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
## 0.7498623 1.0193922
```

```
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_probability <- mean(results['is_in', ])
    average_length <- mean(results['length', ])
```

```
    print(paste0('Simulation results for ', attr(ki_function, 'name'), ':'))
    print(paste0('Contains true parameter: ', in_probability))
    print(paste0('Average length: ', average_length))
    cat('\n')
  }
}
simulation(10)
```

```
## [1] "Simulation results for ki_i:"
## [1] "Contains true parameter: 0.9081"
## [1] "Average length: 0.589939689611312"
##
## [1] "Simulation results for ki_i_slutsky:"
## [1] "Contains true parameter: 0.8326"
## [1] "Average length: 0.250714504597298"
##
## [1] "Simulation results for ki_ii:"
## [1] "Contains true parameter: 0.9542"
## [1] "Average length: 0.373973887308667"
##
## [1] "Simulation results for ki_ii_slutsky:"
## [1] "Contains true parameter: 0.9557"
## [1] "Average length: 0.387248157862254"
```

Man erkennt, dass die Verfahren *ki_i* und *ki_i_slutsky* das Konfidenzniveau bei einer Stichprobengröße von $n = 10$ nicht einhalten. Die Intervalle von *ki_i_slutsky* sind überdies weniger als halb so lang wie die von *ki_i* und enthalten den wahren Parameter nur in knapp 83% der Fällen.

Wendet man allerdings Fishers Z-Transformation an, wie es bei den Verfahren *ki_ii* und *ki_ii_slutsky* geschehen ist, so erhält man selbst bei einer Stichprobengröße von nur 10 Elementen Konfidenzintervalle, welche das geforderte Niveau einhalten. Die Länge dieser Intervalle ist überdies deutlich kürzer, als die ohne Transformation. Es lässt sich daher vermuten, dass bei kleinen Stichproben die Intervalle, welche mithilfe von Fishers Z-Transformation bestimmt worden sind, vorzuziehen sind. Ob man die Intervallgrenzen nun numerisch oder mithilfe von Slutsky bestimmt, scheint unter Betrachtung der Simulationsergebnisse hier keinen Unterschied zu machen. Berücksichtigt man allerdings, dass die numerische Berechnung der Grenzen etwas rechenaufwändiger ist, sind die Intervalle der Funktion *ki_ii_slutsky* unter diesem Aspekt zu bevorzugen.

c)

```
simulation(500)
```

```
## [1] "Simulation results for ki_i:"
## [1] "Contains true parameter: 0.9496"
## [1] "Average length: 0.034097489331776"
##
## [1] "Simulation results for ki_i_slutsky:"
## [1] "Contains true parameter: 0.9482"
## [1] "Average length: 0.0333615206945587"
##
## [1] "Simulation results for ki_ii:"
## [1] "Contains true parameter: 0.9502"
## [1] "Average length: 0.0336261418713686"
##
```

```
## [1] "Simulation results for ki_ii_slutsky:"
## [1] "Contains true parameter: 0.9449"
## [1] "Average length: 0.0335976475511064"
```

```
simulation(1000)
```

```
## [1] "Simulation results for ki_i:"
## [1] "Contains true parameter: 0.9507"
## [1] "Average length: 0.0238432212090178"
##
## [1] "Simulation results for ki_i_slutsky:"
## [1] "Contains true parameter: 0.9481"
## [1] "Average length: 0.0235794101641961"
##
## [1] "Simulation results for ki_ii:"
## [1] "Contains true parameter: 0.9482"
## [1] "Average length: 0.0236496448009481"
##
## [1] "Simulation results for ki_ii_slutsky:"
## [1] "Contains true parameter: 0.9489"
## [1] "Average length: 0.0236633692113696"
```

Für größere Stichprobenumfänge von $n = 500$ oder $n = 1000$ Elementen scheinen die Unterschiede zwischen den Verfahren zu verschwinden. Betrachtet man die Ergebnisse der Simulation, so scheinen alle Intervalle bei in etwa gleicher Intervallbreite das geforderte Niveau einzuhalten. Diese Beobachtung lässt sich vermutlich dadurch erklären, dass die Approximation durch die Normalverteilung für große n besser funktioniert, als bei kleineren Stichproben.