BIPnet_EDA

Aidan Neher

2023-03-03

Download R Package

This file uses Thierry's R Package BIPnet. Here's BIPnet's documentation.

For successful installation, I needed gsl (gnu-scientific-library) installed on my machine. For Mac OS, I did so manually in the terminal with homebrew's command `brew install gsl`.

Then, I ran the code commented out below to install BIPnet from GitHub.

```
# install.packages("devtools")
# library(devtools)
# install_github('chekouo/BIPnet')
```

Explore BIPnet's inputs and outputs with simulated data

```
library(BIPnet)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
simulation_results <- Simulate(setting=1)
simList <- list(simulation_results$Y,
              simulation_results$X1,
              simulation_results$X2)
bip_results <- BIP(dataList=simList,
    IndicVar=c(1,0,0), # clinical outcome, omics view 1, omics view 2
    Method="BIP") # no network information at first

# TODO: Layer a tidy function for getting variable selection criteria results
TrueVar1 <- simulation_results$TrueVar1
IndicVar = bip_results$IndicVar
Np = length(IndicVar)
for (m in 1:Np) {
  if (IndicVar[m]==0){
    print(ComputeVarCriteria(bip_results$VarSelMeanGlobal[[m]], TrueVar1))
  }
}
```

```
## $FalsePosRate
## [1] 0
##
## $FalseNegRate
## [1] 0
##
## $F1measure
## [1] 100
##
## $AUC
## [1] 1
##
## $FalsePosRate
## [1] 0.5
##
## $FalseNegRate
## [1] 0
##
## $F1measure
## [1] 99.0099
##
## $AUC
## [1] 1
```

```
# TODO: Split into training and test set prior to obtaining prediction results below
# Resolve Error in h(simpleError(msg, call)) :
# error in evaluating the argument 'x' in selecting a method for function 't': dims [product 200] do no
# prediction_results <- BIPpredict(dataListNew = dataList, Result = bip_results, meth = "BMA")
```

Load ABCD data and look into the missingness of the covariates

Of those with complete imaging data, what are the percentages for a given covariate?

```
tidy_data <- readRDS("data/2023-02-24-tidy_data.RDS")
cross_data <- tidy_data %>%
  split(f = tidy_data$eventname)
# check for NAs
get_na_props <- function(data_t) {
  n_obs <- nrow(data_t)
  na_counts <- is.na(data_t) %>% colSums
  na_counts <- na_counts[which(na_counts > 0)]
  na_percents <- 100 * na_counts / n_obs
  output <- list(n_obs, na_percents)
  names(output) <- c("n_observations", "percent_missingness")
  return(output)
}
lapply(cross_data, get_na_props)
```

```
## $baseline_year_1_arm_1
## $baseline_year_1_arm_1$n_observations
## [1] 11760
##
## $baseline_year_1_arm_1$percent_missingness
##         demo_comb_income_v2                demo_ethn_v2
##                 11.17346939                 11.23299320
##        demo_prnt_marital_v2                   race_white
##                 11.17346939                  11.15646259
##                  race_black                  race_native
##                 11.15646259                  11.15646259
##       race_pacific_islander                   race_asian
##                 11.15646259                  11.15646259
##                  race_other        demo_prnt_highest_ed
##                 11.15646259                  11.15646259
##                   abcd_site                rel_family_id
##                 11.15646259                  11.15646259
##                  outcome_si outcome_internalizing_score
##                  0.61224490                   0.05952381
##
##
## $`2_year_follow_up_y_arm_1`
## $`2_year_follow_up_y_arm_1`$n_observations
## [1] 7827
##
## $`2_year_follow_up_y_arm_1`$percent_missingness
##         demo_comb_income_v2                 demo_ethn_v2
##                  10.0932669                   10.1699246
##        demo_prnt_marital_v2                   race_white
##                  10.0932669                   10.0932669
##                  race_black                  race_native
##                  10.0932669                   10.0932669
##       race_pacific_islander                   race_asian
##                  10.0932669                   10.0932669
##                  race_other        demo_prnt_highest_ed
##                  10.0932669                   10.0932669
##                   abcd_site                rel_family_id
##                  10.0932669                   10.0932669
##                  outcome_si outcome_internalizing_score
##                   0.6899195                   13.8494953
```

Benchmark running Thierry's code with baseline subset

Key questions:

- What's the run time using BIP out of the box?
- What results might we yield cross-sectionally?

```
data <- cross_data$baseline_year_1_arm_1 %>% ungroup
data <- data[complete.cases(data),] # TODO: deal with missingness in model
outcome <- data$outcome_internalizing_score # start with regression
cortical_thickness <- data %>%
  select(starts_with("smri_thick"))
cortical_area <- data %>%
  select(starts_with("smri_area"))
covariates <- data %>%
  select(-c("subjectkey", "eventname",
            starts_with("smri"), starts_with("outcome"),
            "abcd_site", "rel_family_id")) %>% # TODO: Account for fixed effects in model
  sapply(as.numeric) # convert to numeric for BIPnet

dataList <- list(outcome,
                 cortical_thickness,
                 cortical_area,
                 covariates) %>% lapply(as.matrix)

dataList %>% lapply(class)
```

```
## [[1]]
## [1] "matrix" "array"
##
## [[2]]
## [1] "matrix" "array"
##
## [[3]]
## [1] "matrix" "array"
##
## [[4]]
## [1] "matrix" "array"
```

```
dataList %>% lapply(dim)
```

```
## [[1]]
## [1] 10371     1
##
## [[2]]
## [1] 10371    68
##
## [[3]]
## [1] 10371    68
##
## [[4]]
## [1] 10371    12
```

```
# get training sample
train_index <- sample(1:nrow(data), 1000)
get_matrix_subset <- function(matrix, index) { matrix[index,] %>% as.matrix }
trainList <- dataList %>%
  lapply(get_matrix_subset, index = train_index)

trainList %>% lapply(dim)
```

```
## [[1]]
## [1] 1000    1
##
## [[2]]
## [1] 1000   68
##
## [[3]]
## [1] 1000   68
##
## [[4]]
## [1] 1000   12
```

```
start_time <- Sys.time()

bip_results <- BIP(dataList=trainList[1:3],
    IndicVar=c(1,0,0), # clinical outcome, omics view 1, omics view 2
    Method="BIP") # no covariates/ network information for now

end_time <- Sys.time()

end_time-start_time
```

```
## Time difference of 2.622111 mins
```

```
IndicVar <- bip_results$IndicVar
# check percentage of variables selected at least once in 95% of models
VarSelMeanGlobalOmics_95Prop <- bip_results$VarSelMeanGlobal[which(IndicVar==0)] %>%
  lapply(function(x) {
    n_selected <- (x >= 0.95) %>% sum
    n_vars <- length(x)
    n_selected / n_vars
    })
names(VarSelMeanGlobalOmics_95Prop) <- c("thickness", "surface_area")
print(VarSelMeanGlobalOmics_95Prop)
```

```
## $thickness
## [1] 1
##
## $surface_area
## [1] 1
```

```
# TODO: Inclusion of covariates caused the following error:
# Error in BIP(dataList = trainList, IndicVar = c(1, 0, 0, 2), Method = "BIP") :
  # NA/NaN/Inf in foreign function call (arg 6)
```