Spearman's test: introduction

The Spearman's correlation coefficient is simply the Pearson's correlation between the ranks of the x_i's and the y_i's. Let $\mathbf{x} = x_1, ..., x_n$ and $\mathbf{y} = y_1, ..., y_n$ If $(r_1, ...r_n) = rank(x_1, ..., x_n)$ and $(s_1, ...s_n) = rank(y_1, ..., y_n)$, then the Spearman's correlation coefficient ρ_{xy} between $\mathbf{x}$ and $\mathbf{y}$ is defined as

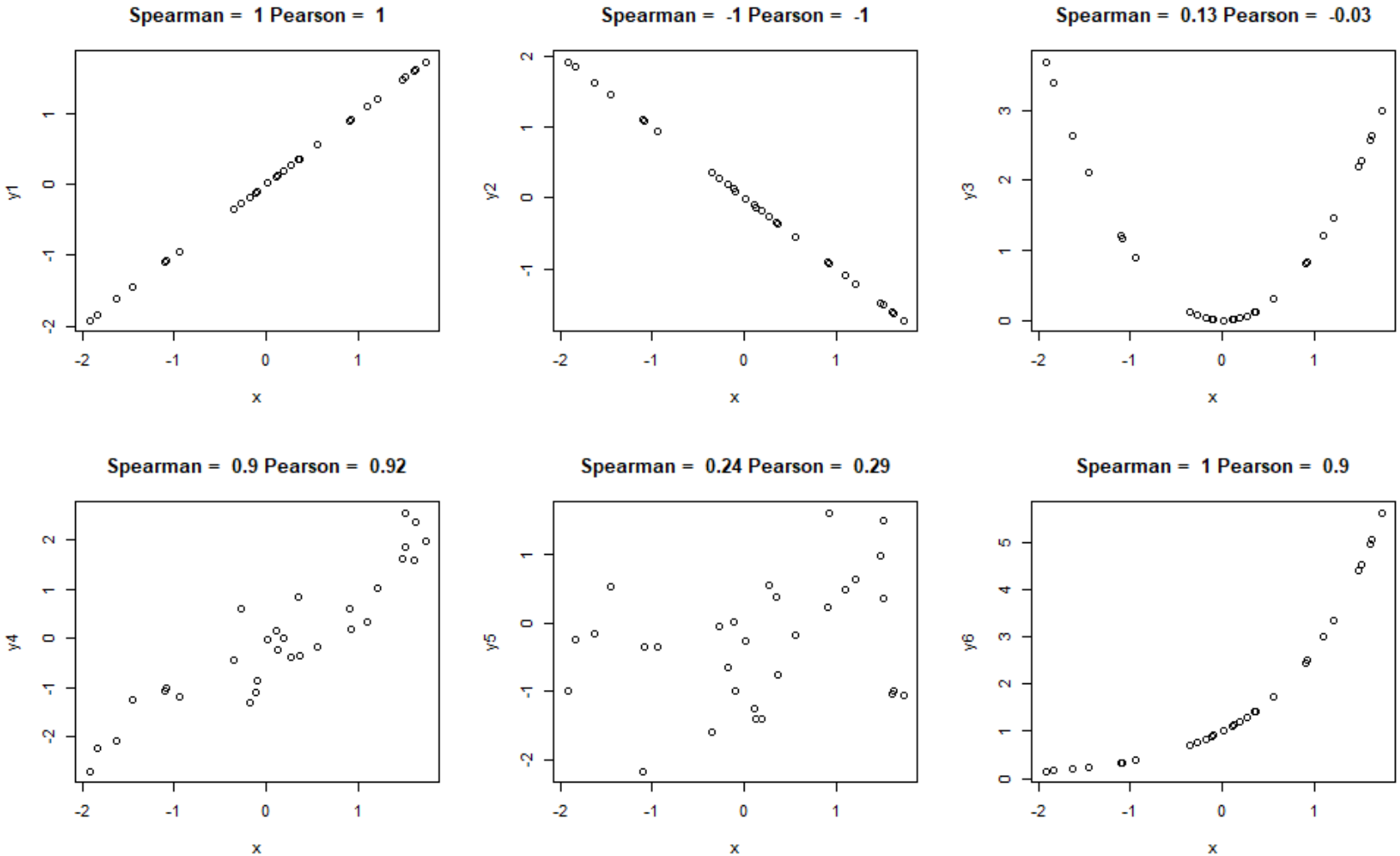
$$\rho_{xy} = \frac{\sum_{i=1}^{n}(r_i - \bar{r})(s_i - \bar{s})}{\sqrt{\sum_{i=1}^{n}(r_i - \bar{r})^2 \sum_{i=1}^{n}(s_i - \bar{s})^2}}$$

with $\bar{r} = \bar{s} = (n+1)/2$. When there are no ties, this expression can be reduced to

$$\rho_{xy} = 1 - \frac{6\sum_{i=1}^{n}(r_i - s_i)^2}{n(n^2 - 1)}$$

The coefficient ρ_{xy} takes values within the interval $[-1, 1]$, -1 being perfect negative correlation and 1 being perfect positive correlation.

Spearman vs Pearson correlation



Asymptotic distribution

For large n, then we have that $\sqrt{n-1}\rho_{xy} \sim N(0,1)$. Moreover, we also have, for large n, that

$$\sqrt{n-2}\frac{\rho_{xy}}{\sqrt{1-\rho_{xy}^2}} \sim t_{n-2}$$

This is useful to set up an hypothesis test and compute asymptotic p-values associated with the test having null hypothesis $H_0 : \rho_{xy} = 0$ versus $H_1 : \rho_{xy} \neq 0$ for the bilateral test.

Example 1

Example 1: Compute manually and in R the Spearman's correlation coefficient of the data shown in the below table.

n	x_i	y_i	r_i	s_i	$r_i - s_i$	$(r_i - s_i)^2$
1	273	288	3	3	0	0
2	285	284	7	2	-5	25
3	270	298	1	6	5	25
4	272	271	2	1	-1	1
5	278	307	5	9	4	16
6	276	301	4	8	4	16
7	291	299	9	7	-2	4
8	290	295	8	5	-3	9
9	279	293	6	4	-2	4
10	292	352	10	10	0	0

Example 2

In this example, the Spearman's correlation coefficient is equal to

$$\rho_{xy} = 1 - \frac{6\sum_{i=1}^{n}(r_i - s_i)^2}{n(n^2-1)} = 1 - \frac{6 * 100}{10(10^2-1)} = 1 - \frac{600}{990} \approx 0.394$$

Example 2: Create a program in R and Python that implements the Spearman's correlation coefficient. Then, for $n = 100$ values for the x_i's and the y_i's, generated from $U_{[0,1]}$, use the large sample approximations to compute asymptotic p-values at level of significance $\alpha = 5\%$ and compare the results with the output of **cor.test(x,y, method= 'spearman', conf.int = 0.95, alternative = 'two.sided')** in R and the equivalent inbuilt function in Python.

R code for example 2 (1/2)

```r
# data
x <- c(273, 285, 270, 272, 278, 276, 291, 290, 279, 292)
y <- c(288, 284, 298, 271, 307, 301, 299, 295, 293, 352)

# Spearman's correlation coefficient
cor.test(x, y, method = 'spearman', conf.level = 0.95,
         alternative = 'two.sided')

# Spearman's rank correlation rho
#
# data:  x and y
# S = 100, p-value = 0.2629
# alternative hypothesis: true rho is not equal to 0
# sample estimates:
#       rho
# 0.3939394

plot(x,y)


# function that computes the Spearman's correlation coefficient
spearman <- function(x, y) {
  n <- length(x)
  numerator <- sum((rank(x) - (n+1)/2)*(rank(y) - (n+1)/2))
  denominator <- sqrt( sum((rank(x) - (n+1)/2)^2)*sum((rank(y) - (n+1)/2)^2))
  return(numerator/denominator)
}

spearman(x = x, y = y)
# [1] 0.3939394
```

R code for example 2 (2/2)

```r
# examples of p-value calculation using large-sample approximations
set.seed(2021)
n=100 ; x=runif(n) ; y=runif(n)
cor.test(x, y, method = 'spearman', conf.level = 0.95,
         alternative = 'two.sided')
#
# Spearman's rank correlation rho
#
# data:  x and y
# S = 160598, p-value = 0.7194
# alternative hypothesis: true rho is not equal to 0
# sample estimates:
#       rho
# 0.03631563

# p-value with standard Normal approximation
2*pnorm(q=(sqrt(n-1) * spearman(x = x, y = y)), mean = 0, sd =1,
           lower.tail = FALSE)
# [1] 0.7178483

# p-value with Student's approximation
2*pt(q = (sqrt(n-2) * (spearman(x = x, y = y) / sqrt( 1 - (spearman(x = x, y = y
     ))^2))),
        df = n-2,
        lower.tail = FALSE)
# [1] 0.7198129
```

Python code for example 2 (1/2)

```
1  import numpy as np
2
3  x = ([273, 285, 270, 272, 278, 276, 291, 290, 279, 292])
4  y = ([288, 284, 298, 271, 307, 301, 299, 295, 293, 352])
5
6  from scipy import stats
7  stats.spearmanr(x,y)
8
9  from scipy.stats import rankdata
10
11 def spearman(x,y):
12     n = len(x)
13     numerator = np.sum((rankdata(x, method='min') - (n+1)/2)*(rankdata(y, method
        ='min') - (n+1)/2))
14     denominator = (( np.sum(   np.square(rankdata(x, method='min') - (n+1) / 2)
        )) *np.sum( np.square(rankdata(y, method='min') - (n+1)/2))  )**0.5
15     return(numerator/denominator)
16
17 spearman(x,y)
```

Python code for example 2 (2/2)

```
# examples of p-value calculation using large-sample approximations
np.random.seed(2023)
n=100
x=np.random.uniform(0,1,n)
y=np.random.uniform(0,1,n)
stats.spearmanr(x,y, alternative='two-sided')

# SignificanceResult(statistic=0.020774077407740774, pvalue=0.8374540868726486)

from scipy.stats import norm

# p-value with standard Normal approximation
2*norm.sf(x=((n-1)**0.5 * spearman(x = x, y = y)), loc=0, scale=1)

# 0.8362445788532031

from scipy.stats import t

# p-value with student approximation
2*t.sf(x=((n-1)**0.5 * spearman(x = x, y = y)), df=n, loc=0, scale=1)

# 0.8366648646693274
```

References

Bagdonavičius V., Kruopis J., Nikulin M. S., Non-parametric Tests for Complete Data (2011), Wiley, ISBN 978-1-84821-269-5 (hard-back)

The R Project for Statistical Computing:
https://www.r-project.org/

Python:
https://www.python.org/

course notes