

# Basic R tutorial

February 22, 2024

## Contents

<b>Problem 3: Learning Bayesian networks from protein data</b>	<b>1</b>
Preparation to run the code . . . . .	1
Loading data . . . . .	1
Variables and observations . . . . .	1
Visualisation of transformed data . . . . .	1
Defining functions . . . . .	2
Default parameters . . . . .	3
Different parameters . . . . .	3
Render this .rmd into a pdf . . . . .	4

## Problem 3: Learning Bayesian networks from protein data

### Preparation to run the code

Setting seed for reproducibility and loading packages.

```
set.seed(42)
library("GGally")
library("BiDAG")
```

### Loading data

```
data <- read.table("2005_sachs_2_cd3cd28icam2_log_std.csv", sep="," , header=TRUE)
```

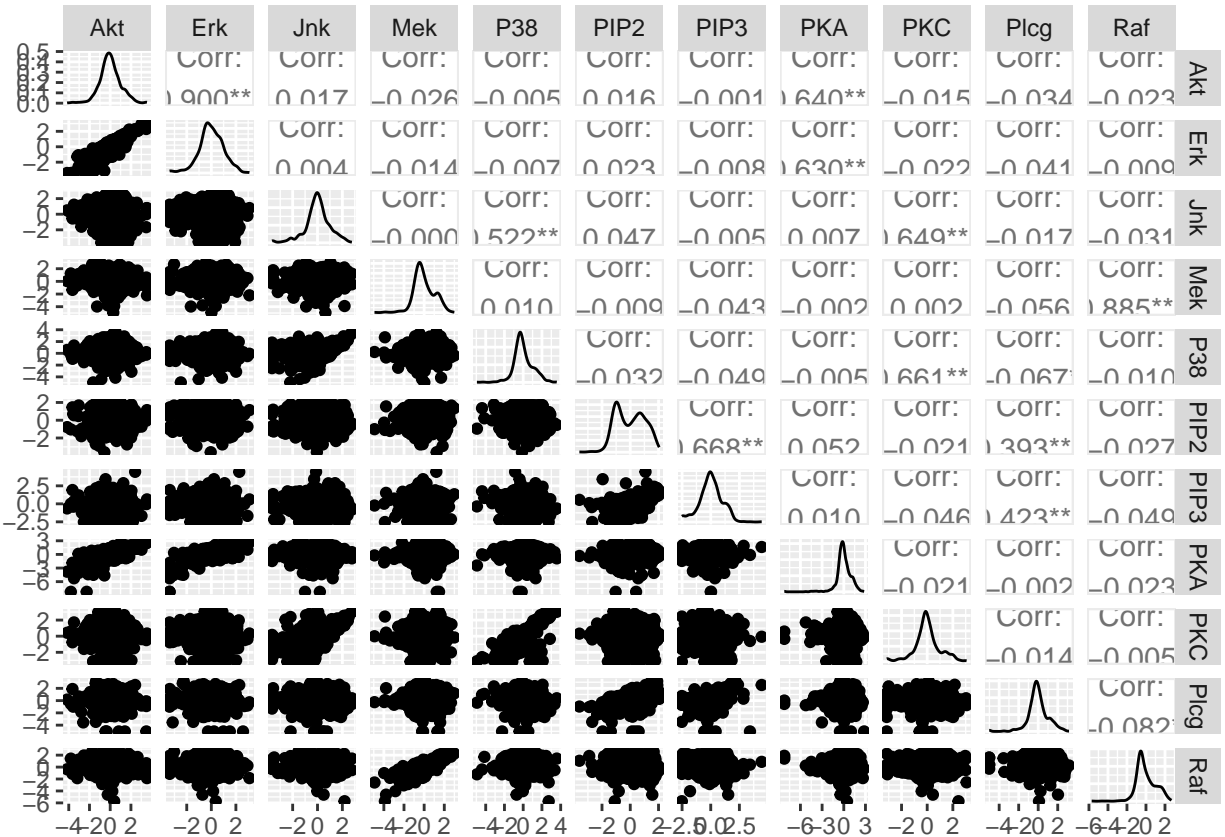
### Variables and observations

```
num_variables <- ncol(data)
num_observations <- nrow(data)
```

### Visualisation of transformed data

TODO: must resize the plot

```
pair_plot <- ggpairs(data, progress=FALSE)
pair_plot
```



TODO: execute the following routine with different parameters (10 times for each change of parameter)

TODO: include possibility of parallelization

## Defining functions

```
splitting_data <- function(data) {
  # Shuffling data
  indices <- 1:nrow(data)
  indices <- sample(length(indices))

  # Splitting data
  train_size <- ceiling(nrow(data)*0.8)
  train_indices <- indices[1:train_size]
  test_indices <- indices[(train_size+1):(length(indices))]

  # Checking if there is no overlap
  if(length(unique(c(train_indices, test_indices))) != length(c(train_indices, test_indices))) {
    print("Overlap!")
  }

  train_data <- data[row.names(data) %in% train_indices, ]
  test_data <- data[row.names(data) %in% test_indices, ]

  split_data <- list("train_data"=train_data, "test_data"=test_data)
```

```

    return(split_data)
}

training_BN <- function(data, bgepar) {
  init_score_par <- scoreparameters("bge", data$train_data, bgepar)
  learnt_BN <- iterativeMCMC(init_score_par, verbose=FALSE)

  return(learnt_BN)
}

testing_BN <- function(data, BN, bgepar) {
  test_score_par <- scoreparameters("bge", data$test_data, bgepar)
  test_score <- scoreagainstDAG(test_score_par, BN$DAG)
  return(mean(test_score))
}

```

## Default parameters

```

bgepar <- list(am=1, aw=NULL, edgepf=1)
split_data <- splitting_data(data)
learnt_BN <- training_BN(split_data, bgepar)
mean_test_score <- testing_BN(split_data, learnt_BN, bgepar)
print(paste0("Average BGe score on testing data: ", mean_test_score))

```

```
## [1] "Average BGe score on testing data: -12.1192090033328"
```

## Different parameters

```

ams <- c(10^(-5), 10^(-3), 10^(-1), 10, 10^2)

for(am in ams) {
  print(paste0("Running BN training with a_m: ", am))
  bgepar <- list(am=am, aw=NULL, edgepf=1)
  split_data <- splitting_data(data)
  learnt_BN <- training_BN(split_data, bgepar)
  mean_test_score <- testing_BN(split_data, learnt_BN, bgepar)
  print(paste0("Average BGe score on testing data: ", mean_test_score))
}

```

```

## [1] "Running BN training with a_m: 1e-05"
## [1] "Average BGe score on testing data: -12.3484288167485"
## [1] "Running BN training with a_m: 0.001"
## [1] "Average BGe score on testing data: -12.8230694678731"
## [1] "Running BN training with a_m: 0.1"
## [1] "Average BGe score on testing data: -12.5901244279439"
## [1] "Running BN training with a_m: 10"
## [1] "Average BGe score on testing data: -13.0520355093837"
## [1] "Running BN training with a_m: 100"
## [1] "Average BGe score on testing data: -13.6082779927528"

```

Render this .rmd into a pdf

```
library(rmarkdown)
render("1/1.Rmd", pdf_document(TRUE), "1.pdf") # TRUE adds table of content
```