DATA 501: Assignment 1 solution

Dr David Huijser Cotton Building 542
David.Huijser@vuw.ac.nz

Shapiro-Wilk Test

For this assignment we are asked to calculated the W test statistics defined in (Shapiro and Wilk 1965) as

$$W = \frac{[\sum_i a_i y_{(i)}]^2}{\sum_i (y_i - \overline{y})^2} \tag{1}$$

where $y' = (y_1, ..., y_n)$ is a (row) vector[1] of random observations, and $a' = (a_1, ..., a_n)$ is (row) vector with the co-efficients which we will be explained later. In the application of this test we distinguish between elements in the sample x_i and elements in the ordered sample $x_(i)$ by using round brackets on the index i, where the index (i) indicates the i^{th} order of the statistic.

The rational behind the Shapiro-Wilk test is that if we have a vector of observed random values from an unknown distribution $y' = (y_1, ...y_n)$ we can create the order statistics $y_{(1)} \leq y_{(2)} \leq .... \leq y_{(n)}$. As a comparison we can draw a set of random values $x_1, x_2, ...x_n$ from a normal distribution $N(0, 1)$, and take their ordered statistics $x_{(1)} \leq x_{(2)} \leq ...x_{(n)}$. Then we can investigate the correlation between the ordered observed random values $y_{(1)}, y_{(2)}, ..., y_{(n)}$ with the expected values of the ordered statistics $Ex_{(1)}, Ex_{(2)}...Ex_{(n)}$. If the order statistics of $y_{(i)}$ are well correlated with the expected standard normal order statistics we should obtain a correlation close to 1.

Calculation expectations of ordered statistics is not a trivial problem.
The expectation of the of i^{th} order statistics is given by (Royston 1982a)

$$E(x_{(i)}) = \int_{-\infty}^{\infty} x \cdot f_{X_{(i)}}(x) dx \tag{2}$$

where $f_{X_{(i)}}$ the probability density function of the i-th order statistic $f_{X_{(i)}}$ (Casella and Berger 2002).

[1] we follow the convention to use the prime symbol to indicate a transpose. Therefore we can distinguish row vector y' from column vector y.

The expectations of $x_{(j)}$ depends on the length of the vector n through the probability density function of the i-th order statistic $f_{X_{(i)}}$:

$$f_{x_{(i)}} = \frac{n!}{(i-1)!(n-i)!} f_x(x)[F_x(x)]^{i-1}[1-F_x(x)]^{n-i} \tag{3}$$

where $F_x(x)$ is the cumulative distribution function and $f_x(x)$ is the probability density function. The integral in equation 2 can in general not be done exactly, this is done by a numerical integration or simulation.

Several papers have been written regarding different techniques, algorithms, approximations and corrections that could be implemented to improve accuracy and speed(Royston 1982b). The most convenient approximation is the fact that for a normal distribution, the expected

values of the order statistics can be approximated using the quantile function of the normal distribution with a correction term α.

$$E(x_{(i)}) \approx \Phi^{-1}\left(\frac{i-\alpha}{n-2\alpha+1}\right) \tag{4}$$

where Φ^{-1} is the quantile function (inverse CDF) of the standard normal distribution, i is the rank of the order statistics, n is the sample size and α is the correction factor. Empirically the correction factor has been set to $\alpha = 0.375$, therefore the expectation of the of i^{th} order statistics Blom (1960)

$$E(x_{(i)}) \approx \Phi^{-1}\left(\frac{i-0.375}{n+0.25}\right) \tag{5}$$

The reason why this approximation works is that the order statistics of a sample from a continuous distribution are approximately evenly spaced when transformed by the CDF of that distribution.

In the literature (Shapiro and Wilk 1965), the expected values of the order statistics from a i.i.d random variable sample from a standard normal distribution $Ex_{(1)}, Ex_{(2)}...Ex_{(n)}$ is referred to as vector $m' = (m_1, m_2..m_n)$. The Shapiro-Wilk test also takes into account co-variances between different using the covariance matrix V, because $x_{(j)}$ tends to be strongly correlated with $x_{(j-1)}$).
Using m‘ and V we are able to define coefficients $a' = (a_1, ...a_n)$ referred to by Royston (1982b) as the normalized “best linear unbiased coefficients. These can calculated, although there are also tables available for this. The vector of co-efficient a' is calculated using the following equations:

$$a' = (a_1, a_2, ...a_n) = \frac{m'V^{-1}}{C} \tag{6}$$

where C is the vector norm:

$$C = \sqrt{m'V^{-1}V^{-1}m} \tag{7}$$

The problem with this approach is that V is not guaranteed to be singular. In matter of factor V is often singular and therefore the inverse V^{-1} can not be found. One work around is to use the pseudo inverse instead.

In practice the calculation of the Shapiro_Wilk statistic can be obtained in four steps:

1. Sort the data
2. Calculate the expected values of the order statistics of a sample drawn from a $N(0, 1)$.
3. Compute the coefficients a
   a. using the pseudo-inverse of V to calculate a
   b. using the approximations using m and g to calculate a
4. Calculate the Shapiro-Wilk statistic.

Here are two versions of possible implementatins.

```
my_shapiro_wilk_LA <- function(x) {

  # step 1 (sort data)
  sorted_x <- sort(x)
  n <-length(x)

  # step 2 (obtain ordered sample from N(0,1) distribution with a correction)
  m <-  matrix(qnorm((1:n - 0.375) / (n + 0.25)), nrow = n, byrow = TRUE)

  # step 3 Find co-efficients using LA
  V <- (1 / (n - 1)) * (m %*% t(m))

  # Use Pseudo-inverse instead of inverse to avoid SVD issues
  invV <- pseudoinverse(V)
  C <- sqrt(t(m)%*%invV%*%invV%*%m)[1]
  a <- t(m)%*%invV/C
  W <- sum(a*sorted_x)**2/(sum( (x-mean(x))**2))
  return(W)
}
```

```
my_shapiro_wilk <- function(x) {

  # find length of array
  n <-length(x)

  # step 1 (order data)
  ordered <- sort(x)

  # step 2 (obtain ordered sample from N(0,1) distribution with a correction)
  m <- qnorm((1:n - 0.375) / (n + 0.25))

  # step 3 - use approximation to obtain coefficients vector a (except a[1] and a[n])
```

```
  a <- 2*m
  a[1] <- 0
  a[n] <- 0

  # Using Stirling approximation from Royston 1982 to calculate g
  if (n < 20){
    t <- n-1
    g <- ((6*t+7)/(6*t+13))*sqrt( (exp(1)/(t+2))*((t+1)/(t+2))**(t-2) )
  }else
  {
    t <- n
    g <- ((6*t+7)/(6*t+13))*sqrt( (exp(1)/(t+2))*((t+1)/(t+2))**(t-2) )
  }

  # There are two common was to calculate `g`
  # Use Gamma-function to calculate g
  if (n < 20){
    g <- gamma(0.5*n)/(sqrt(2)*gamma(0.5*n+0.5))
  }else
  {
    g <- gamma(0.5*n+0.5)/(sqrt(2)*gamma(0.5*n+1))
  }

  # Use g to find a[1] and a[n]
  a_start <- sqrt((g/(1-2*g))*sum(a**2))
  a[1] <--a_start
  a[n] <-a_start

  # normalize a
  a <- a / sqrt(sum(a^2))

  # Step 4 Calculate W
  X_bar <- mean(ordered)
  numerator <- (sum(a * ordered))^2
  numerator
  denominator <- sum((ordered - X_bar)^2)
  return(numerator/denominator)
}
```

The paper (Shapiro and Wilk 1965) has a few examples to serve as test-cases from the paper. We will compare these result with the result in the paper as well as results of the sapiro-wilk test using the standard R package.

```
library(corpcor) # needed for the pseudo-inverse

# Example 1: (Sapiro-Wilk paper, page 16, W=0.9530)
x <- c(6,1,-4,8,-2,5,0)
print("W=0.9530 (according to paper)")
```

```
## [1] "W=0.9530 (according to paper)"
```

```
W <- as.numeric(shapiro.test(x)$statistic)
print(paste("W (test)",W))
```

```
## [1] "W (test) 0.953475858466522"
```

```
W <- my_shapiro_wilk(x)
print(paste("W (approx)",W))
```

```
## [1] "W (approx) 0.950707547372675"
```

```
W <- my_shapiro_wilk_LA(x)
print(paste("W (LA)",W))
```

```
## [1] "W (LA) 0.969206617042761"
```

```
# Example 2: (Sapiro-Wilk paper, page 16, W=0.79)
x <- c(148,154,158,160,161,162,166,170,182,195,236)
print("W=0.79 (according to paper)")
```

```
## [1] "W=0.79 (according to paper)"
```

```
W <- as.numeric(shapiro.test(x)$statistic)
print(paste("W (test)",W))
```

```
## [1] "W (test) 0.788814694835387"
```

```
W <- my_shapiro_wilk(x)
print(paste("W (approx)",W))
```

```
## [1] "W (approx) 0.785865925532141"
```

```
W <- my_shapiro_wilk_LA(x)
print(paste("W (LA)",W))
```

```
## [1] "W (LA) 0.771381939646386"
```

```
# Example 3: (Sapiro-Wilk paper, page 16, W=0.943)
x <- c(303,338,406,457,461,469,474,489,515,583)
print("W=0.943 (according to paper)")
```

```
## [1] "W=0.943 (according to paper)"
```

```
W <- as.numeric(shapiro.test(x)$statistic)
print(paste("W (test)",W))
```

```
## [1] "W (test) 0.942675463966398"
```

```
W <- my_shapiro_wilk(x)
print(paste("W (approx)",W))
```

```
## [1] "W (approx) 0.941641166119972"
```

```
W <- my_shapiro_wilk_LA(x)
print(paste("W (LA)",W))
```

```
## [1] "W (LA) 0.935790560435273"
```

In the final R script we will put these two versions together. The script has one required input which is the data `x` and one optional input `approx` which is a logical (or boolean) to indicate which method should be used to calculate the coefficient vector a. The default of the optional value is `TRUE` which will use the approximations described in the paper, and the switch is `FALSE` the script will use the linear-algebra approach, and calculate the co-efficient vector `a` using the vector `m` and co-variance matrix `V`.

```
my_shapiro_wilk_test <- function(x,approx=TRUE) {

  # Input test
  if (any(is.na(x)))
      stop(paste("There seems to be at least one NA value in the data."))

  if (any(is.infinite(x)))
     stop(paste("There seems to be at least one infinity value in the data."))

  #if (typeof(x)=="double")) print("There seems to be at least one infinity value in t

  if ( (NROW(x)!=1) & (NCOL(x)!=1))
      stop(paste("There seems to be a problem with the dimensions of the data."))

  if (length(x) < 3)
      stop(paste("The data does not have enough elements."))

  if (length(x) >2000)
      stop(paste("The data has too many elements."))

  # Choose which type of calculation is preferred.
  if (approx)
```

```
  {
    W= my_shapiro_wilk(x)
  }else
  {
    W= my_shapiro_wilk_LA(x)
  }
   return(W)
}
```

Now we can create test-script with the following lines.

```
library(testthat)
```

```
## Warning: package 'testthat' was built under R version 4.3.2
```

```
context("Testing Shapiro-Wilk test")

## Invalid inputs testing
test_that("Our script gives helpful errors for invalid inputs", {

  expect_error(my_shapiro_wilk_test(c(303,NA,406,457,461,469,474,489,515,583),TRUE),"Th

  expect_error(my_shapiro_wilk_test(c(303,Inf,406,457,461,469,474,489,515,583),TRUE),"T

  expect_error(my_shapiro_wilk_test(matrix(c(303,338,406,457,461,469,474,489,515,583),

  expect_error(my_shapiro_wilk_test(matrix(c(303,338,406,457,461,469,474,489,515,583),

  expect_error(my_shapiro_wilk_test(c(303,338),TRUE),"The data does not have enough ele

  expect_error(my_shapiro_wilk_test(rnorm(2001),TRUE),"The data has too many elements."

  })
```

```
## Test passed
```

```
  ## Valid inputs testing
  test_that("Test if the script work with or without the approx argument", {
  expect_silent(my_shapiro_wilk_test(c(303,338,406,457,461,469,474,489,515,583)))
  expect_silent(my_shapiro_wilk_test(c(303,338,406,457,461,469,474,489,515,583),TRUE))
  expect_silent(my_shapiro_wilk_test(c(303,338,406,457,461,469,474,489,515,583),FALSE))
  })
```

```
## Test passed
```

```
  ## Functionality testing (not much to do here.)
```

```
test_that("Test if script finds approximate the right values", {

# Example 1: (Sapiro-Wilk paper, page 16, W=0.9530)
expect_equal(my_shapiro_wilk_test(c(6,1,-4,8,-2,5,0)),0.953,tolerance=0.02)
expect_equal(my_shapiro_wilk_test(c(6,1,-4,8,-2,5,0),FALSE),0.953,tolerance=0.02)

# Example 2: (Sapiro-Wilk paper, page 16, W=0.79)
expect_equal(my_shapiro_wilk_test(c(148,154,158,160,161,162,166,170,182,195,236)),0.7
expect_equal(my_shapiro_wilk_test(c(148,154,158,160,161,162,166,170,182,195,236),FALS

# Example 3: (Sapiro-Wilk paper, page 16, W=0.943)
expect_equal(my_shapiro_wilk_test(c(303,338,406,457,461,469,474,489,515,583)),0.943,t
expect_equal(my_shapiro_wilk_test(c(303,338,406,457,461,469,474,489,515,583),FALSE),0

})
```

```
## Test passed
```

The link to all code and scripts can be found here: https://github.com/DavidHuijser/DATA501_SP

The final part is self evaluation. The values obtained from the script can deviate from the examples in the paper as well as the `R` package. To delve into the origin of these deviation is beyond the aim of this assignment.

This script does not have all possible tests available. - For example if you would use this could with a string as argument at it you would return an inaccurate/wrong error. - For example the calculation of `g` can produce NAs (`x <- c(2, 3, 4, 5)`, `x<- c(100, 100, -100, 100)`) - For example the calculation can return errors or NA-error (`x <- c(0, 0, 0, 0)`) - For example the calculation can lead to memory issues (`x <- rbnorm(2000)*100000`)

References

Blom, N. L. J. 1960. *The Incorporated Statistician* 10 (1): 53–55.

Casella, G, and R. L Berger. 2002. *Statistical Inference*.

Royston, J. P. 1982a. "Algorithm AS 177: Expected Normal Order Statistics (Exact and Approximate)." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31 (2): 161–65.

———. 1982b. "An Extension of Shapiro and Wilk's w Test for Normality to Large Samples." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31 (2): 115–24.

Shapiro, S.S, and M. B. Wilk. 1965. "An Analysis of Variance Test for Normality (Complete Samples)." *Biometrika* 52 (3/4): 591–611.