Network Inference Methods in R

DJ Passey

Jan 4, 2023

```
library(renv)
```

```
##
## Attaching package: 'renv'
```

```
## The following objects are masked from 'package:stats':
##
##     embed, update
```

```
## The following objects are masked from 'package:utils':
##
##     history, upgrade
```

```
## The following objects are masked from 'package:base':
##
##     autoload, load, remove
```

```
library(here)
```

```
## here() starts at /Users/djpassey/Code/graphinference
```

```
# Activate R virtual environment
# The `here()` function should locate the top level
# directory of the enclosing git repository called `graphinference`
proj_dir = here()
print(cat("Double check that project directory\n",
          "is named graphinference:\n",
          "`proj_dir =`", proj_dir, "\n"))
```

```
## Double check that project directory
##  is named graphinference:
##  `proj_dir =` /Users/djpassey/Code/graphinference
## NULL
```

```
renv::activate(here())

# Activate python virtual environment
# This will not work unless the python virtual environment is named
# `graphinf_venv` and located in the first level of the repo.
#
# Follow the instructions in `graphinference/README.md` to install
# the python virtual environment properly.
library(reticulate)
```

```
##
## Attaching package: 'reticulate'
```

```
## The following object is masked from 'package:renv':
##
##     use_python
```

```
reticulate::use_python(paste0(proj_dir, "/graphinf_venv/bin/python3"))

library(graphicalVAR)
```

```
import networkx as nx
import numpy as np
import random

import graphinference.libs.qspems_sim_core as qspems_sim_core

random.seed(111)
np.random.seed(111)
rng = np.random.RandomState(111)
```

Exploring Graph Inference Methods in R

This notebook explores techniques from R to infer a network underlying time series data.

Generate Test Data

We use a stochastic dynamical system to generate a time series where the variables influence each other according to a fixed adjacency matrix.

The underlying stochastic model is an order one vector auto-regressive process of the form

$$\mathbf{x}_t = \epsilon A \mathbf{x}_{t-1} + \mathbf{e}_t$$

where ϵ is the coupling strength, A is an adjacency matrix and $\mathbf{e}_t$ is the multivariate Gaussian error term with mean $\mathbf{0}$ and covariance $\frac{\sigma^2 \tau}{n} I$. Here τ is the characteristic time, σ is the intrinsic noise strength and n is the dimension of the time series.

```
# Erdos Renyi network
num_nodes = 5
mean_degree = 2
edge_prob = mean_degree / num_nodes
er = nx.erdos_renyi_graph(num_nodes, edge_prob, seed=rng)
adj = nx.adjacency_matrix(er).toarray()

# Stochastic model to generate time series. Compare to the
# jupyter notebook "../Jupyter/NetworkInferenceMethods.ipynb"
```

```
## <string>:1: FutureWarning: adjacency_matrix will return a scipy.sparse array instead of a matrix in
```

```
stochastic_timeseries = qspems_sim_core.sim(adj, dt=1.0, sigma=0.1)
```

Apply GraphicalVAR

Next we use the graphical VAR method to try to fit a type of vector auto regressive model and fit a network to that model.

```r
# Run graphical VAR
gVAR_of_qpsems_sim <- graphicalVAR(t(py$stochastic_timeseries), gamma = 0)
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## i The deprecated feature was likely used in the graphicalVAR package.
##   Please report the issue to the authors.
```

```
##   |                                                                              |
```

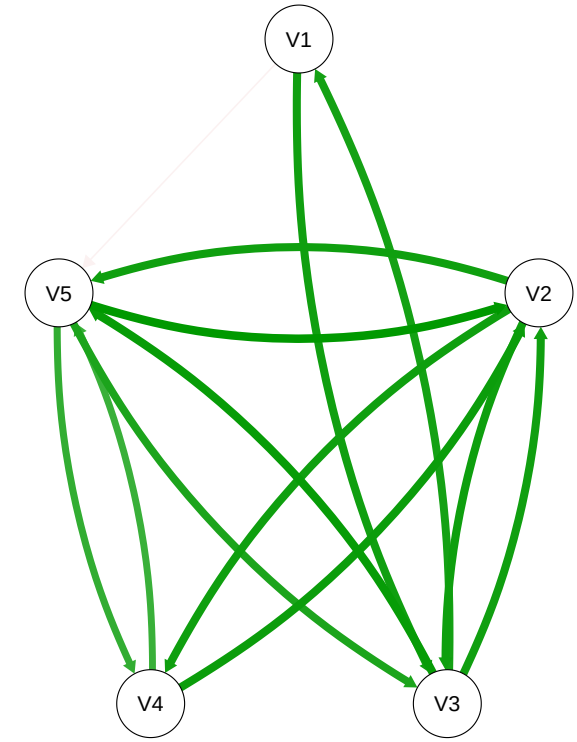
```
## Warning in Rothmana(data_l, data_c, lambdas$beta[i], lambdas$kappa[i],
## regularize_mat_beta = regularize_mat_beta, : Model did NOT converge in outer
## loop
```

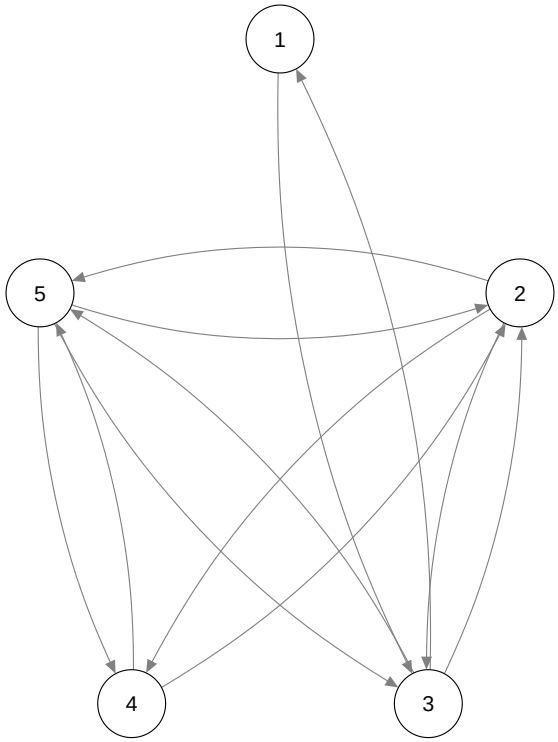
```
##   |                                                                              |=======
```

```r
# Plot the predicted network on the left
layout(t(1:2))
plot(gVAR_of_qpsems_sim, "PDC", layout="circle")
# Save PDC matrix
qpsems_PDC = gVAR_of_qpsems_sim$PDC

# Replace with adj matrix that underpinned the simulation
gVAR_of_qpsems_sim$PDC = py$adj

# Plot the true network on the right
plot(gVAR_of_qpsems_sim, "PDC", layout="circle")
```





Graphical VAR correctly identifies every edge in the network! If we view this notebook as a continuation of `../Jupyter/NetworkInferenceMethods.ipynb` then every network method investigated so far has correctly reproduced the underlying network in the first order vector auto-regressive model.

Apply gimme (unified structural equation model)

```
# temporary gimme input and output directories relative to project directory
gimme_input_dir <- "/Notebooks/R/tmp/gimme_input/"
gimme_output_dir <- "/Notebooks/R/tmp/gimme_output/"

dir.create(paste0(proj_dir, "/Notebooks/R/tmp"))
```

```
## Warning in dir.create(paste0(proj_dir, "/Notebooks/R/tmp")):
## '/Users/djpassey/Code/graphinference/Notebooks/R/tmp' already exists
```

```
dir.create(paste0(proj_dir, gimme_input_dir))
```

```
## Warning in dir.create(paste0(proj_dir, gimme_input_dir)):
## '/Users/djpassey/Code/graphinference/Notebooks/R/tmp/gimme_input' already exists
```

```
dir.create(paste0(proj_dir, gimme_output_dir))
```

```
## Warning in dir.create(paste0(proj_dir, gimme_output_dir)):
## '/Users/djpassey/Code/graphinference/Notebooks/R/tmp/gimme_output' already
## exists
```

```
# Save file to
np.savetxt(
  r.proj_dir + r.gimme_input_dir + "stochastic_ts.csv",
  stochastic_timeseries.T,
```

```
  delimiter=","
)
```

```
fit <- gimme::gimmeSEM(
  data = paste0(proj_dir, gimme_input_dir),
  out = paste0(proj_dir, gimme_output_dir),
  sep = ",",
  header = FALSE,
  ar = TRUE,
  plot = TRUE,
  subgroup = FALSE,
  paths = NULL,  # option to list paths that will be group-level (semi-confirmatory)
  groupcutoff = .75,  # the proportion that is considered the majority at the group level
  subcutoff = .5  # the proportion that is considered the majority at the subgroup level
)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## group-level search
## individual-level pruning, subject 1 (stochastic_ts)
## individual-level search, subject 1 (stochastic_ts)
```

```
## gimme finished running normally
## output is stored in /Users/djpassey/Code/graphinference/Notebooks/R/tmp/gimme_output/
```

```
gimme_plot_file = paste0(proj_dir, gimme_output_dir, "summaryPathsPlot.pdf")
knitr::include_graphics(gimme_plot_file)
```

It's interesting that the unified SEM model, one, makes the distinction between lagged (dashed edge) and contemporaneous (solid edges) in the model. However, it does discover some spurious edges, unlike any other method that we have explored so far.

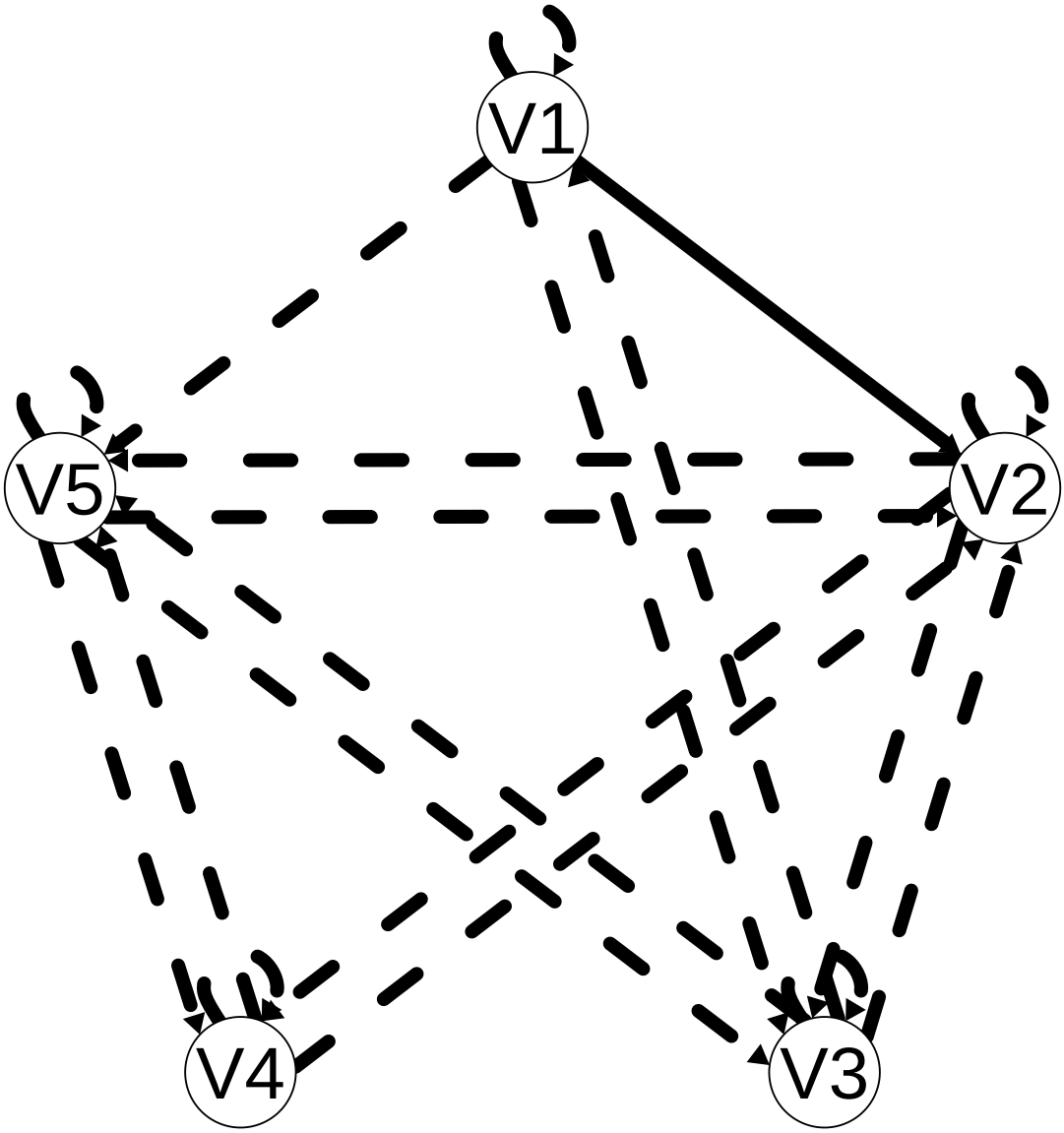


Figure 1: Network predicted by gimme (unified structural equation model). Dashed lines indicate lag-1 relationships and solid indicate contemporaneous relationships.