

# Ssuis\_Analysis-Manuscript

Brittany L Morgan Bustamante

Updated: 2025-12-17

## 1. Data Wrangling

```
# subset data
suis.pic <- filter(suis.df, source == "PIC")

# isolate identifying variables
suis.pic2 <- suis.pic %>%
  dplyr::select(source, result_name, interpretation,,
                site_name, siteID, yearrec, accessionID, animal_tag) %>%
  distinct(site_name, siteID, accessionID, animal_tag, yearrec) %>%
  dplyr::mutate(sampleflag = row_number())

# long to wide
suis.pic3 <- suis.pic %>%
  left_join(suis.pic2, c("site_name", "siteID", "accessionID", "yearrec",
                        "animal_tag")) %>%
  dplyr::select(sampleflag, source, result_name, interpretation,
                site_name, siteID, yearrec, accessionID, animal_tag) %>%
  pivot_wider(names_from = result_name, values_from = c(interpretation))

# isolate identifying variables - sensitivity analysis
sens_suis.pic2 <- suis.pic %>%
  dplyr::select(source, result_name, sens_interpretation,,
                site_name, siteID, yearrec, accessionID, animal_tag) %>%
  distinct(site_name, siteID, accessionID, animal_tag, yearrec) %>%
  dplyr::mutate(sampleflag = row_number())

# long to wide - sensitivity analysis
sens_suis.pic3 <- suis.pic %>%
  left_join(sens_suis.pic2, c("site_name", "siteID", "accessionID", "yearrec",
                            "animal_tag")) %>%
  dplyr::select(sampleflag, source, result_name, sens_interpretation,
                site_name, siteID, yearrec, accessionID, animal_tag) %>%
  pivot_wider(names_from = result_name, values_from = c(sens_interpretation))
```

## 2. Descriptive Statistics

```

MICResults <- suis.df %>%
  group_by(source, result_name, interpretation) %>%
  dplyr::summarize(n_samples = n()) %>%
  ungroup()

```

Will remove Ampicillin, Florfenicol, and Trimethoprim-Sulphamethoxazole because less than ten isolates showed phenotypic susceptibility/resistance for each.

### 3. Analytic Dataframe

```

# select AMDs with large enough cell counts
library(gtsummary)

suis.amd <- suis.pic3 %>%
  select(siteID, Ceftiofur, Chlortetracycline, Clindamycin, Enrofloxacin,
         Gentamicin, Neomycin, Oxytetracycline, Penicillin, Spectinomycin,
         Sulphadimethoxine, Tetracycline, Tiamulin, Tilmicosin)
# suis.amd %>%tbl_summary()

# create factors for variables
suis.amd$siteID <- (as.factor(suis.amd$siteID))
suis.amd$CEF <- (as.factor(suis.amd$Ceftiofur))
suis.amd$ENR <- (as.factor(suis.amd$Enrofloxacin))
suis.amd$GEN <- (as.factor(suis.amd$Gentamicin))
suis.amd$NEO <- (as.factor(suis.amd$Neomycin))
suis.amd$PEN <- (as.factor(suis.amd$Penicillin))
suis.amd$SPC <- (as.factor(suis.amd$Spectinomycin))
suis.amd$SUL <- (as.factor(suis.amd$Sulphadimethoxine))
suis.amd$TIA <- (as.factor(suis.amd$Tiamulin))
suis.amd$TIL <- (as.factor(suis.amd$Tilmicosin))
suis.amd$CHL <- (as.factor(suis.amd$Chlortetracycline))
suis.amd$CLN <- (as.factor(suis.amd$Clindamycin))
suis.amd$OXY <- (as.factor(suis.amd$Oxytetracycline))
suis.amd$TET <- (as.factor(suis.amd$Tetracycline))

# keep only variables for analysis
pic.analyze <- as.data.frame(suis.amd) %>%
  select(siteID, CEF, ENR, GEN, NEO, PEN, SPC, SUL, TIA, TIL, CHL, CLN, OXY, TET)

# pic.analyze %>%tbl_summary()

# total number of sites
sites <- distinct(suis.pic3, siteID)

# drop sites with no retained data
pic.analyze <- pic.analyze %>%
  mutate(siteID = droplevels(siteID))

```

## 4. Bayesian Network Analysis

Will use the structure learning algorithm, tabu. Tabu is a score-based algorithm, a modified greedy hill-climbing search. Score-based algorithms use goodness-of-fit scores as objective functions to maximize fit score (in this case, BIC).

To smooth over missing data, *Structural.em* is used. Structural.em is the “Expectation-Maximization” algorithm, an iterative procedure computing the expected sufficient statistics conditional on the observed data using belief propagation, then applying complete-data learning methods using expected sufficient statistic. EM starts from the model and estimates/identifies the sufficient statistics needed to estimate its parameters and completes those statistics by averaging over the missing values.

Two methods of imputation will be tested: parents and Bayes. Parents method computes predicted values by plugging in new values for the parents node in the local probability distribution of “node” extracted from “fitted” DAG. Bayes method computes predicted values by averaging likelihood weighting simulations performed using all the nodes as evidence. The default number of random samples drawn is 500.

To account for clustering of observations at the siteID level, we create tiers that allow siteID to be a background or context variable, now be the child of any lower-level variable (blocklist) and allows the lower-level variables to connect among themselves.

```
# create levels
amd_vars <- c("CEF", "ENR", "GEN", "NEO", "PEN", "SPC", "SUL",
             "TIA", "TIL", "CHL", "CLN", "OXY", "TET")

tiers <- list(c("siteID"), amd_vars)

bl <- tiers2blocklist(tiers)

library(bnlearn)

# impute with parents method
pic.tabu <- structural.em(pic.analyze, maximize = "tabu",
                            maximize.args = list(score = "bic",
                                                  blacklist = bl),
                            impute = "parents", return.all = TRUE)

# impute with bayes method
pic.tabu2 <- structural.em(pic.analyze, maximize = "tabu",
                            maximize.args = list(score = "bic",
                                                  blacklist = bl),
                            impute = "bayes-lw", return.all = TRUE)
```

Bootstrap analysis identifies which arcs are the strongest and most consistent. Will run 10,000 iterations. For each sample, the previous steps are repeated, and a separate network is learned using the score-based learning algorithm and BIC test. The *boot.strength* function returns the strength of connection for each pair of nodes (i.e., how frequently the connection is observed). This information is used to build a consensus network, defined as a network containing arcs having a strength >50%.

There will be lots of warnings that pop up because some sites have few data points and it just means some bootstrap samples don't include all sites. So, we suppress the warnings

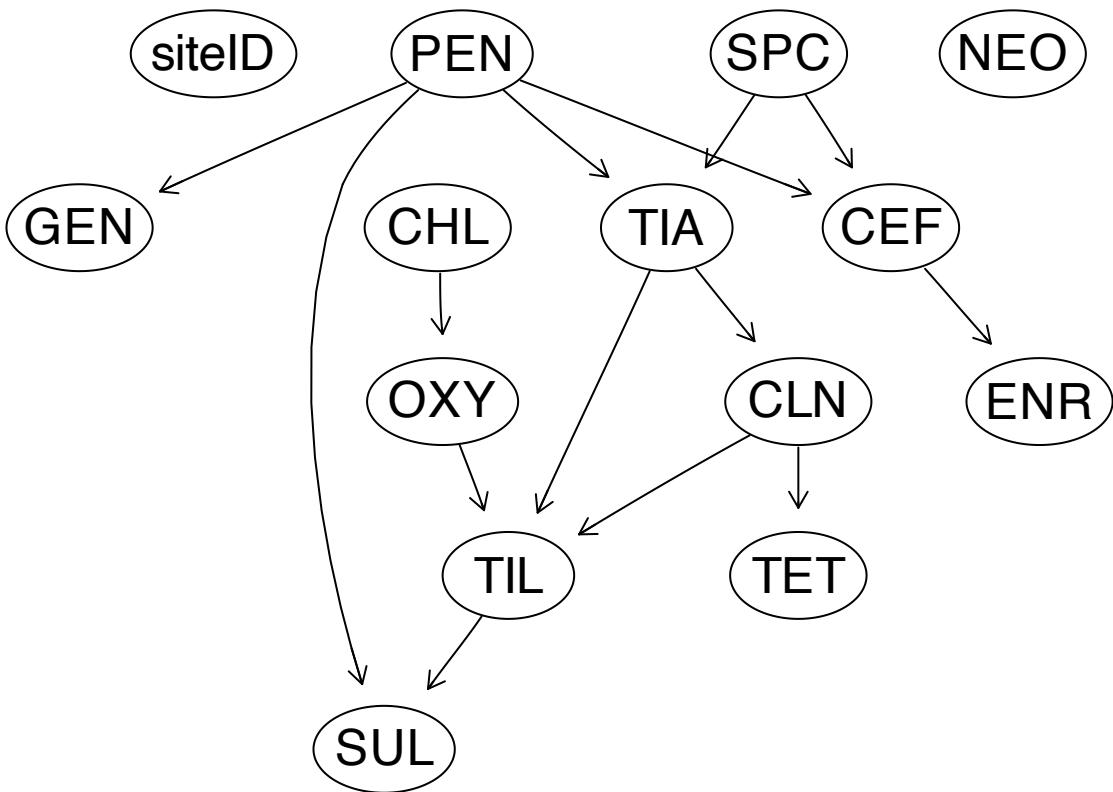
```
# bootstrap - parents
pic.boot = suppressWarnings(
  boot.strength(pic.tabu$imputed, R = 10000, algorithm = "tabu",
```

```

algorithm.args = list(score = "bic",
                      blacklist = bl))

# take arcs with strength of threshold and return average consensus network - parents
pic.avg = averaged.network(pic.boot, threshold = 0.5)
graphviz.plot(pic.avg, shape = "ellipse", main = NULL, sub = "Bayesian Network Analysis - S. suis isolates from PIC w/ parents imputation")

```

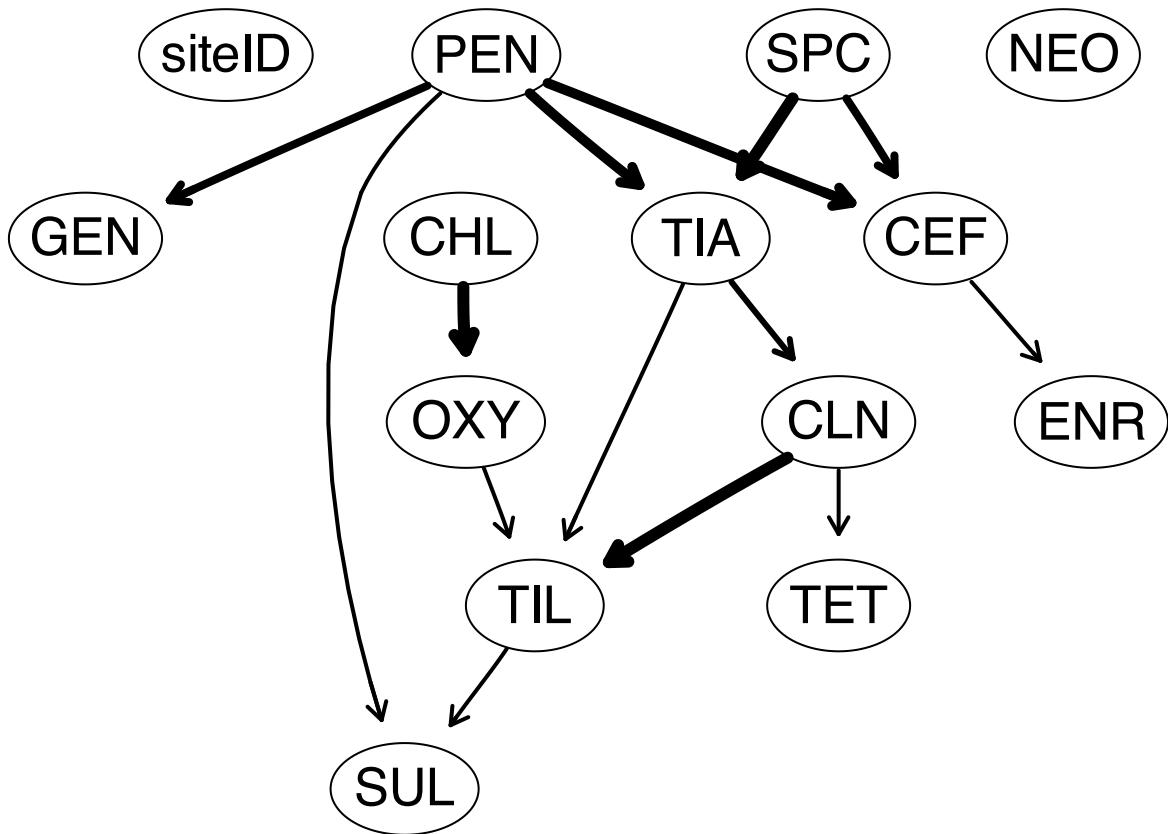


Bayesian Network Analysis - S. suis isolates from PIC w/ parents imputation

```

# plot DAG with strength of association information - parents
strength.plot(pic.avg, pic.boot, shape = "ellipse")

```

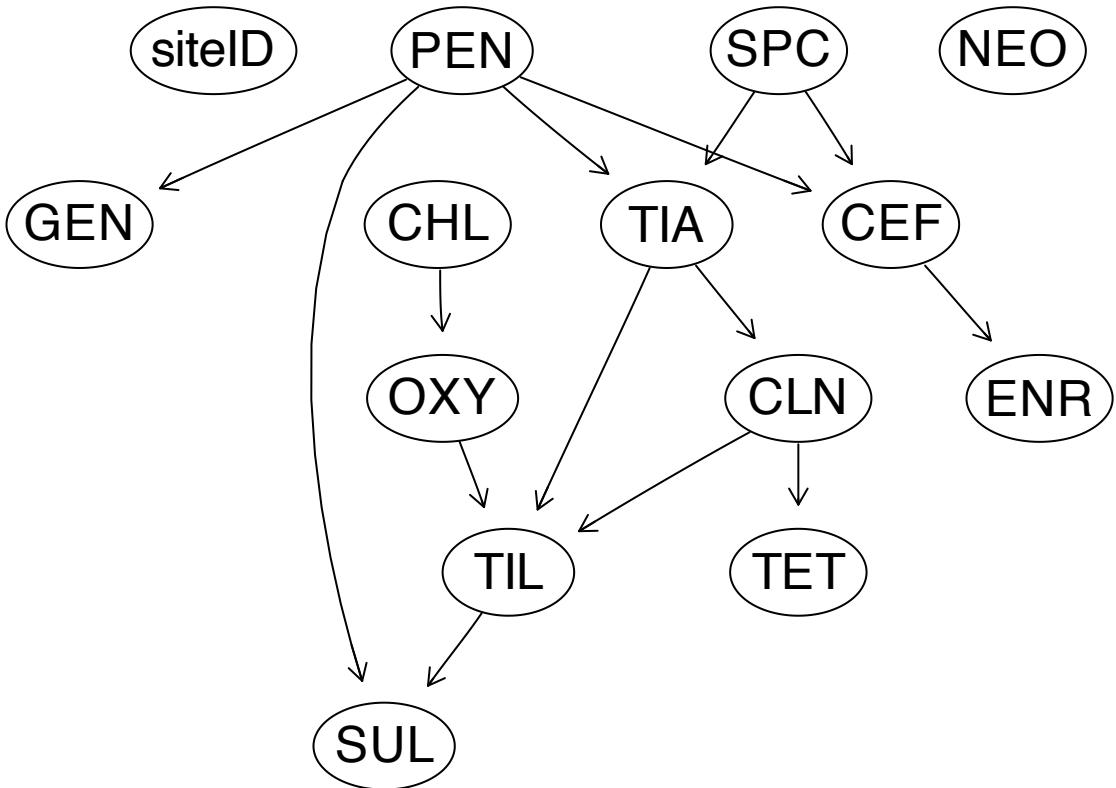


```

# bootstrap - bayes
pic.boot2 = suppressWarnings(boot.strength(
  pic.tabu2$imputed, R = 10000, algorithm = "tabu",
  algorithm.args = list(score = "bic",
    blacklist = bl)))

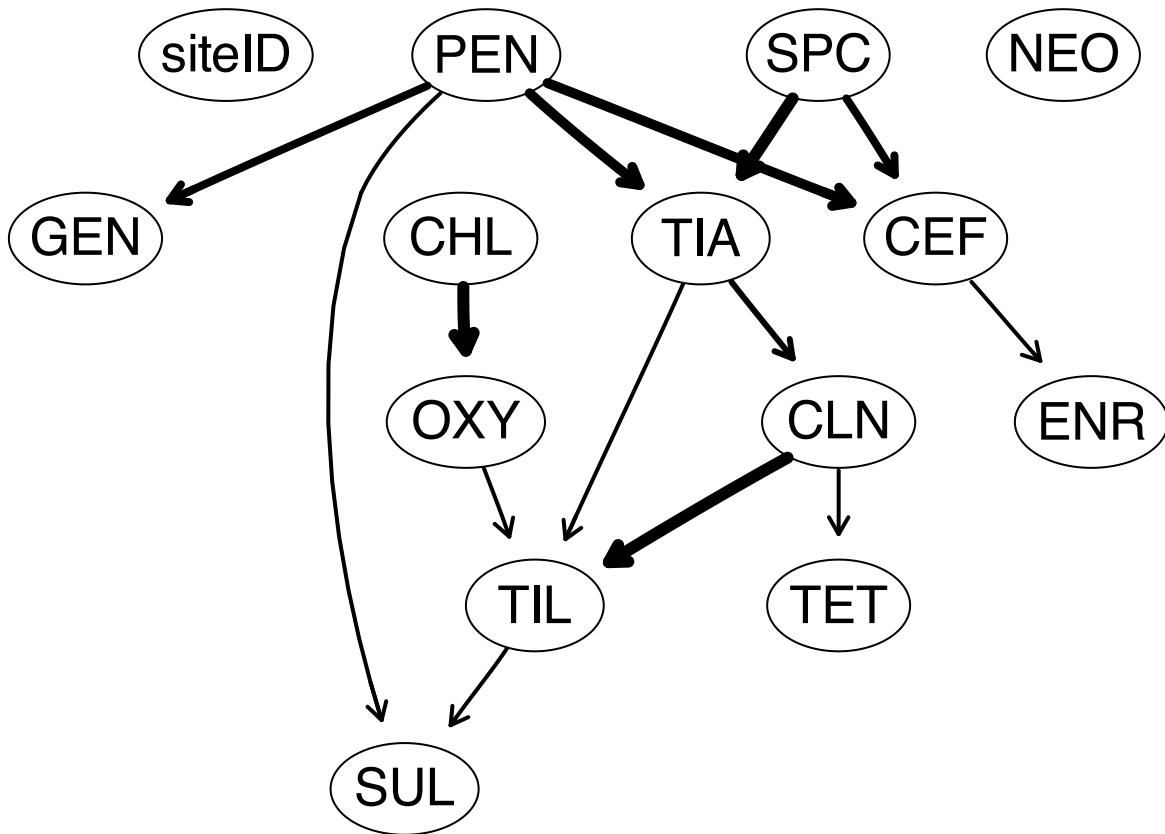
# take arcs with strength of threshold and return average consensus network - bayes
pic.avg2 = averaged.network(pic.boot2, threshold = 0.5)
graphviz.plot(pic.avg2, shape = "ellipse", main = NULL, sub = "Bayesian Network Analysis - S. suis isolate")

```



Bayesian Network Analysis - *S. suis* isolates from PIC w/ bayes imputation

```
# plot DAG with strength of association information - bayes
strength.plot(pic.avg2, pic.boot2, shape = "ellipse")
```



Undirected strength plot using Bayes imputation method for publication

```

library(igraph)
library(ggraph)
library(scales)

# convert boot result + averaged network into an undirected graph
nodes_df <- data.frame(name = nodes(pic.avg2))
boot_df  <- as.data.frame(pic.boot2)
avg_arcs <- as.data.frame(arcs(pic.avg2))

edges <- avg_arcs %>%
  left_join(boot_df[, c("from", "to", "strength")],
            by = c("from", "to"))

g <- graph_from_data_frame(edges, directed = FALSE, vertices = nodes_df)

# strength plot without arrows
set.seed(123) # for reproducible layout

# create layout
lay <- create_layout(g, layout = "nicely")

# find the row corresponding to siteID
idx_site <- which(lay$name == "siteID")
  
```

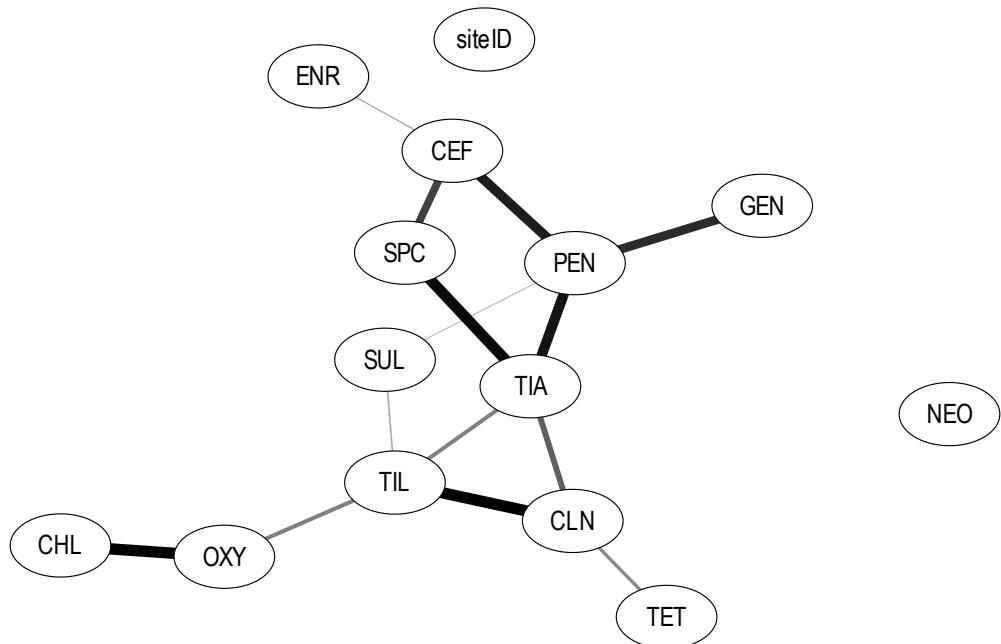
```

# compute a horizontal center for the main network (all non-siteID nodes)
center_x <- mean(lay$x[lay$name != "siteID"])

# put siteID above the rest
lay$x[idx_site] <- center_x
lay$y[idx_site] <- max(lay$y) + 0.35

# plot
ggraph(lay) +
  geom_edge_link(aes(width = strength, alpha = strength),
                 color = "black") +
  geom_node_circle(aes(r = 0.3),
                   fill = "white",
                   color = "black",
                   size = 0.2) +
  geom_node_text(aes(label = name),
                size = 3.5) +
  scale_edge_width(range = c(0.2, 2)) +
  scale_edge_alpha(range = c(0.3, 1)) +
  guides(edge_width = "none", edge_alpha = "none") +
  theme_graph()

```



```
# ggsave("full_network_plot.png", width = 8, height = 6, dpi = 300)
```

## 5. Parameter Learning

```
# fit the parameters - maximum likelihood method
pic.fit = bn.fit(pic.avg, pic.analyze, method = "bayes")

# PIC probability table
# pic.fit

set.seed(112)

# PEN - GEN
cpquery(pic.fit, (PEN == "Resistant"), GEN == "Resistant")

## [1] 0.8058036

cpquery(pic.fit, (PEN == "Resistant"), GEN == "Susceptible")

## [1] 0.07248954

cpquery(pic.fit, (PEN == "Susceptible"), GEN == "Resistant")

## [1] 0.1924883

cpquery(pic.fit, (PEN == "Susceptible"), GEN == "Susceptible")

## [1] 0.9186483

set.seed(134)
cpquery(pic.fit, (GEN == "Resistant"), PEN == "Resistant")

## [1] 0.345735

cpquery(pic.fit, (GEN == "Resistant"), PEN == "Susceptible")

## [1] 0.009798401

cpquery(pic.fit, (GEN == "Susceptible"), PEN == "Resistant")

## [1] 0.6524113

cpquery(pic.fit, (GEN == "Susceptible"), PEN == "Susceptible")

## [1] 0.9890358
```

```

set.seed(113)

# PEN - CEF
cpquery(pic.fit, (PEN == "Resistant"), CEF == "Resistant")

## [1] 0.7740741

cpquery(pic.fit, (PEN == "Resistant"), CEF == "Susceptible")

## [1] 0.09154495

cpquery(pic.fit, (PEN == "Susceptible"), CEF == "Resistant")

## [1] 0.2906977

cpquery(pic.fit, (PEN == "Susceptible"), CEF == "Susceptible")

## [1] 0.9047766

set.seed(114)

# SUL - PEN
cpquery(pic.fit, (SUL == "Resistant"), PEN == "Resistant")

## [1] 0.9239033

cpquery(pic.fit, (SUL == "Resistant"), PEN == "Susceptible")

## [1] 0.6613408

cpquery(pic.fit, (SUL == "Susceptible"), PEN == "Resistant")

## [1] 0.07201459

cpquery(pic.fit, (SUL == "Susceptible"), PEN == "Susceptible")

## [1] 0.32953

set.seed(114)

# SUL - PEN - TIL
cpquery(pic.fit, (SUL == "Resistant"), PEN == "Resistant" & TIL == "Resistant")

## [1] 0.8890511

```

```

cpquery(pic.fit, (SUL == "Resistant"), PEN == "Resistant" & TIL == "Susceptible")

## [1] 0.9850746

cpquery(pic.fit, (SUL == "Resistant"), PEN == "Susceptible" & TIL == "Resistant")

## [1] 0.7969303

cpquery(pic.fit, (SUL == "Resistant"), PEN == "Susceptible" & TIL == "Susceptible")

## [1] 0.415522

cpquery(pic.fit, (SUL == "Susceptible"), PEN == "Resistant" & TIL == "Susceptible")

## [1] 0.01864802

cpquery(pic.fit, (SUL == "Susceptible"), PEN == "Susceptible" & TIL == "Susceptible")

## [1] 0.5881154

cpquery(pic.fit, (SUL == "Resistant"), PEN == "Susceptible" | TIL == "Susceptible")

## [1] 0.678123

set.seed(114)

# PEN - SUL
cpquery(pic.fit, (PEN == "Resistant"), SUL == "Resistant")

## [1] 0.147555

cpquery(pic.fit, (PEN == "Resistant"), SUL == "Susceptible")

## [1] 0.01941119

cpquery(pic.fit, (PEN == "Resistant"), SUL == "Resistant" & TIL == "Resistant")

## [1] 0.1115081

cpquery(pic.fit, (PEN == "Resistant"), SUL == "Susceptible" & TIL == "Resistant")

## [1] 0.04631083

```

```

cpquery(pic.fit, (PEN == "Resistant"), SUL == "Resistant" & TIL == "Susceptible")

## [1] 0

cpquery(pic.fit, (PEN == "Resistant"), SUL == "Susceptible" & TIL == "Susceptible")

## [1] 0.002878526

cpquery(pic.fit, (SUL == "Resistant"), TIL == "Resistant" & PEN == "Susceptible")

## [1] 0.7912179

set.seed(115)

# PEN - TIA
cpquery(pic.fit, (TIA == "Resistant"), PEN == "Resistant")

## [1] 0.7117278

cpquery(pic.fit, (TIA == "Resistant"), PEN == "Susceptible")

## [1] 0.1463196

cpquery(pic.fit, (TIA == "Susceptible"), PEN == "Resistant")

## [1] 0.2975133

cpquery(pic.fit, (TIA == "Susceptible"), PEN == "Susceptible")

## [1] 0.8495446

cpquery(pic.fit, (PEN == "Resistant"), TIA == "Resistant")

## [1] 0.3724042

cpquery(pic.fit, (PEN == "Resistant"), TIA == "Susceptible")

## [1] 0.04111842

cpquery(pic.fit, (PEN == "Susceptible"), TIA == "Susceptible")

## [1] 0.9532425

```

```

set.seed(167)
cpquery(pic.fit, (TIA == "Resistant"), PEN == "Resistant" & SPC == "Resistant")

## [1] 0.7045455

cpquery(pic.fit, (TIA == "Resistant"), PEN == "Susceptible" & SPC == "Resistant")

## [1] 0.6586587

cpquery(pic.fit, (TIA == "Resistant"), PEN == "Resistant" & SPC == "Susceptible")

## [1] 0.7275676

cpquery(pic.fit, (TIA == "Susceptible"), PEN == "Susceptible" & SPC == "Susceptible")

## [1] 0.9143038

set.seed(194)
cpquery(pic.fit, (CLN == "Resistant"), TIA == "Resistant")

## [1] 0.8964236

cpquery(pic.fit, (TIA == "Resistant"), CLN == "Resistant")

## [1] 0.2449695

cpquery(pic.fit, (TIA == "Susceptible"), CLN == "Suseptible")

## [1] 0

cpquery(pic.fit, (TIA == "Resistant"), PEN == "Resistant" & CLN == "Resistant")

## [1] 0.7097458

cpquery(pic.fit, (TIA == "Resistant"), PEN == "Susceptible" & CLN == "Resistant")

## [1] 0.1708786

cpquery(pic.fit, (TIA == "Resistant"), PEN == "Resistant" & CLN == "Susceptible")

## [1] 0.538961

cpquery(pic.fit, (TIA == "Susceptible"), PEN == "Resistant" & CLN == "Susceptible")

## [1] 0.4907975

```

```

cpquery(pic.fit, (TIA == "Susceptible"), PEN == "Susceptible" & CLN == "Susceptible")

## [1] 0.9179559

set.seed(116)

# SUL - TIL
cpquery(pic.fit, (TIL == "Resistant"), SUL == "Resistant")

## [1] 0.7608601

cpquery(pic.fit, (TIL == "Resistant"), SUL == "Susceptible")

## [1] 0.4151684

cpquery(pic.fit, (TIL == "Susceptible"), SUL == "Resistant")

## [1] 0.234456

cpquery(pic.fit, (TIL == "Susceptible"), SUL == "Susceptible")

## [1] 0.5734473

cpquery(pic.fit, (TIL == "Resistant"), SUL == "Resistant" & CLN == "Susceptible")

## [1] 0.0187747

cpquery(pic.fit, (TIL == "Resistant"), SUL == "Resistant" & CLN == "Resistant")

## [1] 0.8921371

set.seed(117)

# TIL - CLN
cpquery(pic.fit, (TIL == "Resistant"), CLN == "Resistant")

## [1] 0.8420718

cpquery(pic.fit, (TIL == "Resistant"), CLN == "Susceptible")

## [1] 0.01325411

cpquery(pic.fit, (TIL == "Susceptible"), CLN == "Resistant")

## [1] 0.1589921

```

```

cpquery(pic.fit, (TIL == "Susceptible"), CLN == "Susceptible")

## [1] 0.9857863

cpquery(pic.fit, (TIL == "Resistant"), CLN == "Resistant" & SUL == "Resistant")

## [1] 0.887331

cpquery(pic.fit, (TIL == "Resistant"), CLN == "Susceptible" & SUL == "Resistant")

## [1] 0.0281407

cpquery(pic.fit, (TIL == "Resistant"), CLN == "Resistant" & SUL == "Susceptible")

## [1] 0.6961094

set.seed(118)

# TIA - SPC
cpquery(pic.fit, (TIA == "Resistant"), SPC == "Resistant")

## [1] 0.676259

cpquery(pic.fit, (TIA == "Resistant"), SPC == "Susceptible")

## [1] 0.1520665

cpquery(pic.fit, (TIA == "Susceptible"), SPC == "Resistant")

## [1] 0.3463936

cpquery(pic.fit, (TIA == "Susceptible"), SPC == "Susceptible")

## [1] 0.8464221

cpquery(pic.fit, (SPC == "Resistant"), TIA == "Resistant")

## [1] 0.3582296

cpquery(pic.fit, (SPC == "Susceptible"), TIA == "Susceptible")

## [1] 0.9535208

```

```

set.seed(119)

# CLN - TET
cpquery(pic.fit, (CLN == "Resistant"), TET == "Resistant")

## [1] 0.8151992

cpquery(pic.fit, (CLN == "Resistant"), TET == "Susceptible")

## [1] 0.4503968

cpquery(pic.fit, (CLN == "Susceptible"), TET == "Resistant")

## [1] 0.1807645

cpquery(pic.fit, (CLN == "Susceptible"), TET == "Susceptible")

## [1] 0.5908257

cpquery(pic.fit, (TET == "Resistant"), CLN == "Resistant")

## [1] 0.9472874

cpquery(pic.fit, (TET == "Resistant"), CLN == "Susceptible")

## [1] 0.7192429

set.seed(120)

# SPC - CEF
cpquery(pic.fit, (SPC == "Resistant"), CEF == "Resistant")

## [1] 0.562724

cpquery(pic.fit, (SPC == "Resistant"), CEF == "Susceptible")

## [1] 0.1012971

cpquery(pic.fit, (SPC == "Susceptible"), CEF == "Resistant")

## [1] 0.4496124

cpquery(pic.fit, (SPC == "Susceptible"), CEF == "Susceptible")

## [1] 0.8988984

```

```

# CEF - SPC
cpquery(pic.fit, (CEF == "Resistant"), SPC == "Resistant")

## [1] 0.1418685

cpquery(pic.fit, (CEF == "Resistant"), SPC == "Susceptible")

## [1] 0.01467269

cpquery(pic.fit, (CEF == "Susceptible"), SPC == "Resistant")

## [1] 0.864818

cpquery(pic.fit, (CEF == "Susceptible"), SPC == "Susceptible")

## [1] 0.9853488

set.seed(121)

# OXY - CHL
cpquery(pic.fit, (OXY == "Resistant"), CHL == "Resistant")

## [1] 0.9976866

cpquery(pic.fit, (OXY == "Resistant"), CHL == "Susceptible")

## [1] 0.1613663

cpquery(pic.fit, (OXY == "Susceptible"), CHL == "Resistant")

## [1] 0.002174438

cpquery(pic.fit, (OXY == "Susceptible"), CHL == "Susceptible")

## [1] 0.842164

cpquery(pic.fit, (OXY == "Resistant"), CHL == "Resistant" & TIL == "Resistant")

## [1] 0.9993318

cpquery(pic.fit, (OXY == "Resistant"), CHL == "Resistant" & TIL == "Susceptible")

## [1] 0.9950606

```

```

cpquery(pic.fit, (CHL == "Resistant"), OXY == "Resistant")

## [1] 0.9685012

cpquery(pic.fit, (CHL == "Resistant"), OXY == "Susceptible")

## [1] 0.009162304

set.seed(122)

# CEF - PEN
cpquery(pic.fit, (CEF == "Resistant"), PEN == "Resistant")

## [1] 0.1949627

cpquery(pic.fit, (CEF == "Susceptible"), PEN == "Susceptible")

## [1] 0.9940516

```

## 6. Sensitivity Analysis

Same steps as above, but grouping intermediate with resistant rather than susceptible

```

# select AMDs with large enough cell counts
sens_suis.amd <- sens_suis.pic3 %>%
  select(siteID, Ceftiofur, Chlortetracycline, Clindamycin, Enrofloxacin,
         Gentamicin, Neomycin, Oxytetracycline, Penicillin, Spectinomycin,
         Sulphadimethoxine, Tetracycline, Tiamulin, Tilmicosin)

# create factors for variables
sens_suis.amd$siteID <- (as.factor(sens_suis.amd$siteID))
sens_suis.amd$CEF <- (as.factor(sens_suis.amd$Ceftiofur))
sens_suis.amd$ENR <- (as.factor(sens_suis.amd$Enrofloxacin))
sens_suis.amd$GEN <- (as.factor(sens_suis.amd$Gentamicin))
sens_suis.amd$NEO <- (as.factor(sens_suis.amd$Neomycin))
sens_suis.amd$PEN <- (as.factor(sens_suis.amd$Penicillin))
sens_suis.amd$SPC <- (as.factor(sens_suis.amd$Spectinomycin))
sens_suis.amd$SUL <- (as.factor(sens_suis.amd$Sulphadimethoxine))
sens_suis.amd$TIA <- (as.factor(sens_suis.amd$Tiamulin))
sens_suis.amd$TIL <- (as.factor(sens_suis.amd$Tilmicosin))
sens_suis.amd$CHL <- (as.factor(sens_suis.amd$Chlortetracycline))
sens_suis.amd$CLN <- (as.factor(sens_suis.amd$Clindamycin))
sens_suis.amd$OXY <- (as.factor(sens_suis.amd$Oxytetracycline))
sens_suis.amd$TET <- (as.factor(sens_suis.amd$Tetracycline))

# keep only variables for analysis
pic.sensitivity <- as.data.frame(sens_suis.amd) %>%
  select(siteID, CEF, ENR, GEN, NEO, PEN, SPC, SUL, TIA, TIL, CHL, CLN, OXY, TET)

```

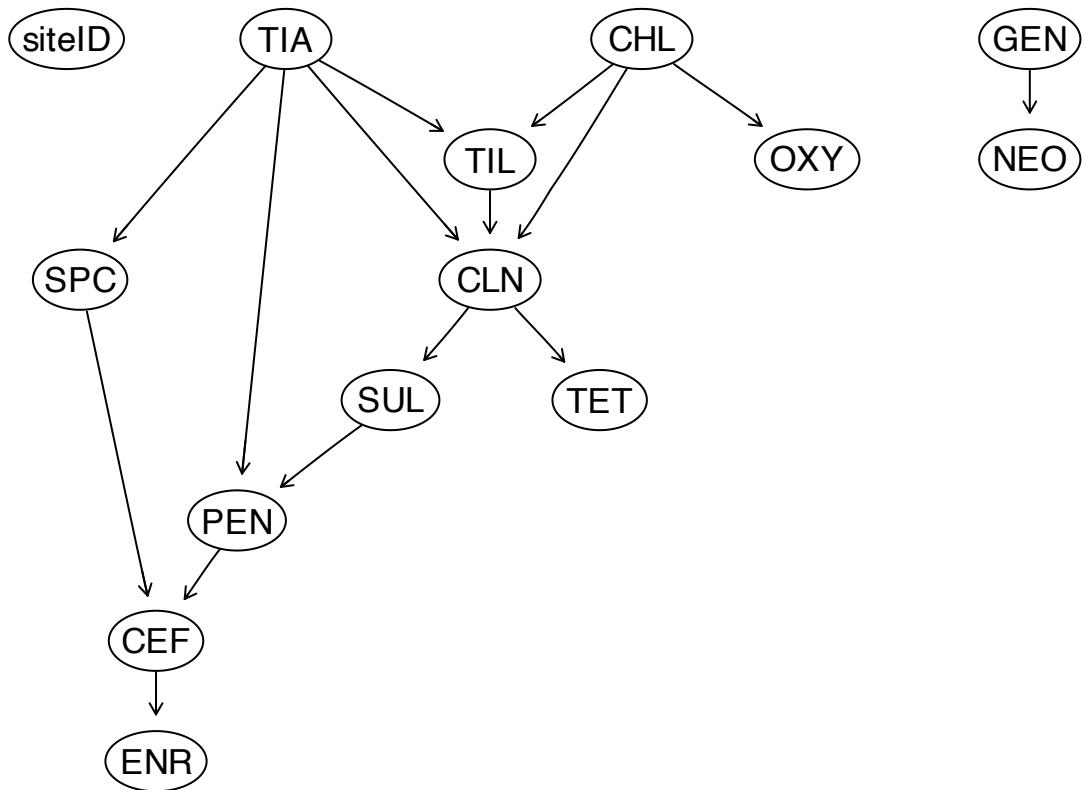
```

# impute with bayes method
pic.tabu2_sens <- structural.em(pic.sensitivity, maximize = "tabu",
                                 maximize.args = list(score = "bic",
                                                       blacklist = bl),
                                 impute = "bayes-lw", return.all = TRUE)

# bootstrap - parents
pic.boot_sens = suppressWarnings(boot.strength(pic.tabu2_sens$imputed, R = 10000,
                                                algorithm = "tabu",
                                                algorithm.args = list(score = "bic", blacklist = bl)))

# take arcs with strength of threshold and return average consensus network - parents
pic.avg_sens = averaged.network(pic.boot_sens, threshold = 0.5)
graphviz.plot(pic.avg_sens, shape = "ellipse", main = NULL, sub = "Bayesian Network Analysis - S. suis")

```

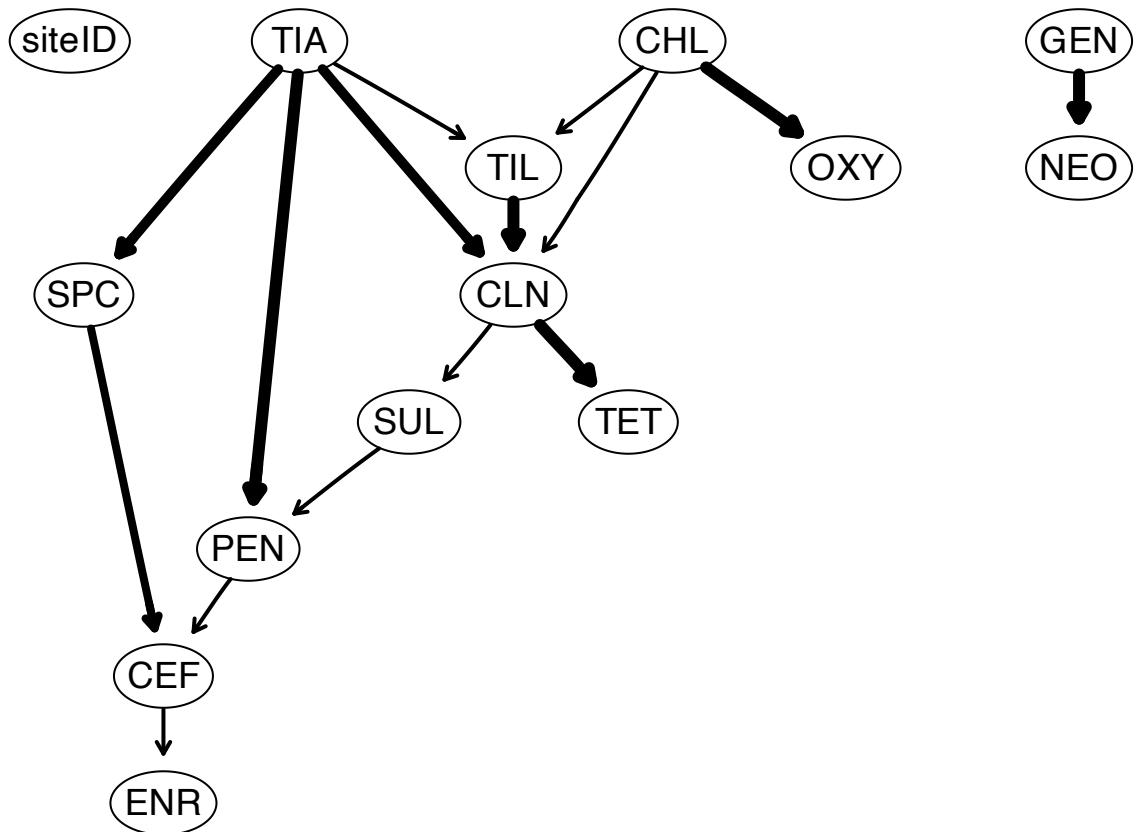


Bayesian Network Analysis - S. suis isolates from PIC w/ parents imputation

```

# plot DAG with strength of association information - parents
strength.plot(pic.avg_sens, pic.boot_sens, shape = "ellipse")

```

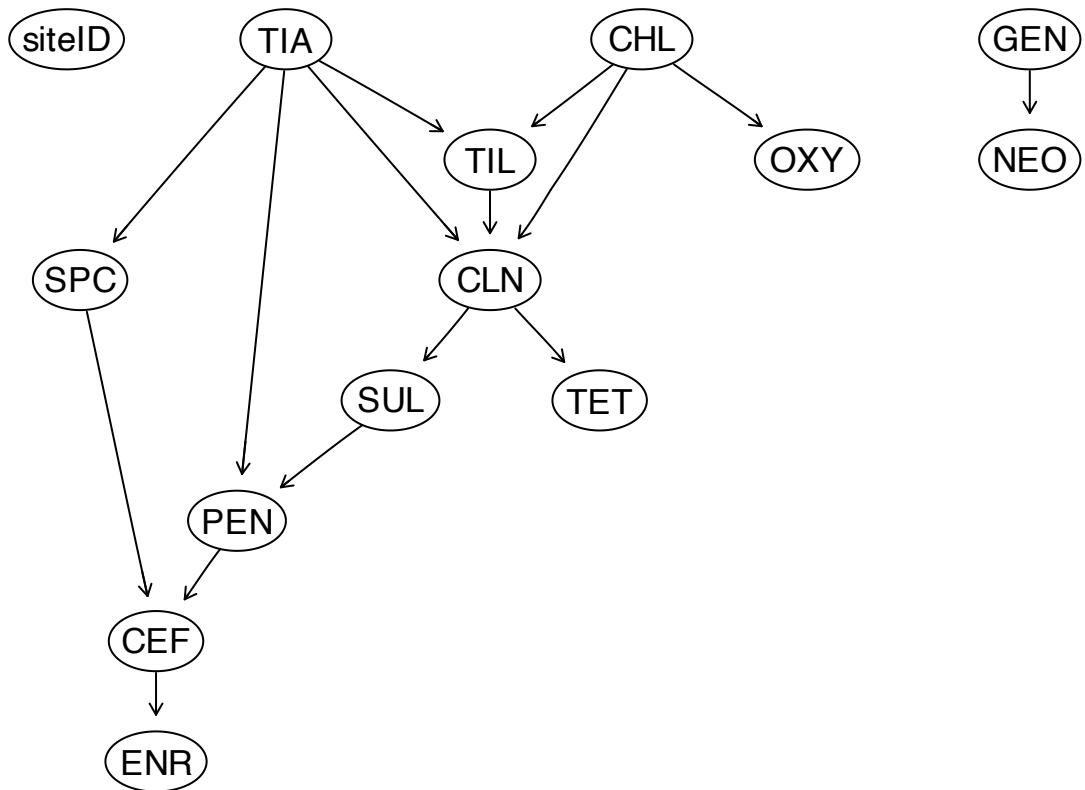


```

# bootstrap - bayes
pic.boot2_sens = boot.strength(pic.tabu2_sens$imputed, R = 10000, algorithm = "tabu",
                                algorithm.args = list(score = "bic"))

# take arcs with strength of threshold and return average consensus network - bayes
pic.avg2_sens = averaged.network(pic.boot2_sens, threshold = 0.5)
graphviz.plot(pic.avg2_sens, shape = "ellipse", main = NULL, sub = "Bayesian Network Sensitivity Analysis")

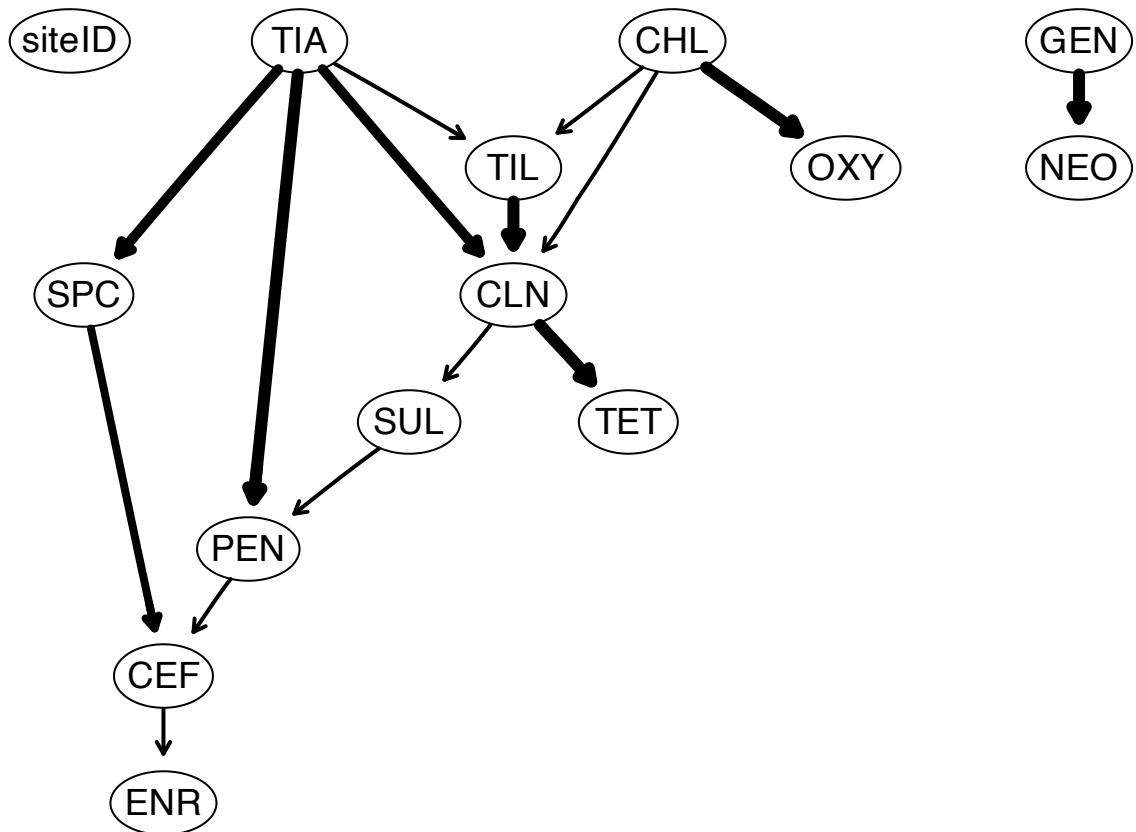
```



Bayesian Network Sensitivity Analysis - *S. suis* isolates from PIC w/ bayes imputation

```

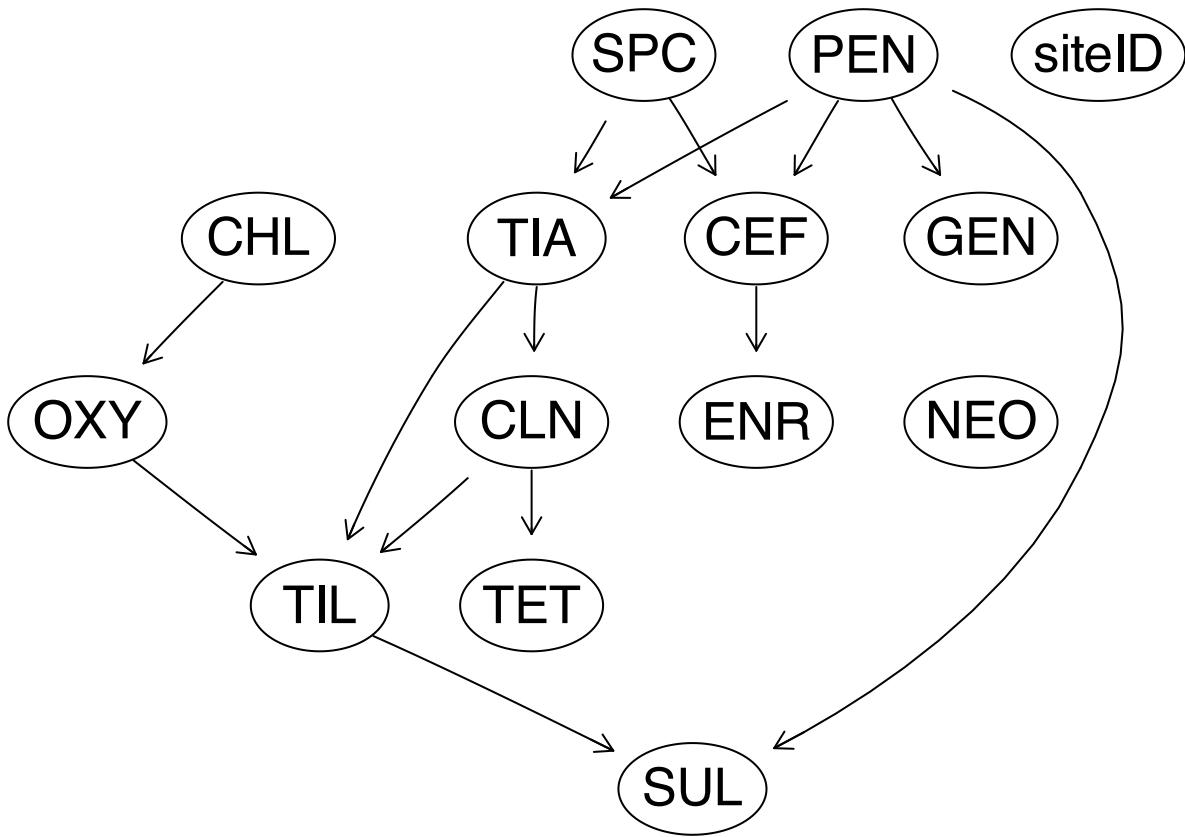
# plot DAG with strength of association information - bayes
strength.plot(pic.avg2_sens, pic.boot2_sens, shape = "ellipse")
  
```

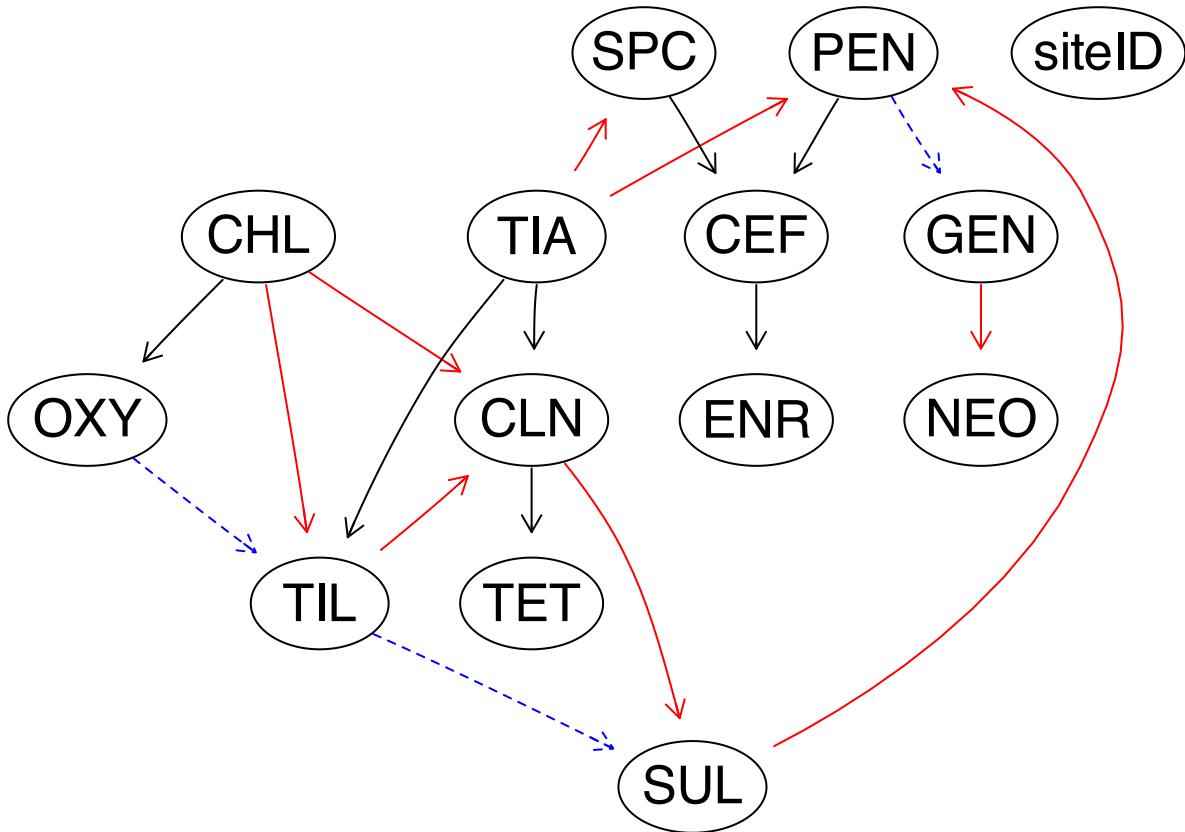


```

# compare graphs
graphviz.compare(pic.avg2, pic.avg2_sens,
                 shape = "ellipse")

```





## 7. Subnetworks

```

# df
yearly <- suis.pic3 %>%
  select(yearrec, siteID, Ceftiofur, Chlortetracycline, Clindamycin, Enrofloxacin,
         Gentamicin, Neomycin, Oxytetracycline, Penicillin, Spectinomycin,
         Sulphadimethoxine, Tetracycline, Tiamulin, Tilmicosin)
# yearly %>% tbl_summary()

# will combine 2014-2018, and 2019-2021

```

```

# create factors for variables
yearly$siteID <- (as.factor(suis.pic3$siteID))
yearly$CEF <- (as.factor(suis.pic3$Ceftiofur))
yearly$CHL <- (as.factor(suis.pic3$Chlortetracycline))
yearly$CLN <- (as.factor(suis.pic3$Clindamycin))
yearly$ENR <- (as.factor(suis.pic3$Enrofloxacin))
yearly$GEN <- (as.factor(suis.pic3$Gentamicin))
yearly$NEO <- (as.factor(suis.pic3$Neomycin))
yearly$OXY <- (as.factor(suis.pic3$Oxytetracycline))
yearly$PEN <- (as.factor(suis.pic3$Penicillin))
yearly$SPC <- (as.factor(suis.pic3$Spectinomycin))
yearly$SUL <- (as.factor(suis.pic3$Sulphadimethoxine))
yearly$TET <- (as.factor(suis.pic3$Tetracycline))

```

```

yearly$TIA <- (as.factor(suis.pic3$Tiamulin))
yearly$TIL <- (as.factor(suis.pic3$Tilmicosin))
yearly$year <- factor(suis.pic3$yearrec, order = TRUE,
                      levels = c(2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021),
                      labels = c("2014-2018", "2014-2018", "2014-2018", "2014-2018",
                                "2014-2018", "2019-2021", "2019-2021", "2019-2021"))

# keep only variables for analysis
year.analyze <- as.data.frame(yearly) %>%
  select(siteID, CEF, CHL, CLN, ENR, GEN, NEO, OXY, PEN, SPC, SUL, TET, TIA, TIL, year)

year.tbl <- year.analyze %>%
  select(siteID, year, CEF, CHL, CLN, ENR, GEN, NEO, OXY, PEN, SPC, SUL, TET, TIA, TIL) %>%
 tbl_summary(
  by = year)

# split by year
analyze.2014 <- year.analyze %>%
  filter(year == "2014-2018") %>%
  select(-year)

analyze.2019 <- year.analyze %>%
  filter(year == "2019-2021") %>%
  select(-year)

# table summary 2014-2018
# analyze.2014 %>%tbl_summary()

# table summary 2019-2021
# analyze.2019 %>%tbl_summary()

```

Build network for each year using Bayes method imputation and structural.em

```

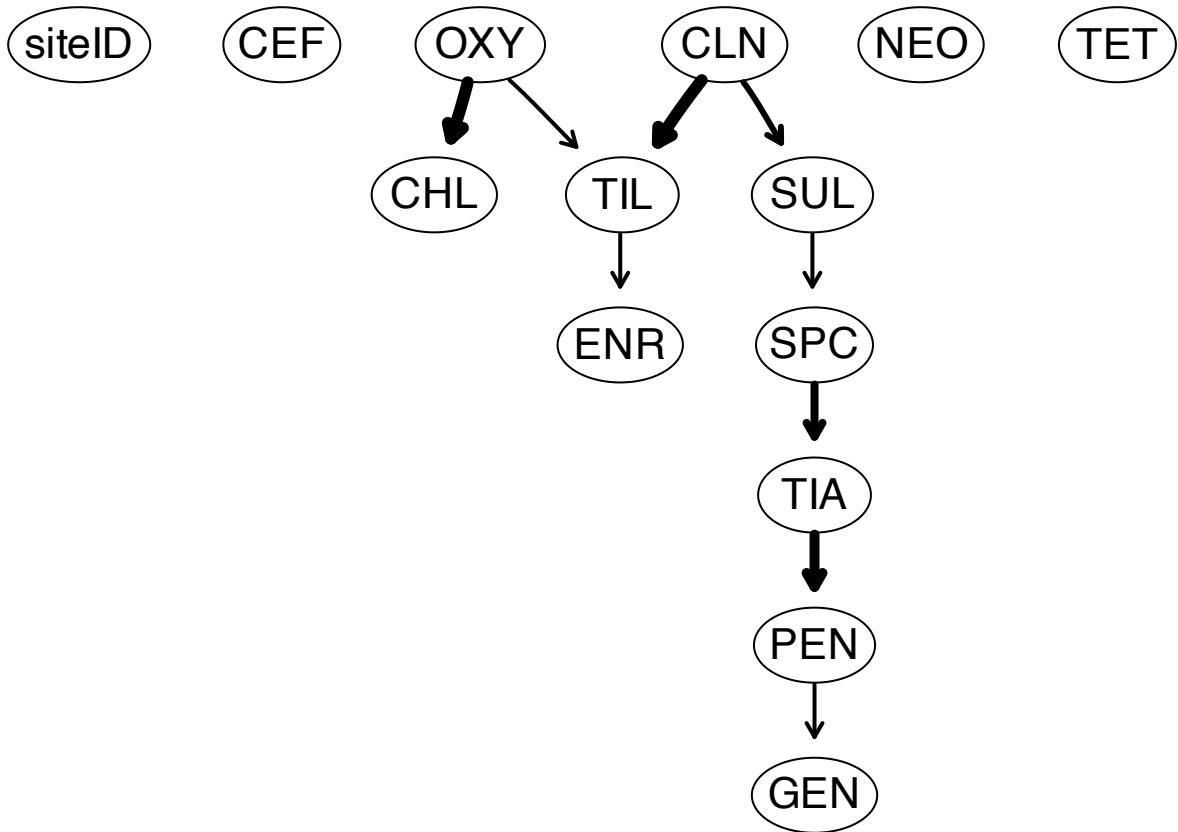
# 2014-2018
graph.2014 <- structural.em(analyze.2014, maximize = "tabu",
                               maximize.args = list(score = "bic",
                                                     blacklist = bl),
                               impute = "bayes-lw", return.all = TRUE)

# bootstrap
boot.2014 = boot.strength(graph.2014$imputed, R = 10000, algorithm = "tabu",
                           algorithm.args = list(score = "bic",
                                                 blacklist = bl))

# take arcs with strength of threshold and return average consensus network - bayes
avg.2014 = averaged.network(boot.2014, threshold = 0.5)

strength.plot(avg.2014, boot.2014, shape = "ellipse")

```



```

# fit the parameters - maximum likelihood method
fit.2014 = bn.fit(avg.2014, analyze.2014, method = "bayes")

# 2014-2018 probability table
fit.2014

## 
##   Bayesian network parameters
## 
##   Parameters of node siteID (multinomial distribution)
## 
## Conditional probability table:
##      25903      25960      26031      26075      26102      26146
## 6.675824e-02 4.175824e-02 8.424908e-03 8.424908e-03 2.509158e-02 8.424908e-03
##      26250      26270      26291      26294      26304      26312
## 2.509158e-02 8.424908e-03 8.424908e-03 1.675824e-02 8.424908e-03 8.424908e-03
##      26376      26667      26685      26876      27005      27023
## 8.424908e-03 8.424908e-03 2.509158e-02 8.424908e-03 9.157509e-05 8.424908e-03
##      27124      27209      27234      27260      27311      27323
## 8.424908e-03 1.675824e-02 8.424908e-03 9.157509e-05 8.424908e-03 1.675824e-02
##      27378      27411      27445      27470      27519      27545
## 8.424908e-03 8.424908e-03 8.424908e-03 4.175824e-02 1.675824e-02 1.675824e-02
##      32071      32181      33158      33228      33246      33252
## 1.675824e-02 5.009158e-02 8.424908e-03 9.157509e-05 8.424908e-03 8.424908e-03
##      33277      33363      33402      33452      33652      33657
## 8.424908e-03 8.424908e-03 8.424908e-03 8.424908e-03 1.675824e-02 8.424908e-03

```

```

##      33786      33787      33788      34337      34370      34373
## 1.675824e-02 8.424908e-03 1.675824e-02 8.424908e-03 3.342491e-02 8.424908e-03
##      34382      34431      36070      36177      40584      41916
## 1.675824e-02 8.424908e-03 9.157509e-05 1.675824e-02 8.424908e-03 8.424908e-03
##      41917      41947      41948      42199      42822      43344
## 1.675824e-02 1.675824e-02 1.675824e-02 8.424908e-03 8.424908e-03 8.424908e-03
##      44911      46612      49123      54764      56373      56433
## 8.424908e-03 8.424908e-03 8.424908e-03 2.509158e-02 8.424908e-03 8.424908e-03
##      57199      57212      59013      59164      61034      61136
## 8.424908e-03 1.675824e-02 9.157509e-05 8.424908e-03 8.424908e-03 9.157509e-05
##      62981      64319      64440      65291      65637      65642
## 9.157509e-05 8.424908e-03 9.157509e-05 8.424908e-03 8.424908e-03 8.424908e-03
##      65647      65652      65664      66744      67342      67356
## 8.424908e-03 9.157509e-05 9.157509e-05 9.157509e-05 9.157509e-05 1.675824e-02
##      67506      68169      68542      68780      69282      70587
## 9.157509e-05 9.157509e-05 9.157509e-05 9.157509e-05 9.157509e-05 9.157509e-05
##      72026
## 9.157509e-05
##
##      Parameters of node CEF (multinomial distribution)
##
## Conditional probability table:
##      Resistant Susceptible
##      0.0125      0.9875
##
##      Parameters of node CHL (multinomial distribution)
##
## Conditional probability table:
##      OXY
##      CHL      Resistant Susceptible
##      Resistant  0.96448087  0.01612903
##      Susceptible 0.03551913  0.98387097
##
##      Parameters of node CLN (multinomial distribution)
##
## Conditional probability table:
##      Resistant Susceptible
##      0.7625      0.2375
##
##      Parameters of node ENR (multinomial distribution)
##
## Conditional probability table:
##      TIL
##      ENR      Resistant Susceptible
##      Resistant  0.090062112 0.006329114
##      Susceptible 0.909937888 0.993670886
##
##      Parameters of node GEN (multinomial distribution)
##
## Conditional probability table:
##      PEN

```

```

## GEN          Resistant Susceptible
##   Resistant    0.23684211  0.02036199
##   Susceptible  0.76315789  0.97963801
##
##   Parameters of node NEO (multinomial distribution)
##
## Conditional probability table:
##   Resistant Susceptible
##   0.6291667   0.3708333
##
##   Parameters of node OXY (multinomial distribution)
##
## Conditional probability table:
##   Resistant Susceptible
##   0.8551402   0.1448598
##
##   Parameters of node PEN (multinomial distribution)
##
## Conditional probability table:
##   TIA
## PEN          Resistant Susceptible
##   Resistant    0.37179487  0.02238806
##   Susceptible  0.62820513  0.97761194
##
##   Parameters of node SPC (multinomial distribution)
##
## Conditional probability table:
##   SUL
## SPC          Resistant Susceptible
##   Resistant    0.14150943  0.00617284
##   Susceptible  0.85849057  0.99382716
##
##   Parameters of node SUL (multinomial distribution)
##
## Conditional probability table:
##   CLN
## SUL          Resistant Susceptible
##   Resistant    0.7896175   0.2543860
##   Susceptible  0.2103825   0.7456140
##
##   Parameters of node TET (multinomial distribution)
##
## Conditional probability table:
##   Resistant Susceptible
##   0.75          0.25
##
##   Parameters of node TIA (multinomial distribution)
##
## Conditional probability table:
##   SPC

```

```

## TIA          Resistant Susceptible
##   Resistant    0.6304348   0.1129032
##   Susceptible  0.3695652   0.8870968
##
##   Parameters of node TIL (multinomial distribution)
##
## Conditional probability table:
##
## , , OXY = Resistant
##
##           CLN
## TIL          Resistant Susceptible
##   Resistant    0.95614035  0.00617284
##   Susceptible  0.04385965  0.99382716
##
## , , OXY = Susceptible
##
##           CLN
## TIL          Resistant Susceptible
##   Resistant    0.50000000  0.03846154
##   Susceptible  0.50000000  0.96153846

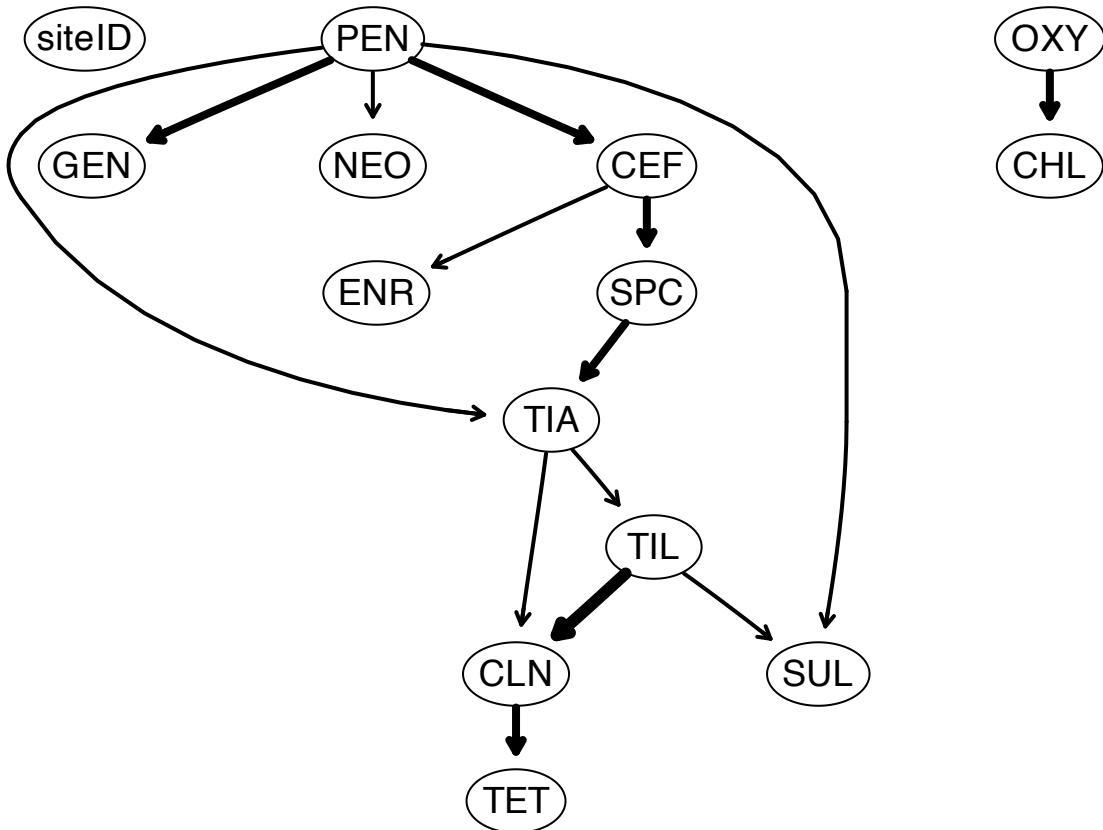
# 2019-2021
graph.2019 <- structural.em(analyze.2019, maximize = "tabu",
                               maximize.args = list(score = "bic",
                                                    blacklist = bl),
                               impute = "bayes-lw", return.all = TRUE)

# bootstrap
boot.2019 = boot.strength(graph.2019$imputed, R = 10000, algorithm = "tabu",
                           algorithm.args = list(score = "bic",
                                                 blacklist = bl))

# take arcs with strength of threshold and return average consensus network - bayes
avg.2019 = averaged.network(boot.2019, threshold = 0.5)

strength.plot(avg.2019, boot.2019, shape = "ellipse")

```



```

# fit the parameters - maximum likelihood method
fit.2019 = bn.fit(avg.2019, analyze.2019, method = "bayes")

# 2019-2021 probability table
fit.2019

## 
## Bayesian network parameters
## 
## Parameters of node siteID (multinomial distribution)
## 
## Conditional probability table:
##      25903      25960      26031      26075      26102      26146
## 2.135453e-02 7.170135e-03 7.793625e-05 7.793625e-05 1.426233e-02 7.793625e-05
##      26250      26270      26291      26294      26304      26312
## 7.170135e-03 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05
##      26376      26667      26685      26876      27005      27023
## 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05 7.170135e-03 7.793625e-05
##      27124      27209      27234      27260      27311      27323
## 2.135453e-02 7.170135e-03 7.793625e-05 7.170135e-03 7.793625e-05 7.793625e-05
##      27378      27411      27445      27470      27519      27545
## 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05 7.170135e-03 7.170135e-03
##      32071      32181      33158      33228      33246      33252
## 7.793625e-05 3.553893e-02 7.170135e-03 7.170135e-03 7.793625e-05 7.793625e-05
##      33277      33363      33402      33452      33652      33657
## 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05 1.426233e-02 7.793625e-05

```

```

##      33786      33787      33788      34337      34370      34373
## 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05 7.170135e-03 7.793625e-05
##      34382      34431      36070      36177      40584      41916
## 7.793625e-05 7.793625e-05 7.170135e-03 7.793625e-05 7.793625e-05 7.793625e-05
##      41917      41947      41948      42199      42822      43344
## 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05 7.793625e-05
##      44911      46612      49123      54764      56373      56433
## 7.793625e-05 7.170135e-03 7.793625e-05 7.793625e-05 7.793625e-05 7.170135e-03
##      57199      57212      59013      59164      61034      61136
## 7.793625e-05 7.793625e-05 7.170135e-03 7.793625e-05 7.793625e-05 7.170135e-03
##      62981      64319      64440      65291      65637      65642
## 7.170135e-03 7.793625e-05 7.170135e-03 7.793625e-05 7.793625e-05 7.793625e-05
##      65647      65652      65664      66744      67342      67356
## 7.793625e-05 7.170135e-03 7.170135e-03 7.170135e-03 1.426233e-02 7.793625e-05
##      67506      68169      68542      68780      69282      70587
## 7.170135e-03 6.809290e-01 7.170135e-03 1.426233e-02 7.170135e-03 7.170135e-03
##      72026
## 7.170135e-03
##
##      Parameters of node CEF (multinomial distribution)
##
## Conditional probability table:
##
##      PEN
## CEF      Resistant Susceptible
##   Resistant 0.47435897 0.01028807
##   Susceptible 0.52564103 0.98971193
##
##      Parameters of node CHL (multinomial distribution)
##
## Conditional probability table:
##
##      OXY
## CHL      Resistant Susceptible
##   Resistant 0.96666667 0.10000000
##   Susceptible 0.03333333 0.90000000
##
##      Parameters of node CLN (multinomial distribution)
##
## Conditional probability table:
##
## , , TIL = Resistant
##
##      TIA
## CLN      Resistant Susceptible
##   Resistant 0.992753623 0.998381877
##   Susceptible 0.007246377 0.001618123
##
## , , TIL = Susceptible
##
##      TIA
## CLN      Resistant Susceptible
##   Resistant 0.764150943 0.042857143
##   Susceptible 0.235849057 0.957142857

```

```

##
##
## Parameters of node ENR (multinomial distribution)
##
## Conditional probability table:
##
##          CEF
## ENR      Resistant Susceptible
##   Resistant 0.50000000 0.03256705
##   Susceptible 0.50000000 0.96743295
##
## Parameters of node GEN (multinomial distribution)
##
## Conditional probability table:
##
##          PEN
## GEN      Resistant Susceptible
##   Resistant 0.371794872 0.002057613
##   Susceptible 0.628205128 0.997942387
##
## Parameters of node NEO (multinomial distribution)
##
## Conditional probability table:
##
##          PEN
## NEO      Resistant Susceptible
##   Resistant 0.4230769 0.7757202
##   Susceptible 0.5769231 0.2242798
##
## Parameters of node OXY (multinomial distribution)
##
## Conditional probability table:
##   Resistant Susceptible
##     0.75      0.25
##
## Parameters of node PEN (multinomial distribution)
##
## Conditional probability table:
##   Resistant Susceptible
##     0.1382979 0.8617021
##
## Parameters of node SPC (multinomial distribution)
##
## Conditional probability table:
##
##          CEF
## SPC      Resistant Susceptible
##   Resistant 0.78571429 0.07854406
##   Susceptible 0.21428571 0.92145594
##
## Parameters of node SUL (multinomial distribution)
##
## Conditional probability table:
##

```

```

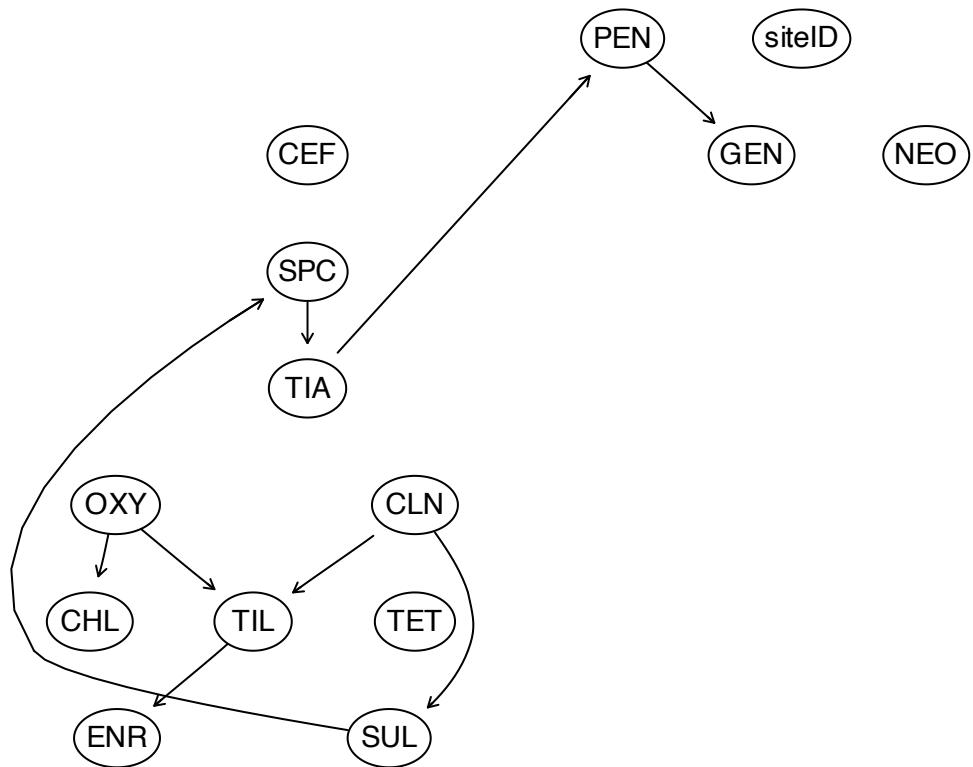
## , , TIL = Resistant
##
##          PEN
## SUL      Resistant Susceptible
##   Resistant  0.92105263  0.77826087
##   Susceptible 0.07894737  0.22173913
##
## , , TIL = Susceptible
##
##          PEN
## SUL      Resistant Susceptible
##   Resistant  0.97619048  0.48581560
##   Susceptible 0.02380952  0.51418440
##
##
##          CLN
## TET      Resistant Susceptible
##   Resistant  0.9589372   0.7105263
##   Susceptible 0.0410628   0.2894737
##
##          Parameters of node TET (multinomial distribution)
##
## Conditional probability table:
##
##          CLN
## TET      Resistant Susceptible
##   Resistant  0.9589372   0.7105263
##   Susceptible 0.0410628   0.2894737
##
##          Parameters of node TIA (multinomial distribution)
##
## Conditional probability table:
##
## , , SPC = Resistant
##
##          PEN
## TIA      Resistant Susceptible
##   Resistant  0.74242422  0.6951220
##   Susceptible 0.2575758   0.3048780
##
## , , SPC = Susceptible
##
##          PEN
## TIA      Resistant Susceptible
##   Resistant  0.6333333   0.1000000
##   Susceptible 0.3666667   0.9000000
##
##
##          TIA
## TIL      Resistant Susceptible
##   Resistant  0.5793651   0.7511416
##   Susceptible 0.4206349   0.2488584

```

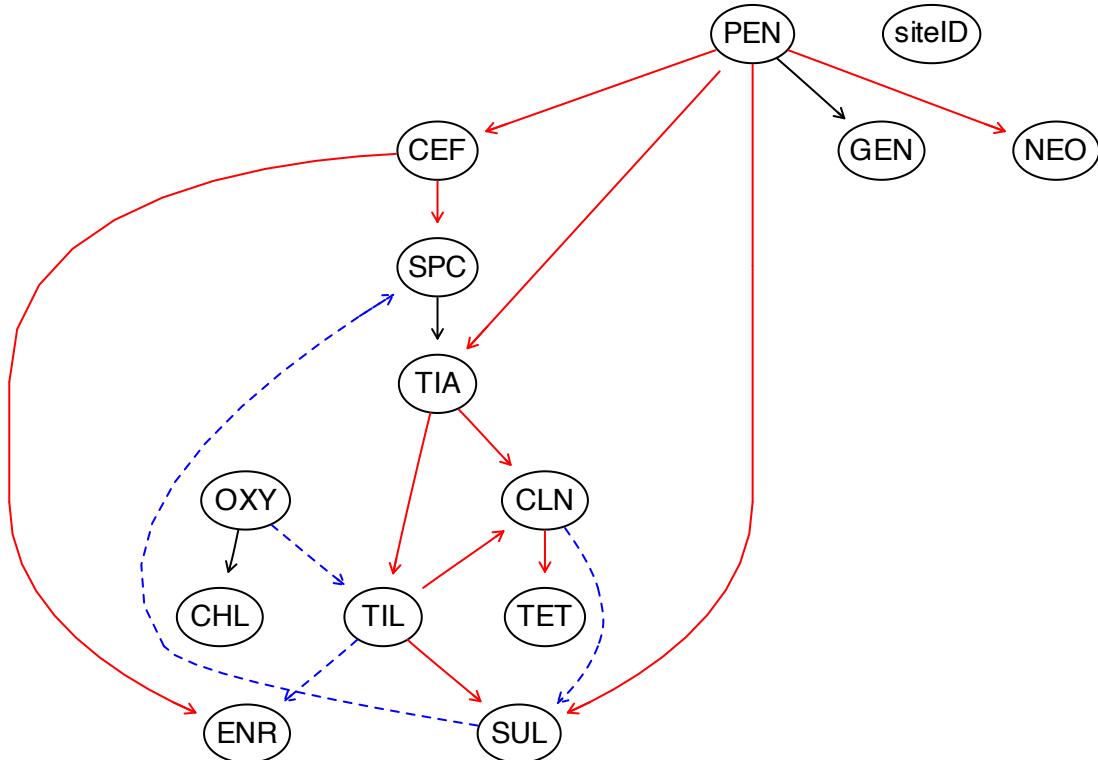
Combine graphs for visual comparison

```
graphviz.compare(avg.2014, avg.2019, shape = "ellipse",
                 main= c("averaged DAG - 2014-2018",
                        "averaged DAG - 2019-2021"))
```

averaged DAG - 2014-2018



averaged DAG - 2019-2021



remove arc directions for publication

```

## 2014-2018

# convert boot result + averaged network into an undirected graph
nodes_df <- data.frame(name = nodes(avg.2014))
boot_df  <- as.data.frame(boot.2014)
avg_arcs <- as.data.frame(arcs(avg.2014))

edges <- avg_arcs %>%
  left_join(boot_df[, c("from", "to", "strength")],
            by = c("from", "to"))

g <- graph_from_data_frame(edges, directed = FALSE, vertices = nodes_df)

# strength plot without arrows
set.seed(123) # for reproducible layout

# create layout
lay <- create_layout(g, layout = "nicely")

# find the row corresponding to siteID
idx_site <- which(lay$name == "siteID")

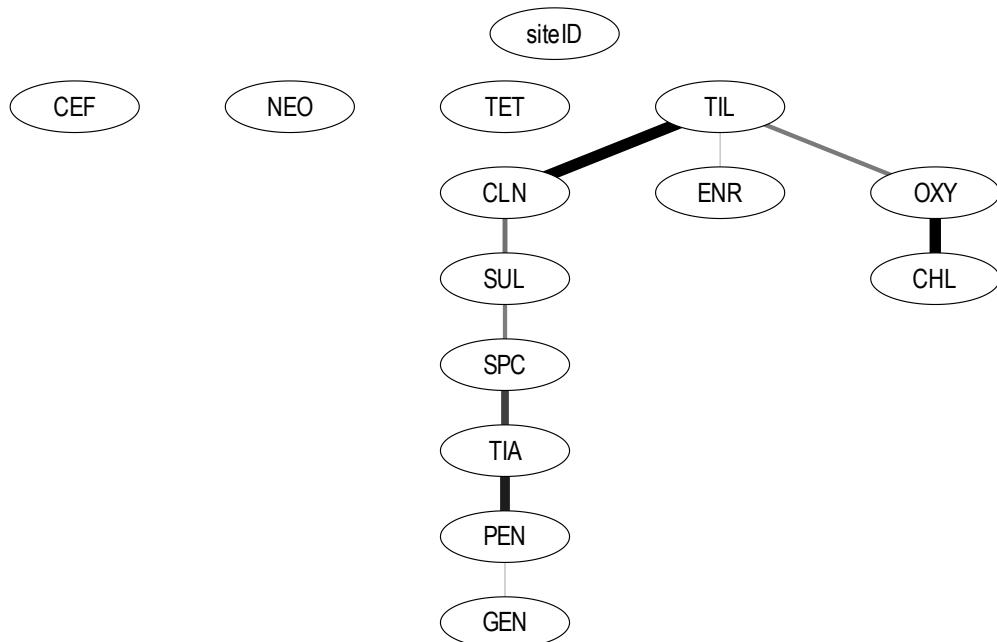
# compute a horizontal center for the main network (all non-siteID nodes)
center_x <- mean(lay$x[lay$name != "siteID"])

```

```

# put siteID above the rest
lay$x[idx_site] <- center_x
lay$y[idx_site] <- max(lay$y) + 0.85
# plot
ggraph(lay) +
  geom_edge_link(aes(width = strength, alpha = strength),
                 color = "black") +
  geom_node_circle(aes(r = 0.3),
                   fill = "white",
                   color = "black",
                   size = 0.2) +
  geom_node_text(aes(label = name),
                size = 3.5) +
  scale_edge_width(range = c(0.2, 2)) +
  scale_edge_alpha(range = c(0.3, 1)) +
  guides(edge_width = "none", edge_alpha = "none") +
  theme_graph()

```



```
# ggsave("2014_network_plot.png", width = 8, height = 6, dpi = 300)
```

```
## 2019-2021
```

```
# convert boot result + averaged network into an undirected graph
nodes_df <- data.frame(name = nodes(avg.2019))
```

```

boot_df <- as.data.frame(boot.2019)
avg_arcs <- as.data.frame(arcs(avg.2019))

edges <- avg_arcs %>%
  left_join(boot_df[, c("from", "to", "strength")],
            by = c("from", "to"))

g <- graph_from_data_frame(edges, directed = FALSE, vertices = nodes_df)

# strength plot without arrows
set.seed(123) # for reproducible layout

# create layout
lay <- create_layout(g, layout = "nicely")

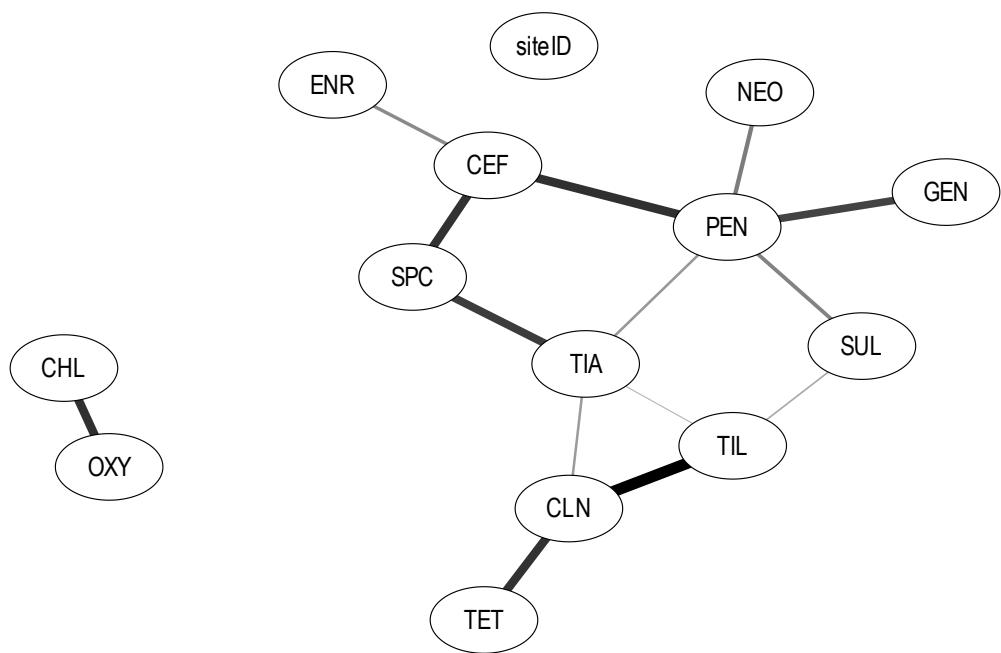
# find the row corresponding to siteID
idx_site <- which(lay$name == "siteID")

# compute a horizontal center for the main network (all non-siteID nodes)
center_x <- mean(lay$x[lay$name != "siteID"])

# put siteID above the rest
lay$x[idx_site] <- center_x
lay$y[idx_site] <- max(lay$y) + 0.35

# plot
ggraph(lay) +
  geom_edge_link(aes(width = strength, alpha = strength),
                 color = "black") +
  geom_node_circle(aes(r = 0.3),
                   fill = "white",
                   color = "black",
                   size = 0.2) +
  geom_node_text(aes(label = name),
                size = 3.5) +
  scale_edge_width(range = c(0.2, 2)) +
  scale_edge_alpha(range = c(0.3, 1)) +
  guides(edge_width = "none", edge_alpha = "none") +
  theme_graph()

```



```
# ggsave("2019_network_plot.png", width = 8, height = 6, dpi = 300)
```