**Manual**

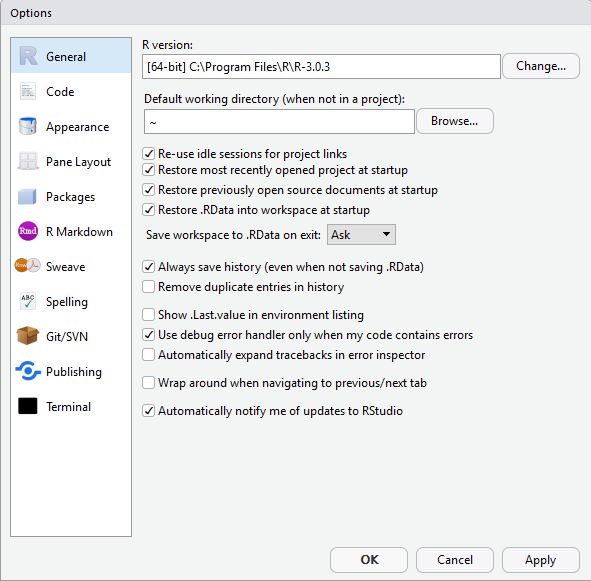
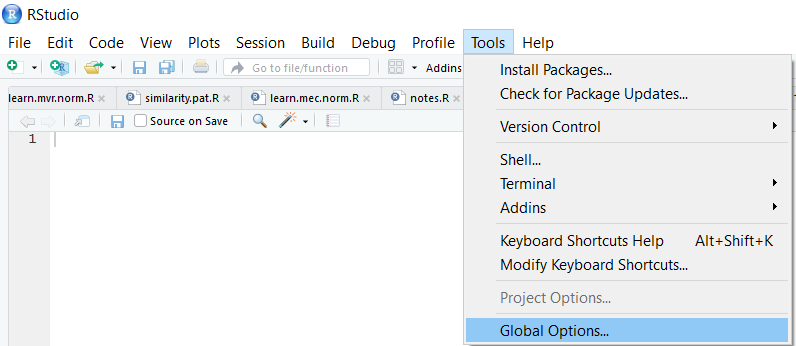
This manual tells you how to use the functions for learning AMP chain graphs (and as a special case, Bayesian networks) via decomposition approach and also the PC-like algorithm.

**RUNNING THE R CODE**:

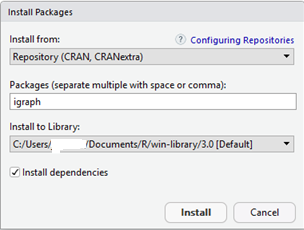
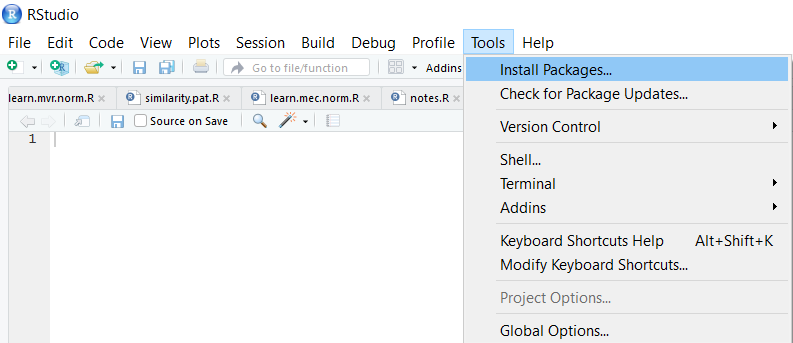
1. Install [R-3.0.3 for Windows (32/64 bit)](https://cran.r-project.org/bin/windows/base/old/3.0.3/).

**NOTICE**: Since LCD R package is not compatible with R versions greater than R-3.0.3, you need to install this version of R. In addition, since R versions can be installed side-by-side on a system, do not worry about the installing this version of R on your machine.

1. Install [RStudio 1.1.463 - Windows Vista/7/8/10](https://www.rstudio.com/products/rstudio/download/).
2. Run the RStudio software and make sure that you are using the right version of R i.e., R-3.0.3:



1. Install the following R packages: igraph, ggm, and lcd. Make sure that the Install Dependencies option is active (tick the box).



1. Install the pcalg R package and its dependencies. Also, copy and paste the following lines in your RStudio Console panel, respectively and press Enter:

source("https://bioconductor.org/biocLite.R")

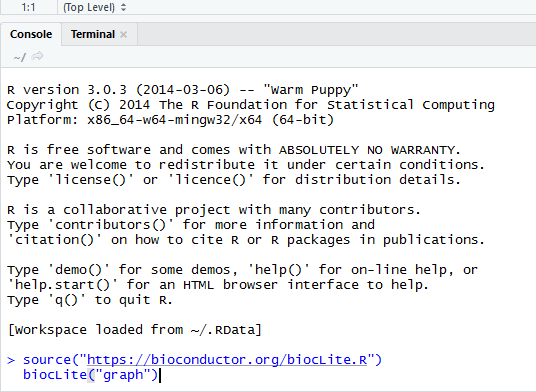
biocLite("graph")

source("https://bioconductor.org/biocLite.R")

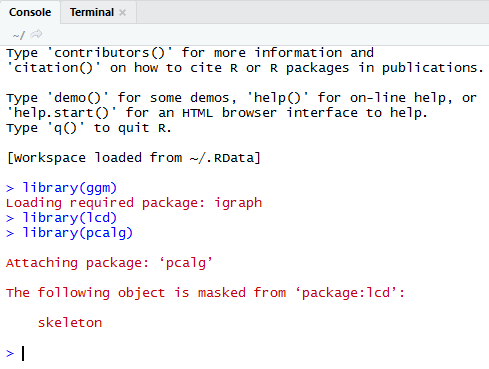
biocLite("RBGL")

source("https://bioconductor.org/biocLite.R")

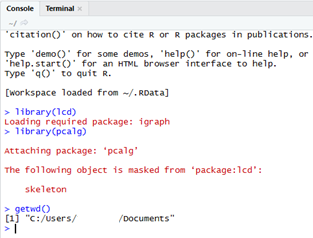
biocLite("Rgraphviz")



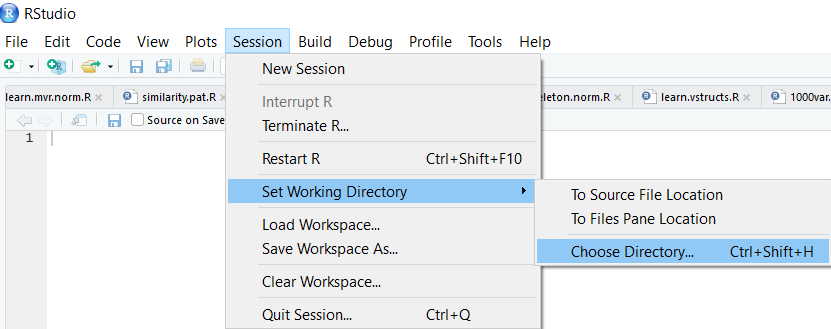
1. Load the following libraries: ggm, lcd and pcalg



1. R is always pointed at a directory on your computer. You can find out which directory by running the getwd (get working directory) function; this function has no arguments.



To change your working directory, use the following instruction and specify the path to the desired folder.



1. Download the R and csv files, and put them in the working directory.

**Examples**

1. **Gaussian Case**

#load the R code source

source("AMPCGs2019.R")

# copy & paste the following lines in the console panel

#of the RStudio

dag <- matrix(c( 0, 1, 1, 0, 0,0,

0, 0, 0, 1, 0,0,

0, 0, 0, 0, 1,0,

0, 0, 0, 0, 1,0,

0, 0, 0, 0, 0,1,

0,0,0,0,0,0),

6,6, byrow = TRUE)

N <- c("a","b","c","d","e","f")

dimnames(dag) <- list(N, N)

#plot “dag” from the R package lcd

draw(dag)

#check whether “toy.graph” is a chain graph or not

is.chaingraph(dag)

#First, put the “DAG.csv” file in your workspace. This file contains 3000 ra-ndom samples of the DAG mentioned above. Then read the file:

cg.data<-read.csv("DAG.csv")

#Learn the chain graph structure via LCD-like algorithm

ampcg<-learn.original.amp.normLCD(cg.data,p.value=0.05)

#print the result in the console panel of the RStudio

ampcg

#plot the learned CG

draw(ampcg)

#compare the learned CG to the true CG

comp.cgs(dag,ampcg)

#Learn largest deflagged graph (LDCG) structure

ldcg<-Largest\_DeflaggedAMPCG(ampcg)

#print the result in the console panel of the RStudio

Ldcg

#plot the learned LDCG

draw(ldcg)

#compare the learned LDCG to the true LDCG (dag)

comp.cgs(dag,ldcg)

#Learn the chain graph structure via PC-like algorithm

ampcg<-learn.amp.normPC(cg.data,p.value=0.05,method ="stable")

#print the result in the console panel of the RStudio

ampcg

#plot the learned CG

draw(ampcg)

#compare the learned CG to the true CG

comp.cgs(dag,ampcg)

#Learn largest deflagged graph (LDCG) structure

ldcg<-Largest\_DeflaggedAMPCG(ampcg)

#print the result in the console panel of the RStudio

Ldcg

#plot the learned LDCG

draw(ldcg)

#compare the learned LDCG to the true LDCG (dag)

comp.cgs(dag,ldcg)

1. **Discrete Case**

#First, put the “asia.csv” and “adj\_asia.csv” file in your workspace. This “asia.csv” file contains 5000 random samples of the ASIA network. The “adj\_asia.csv” file contains the adjacency matrix of the ASIA network.

#Read the files:

ds<-read.csv("asia.csv")

adj\_asia<-read.csv("adj\_asia.csv")

ref<-as(adj\_asia,"matrix")

rownames(ref)<-colnames(ref)

#plot the ASIA net

draw(ref)

#Learning using the stable PC-like algorithm:

cg<-learn.amp.multinomPC(ds,p.value=0.05,method = "stable")

#plot the learned net

draw(cg)

#compare the learned net to the ASIA net

comp.cgs(ref,cg)

#Learning using the LCD-like algorithm:

cg<-learn.amp.multinomLCD(ds,p.value = 0.05)

#plot the learned net

draw(cg)

#compare the learned net to the ASIA net

comp.cgs(ref,cg)