STAT-6494 Advanced Statistical Computing with R

Homework 4

Wenjie Wang

10 March 2016

1 Exercise: Spearman's rho for t-copula

Consider a t-copula with two parameters: degrees of freedom df and dispersion r. For any given df, Spearman's rho is a function of ρ, but this function has no closed-form. One way to approximate this function is through sample Spearman's ρ on a grid of r values in $[-1, 1]$. A spline approximation can be fitted to the pairs of $\hat{\rho}$ and r (Kojadinovic and Yan 2010, Appendix).

1. Create a grid of $(\hat{\rho}, r)$ pairs for a grid of r values with sample size $n = 1000$, with df 5.
2. Find a spline function that approximates the true function $\rho(r)$.
3. Repeat with df 10, and overlay the two functions.
4. It is known that $\rho(-1) = -1$, $\rho(0) = 0$, and $\rho(1) = 1$. Can you incorporate these in the fitting?

1.1 Part 1.

With the help of **R** package **copula**, we may easily get random numbers from bivariate t-copula for given degree of freedom, df, and dispersion r. Then the sample Spearman correlation coefficient $\hat{\rho}$ is computed for each sample. Function `oneRho` and `rRho` are implemented to get one ρ at given r and the mean ρ of replicates at each grid point of r values, respectively.

```
library(copula)
oneRho <- function (r, df, nSample) {
    if (r == 0L) return(0)
    obj <- tCopula(r, dim = 2, dispstr = "ex", df = df, df.fixed = TRUE)
    sampMat <- rCopula(nSample, obj)
    cor(sampMat[, 1], sampMat[, 2], method = "spearman")
}
rRho <- function (rVec, df = 5, nSample = 1000) {
    rhoVec <- sapply(rVec, oneRho, df = df, nSample = nSample)
    data.frame(r = rVec, rho = rhoVec)
}
```

In part 1, the degree of freedom is fixed to be 5. The sample size is fixed as 1,000. We take a grid of r values in $[-1, 1]$ by step 0.01.

```
set.seed(1216)
rVec <- seq(- 1, 1, by = 0.01)
resMat1 <- rRho(rVec, df = 5)
```

We may plot the mean ρ over the grid of r as follows:

```
library(ggplot2)
resDat1 <- data.frame(resMat1, df = 5)
(p1 <- ggplot(resDat1, aes(x = r, y = rho)) +
     geom_point(color = "royalblue", size = 0.8, alpha = 0.8) +
     geom_abline(slope = 1, color = "black", linetype = 3) +
     ylab("Sample Spearman's rho") + theme_bw())
```

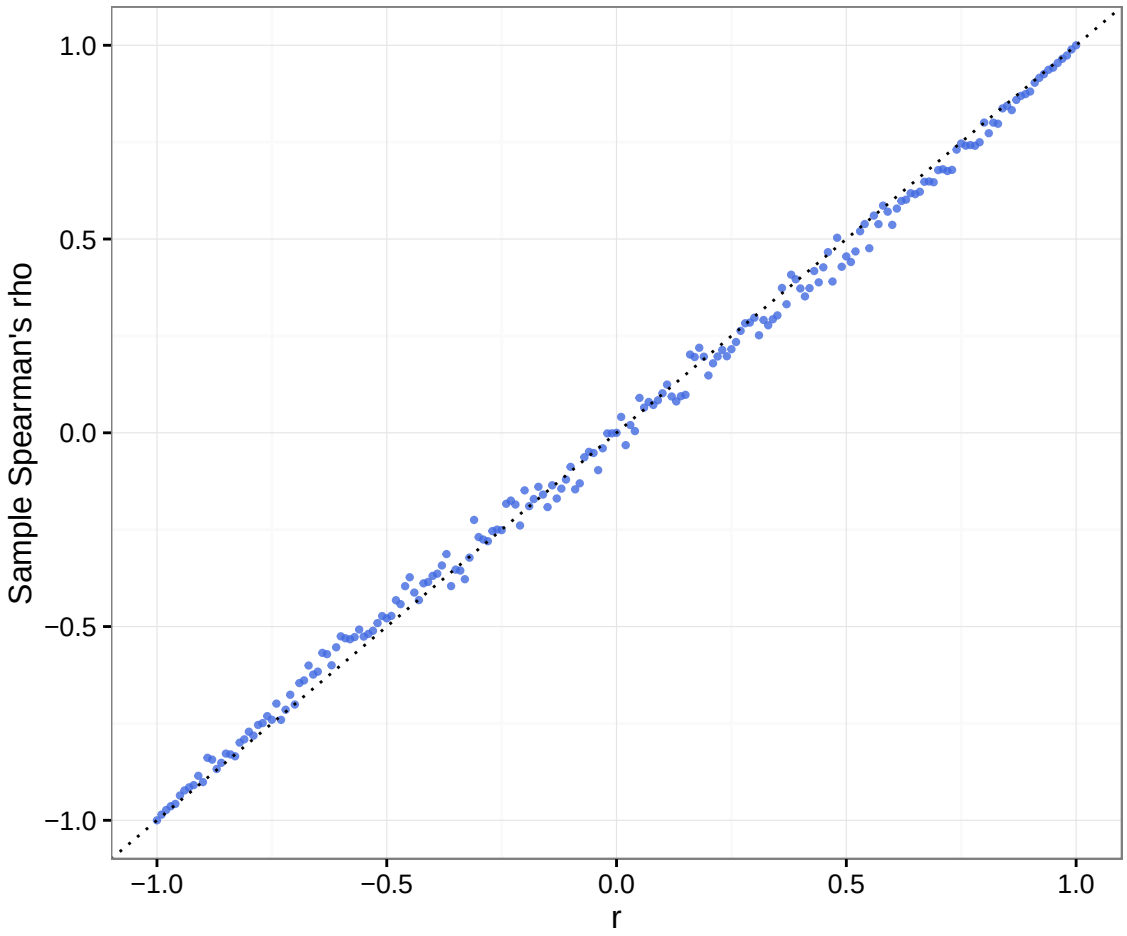

Figure 1: Sample Spearman's rho (blue solid curve) and reference line with slope 1 and intercept 0 (black dotted line).

1.2 Part 2.

From Figure 1, We may observe that the mean Spearman's rho curve looks quite linear. Therefore, penalized smoothing cubic spline is probably a good choice since it penalizes the second derivatives. The tuning parameter is left to be selected by GCV.

```
(smoothFit1 <- with(resDat1, smooth.spline(x = r, y = rho)))
```

```
Call:
smooth.spline(x = r, y = rho)
```

```
Smoothing Parameter  spar= 0.8620881  lambda= 0.002407559 (13 iterations)
Equivalent Degrees of Freedom (Df): 7.031296
Penalized Criterion: 0.1146031
GCV: 0.0006122502
```

```r
plot(rho ~ r, data = resDat1, col = "gray",
     xlab = expression(r), ylab = expression(rho))
lines(smoothFit1, col = "blue")
abline(a = 0, b = 1, col = 1, lty = 3)
legend("bottomright", col = c("blue", "gray", "black"),
       legend = c("Fitted spline", "Mean Spearman rho",
                  expression("Reference line"~y==x)),
       lty = c(1, NA, 3), pch = c(NA, 1, NA), cex = 0.8)
```

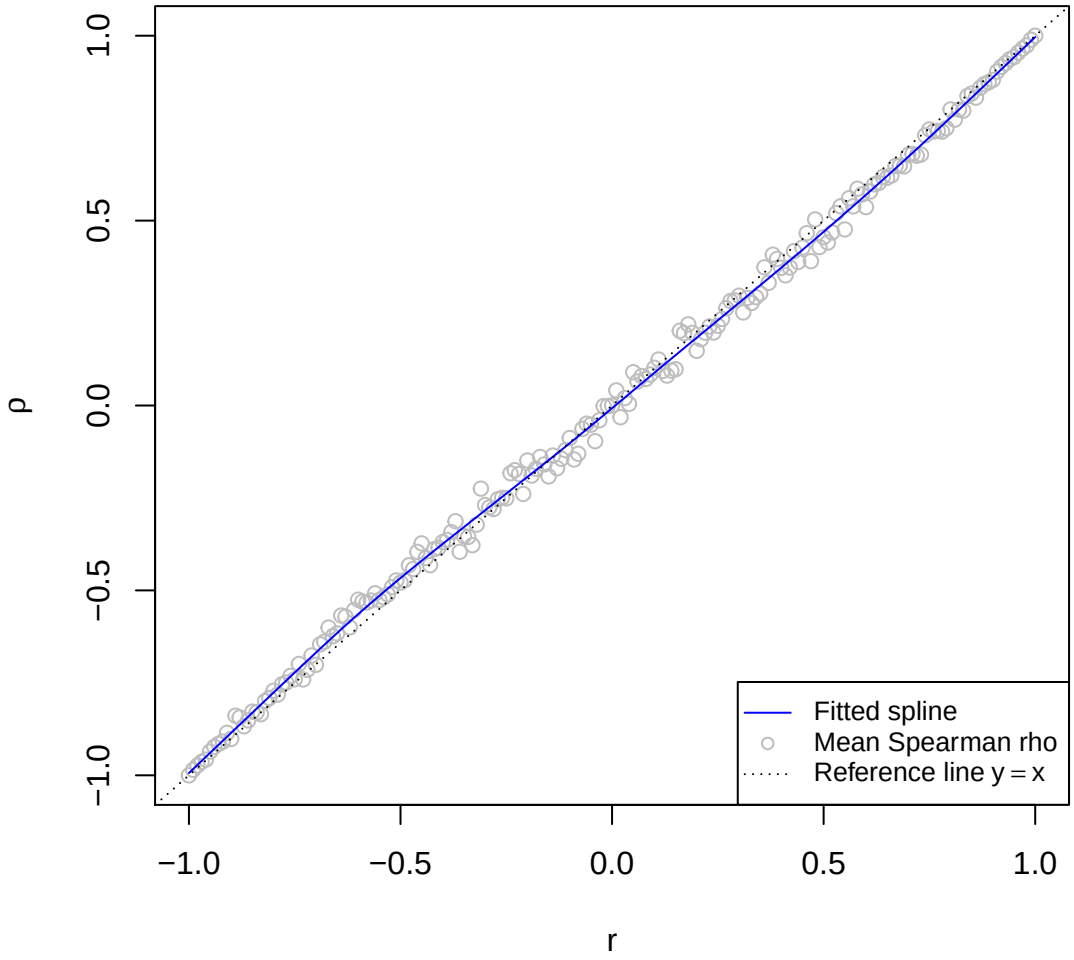

Figure 2: Fitted Spearman's rho curve by penalized spline.

The fitted spline function is plotted in Figure 2, from which we may find that the penalized smoothing cubic spline fits the mean Spearman's rho quite well.

1.3 Part 3.

When df is specified as 10, the similar results follow.

```r
set.seed(1216)
resMat2 <- rRho(rVec, df = 10)
```

```r
resDat2 <- data.frame(resMat2, df = 10)
(p2 <- ggplot(resDat2, aes(x = r, y = rho)) +
     geom_point(color = "red", alpha = 0.8, cex = 0.8) +
     geom_abline(slope = 1, color = "black", linetype = 3) +
     ylab("Sample Spearman's rho") + theme_bw())
```

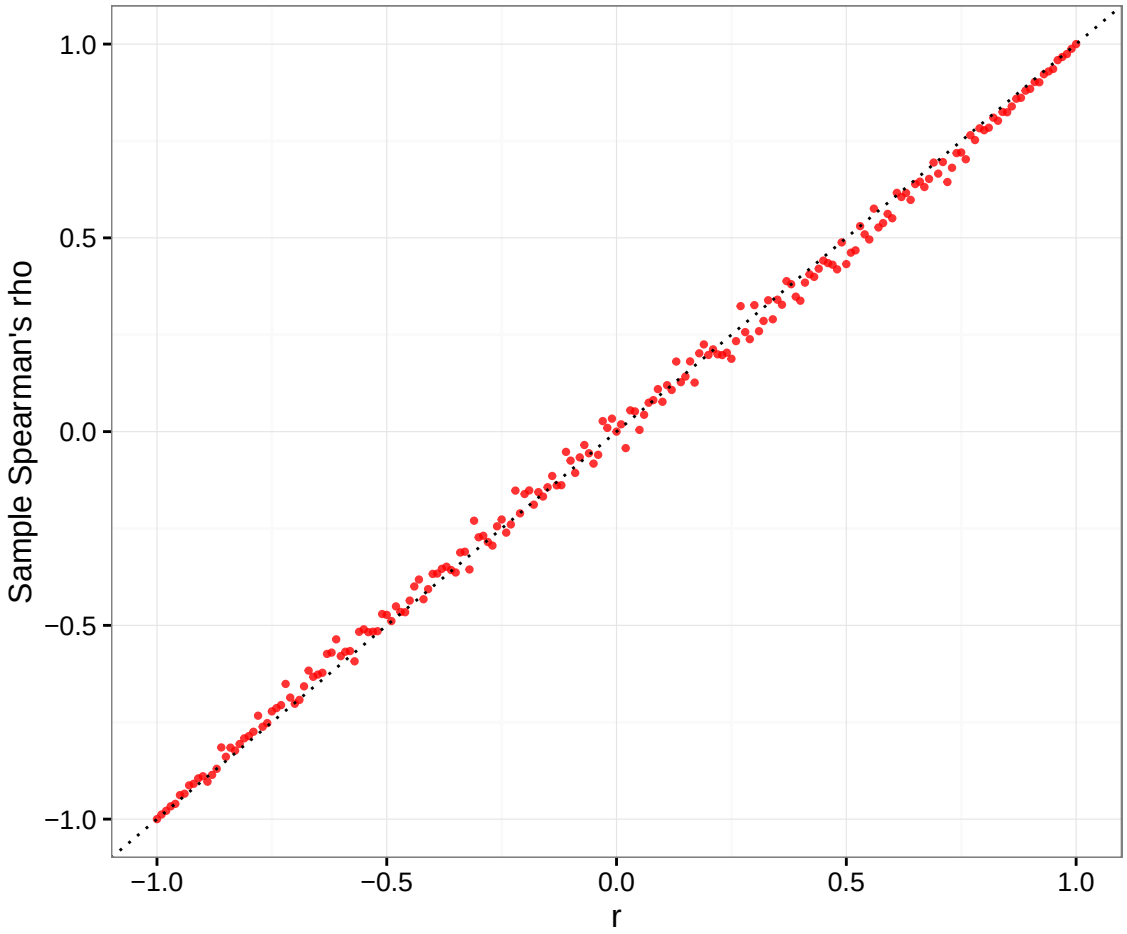

Figure 3: Sample Spearman's rho (red solid curve) and reference line with slope 1 and intercept 0 (black dotted line).

```r
(smoothFit2 <- with(resDat2, smooth.spline(x = r, y = rho)))
```

```
Call:
smooth.spline(x = r, y = rho)

Smoothing Parameter  spar= 0.9026715  lambda= 0.004729128 (10 iterations)
Equivalent Degrees of Freedom (Df): 6.094818
Penalized Criterion: 0.1041976
GCV: 0.000551324
```

```
plot(rho ~ r, data = resDat2, col = "gray",
     xlab = expression(r), ylab = expression(rho))
lines(smoothFit2, col = "red")
abline(a = 0, b = 1, col = 1, lty = 3)
legend("bottomright", col = c("red", "gray", "black"),
       legend = c("Fitted Spearman's rho", "Sample Spearman rho",
                  expression("Reference line"~y==x)),
       lty = c(1, NA, 3), pch = c(NA, 1, NA), cex = 0.8)
```

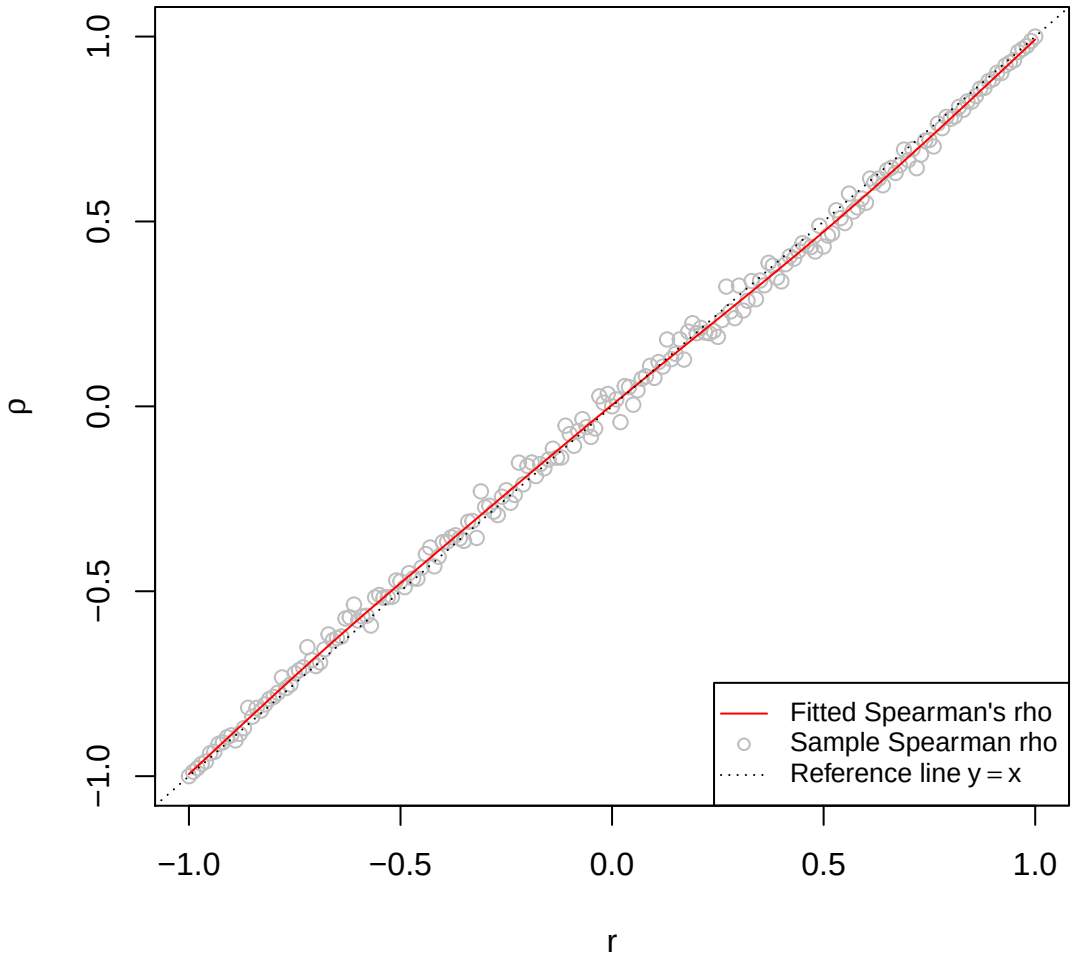

Figure 4: Fitted Spearman's rho curve by penalized spline.

We may further overlay two fitted spline functions as follows:

```
ggDat <- data.frame(r = c(smoothFit1$x, smoothFit2$x),
                    rho = c(smoothFit1$y, smoothFit2$y),
                    df = gl(2, length(rVec), labels = c("5", "10")))
(p3 <- ggplot(ggDat, aes(x = r, y = rho, color = df, linetype = df)) +
     geom_line() + geom_abline(slope = 1, color = 1, linetype = 3) +
     ylab("Fitted Spearman's rho") + theme_bw())
```

The fitted spline function for df= 5 has slightly larger deviation to the reference line $y = x$ than df= 10.

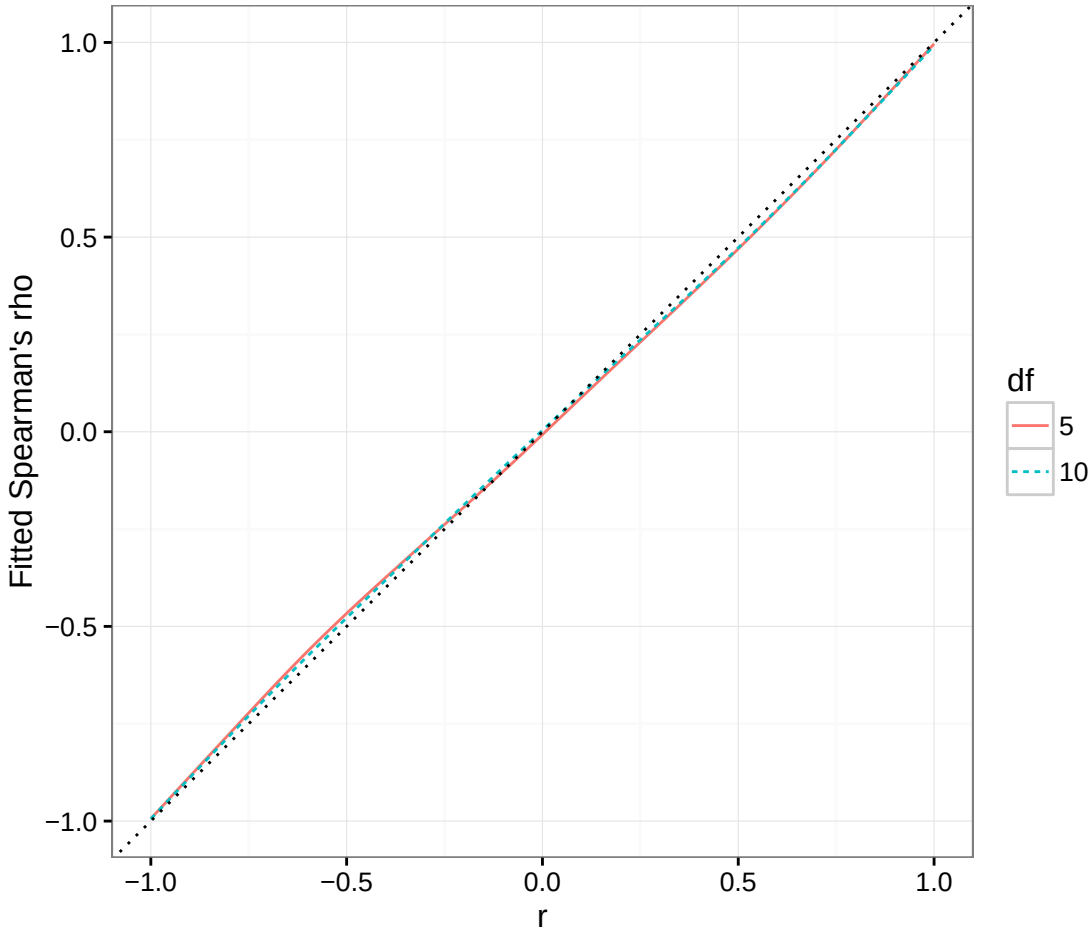

Figure 5: Overlaid spline functions fitted for t-copula with df 5 and 10.

1.4 Part 4.

For simplicity, we fix df= 5 here. It is known that the true function of Spearman's rho is symmetric about zero. Therefore, we may consider fit the smoothing spline ranging from only 0 to 1. The fits over -1 to 0 can be obtained by symmetry.

1.4.1 Weighted smoothing spline

Notice that if r is specified to be 1, the Spearman's rho is necessarily 1. The function `rRho` produces Spearman's rho as 0, if r is 0. Therefore, we consider weighted smoothing spline to constrain fitted curve to be close to $(0, 0)$ and $(1, 1)$. The $(-1, -1)$ will be approached by symmetry. The weight on data points $(0, 0)$ and $(1, 1)$ is set to be $1,000$, which is much larger than the weight on other data points.

```
## generate mirror part from fits over 0 and 1
mirFun <- function (fitObj) {
    if (class(fitObj) == "smooth.spline")
        mirDat <- dat <- with(fitObj, cbind(x, y))
    else if (class(fitObj) == "cobs")
        mirDat <- dat <- with(fitObj, cbind(x, fitted))
    else stop("not applicable")
    mirDat <- - mirDat
    outMat <- rbind(tail(mirDat, -1) , dat)
    out <- outMat[order(outMat[, "x"]), ]
    row.names(out) <- NULL
    colnames(out) <- c("r", "rho")
    data.frame(out)
}
rVec <- seq(0, 1, by = 0.01)
mirDat <- rRho(rVec)
(conFit <- with(mirDat, smooth.spline(x = r, y = rho,
                                      w = c(1e3, rep(1, length(r) - 2), 1e3))))
```

```
Call:
smooth.spline(x = r, y = rho, w = c(1000, rep(1, length(r) -
    2), 1000))

Smoothing Parameter  spar= 0.9553772  lambda= 0.001701693 (9 iterations)
Equivalent Degrees of Freedom (Df): 4.073067
Penalized Criterion: 0.003855976
GCV: 4.145403e-05
```

```
resDat <- mirFun(conFit)
(p4 <- ggplot(resDat, aes(x = r, y = rho)) + geom_line(color = "royalblue") +
     geom_abline(slope = 1, color = 1, linetype = 3, alpha = 0.8) +
     ylab("Fitted Spearman's rho") + xlab("r") + theme_bw())
```

If we check the fitted Spearman's rho at 0 or 1 (-1), we will find that the difference between true value and fitted value is negligible.

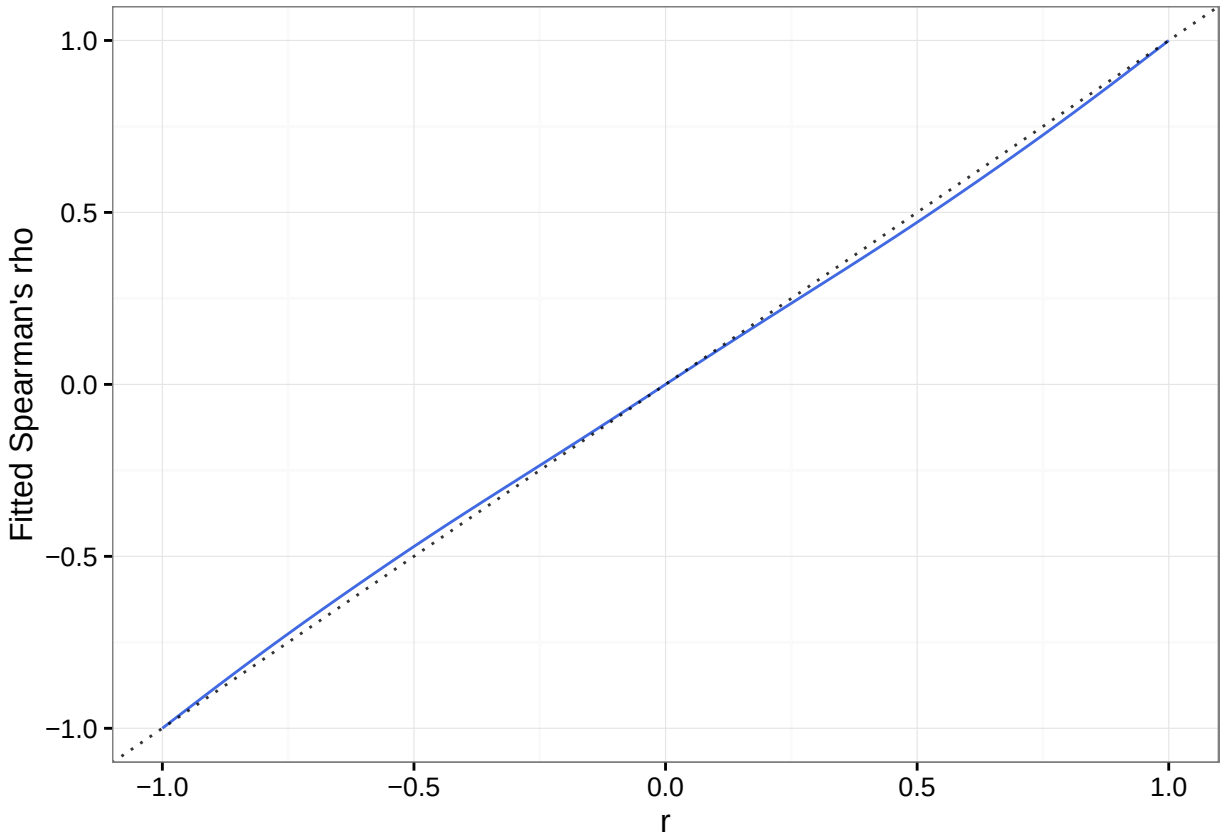

Figure 6: Fitted Spearman's rho by weighted smoothing spline and symmetry.

1.4.2 Qualitatively constrained smoothing splines

Another approach that we may consider is qualitatively constrained smoothing splines (Ng and Maechler 2007). The implementation is available in package **cobs**. The smoothing parameter is set to be chosen by SIC.

```
library(cobs)
con <- matrix(c(0, 0, 0, 0, 1, 1), ncol = 3, byrow = TRUE)
cobsFit <- with(mirDat, cobs(x = r, y = rho, constraint = "convex",
                             pointwise = con, lambda = - 1))
```

```
 Searching for optimal lambda. This may take a while.
   While you are waiting, here is something you can consider
   to speed up the process:
       (a) Use a smaller number of knots;
       (b) Set lambda==0 to exclude the penalty term;
       (c) Use a coarser grid by reducing the argument
       'lambda.length' from the default value of 25.

   The algorithm has converged.  You might
   plot() the returned object (which plots 'sic' against 'lambda')
   to see if you have found the global minimum of the information criterion
   so that you can determine if you need to adjust any or all of
   'lambda.lo', 'lambda.hi' and 'lambda.length' and refit the model.
```

```
resDat <- mirFun(cobsFit)
(p5 <- ggplot(resDat, aes(x = r, y = rho)) + geom_line(color = "red") +
     geom_abline(slope = 1, color = 1, linetype = 3, alpha = 0.8) +
     ylab("Fitted Spearman's rho") + xlab("r") + theme_bw())
```

If we have a close check on the fitted Spearman's rho on 0 or 1 (-1), we may find that the fitted values are not exactly constrained to be the true values. However, the difference is negligible.

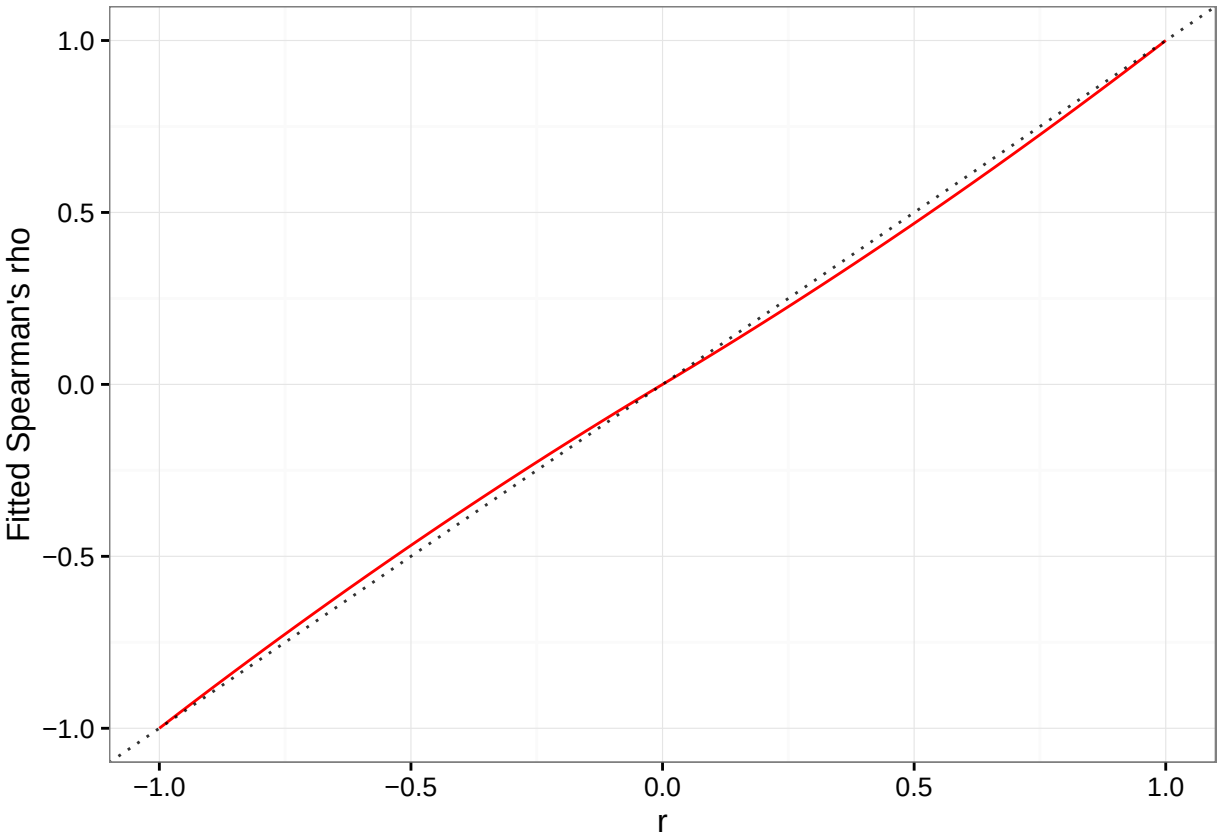

Figure 7: Fitted Spearman's rho by qualitatively constrained smoothing spline and symmetry.

Reference

Kojadinovic, Ivan, and Jun Yan. 2010. "Comparison of Three Semiparametric Methods for Estimating Dependence Parameters in Copula Models." *Insurance: Mathematics and Economics* 47 (1). Elsevier: 52–63.

Ng, Pin, and Martin Maechler. 2007. "A Fast and Efficient Implementation of Qualitatively Constrained Quantile Smoothing Splines." *Statistical Modelling* 7 (4). SAGE Publications: 315–28.