

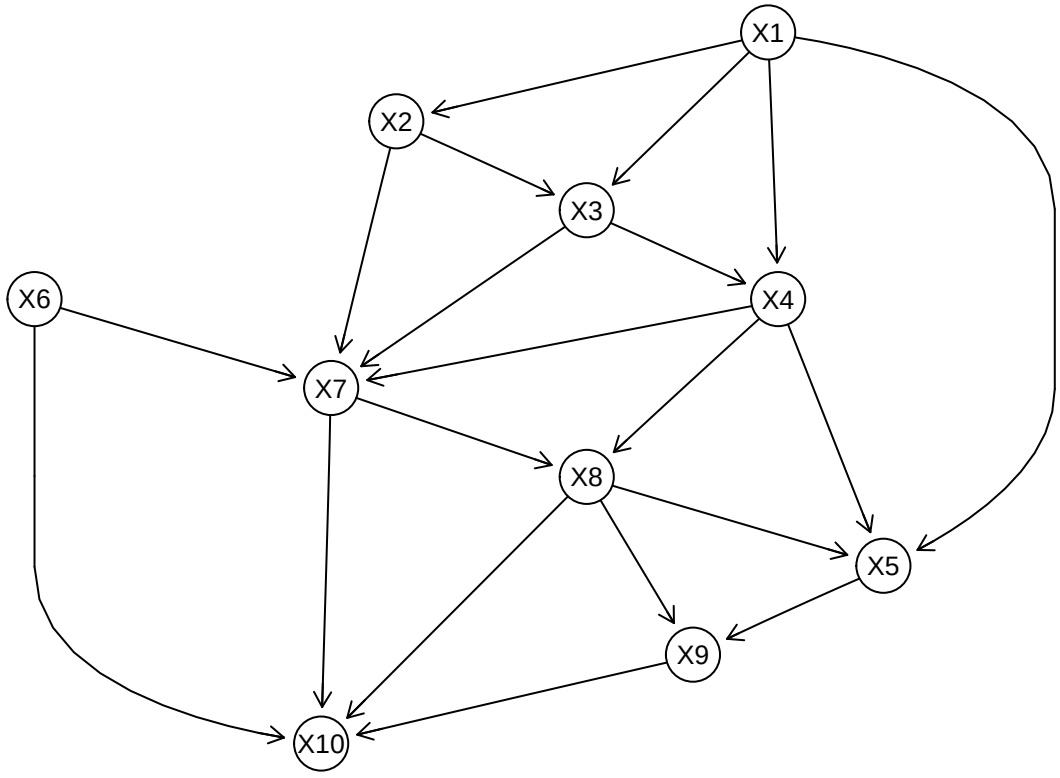
```
## Loading required package: iterators
```

```
## Loading required package: snow
```

```
library(BiDAG)
library(ldags)
simpar <- expand.grid(list(r = 1:25,
                           N = c(100, 300, 1000)))
cl <- makeCluster(4)
doRun <- FALSE
```

Draw a random network based on labeled DAG with 10 nodes.

```
bn <- ldags:::example_bn("LDAG10")
n  <- length(bn)
dindx <- diag(n) == 1
Rgraphviz::plot(bnlearn::as.graphNEL(bn))
```



Run

Define simulation routine.

```
run <- function(bn, r, N, write_to_file = T, verbose = T)  {

  # sample data
  set.seed(r+N)
  data <- bida:::sample_data_from_bn(bn, N)
  nlev <- sapply(bn, function(x) dim(x$prob)[1])

  smpls <- list()
  for (m in c("DAG", "ldag", "tree")) {

    filename <- sprintf("partitionMCMC_%s_N%s_r%02.0f.rds", m, N, r)
    filepath <- here::here("./simulations/LDAG10/MCMCchains/", filename)
    if (file.exists(filepath) && write_to_file) next

    if (verbose) cat(filename)

    # define scoreparameters
    if (tolower(m) == "dag") {
      scorepar <- BiDAG::scoreparameters("bdecat", data = as.data.frame(data), bdecatpar = list(chi = 1
    } else {
      scorepar <- BiDAG::scoreparameters("usr", data = data, usrpar = list(pctesttype = "bdecat"))
```

```
      scorepar$Cvec <- nlev
      scorepar$levels <- lapply(nlev-1, seq.int, from = 0)
      scorepar$lookup <- ldags:::init_lookup_scoretable(length(bn), list(ess = 1))
      scorepar$optPartMethod <- m
     # scorepar$lookup[[m]]  <- rep(list(list()), length(nlev))
      tmp <- rep(list(list()), length(nlev))
      assign(m, tmp, env = scorepar$lookup)

      usrDAGcorescore <- function(j, parentnodes, n, scorepar) {
      ldags:::score_from_lookup(scorepar$data,
                        levels = scorepar$levels,
                        nlev = scorepar$Cvec,
                        j,
                        parentnodes,
                        method = scorepar$optPartMethod,
                        lookup = scorepar$lookup)
      }
      assignInNamespace("usrDAGcorescore", usrDAGcorescore, ns = "BiDAG")
    }


    # sample DAGs using partition MCMC
    # - use try() to avoid simulation stopping, e.g. at "parent-set-size-too-small-errors"
    tic <- Sys.time()
    iterfit <- BiDAG::learnBN(scorepar, "orderIter", hardlimit = 5, scoreout = T, verbose = F)
    smpl     <- try(BiDAG::sampleBN(scorepar, "partition", scoretable = BiDAG::getSpace(iterfit)))
    attr(smpl, "toc") <- (Sys.time()-tic)
    print(Sys.time()-tic)

    if (class(smpl) == "try-error") {
      cat("\nError at sim ", filename, ":\n")
      smpls[[filename]] <- smpl
    } else if (write_to_file) {
      saveRDS(smpl, filepath)
    } else {
      smpls[[filename]] <- smpl
    }
  }
  return(smpls)
}
```

```
if (doRun) {
  # test
  res <- run(bn, 1, 100, F, T)

  if (length(cl) > 1) {
    registerDoSNOW(cl)
    foreach(r = simpar$r, N = simpar$N, m = simpar$method,
         .export = c("get_scorepar"),
         .packages = c("ldags", "BiDAG")) %dopar% run(bn, r, N, T, T)
    stopCluster(cl)
  } else {
    for (i in 1:nrow(simpar)) {
      run(bn, simpar$r[i], simpar$N[i], T, T)
```

```
    }
  }
}
```

Evaluate

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

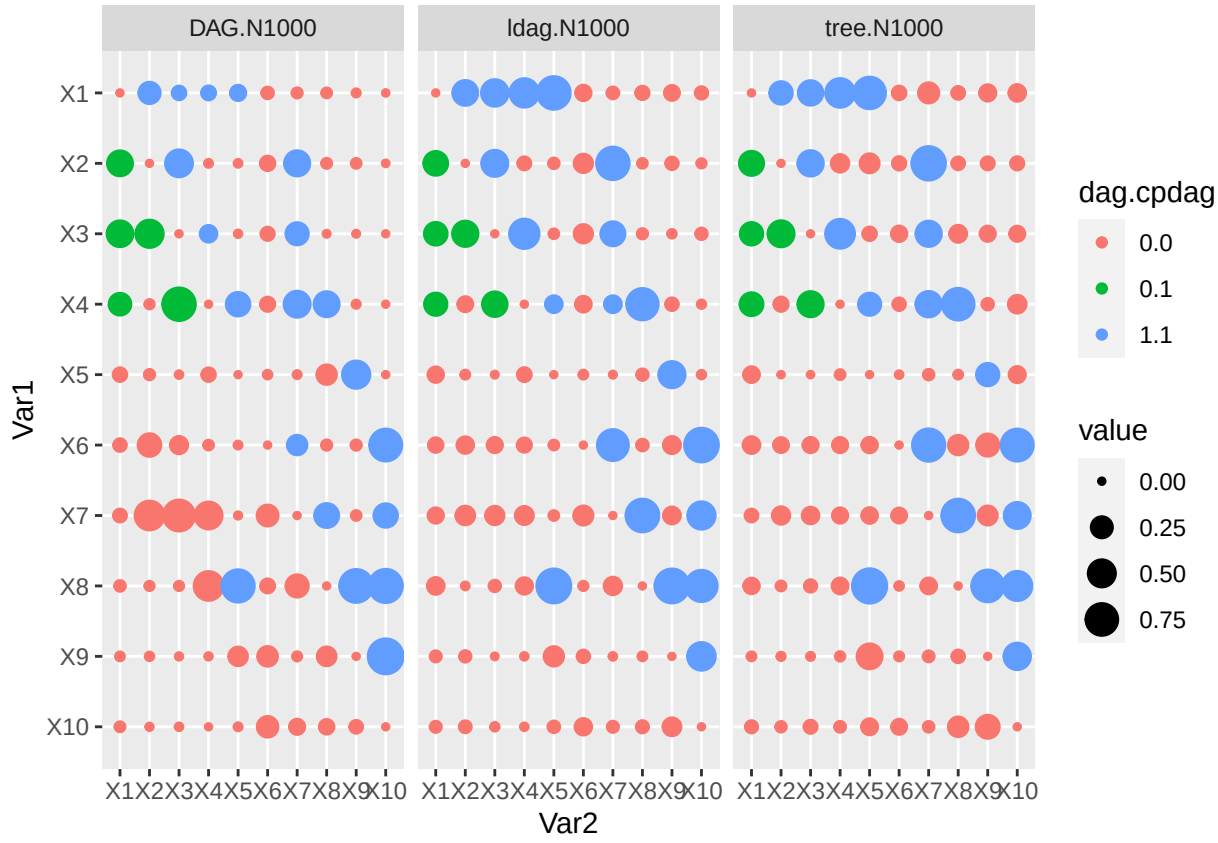
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
filepaths <- list.files(here::here("./simulations/LDAG10/MCMCchains"), pattern = ".rds", full.names = T
indx <- apply(sapply(sprintf("partitionMCMC.+_N%s_r%02.0f.rds", simpar$N, simpar$r), grepl, x = filepath

par <- data.frame(stringr::str_split(filepaths[indx], ".+MCMCchains/|_|.rds", simplify = T)[, 2:5])
colnames(par) <- c("alg", "locals", "N", "r")
par$N <- factor(par$N, c("N100", "N300", "N1000", "N3000"))
```

```
compute_posterior_probs <- function(smpl, burninsamples) {

  # list unique DAGs
  dags <- lapply(smpl$traceadd$incidence[-burninsamples], as.matrix)
  u    <- unique(dags)
  support  <- bida:::rowsum_fast(rep(1/length(dags), length(dags)), dags, u)
  dags <- u
  dmats <- lapply(dags, bida:::descendants)

  list(edgep = Reduce("+", Map("*", dags, support)),
       ancp  = Reduce("+", Map("*", dmats, support)))
}

probs <- lapply(filepaths[indx], function(f) compute_posterior_probs(readRDS(f), seq_len(200)))
edgeps <- lapply(probs, "[[", "edgep")
avgs <- lapply(split(edgeps, par[, c("locals", "N")], drop = T),
                  function(x) Reduce("+", x)/length(x))

plot_edgep <- function(mats, dag, cpdag) {
  df <- cbind(expand.grid(dimnames(mats[[1]])),
              dag.cpdag = interaction(dag, cpdag),
              sapply(mats, c))
  df_long <- tidyr:::pivot_longer(df, names(mats))
  plot <- ggplot(df_long, aes(Var2, Var1, size = value, color = dag.cpdag)) +
    facet_grid(.~name) +
    geom_point() +
    scale_y_discrete(limits=rev)
```

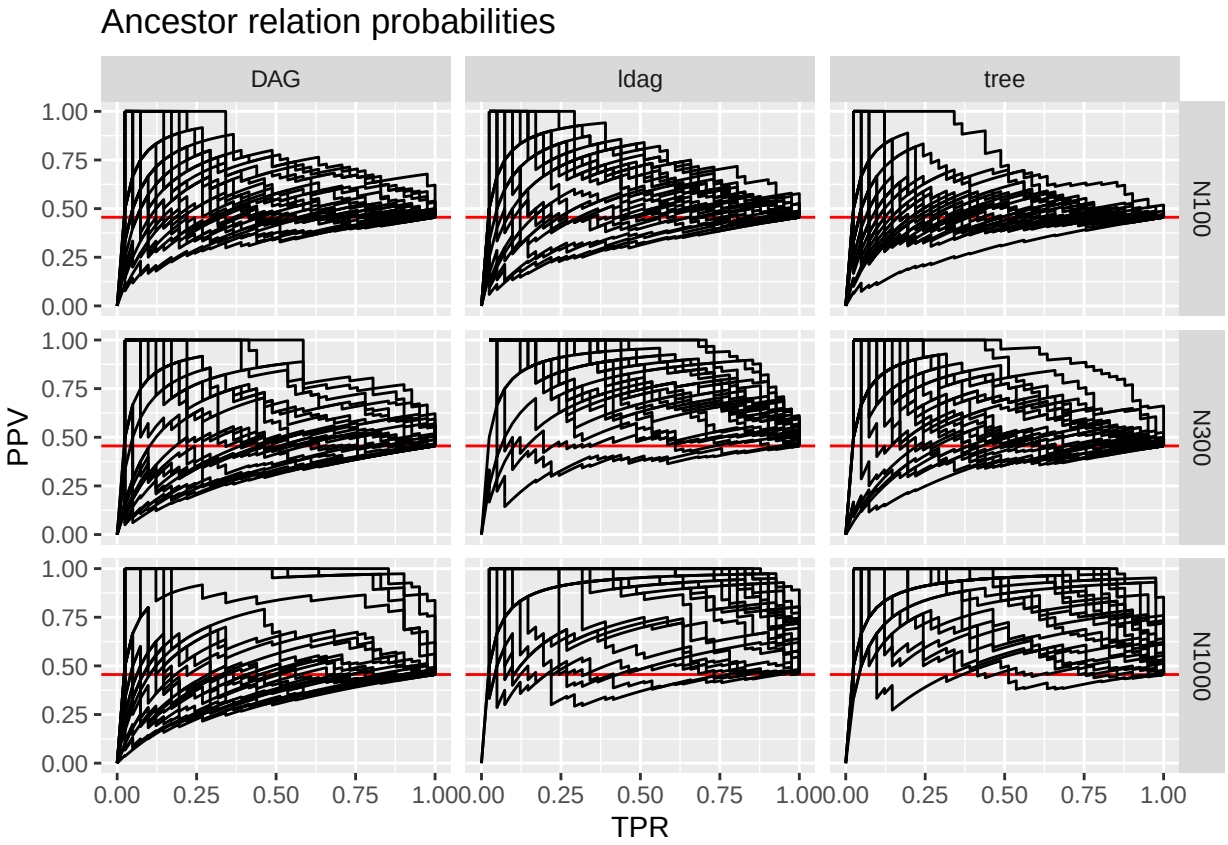
```
  return(plot)
}

plot_edgep(avgs[grepl("N1000", names(avgs))], bnlearn::amat(bn), as(pcalg::dag2cpdag(bnlearn::as.graphN
```

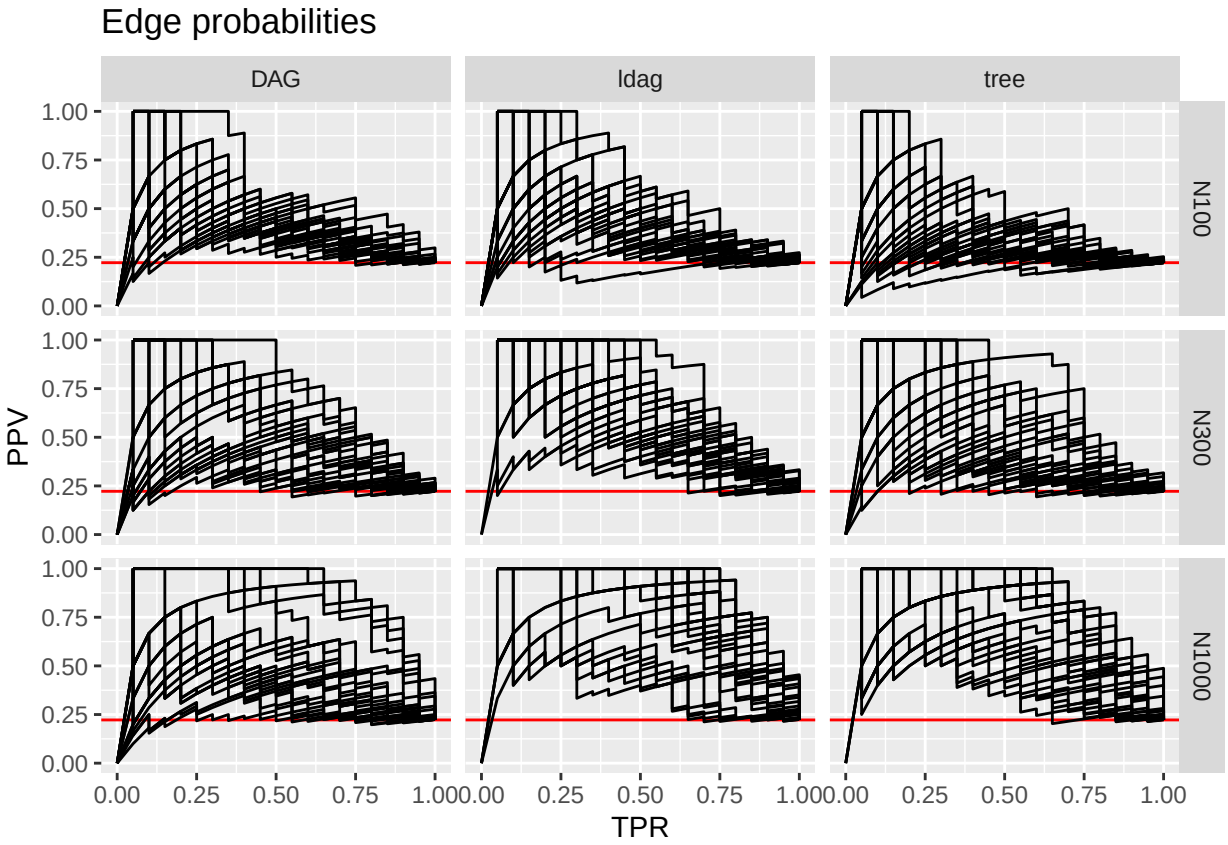


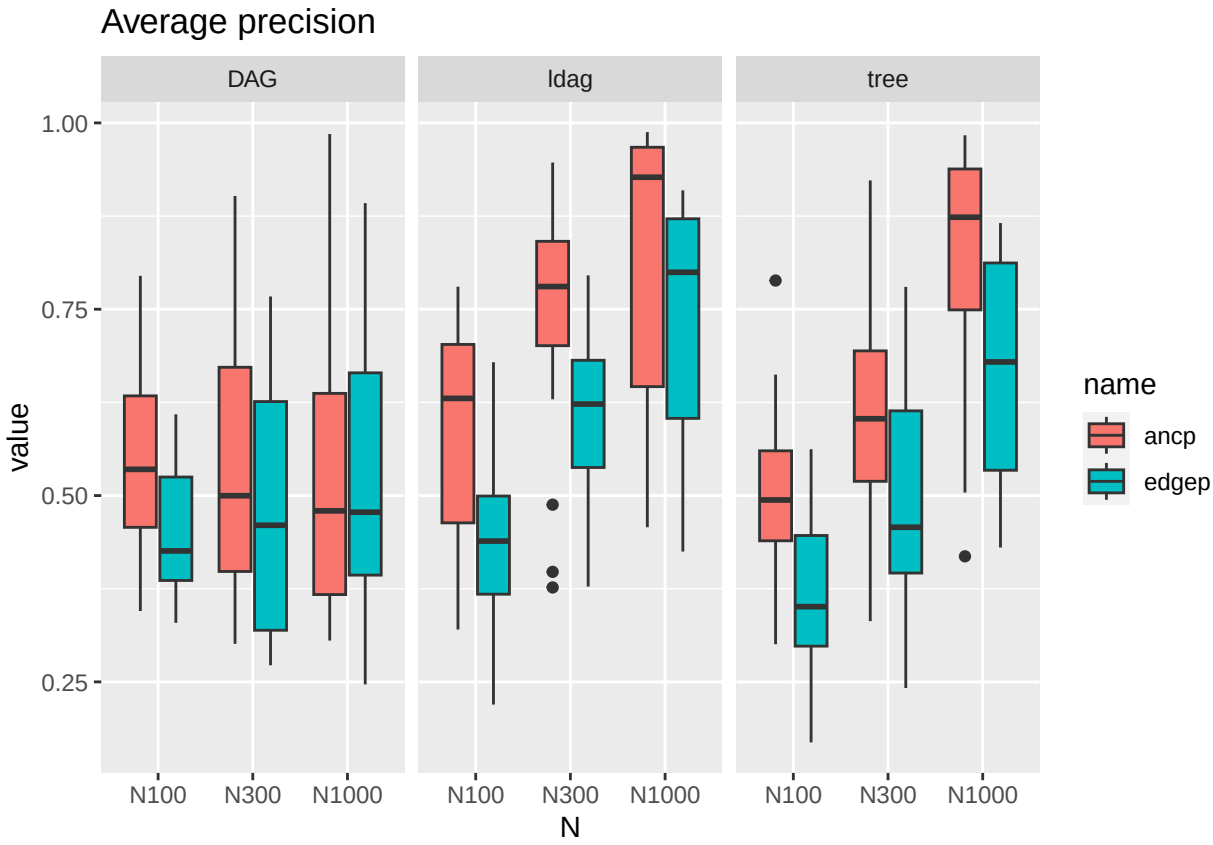

```
# evaluate ----
eval_smpl <- function(smpl, burninsamples, dag, dmat) {
  dindx <- diag(ncol(dag)) == 1

  # list unique DAGs
  dags <- lapply(smpl$traceadd$incidence[-burninsamples], as.matrix)
  u    <- unique(dags)
  support  <- bida:::rowsum_fast(rep(1/length(dags), length(dags)), dags, u)
  dags <- u
  dmats <- lapply(dags, bida:::descendants)

  edgep <- Reduce("+", Map("*", dags, support))[!dindx]
  ancp  <-  Reduce("+", Map("*", dmats, support))[!dindx]

  # compute edge prob and ARP
  list(edgep = compute_prec_recall(edgep, dag[!dindx]),
       ancp =  compute_prec_recall(ancp, dmat[!dindx]),
       avgppv = c(edgep = avgppv(edgep, which(dag[!dindx]==1)),
                  ancp = avgppv(ancp, which(dmat[!dindx]==1))))
}
```

```r
compute_prec_recall <- function(x, y) {
  indx <- order(x + runif(length(x))/1000, decreasing = TRUE)
  tp  <- cumsum(y[indx])
  cbind(x = x[indx], TPR = tp/sum(y), PPV = tp/seq_along(y))
}

# avg ppv
avgppv <- function(x, which_true) {
  np  <- (length(x)-rank(x+runif(length(x))/1000) +1)  # number of instances with equal to or lower val
  pp  <- np[which_true]                                # number of positive predictions at each true pos
  ppv <- seq_along(pp)/sort(pp)                        # precision at each positive
  mean(ppv)
}
```

```r
res <- lapply(filepaths[indx],
              function(f) eval_smpl(readRDS(f), seq_len(200), bnlearn::amat(bn), bida:::descendants(bn))
# res <- list()
# for (f in filepaths[indx]) {
#   res[[f]] <- eval_smpl(readRDS(f), seq_len(200), bnlearn::amat(bn), bida:::descendants(bn))
# }


## ancestor relation
df <- data.frame(par[rep(seq_along(res), each = n*(n-1)), ],
                 do.call(rbind, lapply(res, "[[", "ancp")))

ggplot(df, aes(TPR, PPV, group = interaction(N, r))) +
  facet_grid(N~locals) +
  geom_hline(yintercept = mean(bida:::descendants(bn)[!dindx]), color = "red") +
  geom_line() +
  ggtitle("Ancestor relation probabilities")



## edge probs
df <- data.frame(par[rep(seq_along(res), each = n*(n-1)), ],
                 do.call(rbind, lapply(res, "[[", "edgep")))

ggplot(df, aes(TPR, PPV, group = interaction(N, r))) +
  facet_grid(N~locals) +
  geom_hline(yintercept = mean(bnlearn::amat(bn)[!dindx]), color = "red") +
  geom_line() +
  ggtitle("Edge probabilities")


## avgppv
df <- data.frame(par,
                 do.call(rbind, lapply(res, "[[", "avgppv")))
df_long <- tidyr:::pivot_longer(df, cols = c("edgep", "ancp"))
ggplot(df_long, aes(N, value, fill = name)) +
  facet_grid(.~locals) +
  geom_boxplot() +
  ggtitle("Average precision")
```


Ancestor relation probabilities
DAG
ldag
tree
N100
N300
N1000
PPV
TPR
1.00
0.75
0.50
0.25
0.00


Edge probabilities
DAG
ldag
tree
N100
N300
N1000
PPV
TPR
1.00
0.75
0.50
0.25
0.00


Average precision
DAG
ldag
tree
1.00
0.75
0.50
0.25
value
N100
N300
N1000
N
name
ancp
edgep