Blatt 4

Christian Peters

H8)

Lese zunächst die Daten in Form einer Datenmatrix ein:

```
X <- matrix(c(-0.6, 1.4, 1.0, -0.3, -0.8, 1.2, 0.4, 0.5, -0.6, 0.3, -0.5, 0.9,
              1.5, -0.8, -0.7, 1.0, 0.3, 1.2, -1.4, 0.0), ncol = 2)
colnames(X) <- c('X', 'Y')
X
```

```
##          X    Y
##  [1,] -0.6 -0.5
##  [2,]  1.4  0.9
##  [3,]  1.0  1.5
##  [4,] -0.3 -0.8
##  [5,] -0.8 -0.7
##  [6,]  1.2  1.0
##  [7,]  0.4  0.3
##  [8,]  0.5  1.2
##  [9,] -0.6 -1.4
## [10,]  0.3  0.0
```

Die empirische Korrelation zwischen X und Y ist:

```
cor(X[, 1], X[, 2])
```

```
## [1] 0.8846272
```

a)

Definiere eine Funktion, welche das 0.95% Konfidenzintervall aus i) numerisch ermittelt und berechne anschließend das Ergebnis. Hierzu werden die quadratischen Fehlerterme $(\sqrt{n}(\hat{\rho_n} - \rho)(1-\rho^2)^{-1} - u_{0.025})^2$ sowie $(\sqrt{n}(\hat{\rho_n} - \rho)(1-\rho^2)^{-1} - u_{0.975})^2$ numerisch mithilfe der Funktion *optimize* minimiert, um als Lösungen die Intervallgrenzen zu erhalten.

```
ki_i <- function(X, alpha = 0.05) {
  rho_hat <- cor(X[, 1], X[, 2])
  n <- nrow(X)
  first_bound <- optimize(function(x) {
    (sqrt(n) * (rho_hat - x) / (1 - x**2) - qnorm(alpha/2))**2
  }, lower = 0, upper = 1)$minimum
  second_bound <- optimize(function(x) {
    (sqrt(n) * (rho_hat - x) / (1 - x**2) - qnorm(1-alpha/2))**2
  }, lower = 0, upper = 1)$minimum
  return(c(lower = min(first_bound, second_bound), upper = max(first_bound, second_bound)))
}
attr(ki_i, 'name') <- 'ki_i' # used later for pretty printing
ki_i(X)
```

```
##     lower     upper
## 0.3339548 0.9477318
```

Verfahre für das Konfidenzintervall aus ii) analog:

```
ki_ii <- function(X, alpha = 0.05) {
  rho_hat <- cor(X[, 1], X[, 2])
  n <- nrow(X)
  first_bound <- optimize(function(x) {
    (sqrt(n-3) * (atanh(rho_hat) - atanh(x) - x/(2*(n-1))) - qnorm(alpha/2))**2
  }, lower = 0, upper = 1)$minimum
  second_bound <- optimize(function(x) {
    (sqrt(n-3) * (atanh(rho_hat) - atanh(x) - x/(2*(n-1))) - qnorm(1-alpha/2))**2
  }, lower = 0, upper = 1)$minimum
  return(c(lower = min(first_bound, second_bound), upper = max(first_bound, second_bound)))
}
attr(ki_ii, 'name') <- 'ki_ii'
ki_ii(X)
```

```
##     lower     upper
## 0.5546396 0.9694827
```

Berechne anschließend noch die Konfidenzintervalle nach Slutsky:

```
ki_i_slutsky <- function(X, alpha = 0.05) {
  rho_hat <- cor(X[, 1], X[, 2])
  n <- nrow(X)
  half_length <- qnorm(1-alpha/2) * (1 - rho_hat)**2 / sqrt(n)
  return(c(lower = rho_hat - half_length, upper = rho_hat + half_length))
}
attr(ki_i_slutsky, 'name') <- 'ki_i_slutsky'
ki_i_slutsky(X)
```

```
##     lower     upper
## 0.8763772 0.8928773
```

```
ki_ii_slutsky <- function(X, alpha = 0.05) {
  rho_hat <- cor(X[, 1], X[, 2])
  n <- nrow(X)
  return(c(lower = tanh(atanh(rho_hat) - rho_hat/(2*(n-1)) - qnorm(1-alpha/2)/sqrt(n-3)),
           upper = tanh(atanh(rho_hat) - rho_hat/(2*(n-1)) + qnorm(1-alpha/2)/sqrt(n-3))))
}
attr(ki_ii_slutsky, 'name') <- 'ki_ii_slutsky'
ki_ii_slutsky(X)
```

```
##     lower     upper
## 0.5418112 0.9697635
```

b)

```
simulation <- function(n, N = 10000, rho = 0.9) {
  for (ki_function in c(ki_i = ki_i, ki_i_slutsky, ki_ii, ki_ii_slutsky)) {
    results <- replicate(N, {
      sample <- rmvnorm(n, mean = c(0, 0), sigma = matrix(c(1, rho, rho, 1), nrow = 2))
      ki <- ki_function(sample)
      return(c(is_in = unname(ki['lower'] <= rho && ki['upper'] >= rho),
               length = unname(ki['upper'] - ki['lower'])))
    })
    in_percentage <- mean(results['is_in', ])
    average_length <- mean(results['length', ])
```

```
    print(paste0('Simulation results for ', attr(ki_function, 'name'), ':'))
    print(paste0('% contains true parameter: ', in_percentage))
    print(paste0('average length: ', average_length))
    cat('\n')
  }
}
simulation(10)
```

```
## [1] "Simulation results for ki_i:"
## [1] "% contains true parameter: 0.9114"
## [1] "average length: 0.589024649030024"
##
## [1] "Simulation results for ki_i_slutsky:"
## [1] "% contains true parameter: 0.0759"
## [1] "average length: 0.0236574048045373"
##
## [1] "Simulation results for ki_ii:"
## [1] "% contains true parameter: 0.9527"
## [1] "average length: 0.372500133094973"
##
## [1] "Simulation results for ki_ii_slutsky:"
## [1] "% contains true parameter: 0.9559"
## [1] "average length: 0.385187782292232"
```

c)

```
simulation(500)
```

```
## [1] "Simulation results for ki_i:"
## [1] "% contains true parameter: 0.9486"
## [1] "average length: 0.0341242812822005"
##
## [1] "Simulation results for ki_i_slutsky:"
## [1] "% contains true parameter: 0.0842"
## [1] "average length: 0.00177199685668584"
##
## [1] "Simulation results for ki_ii:"
## [1] "% contains true parameter: 0.948"
## [1] "average length: 0.0336242835355579"
##
## [1] "Simulation results for ki_ii_slutsky:"
## [1] "% contains true parameter: 0.947"
## [1] "average length: 0.0336152859906981"
```

```
simulation(1000)
```

```
## [1] "Simulation results for ki_i:"
## [1] "% contains true parameter: 0.9526"
## [1] "average length: 0.0238436490079966"
##
## [1] "Simulation results for ki_i_slutsky:"
## [1] "% contains true parameter: 0.08"
## [1] "average length: 0.00124701463633903"
##
```

```
## [1] "Simulation results for ki_ii:"
## [1] "% contains true parameter: 0.9505"
## [1] "average length: 0.0236735886916724"
##
## [1] "Simulation results for ki_ii_slutsky:"
## [1] "% contains true parameter: 0.9526"
## [1] "average length: 0.0236390923361666"
```