pcalg: A brief overview — *outdated* 0.x version

Markus Kalisch

March 2009

1 General Introduction

In this document, we will give a short overview of the R-package `pcalg`. The package can be used to analyze dependencies among random variables. The main functions are

- `pcAlgo`: (also see `?pcAlgo`) Use the PC-Algorithm to estimate the skeleton of a DAG. A DAG is a Directed Acyclic Graph and the skeleton of a DAG is the DAG without the arrowheads. In practice, the skeleton will only be a good estimate, if you have way more samples than nodes. If not, you will only get the edges, that show the most obvious "direct" dependencies ("direct" means: The dependence didn't go away, even when conditioning on any subset of the remaining variables). Using the plot option `zvalue.lwd` you can further qualify the edges by choosing their line width according to their reliability in the statistical tests. I.e., thick lines show reliable dependencies. The value of this function is an object of S4 class "pcAlgo" (see `?pcAlgo-class`) on which you can use `plot`, `summary` and `show`.
- `corGraph`: (see also `?corGraph`) Each pair of variables is tested for zero correlation on a given significance level. If a test is rejected, the corresponding edge in the graph is kept. I.e., the graph shows an edge between correlated nodes.

Furthermore, there are some functions that generate random data and perform statistical tests.

2 Some examples

In this section, we will show some simple examples with the most important functions. First, we generate a data set that will be used later on:

```
> library(pcalg)
> ## define parameters
> p <- 10 # number of random variables = nodes in the graph
> n <- 10000 # number of samples
> s <- 0.4 # sparsness of the graph
> ## generate random data
> set.seed(42)
> g <- randomDAG(p,s) # generate a random Directed Acyclic Graph
> d <- rmvDAG(n,g) # generate random samples from the DAG
```

The DAG that was produced by `g <- randomDAG(p,s)` is shown in Figure 1.

`d` contains the datamatrix you would start from in practice. Let's try to estimate the skeleton of the DAG and also compute the correlation graph:

```
> gSkel <- pcAlgo(d,alpha=0.05) # estimate of the skeleton

This function is deprecated and is only kept for backward compatibility. Please use skeleton, pc,

> gCor <- corGraph(d,alpha=0.05) # estimate the correlation graph
```

```
> library(Rgraphviz)
> plot(g) # plot generated DAG
```

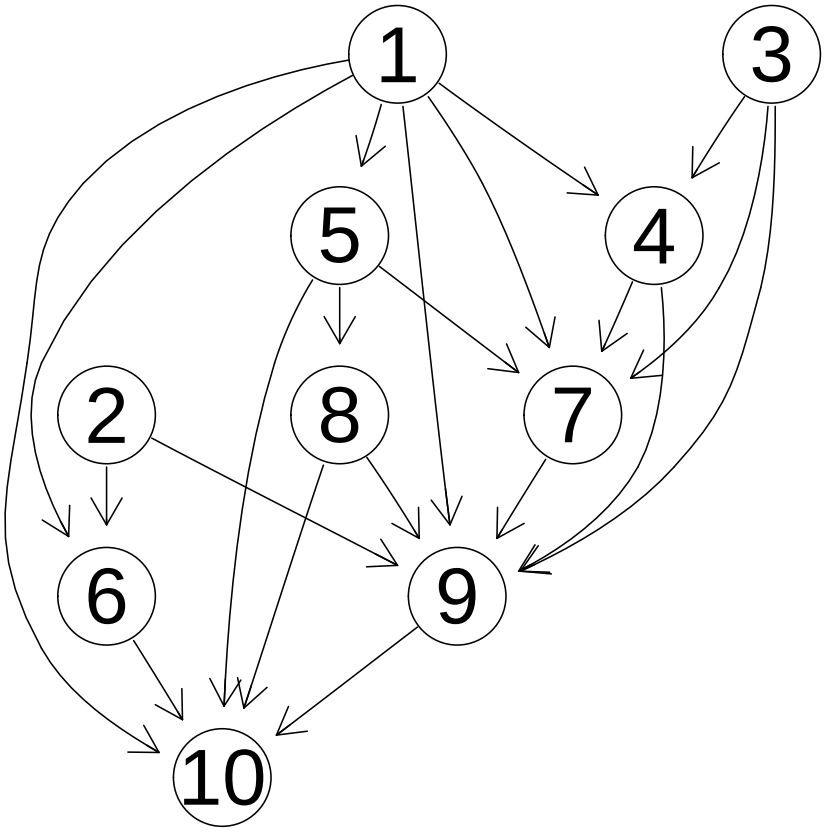

Figure 1: True DAG

```
> plot(gCor)
```

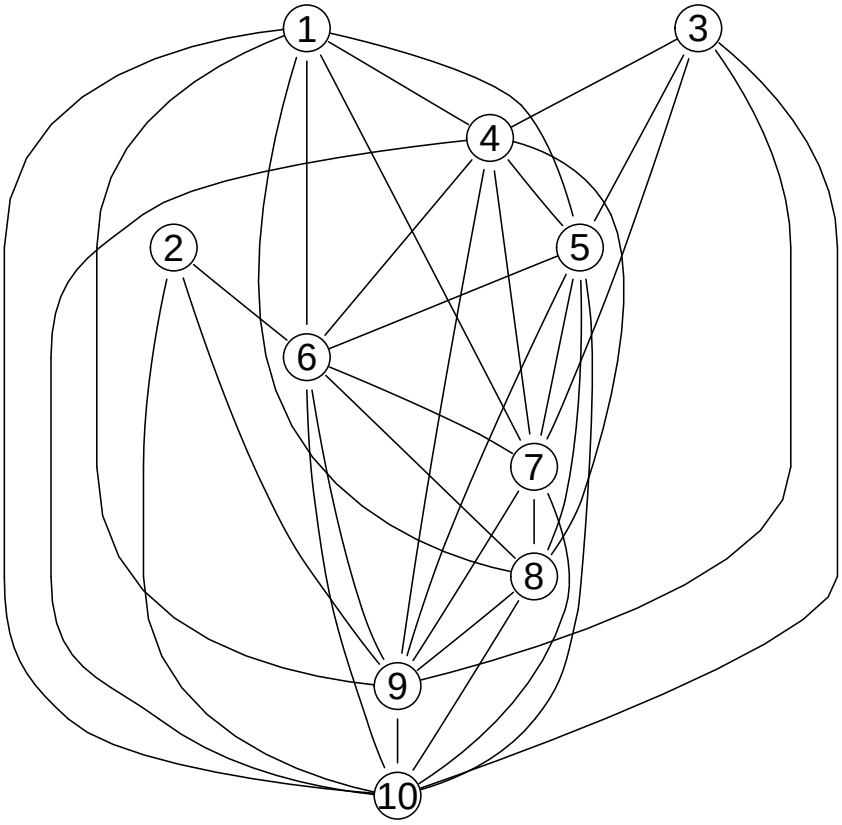

Figure 2: Correlation Graph

The result of `corGraph` is a `graph` object and can be plotted with `plot(gCor)` (see Figure 2).

The result of `pcAlgo` is a `pcAlog` object. To get an overview, you can use the `summary` function:

```
> summary(gSkel)  # shows some facts about the skeleton and the

Object of class 'pcAlgo', from Call:
 pcAlgo(dm = d, alpha = 0.05)

Nmb. edgetests during skeleton estimation:
==========================================
Max. order of algorithm:  5
Number of edgetests from m = 0 up to m = 5 :  81 295 329 277 36 0

Graphical properties of skeleton:
=================================
Max. number of neighbours:  5 at node(s) 9 10
Avg. number of neighbours:  3.2


> plot(gSkel)
```

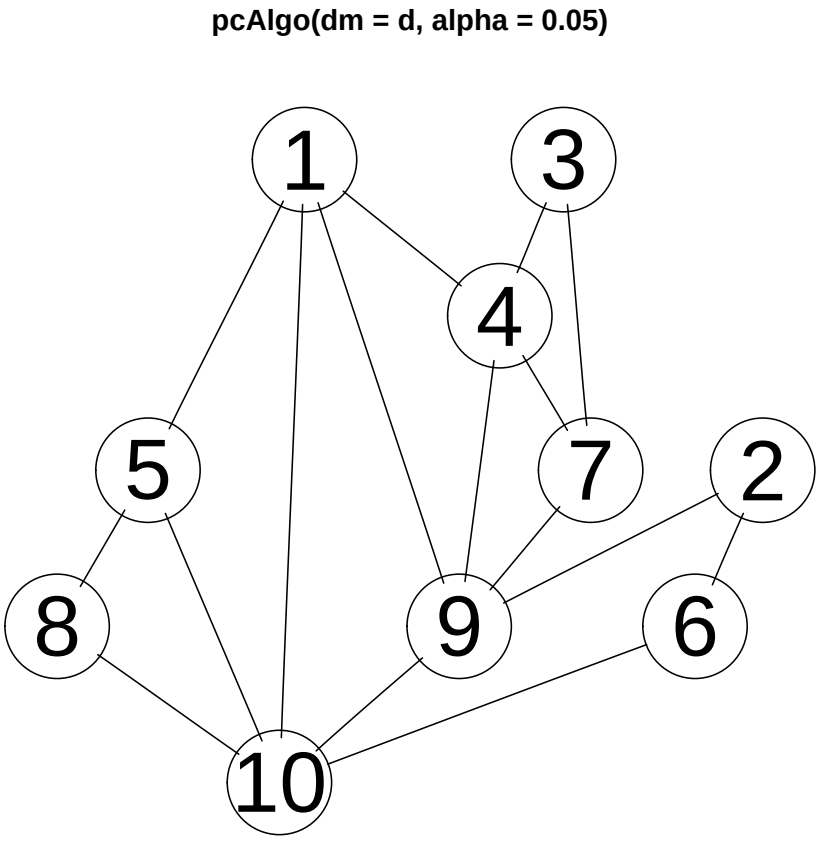

Figure 3: Estimated skeleton

When plotting the `pcAlgo` object by using `plot(gSkel)`, we obtain Figure 3; choosing the option `zvalue.lwd = TRUE`, we can adapt the line width of the graph according to the reliability of the corresponding statistical test (see Figure 4).

To compare the estimate of the skeleton with the true skeleton, use

```
> plot(gSkel, zvalue.lwd=TRUE)
```

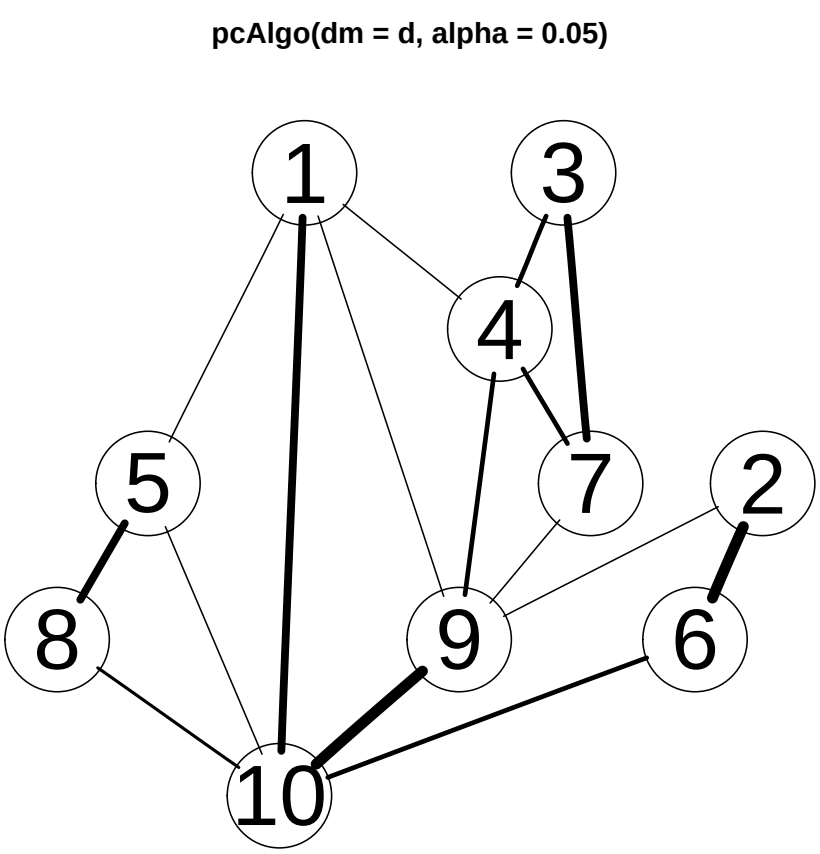

Figure 4: Estimated skeleton using variable line width

```
> compareGraphs(gSkel@graph,g)

      tpr       fpr       tdr
0.7619048 0.0000000 1.0000000
```