Causal discovery demo

```
# Load the required libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.4     v rsample      1.1.1
## v dials        1.2.0     v tune         1.1.1
## v infer        1.0.4     v workflows    1.1.3
## v modeldata    1.1.0     v workflowsets 1.0.1
## v parsnip      1.1.0     v yardstick    1.2.0
## v recipes      1.0.5
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(pcalg)
#library(bnlearn)
library(tidyr)
library(tidyverse)
library(tidymodels)
```

```
#Get Airbnb data
airbnb <- read_csv('data/airbnb-project-msba-sampled-10k.csv')
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 153995 Columns: 100
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (51): listing_url, state, city, name, summary, space, description, pict...
## dbl  (32): id, high_booking, host_id, latitude, longitude, accommodates, bat...
## lgl  (13): host_is_superhost, is_location_exact, requires_license, host_has_...
## date  (4): date, host_since, first_review, last_review
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Selecting variables
airbnb$price = as.numeric(gsub("\\$", "", airbnb$price))
```

```
## Warning: NAs introduced by coercion
```

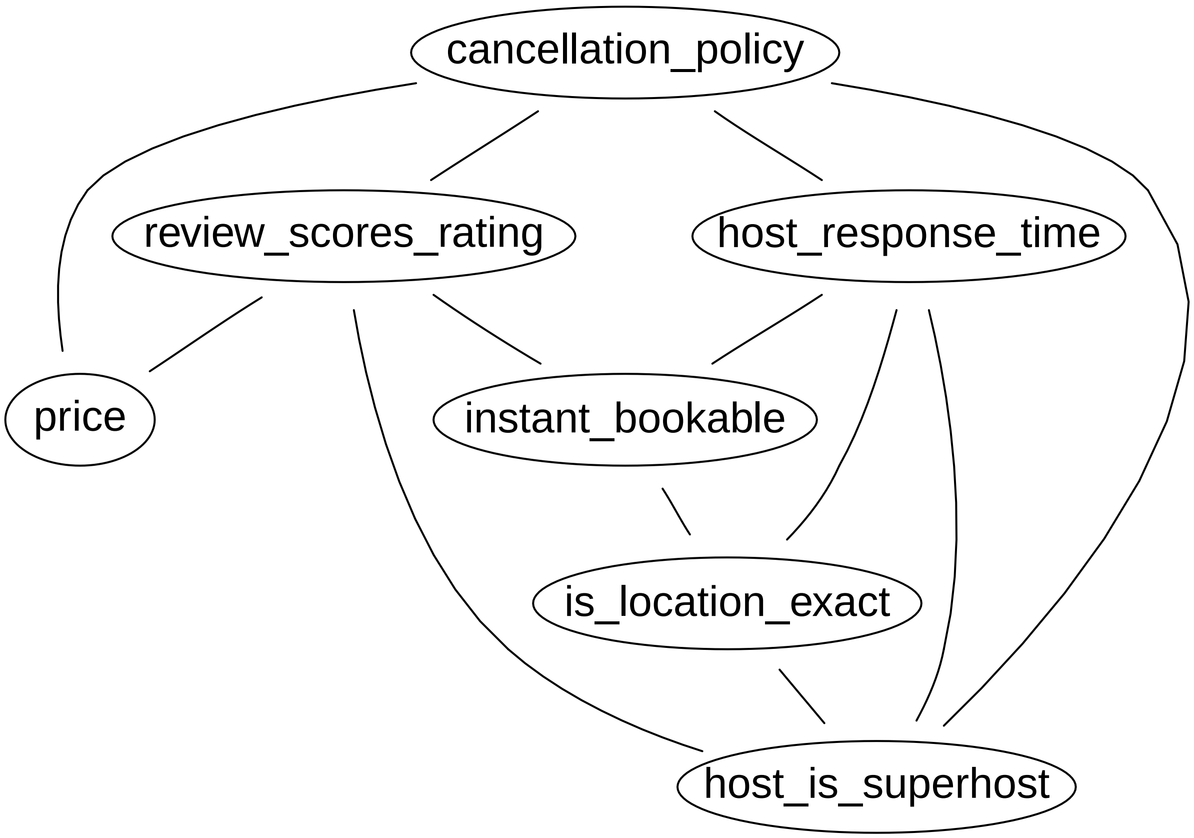
```
airbnb$instant_bookable=as.numeric(airbnb$instant_bookable)
airbnb$host_response_time=as.numeric(as.factor(airbnb$host_response_time))
airbnb$host_is_superhost=as.numeric(airbnb$host_is_superhost)
airbnb$is_location_exact=as.numeric(airbnb$is_location_exact)

df1<-airbnb%>%
  select(cancellation_policy,review_scores_rating,host_response_time,price,instant_bookable,is_location_


df1=drop_na(df1)
df1$cancellation_policy=ifelse(df1$cancellation_policy=="flexible",1,0)
```

```
#Creating Sufficient statistics with pairwise correlation
suffStat <- list(C = cor(df1), n = nrow(df1))

varNames <- colnames(df1)
```

```
#Defining Skeleton
skel.dfc <- skeleton(suffStat, indepTest = gaussCItest, labels = varNames, alpha = 0.01)
```

```
# Graphing the skeleton using a helper function:
mygraph <- function(pcgraph){
  g <- bnlearn::as.bn(pcgraph, check.cycles = FALSE)
  bnlearn::graphviz.plot(g, shape = "ellipse")
}
mygraph(skel.dfc)
```

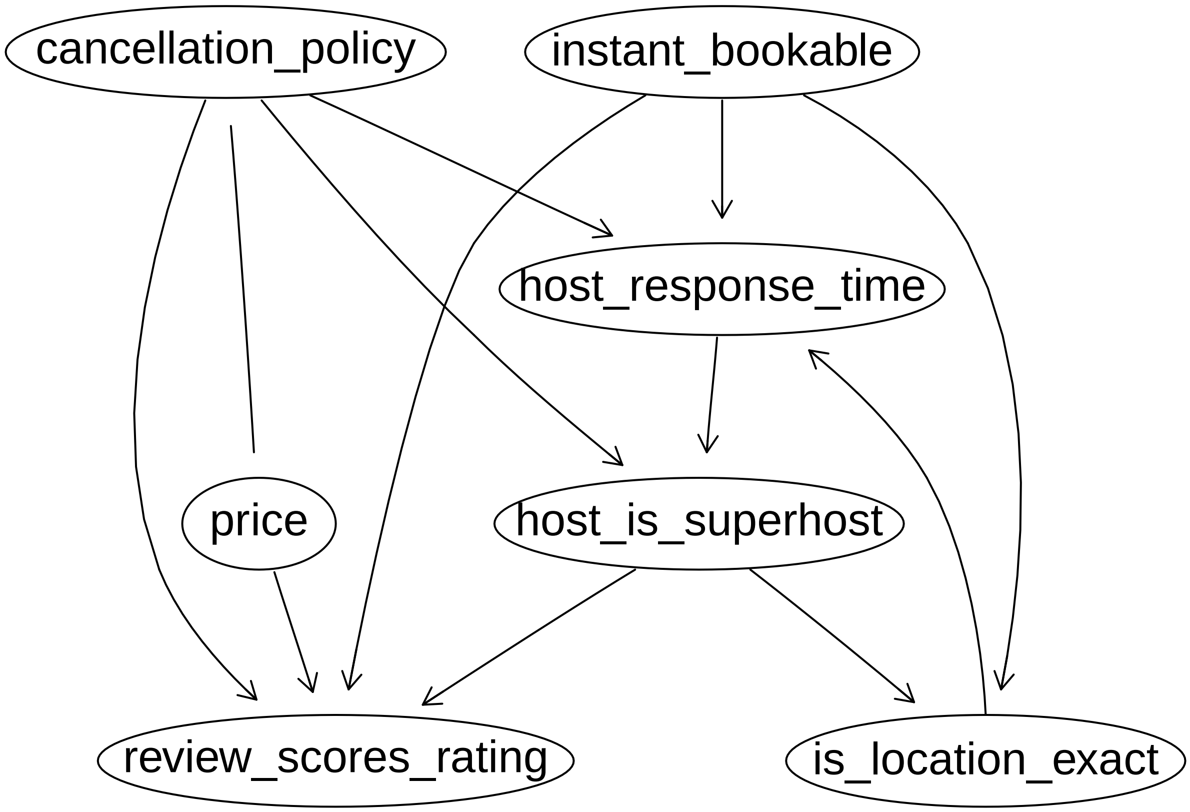
```
## Loading required namespace: Rgraphviz
```



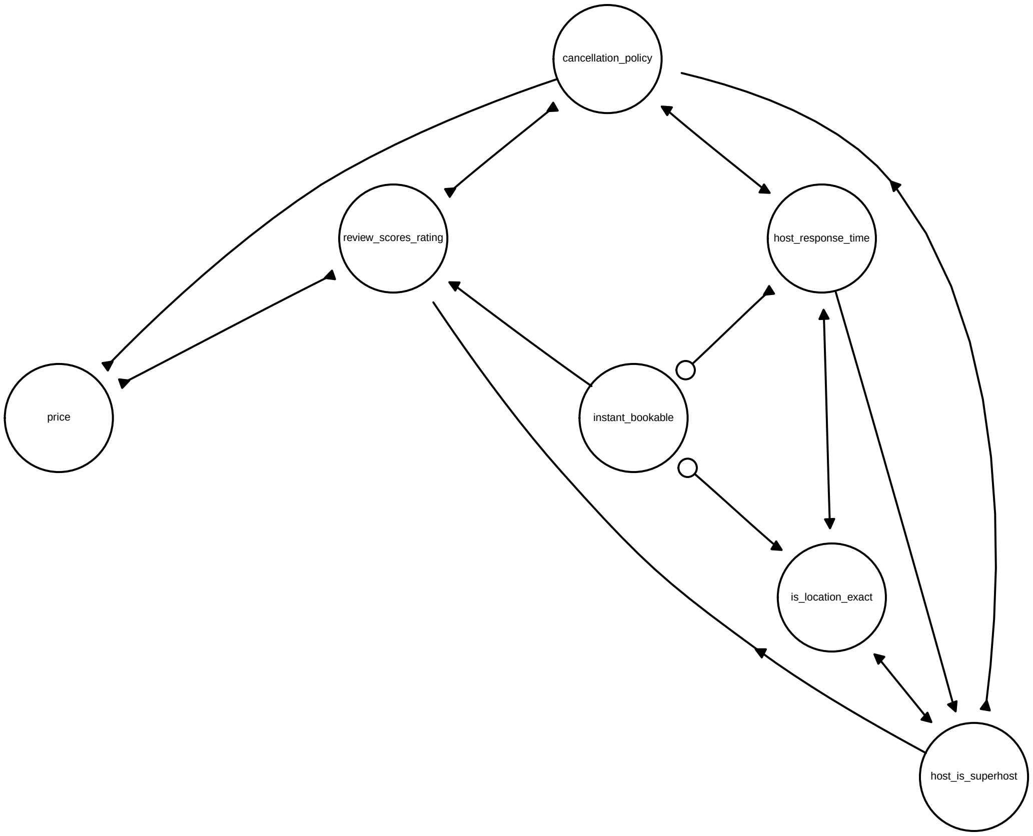
```r
# The PC-algorithm is implemented in function pc(). The arguments follow closely the arguments of skele

start_time <- Sys.time()

pc.dfc <- pc(suffStat, indepTest = gaussCItest, labels = varNames, alpha = 0.01)

end_time <- Sys.time()

end_time - start_time
```

```
## Time difference of 0.07770348 secs
```

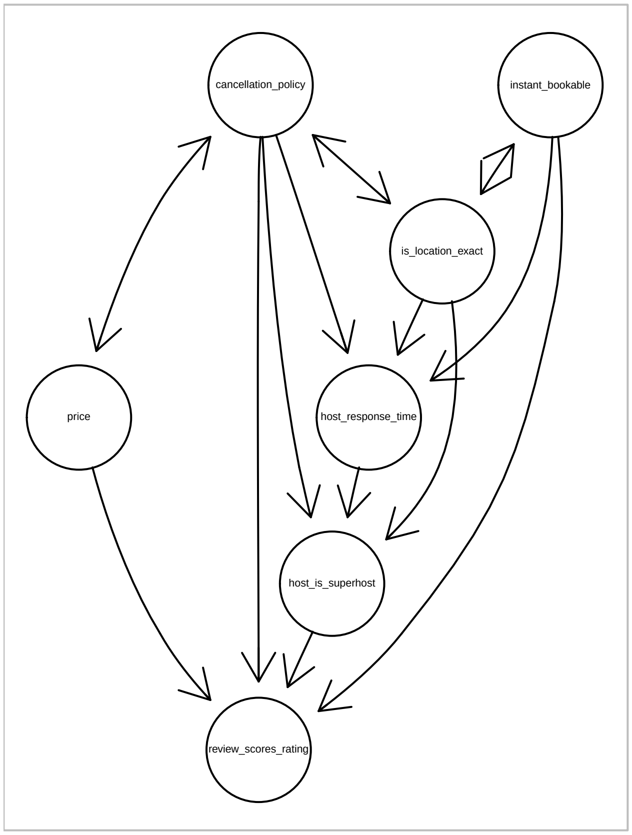
```r
mygraph(pc.dfc)
```



```r
# The RFCI algorithm:

rfci.dfc <- rfci(suffStat, indepTest = gaussCItest, labels = varNames, alpha = 0.01)

plot(rfci.dfc)
```



```
# The GES algorithm:

score <- new("GaussL0penObsScore",df1)

ges.fit <- ges(score)


par(mfrow=1:2)

plot(ges.fit$essgraph, main = ""); box(col="gray")
```


cancellation_policy
instant_bookable
is_location_exact
price
host_response_time
host_is_superhost
review_scores_rating