# Code for 'Inference on rho and r by AJ vdM and PCN Groenewald'

Sean van der Merwe

2024-05-06

## Table of contents



## Introduction

This code defines functions that take an arbitrary bivariate sample and produce various quantities of interest related to the correlation.

### Model definitions

Assuming that the underlying distribution is bivariate normal, we have from Berger & Son (2008) that

$$\rho = \frac{Y}{\sqrt{1+Y^2}}$$
$$\text{where } Y = \frac{r}{\sqrt{1-r^2}}\frac{\sqrt{\chi^2_{n-2}}}{\sqrt{\chi^2_{n-1}}} - \frac{N(0,1)}{\sqrt{\chi^2_{n-1}}}$$

This is both the fiducial distribution and the objective posterior under the right-Haar prior $\pi(\mu_1, \mu_2, \rho, \sigma_1, \sigma_2) \propto \frac{1}{\sigma_1^2(1-\rho^2)}$.

Further, should predictive distributions be of interest, we have that

$$r = \frac{X}{\sqrt{1+X^2}}$$
$$\text{where } X = \frac{\rho}{\sqrt{1-\rho^2}}\frac{\sqrt{\chi^2_{n-1}}}{\sqrt{\chi^2_{n-2}}} + \frac{N(0,1)}{\sqrt{\chi^2_{n-2}}}$$

## Model implementations

### Simulation functions

The first function takes a single sample correlation $r$ and a single sample size $n$, and produces a vector of $\rho$ simulations. The second function takes a vector of $\rho$ simulations and a single sample size $n$ to produce a vector of predictive $r_{future}$ simulations.

```
rho_post_sim <- function(r, n, n_sims = 10000) {
  y <- r*sqrt(rchisq(n_sims, n-2)/rchisq(n_sims, n-1)/(1-(r^2))) -
    rnorm(n_sims)/sqrt(rchisq(n_sims, n-1))
  y/sqrt(y^2 + 1)
}
r_pred_sim <- function(rho_sims, n) {
  n_sims <- length(rho_sims)
  x <- rho_sims*sqrt(rchisq(n_sims, n-1)/rchisq(n_sims, n-2)/(1-(rho_sims^2))) +
    rnorm(n_sims)/sqrt(rchisq(n_sims, n-2))
  x/sqrt(x^2+1)
}
```

### Helper functions for simulations

Some helper functions are defined for working with the simulations.

```
shortestinterval <- function(postsims, width = 0.95) { # Coded by Sean van der Merwe, UFS
  postsims |> sort() -> sorted.postsims
  round(length(postsims)*width) -> gap
  sorted.postsims |> diff(gap) |> which.min() -> pos
  sorted.postsims[c(pos, pos + gap)] }

pvalfunc <- function(sims, target=0) { 2*min(mean(sims<target),mean(sims>target)) }

summary_stats <- function(sims_vector, width = 0.95) {
  v <- sims_vector
  interval <- shortestinterval(v, width)
  sym_int <- quantile(v, c((1-width)/2, (1+width)/2))
  dens <- density(v)
  c(
    Mean = mean(v),
    Lower = interval[1],
    Upper = interval[2],
    p_value = pvalfunc(v),
    Median = median(v),
    Mode = dens$x[which.max(dens$y)],
    L = sym_int[1],
    U = sym_int[2]
  )
}
```

### Functions for working with real data sets

Here we define some functions that help us move to between data sets with multiple columns and the theory for the bivariate normal above.

The functions below assume we have an $n$ by $k$ numeric data matrix, that may contain any number of missing values in any pattern. The correlations are calculated by considering every pair of columns and using the pairwise complete rows of those two columns only. The resulting matrix of correlations is **not** a correlation matrix **if** there are any missing values - it is then only a block of separate correlation estimates.

The first function takes the numeric data matrix (or data frame) and calculates the pairwise complete correlations.

```
correlation_matrix <- function(num_data_matrix) {
  if (any(class(num_data_matrix) %in% "data.frame")) {
    num_data_matrix <- as.matrix(num_data_matrix)
  }
  corrmat <- cor(num_data_matrix, use="pairwise.complete.obs")
  rownames(corrmat) <- colnames(corrmat) <- colnames(num_data_matrix)
  corrmat
}
```

The following function calculates Bayesian two-sided interpretations of the p-values associated with the test $H_0 : \rho = 0$ for all the correlations in the matrix.

```
p_val_matrix <- function(num_data_matrix) {
  corrmat <- correlation_matrix(num_data_matrix)
  nc <- ncol(num_data_matrix)
  p_values <- matrix(0, nc, nc)
  rownames(p_values) <- colnames(p_values) <- colnames(corrmat)
  seq_len(nc-1) |> sapply(\(i) {
    seq((i+1), nc) |> sapply(\(j) {
      n <- sum(!(is.na(num_data_matrix[,i]) | is.na(num_data_matrix[,j])))
      p_values[i, j] <<- rho_post_sim(corrmat[i, j], n) |> pvalfunc()
    })
  })
  p_values + t(p_values) + diag(rep(1, nc))
}
```

The last function uses the *corrplot* function from the *corrplot* package to illustrate the results.

```
corrplot_exact <- function(num_data_matrix, crosssize = 1.8, textsize = 0.9) {
  if (any(class(num_data_matrix) %in% "data.frame")) {
    num_data_matrix <- as.matrix(num_data_matrix)
  }
  corrmat <- correlation_matrix(num_data_matrix)
  pmat <- p_val_matrix(num_data_matrix)
  corrplot::corrplot(corrmat, method = 'color', p.mat = pmat, insig = 'pch', pch.cex = cross
  list(correlations = corrmat, p_values = pmat) |> invisible()
}
```

### Fischer's transformation

The 'standard' approach that has been used for a long time is applied below, in order to compare if desired.

The next function calculates the number of complete observation pairs used to obtain the correlations above. This function is used for efficiency when using *Fischer's transformation* only, not when using the exact approach.

```
n_matrix <- function(num_data_matrix) {
  nc <- ncol(num_data_matrix)
  seq_len(nc) |> sapply(\(i) {
    seq_len(nc) |> sapply(\(j) {
      sum(!(is.na(num_data_matrix[,i]) | is.na(num_data_matrix[,j])))
    })
  })
}
```

The below function uses the *corrplot* function from the *corrplot* package to illustrate the results. It uses *Fischer's transformation* by default to calculate the p-values.

```
corrplot_Fischer <- function(num_data_matrix, crosssize = 1.8, textsize = 0.9) {
  if (any(class(num_data_matrix) %in% "data.frame")) {
    num_data_matrix <- as.matrix(num_data_matrix)
  }
  corrmat <- correlation_matrix(num_data_matrix)
  nmat <- n_matrix(num_data_matrix)
  testmat <- abs(corrmat)
  pmat <- (1-pt((testmat/sqrt(1-testmat^2))*sqrt(nmat-2),(nmat-2)))/2
  corrplot::corrplot(corrmat, method = 'color', type = 'full',
                     order = 'original', p.mat=pmat, insig = 'pch',
                     pch.cex = crosssize, tl.cex = textsize, tl.col='black')
  list(correlations = corrmat, p_values = pmat) |> invisible()
}
```

## Data set applications

Here we apply the above with data sets.

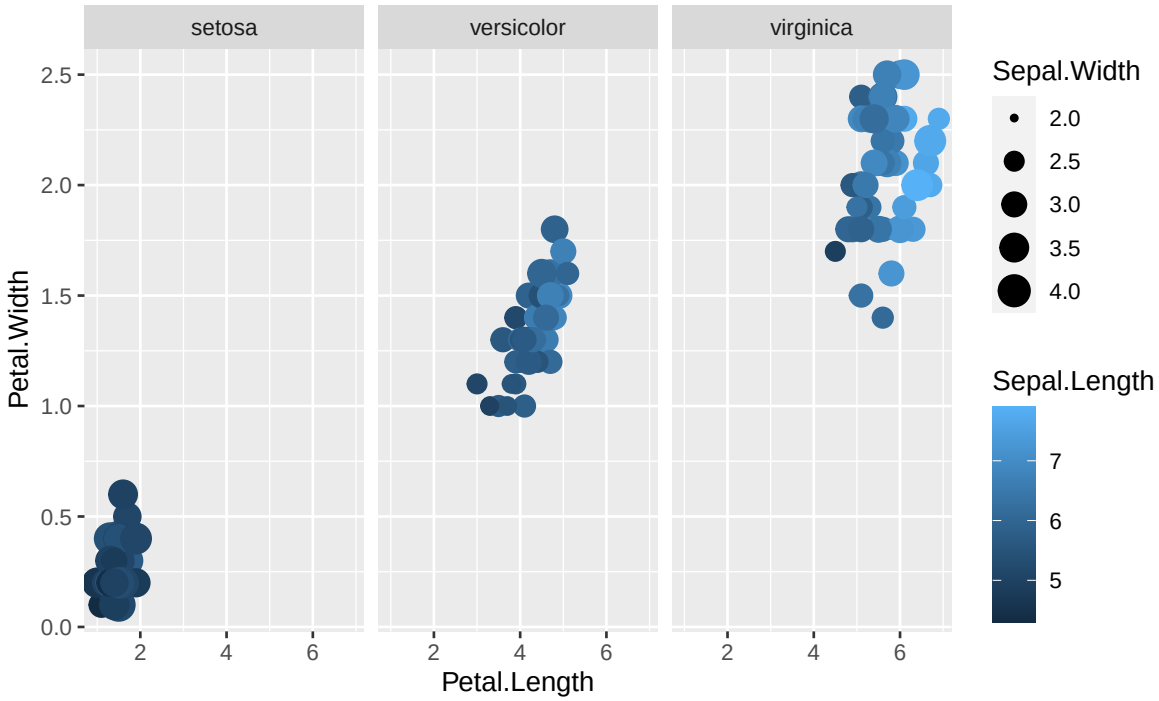
### Iris data

This data set is famous throughout statistics and should work here. Let us examine the correlations within each species.

```
library(tidyverse) |> suppressPackageStartupMessages()
```

Let us illustrate the raw data first.

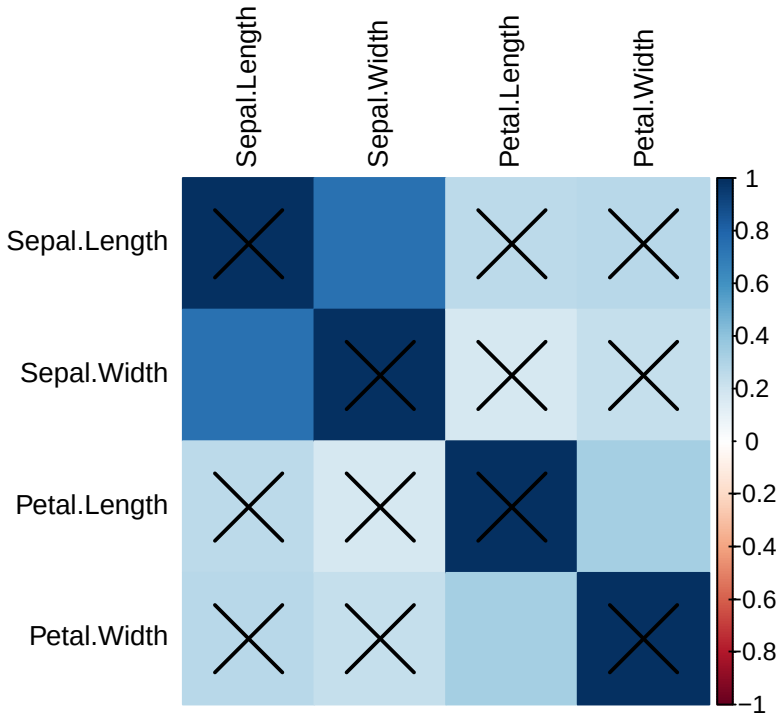
```
iris |> ggplot(aes(x = Petal.Length,
                   y = Petal.Width,
                   colour = Sepal.Length,
                   size = Sepal.Width)) +
  geom_point() + facet_grid(~Species)
```



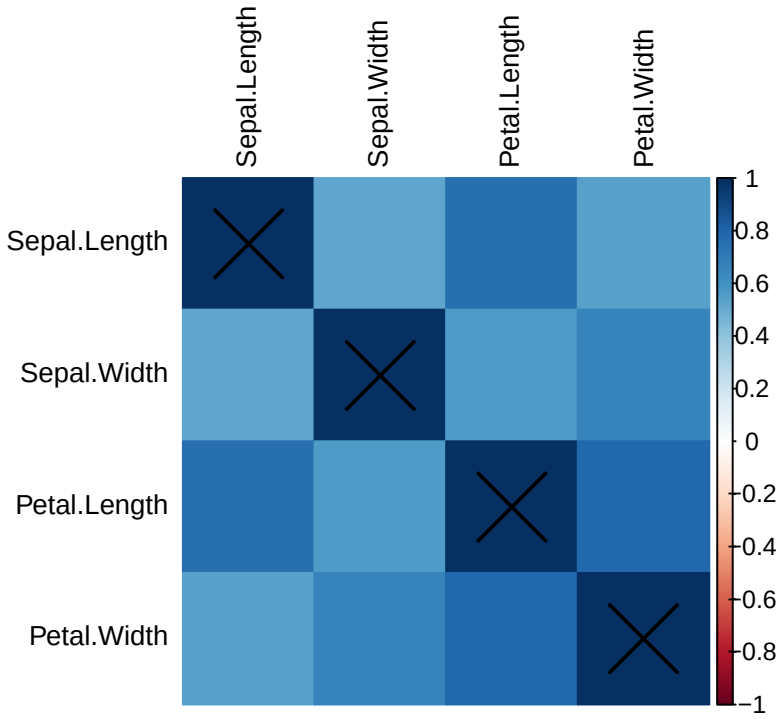

Then the correlations:

```
species_names <- iris$Species |> unique() |> as.character()
species_names |> sapply(\(species_name) {
  cat('\n\n### ', species_name, '\n\n', sep = '')
  num_data_matrix <- iris |>
    filter(Species %in% species_name) |>
    select(-Species)
  num_data_matrix |>
    corrplot_exact(crosssize = 5)
}) |> invisible()
```

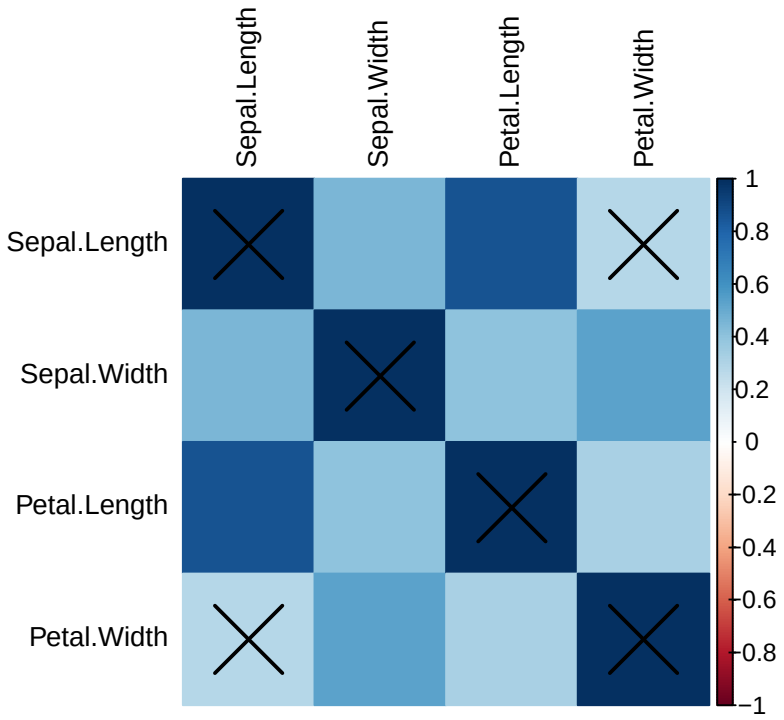
**setosa**


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

**versicolor**


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

**virginica**




### Predictive distribution

Let us consider now the possible correlations to be observed from future setosa flowers. We will focus on the correlation between petal length and width here for illustration.

```
setosa_petals <- iris |>
    filter(Species %in% "setosa") |>
    select(Petal.Length, Petal.Width)
n_obs <- nrow(setosa_petals)
r_obs <- cor(setosa_petals)[1,2]
```

There are 50 observations with a correlation of 0.33163.

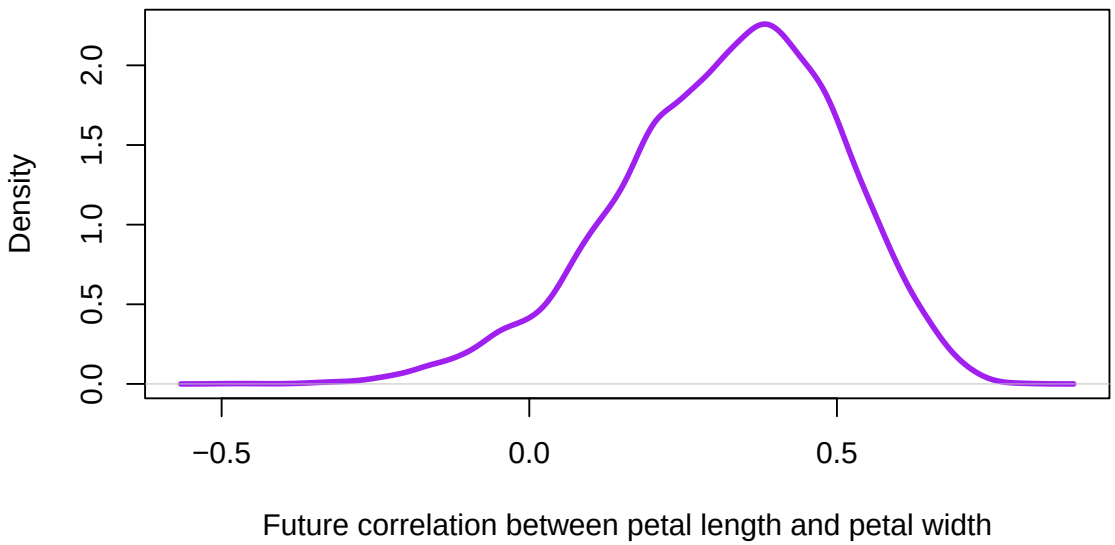
We obtain simulations from the posterior predictive distribution and summarise it.

```
future_setosa_petals <- r_obs |>
  rho_post_sim(n_obs) |>
  r_pred_sim(n_obs)
future_setosa_petals |>
```

```r
  summary_stats() |>
  knitr::kable(digits = 3)
```

| | x |
|---|---|
| Mean | 0.323 |
| Lower | -0.035 |
| Upper | 0.655 |
| p_value | 0.099 |
| Median | 0.339 |
| Mode | 0.382 |
| L.2.5% | -0.066 |
| U.97.5% | 0.631 |

```r
future_setosa_petals |> density() |>
  plot(main='', xlab = 'Future correlation between petal length and petal width',
       lwd = 3, col = 'purple')
```



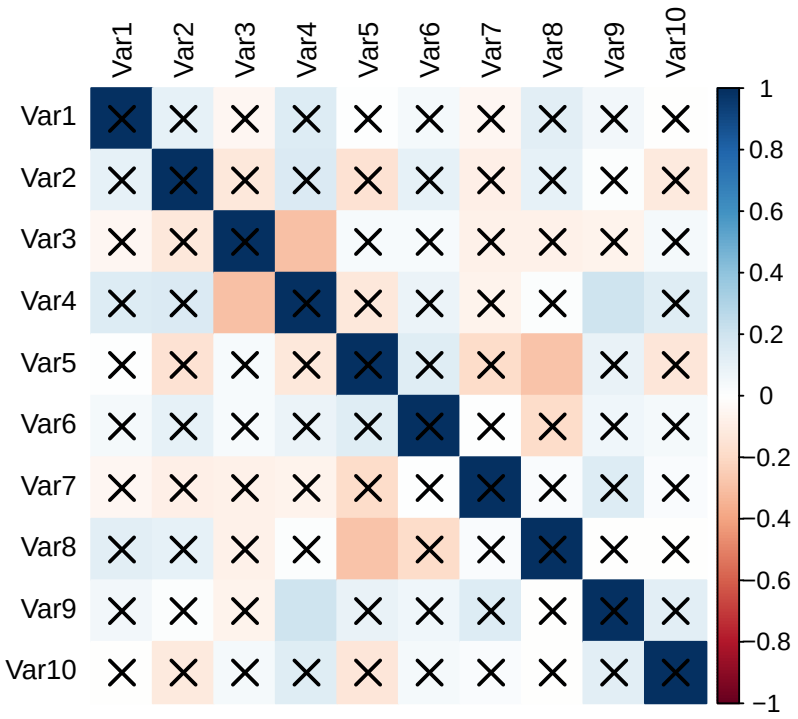
### Independent data set

Here we simulate a data set with no correlations to see how many come up as significant by coincidence.

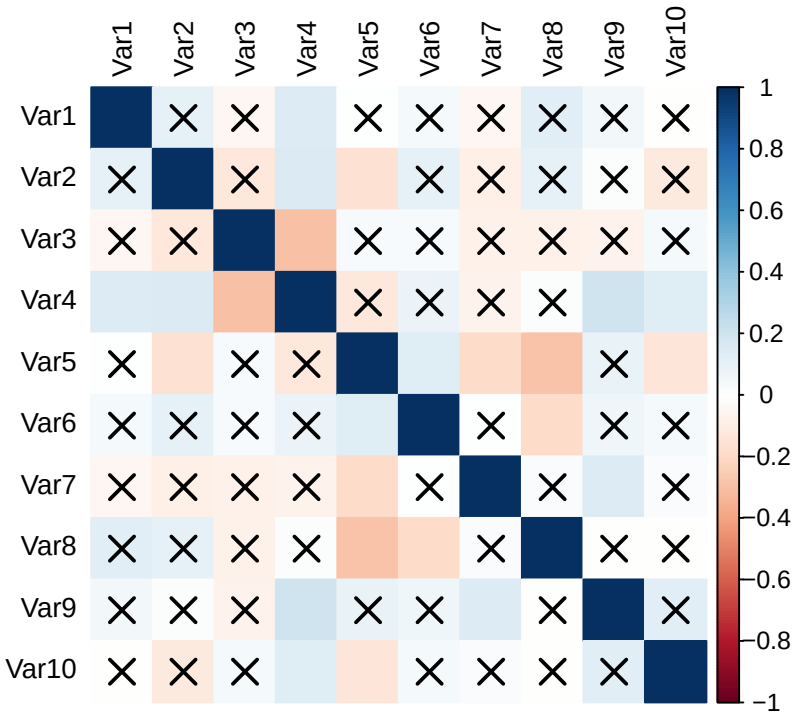
```
set.seed(12345)
X <- rnorm(1000) |> matrix(100, 10)
colnames(X) <- paste0('Var', seq_len(ncol(X)))
```

We plot the correlations using the exact method for p-values and using Fisher's transformation for p-values, in that order.

```
X |> corrplot_exact()
```




```
X |> corrplot_Fischer()
```




**We notice far less false positives from the exact method.**