The ROC Area Under the Curve or Mean Ridit as an Effect-Size: Supplement

Michael Smithson

The Supplement contains an elaboration of several technical points in the paper, a brief overview of currently available software resources for ridits, and details about the examples. The Supplement folder contains the data-files used in the paper.

As mentioned in the paper, the Mann-Whitney U statistic is related to the mean ridit:

$$R = \frac{U}{n_r n_a} \tag{S1}$$

where n_r is the sample size of the reference group and n_a is the sample size of the alternative group.
The variance of the mean ridit therefore is

$$\mathrm{var}(R) = \mathrm{var}(U)/(n_r n_a)^2 . \tag{S2}$$

As mentioned in the paper, R is invariant under multiplication of either the referent or alternative sample frequencies by a constant. However, its variance is not. In fact, even when each sample size is multiplied by scalars that hold the total sample size $n_r + n_a$ constant the variance still will change. Thus, while R is base-rate insensitive, its variance is not.

Jansen (1984) reviews the relationships between R and measures of ordinal association. Most of this review focuses on the case where X is a binary variable crossed with Y, which is a J-category ordinal variable, and one category of X is treated as the reference group. Denote by R^* the mean ridit when the other X category is treated as the reference group, i.e., $R^* = 1 - R$. Jansen reports that Somers' d_{yx} is the difference between these complementary ridits:

$$d_{yx} = R - R^* = 2R - 1. \tag{S3}$$

Relations with Kendall's tau-c and the Goodman-Kruskal gamma are:

$$\tau_c = (R - R^*)\frac{4n_r n_a}{(n_r + n_a)^2} \tag{S4}$$

$$\gamma = \frac{R - R^*}{1 - T_x / n_r n_a}, \tag{S5}$$

where T_x is the number of observations tied on X only.

It also is worth noting that mean ridits of scale items with the entire scale battery as the referent distribution can be interpreted as measuring item "difficulty" in the item-response theoretic sense. Mandal & Dey (2022) compute this kind of mean ridits for 17 ordinal-scaled socioeconomic indicators of vulnerability to climate-change disasters as rated by a sample from coastal districts in the Indian state of Odisha. They use a Kruskal-Wallis test and reject a null hypothesis that the mean ridits do not differ from 0.5, but they do not proceed to the item-difficulty interpretation. This Supplement contains a demonstration in which mean ridits for the scale items from the dq5, a widely used scale for measuring distress symptoms (Batterham, et al., 2016), correlate 0.996 and 0.988 with item difficulty parameters from a Rasch ordinal model (Andrich, 1978) and a partial-credit model (Masters, 1982), respectively.

Overview of AUC and Ridit Software Resources in R

AUC from Raw Data

Several R packages will plot ROC curves and compute the AUC, including `ROCR`, `pROC`, `precrec`, and `MLeval`. Some include confidence intervals for the AUC (e.g., `pROC`).

A disadvantage of the AUC software is that nearly without exception their plots and other outputs are expressed in ROC terms, e.g., "false-positive rates" or "specificity". Likewise, they are limited to two-groups comparisons.

Ridits from Raw Data

There are at least two ways to perform ridit analysis in R (v4.2.1; R Core Team, 2022), via the base R `wilcox.test` function or the `riditttools` package (Bohlman, 2018). They do not always return identical results, although their results are very similar. The `riditttools` package requires counts so it is strict about the distributions being discrete, whereas the `wilcox.test` function can be used to obtain mean ridits from continuous distributions. The `riditttools` package provides mean-ridit standard-error estimates.

Ridits from Models

The most straightforward way to extract mean ridits from GLMs is to derive conditional fitted distributions for the dependent variable from the model parameters and then generate or simulate the fitted CDFs.

Therapy Comparison Example

The R code for this example generates the fake data, and tests the differences between the means and between the medians. Thereafter the ridits, ridit scores, and mean ridits are computed along with an approximate 95% confidence interval for the therapy B mean ridit. Finally, the linearized CDF biplot is generated.

```
# Load libraries
library(riditttools)
library(coin)
# Here are the data and weights (frequencies).
y <- c(1,2,3,4,5,1,2,3,4,5)
x <- c(0,0,0,0,0,1,1,1,1,1)
wt <- c(995,773,427,125,630,719,627,1204,251,149)
#
# Test whether the means or medians differ
# Create long versions of the data
xlong <- c(rep(0,2950),rep(1,2950))
ylong <-
c(rep(1,wt[1]),rep(2,wt[2]),rep(3,wt[3]),rep(4,wt[4]),rep(5,wt[5]),rep(1,wt
[6]),rep(2,wt[7]),rep(3,wt[8]),rep(4,wt[9]),rep(5,wt[10]))
# mean difference test
mod1 <- lm(ylong ~ xlong); summary(mod1)
# Abbreviated output
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.53288    0.02436 103.975   <2e-16 ***
## xlong       -0.04678    0.03445  -1.358    0.175
# median difference test
#
c(median(ylong[1:2950]),median(ylong[2951:5900]))
```

```
median_test(ylong~as.factor(xlong))
# Abbreviated output
##    Asymptotic Two-Sample Brown-Mood Median Test
## data:  ylong by as.factor(xlong) (0, 1)
## Z = -11.004, p-value < 2.2e-16
#
# Now to get the ridits and ridit scores
tab0 <- wt[1:5]; tab1 <- wt[6:10]
mnr <- meanridit(tab1, tab0); mnr
ser <- seridit(tab1, tab0); ser
# Approximat 95 pct CI:
c(mnr-1.96*ser,mnr+1.96*ser)
#
# Get the CDF biplot:
cdfA <-
c(0,sum(wt[1:1]),sum(wt[1:2]),sum(wt[1:3]),sum(wt[1:4]),sum(wt[1:5]))/sum(w
t[1:5])
cdfB <-
c(0,sum(wt[6:6]),sum(wt[6:7]),sum(wt[6:8]),sum(wt[6:9]),sum(wt[6:10]))/sum(
wt[6:10])
plot(cdfB,cdfA, type = "b", pch = 16, main = "therapy cdfs", xlab =
"therapy B", ylab = "therapy A", xlim = c(0,1), ylim = c(0,1))
lines(c(0,1),c(0,1))
```

Impact of Income on Distress (dq5) Linear Regression Example

Linear Regression

```
# Read the data and load the libraries
covid <- read.csv("dq5_income_example_R.csv", header = TRUE)
library(riditools)
#
# Let's also look at the correlation between dq5 and income bracket.
# There are 6 income brackets; a score of 7 = “prefer not to say”.
cor(covid$dq5_Score_w1[covid$income_w1<7],covid$income_w1[covid$income_w1<7
], use = "pairwise.complete.obs")
## [1] -0.06571112
# Run a linear regression:
lmod <-
lm(covid$dq5_Score_w1[covid$income_w1<7]~covid$income_w1[covid$income_w1<7]
); summary(lmod)
# Abbreviated output
## Coefficients:
##                                     Estimate
## (Intercept)                          6.67926
## covid$income_w1[covid$income_w1 < 7] -0.21202
##                                     Std. Error t value
## (Intercept)                            0.37221  17.945
## covid$income_w1[covid$income_w1 < 7]    0.09318  -2.276
##                                     Pr(>|t|)
## (Intercept)                           <2e-16 ***
## covid$income_w1[covid$income_w1 < 7]   0.0231 *
## ---
## Residual standard error: 4.971 on 1194 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.004318,   Adjusted R-squared:  0.003484
## F-statistic: 5.178 on 1 and 1194 DF,  p-value: 0.02305
# Compute the pseudo-Cohen’s d:
```

```
d <- lmod$coefficients[2]/summary(lmod)$sigma; d
## covid$income_w1[covid$income_w1 < 7]
##                          -0.04265312
# Get the mean ridit estimate from this:
pnorm(d/sqrt(2),0,1)
## covid$income_w1[covid$income_w1 < 7]
##                            0.4879696
# To get 95% confidence intervals for the mean ridit,
# we first need them for the regression coefficient and d.
dlo <- (lmod$coefficients[2]-
1.96*sqrt(diag(vcov(lmod)))[2])/summary(lmod)$sigma; dlo
## covid$income_w1[covid$income_w1 < 7]
##                          -0.07939202
dhi <-
(lmod$coefficients[2]+1.96*sqrt(diag(vcov(lmod)))[2])/summary(lmod)$sigma;
dhi
## covid$income_w1[covid$income_w1 < 7]
##                         -0.005914218
# Now get the CI limits for the mean ridit:
pnorm(dlo/sqrt(2),0,1)
## covid$income_w1[covid$income_w1 < 7]
##                            0.4776157
pnorm(dhi/sqrt(2),0,1)
## covid$income_w1[covid$income_w1 < 7]
##                            0.4983316
#
```

Comparing Income Levels

There are two approaches for executing this example in R. The first approach uses the `riditttools` package. It also delivers the standard errors for the mean ridits. The `riditttools` package requires vectors of counts, so these have to be formed before obtaining the mean ridits.

```
# Now we compare adjacent income levels.
# The first approach requires getting the frequency distributions
# of dq5 for each income level.
# Construct the count vectors and assemble into a matrix.
# Get the counts as tables:
tabinc1 <- table(covid$dq5_Score_w1[covid$income_w1==1])
tabinc2 <- table(covid$dq5_Score_w1[covid$income_w1==2])
tabinc3 <- table(covid$dq5_Score_w1[covid$income_w1==3])
tabinc4 <- table(covid$dq5_Score_w1[covid$income_w1==4])
tabinc5 <- table(covid$dq5_Score_w1[covid$income_w1==5])
tabinc6 <- table(covid$dq5_Score_w1[covid$income_w1==6])
#
# Check the lengths
length(tabinc1)
length(tabinc2)
length(tabinc3)
length(tabinc4)
length(tabinc5)
length(tabinc6)
#
# Fill in the gaps to make the lengths equal.
tabinc1 <- c(tabinc1[1:19],0,0)
tabinc3 <- c(tabinc3[1:20],0)
```

```
tabinc5 <- c(tabinc5[1:20],0)
#
# Assemble the vectors into a matrix.
tabinc <- rbind(tabinc1,tabinc2,tabinc3,tabinc4,tabinc5,tabinc6)
#
# Set up a matrix to receive the mean ridits:
# mean ridits
meanriditmat <- matrix(c(rep(0,10)),ncol = 2)
colnames(meanriditmat) <- c("income bracket","mean ridit")
for (i in 1:5){
  meanriditmat[i,1] <- i
  meanriditmat[i,2] <- meanridit(tabinc[i+1,], tabinc[i,])
}
meanriditmat
##      income bracket mean ridit
## [1,]              1  0.4118073
## [2,]              2  0.4878460
## [3,]              3  0.4945191
## [4,]              4  0.4969056
## [5,]              5  0.4940630
#
# Set up a matrix to receive the standard errors:
seriditmat <- matrix(c(rep(0,10)),ncol = 2)
colnames(meanriditmat) <- c("income bracket","std.err.")
# Compute the standard errors for these mean ridits
for (i in 1:5){
  seriditmat[i,1] <- i
  seriditmat[i,2] <- seridit(tabinc[i+1,], tabinc[i,])
}
seriditmat
```

In the second approach, looping the `wilcox.test` function through the income brackets yields the mean ridits.

```
# Get the ridits using the U statistic.
# Tabulate the sizes of the income-bracket subsamples:
inctab <- table(covid$income_w1); inctab
dq5riditmat <- matrix(c(rep(0,10)),ncol = 2)
colnames(dq5riditmat) <- c("income bracket","mean ridit")
for (i in 1:5){
  wilstat <-
wilcox.test(covid$dq5_Score_w1[covid$income_w1==i+1],covid$dq5_Score_w1[cov
id$income_w1==i])
  dq5riditmat[i,1] <- i
  dq5riditmat[i,2] <-wilstat$statistic/(inctab[i+1]*inctab[i])
}
dq5riditmat
##      income bracket mean ridit
## [1,]              1  0.4054718
## [2,]              2  0.4853442
## [3,]              3  0.4945191
## [4,]              4  0.4969056
## [5,]              5  0.4918375
```

We can also compare mean ridits using the 4th income bracket as the reference distribution, using these two approaches.

Ordinal Logistic Regression

Crowd Avoidance Example

This code uses the `VGAM` package (Yee, 2010) to run an OLR model comparing the period 1 and 10 crowd-avoidance behavior ratings. The `ridittools` package is used to compute the sample mean ridit for period 10 relative to period 1 (period 1 is coded 0 and period 10 is coded 1 in the period variable). We then extract the OLR model ridits for period 1 and the mean ridit for period 10 relative to period 1 from the model fitted values.

```
library(VGAM)
library(ridittools)
yougov <- read.csv("YouGovExampleData.csv", header = TRUE)
#
# Get the sample mean ridit and its standard error.
meanridit(yougov$obs[6:10],yougov$obs[1:5])
[1] 0.3316498
seridit(yougov$obs[6:10],yougov$obs[1:5])
[1] 0.02135541
#
# Run the OLR.
ymod1 <- vglm(yougov$score~yougov$period, family = cumulative(parallel = TRUE),
weights = yougov$obs)
summary(ymod1)
# Abbrieviated output:
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept):1  -3.7702     0.2214 -17.032  < 2e-16 ***
## (Intercept):2  -3.3293     0.1941 -17.150  < 2e-16 ***
## (Intercept):3  -2.1454     0.1501 -14.291  < 2e-16 ***
## (Intercept):4  -0.8302     0.1249  -6.645 3.02e-11 ***
## yougov$period   1.2897     0.1655   7.791 6.66e-15 ***
#
# Get the model ridits from the fitted values
ymod1ridit <- c(rep(0,5))
ymod1ridit[1] <- fitted.values(ymod1)[1,1]/2
for (i in 2:5){
  ymod1ridit[i] <- sum(fitted.values(ymod1)[1,1:i-1]) + fitted.values(ymod1)[1,i]/2
}
# Get the model mean ridit
sum(ymod1ridit*fitted.values(ymod1)[6,])
## 0.331266
#
# Now run a non-parallel model:
ymod2 <- vglm(yougov$score~yougov$period, family = cumulative(parallel = FALSE),
weights = yougov$obs)
summary(ymod2)
# Abbreviated output:
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept):1    -4.0673     0.4510  -9.018  < 2e-16 ***
## (Intercept):2    -3.3569     0.3217 -10.435  < 2e-16 ***
## (Intercept):3    -2.5257     0.2216 -11.400  < 2e-16 ***
## (Intercept):4    -0.7855     0.1251  -6.278 3.42e-10 ***
## yougov$period:1   1.6586     0.4970   3.337 0.000847 ***
## yougov$period:2   1.3213     0.3685   3.586 0.000336 ***
## yougov$period:3   1.7877     0.2533   7.057 1.70e-12 ***
## yougov$period:4   1.1663     0.1713   6.809 9.84e-12 ***
```

Mask-Wearing Example

R: This code uses the VGAM package (Yee, 2010) to run an OLR model to predict mask-wearing by age. The fitted values can then be used to generate the model ridits for each value of age. Code for doing this is not provided here, instead a .csv file is provided that contains the ridits. A graph of the model ridits is then produced from this .csv file.

```
# Read data
# This file contains the ridits for each age
agemask <- read.csv("AgePredictsMasks2.csv", header = TRUE)
# This is the raw data
maskwear <- read.csv("MaskWearingByAge_w1_R.csv", header = TRUE)
# Load the VGAM package
library(VGAM)
# Get the non-missing subset of the raw data:
submaskwear <- na.omit(maskwear[maskwear$actFacMsk_w1<6,1:2])
# Run the age OLR model:
vmod1 <- vglm(submaskwear$actFacMsk_w1~ submaskwear$age_w1, family =
cumulative(parallel = TRUE)); summary(vmod1)
# Abbrieviated output:
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept):1      -0.513677   0.155100  -3.312 0.000927 ***
## (Intercept):2      -0.244593   0.153498  -1.593 0.111056
## (Intercept):3       0.469879   0.153058   3.070 0.002141 **
## (Intercept):4       2.461946   0.170675  14.425  < 2e-16 ***
## submaskwear$age_w1 -0.020952   0.003151  -6.649 2.95e-11 ***
#
# Graph the mean ridits by age.
# Get the non-missing subset of the ridits data:
submask <- na.omit(agemask)
# Do the plot
plot(submask$age,submask$mean._ridit, type = "l", main = "", xlab = "age", ylab =
"mean ridit")
lines(c(29,29),c(0.40,submask$mean._ridit[submask$age==29]), lty = 2)
lines(c(46,46),c(0.40,submask$mean._ridit[submask$age==46]))
lines(c(63,63),c(0.40,submask$mean._ridit[submask$age==63]), lty = 2)
lines(c(15,29),c(submask$mean._ridit[submask$age==29],submask$mean._ridit[submask$a
ge==29]), lty = 2)
lines(c(15,46),c(submask$mean._ridit[submask$age==46],submask$mean._ridit[submask$a
ge==46]))
lines(c(15,63),c(submask$mean._ridit[submask$age==63],submask$mean._ridit[submask$a
ge==63]), lty = 2)
```

Generalized Linear Models

Gamma Regression

R: The gamma regression model can be run using the gamlss package.

```
# Load libraries and data
library(gamlss)
library(MASS)
library(lmtest)
library(statmod)
stroop <- read.table("stroopdataRT.txt", header = T)
#
# We will treat the RT variable as having a lower bound of 250 ms.
rt250 <- stroop$StRTOverall - 250
# Elmiinate the NAs in the optimism scale:
findat <-
data.frame(na.omit(cbind(rt250,stroop$Condition,stroop$LifeOrientation)))
colnames(findat) <- c("rt250","Condition","LifeOrientation")
```

```
# Standardize the LOT scale:
zlot <- (findat$LifeOrientation -
mean(findat$LifeOrientation))/sd(findat$LifeOrientation)
# Gamma GLM:
gmod4 <- gamlss(rt250 ~ Condition*zlot, family = GA, data = findat);
summary(gmod4)
# Abbreviated output:
## Mu link function:  log
## Mu Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.57377    0.07942  70.178  < 2e-16 ***
## Condition      -0.25507    0.11261  -2.265  0.02632 *
## zlot            0.04215    0.06518   0.647  0.51977
## Condition:zlot  0.32080    0.10615   3.022  0.00341 **
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.68409    0.07501   -9.12 7.03e-14 ***
```

Next, we generate the model CDFs for each experimental condition, at zlot = 0 and zlot = 1 (i.e., at the covariate mean and one standard deviation above the mean).

```
# Generate the referent CDF, at Condition = 0, zlot = 0:
cdfR <- pGA(seq2,mu = exp(gmod4$mu.coefficients[1]), sigma =
exp(gmod4$sigma.coefficients[1]))
# CDF at Condition = 1, zlot = 0
cdf10 <- pGA(seq2,mu = exp(gmod4$mu.coefficients[1] +
gmod4$mu.coefficients[2]), sigma = exp(gmod4$sigma.coefficients[1]))
# CDF at Condition = 0, zlot = 1
cdf01 <- pGA(seq2,mu = exp(gmod4$mu.coefficients[1] +
gmod4$mu.coefficients[3]), sigma = exp(gmod4$sigma.coefficients[1]))
# CDF at Condition = 1, zlot = 1
cdf11 <- pGA(seq2,mu = exp(gmod4$mu.coefficients[1] +
gmod4$mu.coefficients[2] + gmod4$mu.coefficients[3] +
gmod4$mu.coefficients[4]), sigma = exp(gmod4$sigma.coefficients[1]))
#
```

Finally, we use numerical integration to compute the mean ridits for our simple-effects comparisons.

```
# Compare cdf01 with cdfR
lnth <- length(cdfR)-1
modelridit <- 0.5*cdfR[1]*cdf01[1]
for (i in 1:lnth) {
  modelridit <- modelridit + 0.5*(cdfR[i]+cdfR[i+1])*(cdf01[i+1]-cdf01[i])
}
modelridit01 <- modelridit + 0.5*(1+cdf01[lnth+1])*(1-cdf01[lnth+1])
modelridit01
## [1] 0.5228174
#
# Compare cdf11 with cdf10
modelridit <- 0.5*cdf10[1]*cdf11[1]
for (i in 1:lnth) {
  modelridit <- modelridit + 0.5*(cdf10[i]+cdf10[i+1])*(cdf11[i+1]-
cdf11[i])
}
```

```
modelridit1011 <- modelridit + 0.5*(1+cdf11[lnth+1])*(1-cdf11[lnth+1])
modelridit1011
## [1] 0.6884552
```

Beta Regression and CDF-Quantile Models

R: First, we compare the means and medians of the "likely" and reversed "unlikely" ratings.

```
# Load libraries and read the data:
library(riditttools)
library(coin)
library(betareg)
library(cdfquantreg)
library(MASS)
ipcc <- read.csv("IPCC2009.csv", header = TRUE)
#
# Concatenate the sets of questions:
likely <- c(ipcc$Q4_best,ipcc$Q5_best,ipcc$Q6_best)
unlikely <- 1-c(ipcc$Q8_best,ipcc$Q9_best,ipcc$Q10_best)
# Show the means:
c(mean(likely),mean(unlikely))
## [1] 0.6677728 0.6565770
# Show the medians:
c(median(likely),median(unlikely))
## [1] 0.70 0.75
#
pvar <- c(likely, unlikely)
gvar <- c(rep(0,669),rep(1,669))
# Compare the samples' medians
median_test(pvar~as.factor(gvar))
# Abbreviated output
##  Asymptotic Two-Sample Brown-Mood Median Test
## data:  pvar by as.factor(gvar) (0, 1)
## Z = -5.138, p-value = 2.777e-07
#
# Compare the samples' means
t.test(pvar~as.factor(gvar))
# Abbreviated output
##  Welch Two Sample t-test
## data:  pvar by as.factor(gvar)
## t = 0.93381, df = 1228.4, p-value = 0.3506
#
```

Next, we use the Wilcox test to get the sample mean ridit, using "likely" as the referent distribution.

```
# Get the ridit using Wilcoxon rank sum test:
wilstat <- wilcox.test(unlikely,likely); wilstat$statistic
##      W
## 244501
wilstat$statistic/(length(unlikely)*length(likely))
##         W
## 0.5462965
#
```

Then we use the `betareg` package (Cribari-Neto & Zeileis, 2010) to run a beta regression regression model to compare the two samples, after which we obtain the fitted distributions and the model mean ridit.

```
# Run a beta regression model
# First, shift the zeros and ones away from the boundaries
pvarmod <- pvar; pvarmod[pvar==0] <- 0.001; pvarmod[pvar==1] <- 0.999
bmod1 <- betareg(pvarmod~as.factor(gvar)|as.factor(gvar))
```

```
summary(bmod1)
# Abbreviated output
## Coefficients (mean model with logit link):
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.66646    0.03401  19.596  < 2e-16 ***
## as.factor(gvar)1 -0.16154    0.05335  -3.028  0.00246 **
##
## Phi coefficients (precision model with log link):
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       1.53181    0.05054  30.308   <2e-16 ***
## as.factor(gvar)1 -0.60300    0.06966  -8.656   <2e-16 ***
#
#
# Generate the fitted CDFs:
seq2 <- seq(0,1,0.01)
s1_1 <-
exp(bmod1$coefficients$mean[1])/(1+exp(bmod1$coefficients$mean[1]))*exp(bmod1$coeff
icients$precision[1])
s2_1 <- (1-
(exp(bmod1$coefficients$mean[1])/(1+exp(bmod1$coefficients$mean[1]))))*exp(bmod1$co
efficients$precision[1])
s1_2 <-
exp(sum(bmod1$coefficients$mean))/(1+exp(sum(bmod1$coefficients$mean)))*exp(sum(bmo
d1$coefficients$precision))
s2_2 <- (1-
(exp(sum(bmod1$coefficients$mean))/(1+exp(sum(bmod1$coefficients$mean)))))*exp(sum(
bmod1$coefficients$precision))
cdf1 <- pbeta(seq2,s1_1,s2_1)
cdf2 <- pbeta(seq2,s1_2,s2_2)
pdf1 <- dbeta(seq2,s1_1,s2_1)
pdf2 <- dbeta(seq2,s1_2,s2_2)
#
# Get the approximate model mean ridit
lnth <- length(cdf1)-1
modelridit <- 0.5*cdf1[1]*cdf2[1]
for (i in 1:lnth) {
  modelridit <- modelridit + 0.5*(cdf1[i]+cdf1[i+1])*(cdf2[i+1]-cdf2[i])
}
modelridit <- modelridit + 0.5*(1+cdf2[lnth+1])*(1-cdf2[lnth+1])
modelridit
## [1] 0.4733116
#
# Get the fitted and empirical CDF biplots.
# First, get the empirial CDFs:
tlike <- tabulate(100*likely, nbins=101)
tunlike <- tabulate(100*unlikely, nbins=101)
cdflike <- cumsum(tlike)/sum(tlike)
cdfunlike <- cumsum(tunlike)/sum(tunlike)
#
# Plot them along with the line of equality:
plot(cdf2,cdf1, type = "l", main = "beta reg. model", xlab = "1 - unlikely", ylab =
"likely", xlim = c(0,1), ylim = c(0,1))
lines(c(0,1),c(0,1))
lines(cdfunlike,cdflike, lty = 2)
#
```

Now we use the `cdfquantreg` package (Shou & Smithson, 2019) to run a FTCDFQ regression model to compare the two samples, after which we obtain the fitted distributions and the model mean ridit. The FTCDFQ distribution used here is a 3-parameter arcsinh-arcsinh distribution with the outer-position W function (Smithson & Shou, 2022).

```
# Three-parameter model:
ftmod2 <- cdfquantregFT(pvar ~ as.factor(gvar)|as.factor(gvar), mu.fo = mu.fo ~
as.factor(gvar), fd = "arcsinh", sd = "arcsinh", inner = FALSE, version = "W", data
= pgvar)
summary(ftmod2)
# Abbreviated output:
## Mu coefficients (Location submodel)
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.66962    0.05857  11.433  < 2e-16 ***
## as.factor(gvar)1  0.73620    0.09982   7.376 1.64e-13 ***
##
## Sigma coefficients (Dispersion submodel)
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.79129    0.05898 -13.417  < 2e-16 ***
## as.factor(gvar)1  0.54866    0.08193   6.697 2.13e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta coefficients (Skewness submodel)
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.5156     0.1131  -4.559 5.14e-06 ***
## as.factor(gvar)1   0.5961     0.1493   3.992 6.54e-05 ***
#
#
# Generate the fitted CDFs:
s101 <- seq(0,100,1)/100
cdffit1 <- c(rep(0,101))
for (i in 1:101) {
   cdffit1[i] <- cdfft(s101[i], sigma=exp(coef(ftmod2)[3]),
   theta=coef(ftmod2)[1], fd = "arcsinh", sd = "arcsinh", mu = coef(ftmod2)[5],
inner = FALSE,
   version = "W")
}
cdffit2 <- c(rep(0,101))
for (i in 1:101) {
   cdffit2[i] <- cdfft(s101[i], sigma=exp(coef(ftmod2)[3] + coef(ftmod2)[4]),
   theta=coef(ftmod2)[1] + coef(ftmod2)[2], fd = "arcsinh", sd = "arcsinh", mu =
coef(ftmod2)[5] + coef(ftmod2)[6], inner = FALSE,
   version = "W")
}
#
# Approximate model mean ridit:
lnth <- length(cdffit1)-1
ftmodridit <- 0.5*cdffit1[1]*cdffit2[1]
for (i in 1:lnth) {
  ftmodridit <- ftmodridit + 0.5*(cdffit1[i]+cdffit1[i+1])*(cdffit2[i+1]-
cdffit2[i])
}
ftmodridit <- ftmodridit + 0.5*(1+cdffit2[lnth+1])*(1-cdffit2[lnth+1])
ftmodridit
## [1] 0.5476387
#
# Get the fitted and empirical CDF biplots.
plot(cdffit2,cdffit1, type = "l", main = "FTCDFQ model", xlab = "1 - unlikely",
ylab = "likely")
lines(c(0,1),c(0,1))
lines(cdfunlike,cdflike, lty = 2)
#
```

Finally, Figure S1 shows the fitted beta distributions (dashed-line) and fitted FTCDFQ distributions (solid lines) superimposed on histograms of the data for the likely and

1 – unlikely samples. These graphs confirm the indications from the ridits and CDF biplots that the beta distribution model is misspecified.

```
# Generate the fitted distribution and histograms
# Note that the sigma coefficients must be exponentiated.
# First, the likely data:
uniqdat <- sort(unique(pgvar$pvar[pgvar$gvar == 0]))
densfit <- c(rep(0,length(uniqdat)))
for (i in 1:length(uniqdat)) {
   densfit[i] <- pdfft(uniqdat[i], sigma=exp(coef(ftmod2)[3]),
   theta=coef(ftmod2)[1], fd = "arcsinh", sd = "arcsinh", mu = coef(ftmod2)[5],
inner = FALSE,
   version = "W")
}
truehist(pgvar$pvar[pgvar$gvar == 0], nbins = 101,
main = "likely", xlab = "proportion", ylab = "pdf",
ylim = c(0,4))
lines(uniqdat,densfit,lwd = 2)
lines(seq2,pdf1, lwd = 2, lty = 2)
# Now for the unlikely data:
uniqdat <- sort(unique(pgvar$pvar[pgvar$gvar == 1]))
densfit <- c(rep(0,length(uniqdat)))
for (i in 1:length(uniqdat)) {
   densfit[i] <- pdfft(uniqdat[i], sigma=exp(coef(ftmod2)[3] + coef(ftmod2)[4]),
   theta=coef(ftmod2)[1] + coef(ftmod2)[2], fd = "arcsinh", sd = "arcsinh", mu =
coef(ftmod2)[5] + coef(ftmod2)[6], inner = FALSE,
   version = "W")
}
truehist(pgvar$pvar[pgvar$gvar == 1], nbins = 101,
main = "1 - unlikely", xlab = "proportion", ylab = "pdf",
ylim = c(0,4))
lines(uniqdat,densfit,lwd = 2)
lines(seq2,pdf2, lwd = 2, lty = 2)
#
```

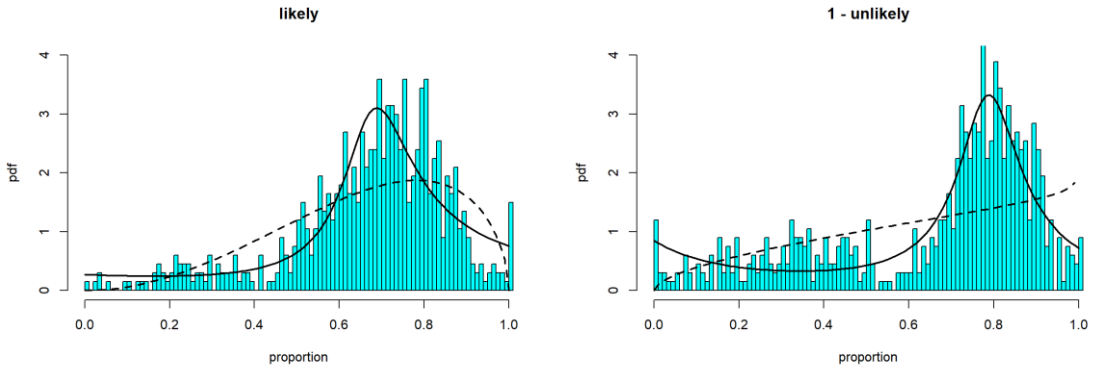

Figure S1. Beta and FTCDFQ Fitted Distribution Plots

Mean Ridit as Item-Difficulty Parameter Example

R: First, we run two graded-response IRT models with difficulty parameters.

```
# Load libraries and read the data:
library(riditools)
library(mirt)
dq5dat <- read.csv("dq5_income_example.csv", header = TRUE)
#
# Run a rsm model (Rasch rating scale model, Andrich 1978)
mod2 <- (mirt(dq5dat[,2:6], 1, verbose = FALSE, itemtype = 'rsm', SE = TRUE))
itemparams2 <- coef(mod2, IRTpars = TRUE, simplify = TRUE); itemparams2$item
```

```
##        a1         b1        b2       b3      b4          c
## dq1_w1  1 -0.8535574 0.3616781 2.539927 4.39275  0.0000000
## dq2_w1  1 -0.8535574 0.3616781 2.539927 4.39275 -0.3726056
## dq3_w1  1 -0.8535574 0.3616781 2.539927 4.39275 -0.7327139
## dq4_w1  1 -0.8535574 0.3616781 2.539927 4.39275 -0.1960120
## dq5_w1  1 -0.8535574 0.3616781 2.539927 4.39275 -0.4134280
# The difficulty parameters are in column c.
#
# Run a grsm model (Muraki 1992)
mod3 <- (mirt(dq5dat[,2:6], 1, verbose = FALSE, itemtype = 'grsm', SE = TRUE))
itemparams3 <- coef(mod3); mod3dif <-
c(itemparams3$dq1_w1[1,6],itemparams3$dq2_w1[1,6],itemparams3$dq3_w1[1,6],itemparam
s3$dq4_w1[1,6],itemparams3$dq5_w1[1,6]); mod3dif
## [1]  0.00000000 -0.15267049 -0.34509988 -0.09052962 -0.17673075
```

Next, we compute the mean ridits for each item with the entire collection of the dq5 items as the reference.

```
# Get the dq5 ridits relative to the entire data-set
# First, construct the item tables
dq5tab <-
rbind(table(dq5dat[,2]),table(dq5dat[,3]),table(dq5dat[,4]),table(dq5dat[,5]),table
(dq5dat[,6]))
# Get the table for the entire data-set:
tottab <- colSums(dq5tab)
# Get the mean ridits
dq5meanridits <- c(rep(0,5))
for (i in 1:5){
  dq5meanridits[i] <- meanridit(dq5tab[i,],tottab)
}
dq5meanridits
## [1] 0.5362586 0.4960221 0.4607704 0.5178188 0.4891546
```

According to the mean ridits, item 1 is the “easiest”, while item 3 is the most “difficult”. Note that the mean ridits for any collection of items on a common scale with the entire collection of the items as the reference distribution are compositional; their mean always is ½.

Finally, we examine the correlations between these mean ridits and the item difficulty parameters from the two graded-response models.

```
# rms model difficulty parameter
cor(dq5meanridits,itemparams2$item[,6])
[1] 0.996778
# grms model difficulty parameter
cor(dq5meanridits,mod3dif)
[1] 0.9878261
#
```

References

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, *43*(4), 561-573.

Batterham, P. J., Sunderland, M., Carragher, N., Calear, A. L., Mackinnon, A. J., & Slade, T. (2016). The Distress Questionnaire-5: population screener for psychological distress was more accurate than the K6/K10. *Journal of clinical epidemiology*, *71*, 35-42. DOI: 10.1016/j.jclinepi.2015.10.005

Bohlman, E. (2018). riditttools: Useful Functions for Ridit Analysis. R package version 0.1, https://CRAN.R-project.org/package=riditttools.

Cribari-Neto, F. & Zeileis, A. (2010). Beta Regression in R. *Journal of Statistical Software*, 34(2), 1-24. doi:10.18637/jss.v034.i02. URL https://doi.org/10.18637/jss.v034.i02.

Jansen, M. E. (1984). Ridit analysis, a review. *Statistica Neerlandica*, *38*(3), 141-158.

John, C.R. (2020). MLeval: Machine Learning Model Evaluation. R package version 0.3, <https://CRAN.R-project.org/package=MLeval>.

Mandal, K., & Dey, P. (2022). Coastal vulnerability analysis and RIDIT scoring of socio-economic vulnerability indicators–A case of Jagatsinghpur, Odisha. *International Journal of Disaster Risk Reduction*, *79*, *103143*.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, *47*(2), 149-174.

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J. C., & Müller, M. (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, *12*(1), 1-8.

Saito, T., & Rehmsmeier, M. (2017). Precrec: fast and accurate precision–recall and ROC curve calculations in R. *Bioinformatics*, *33*(1), 145-147.

Shou, Y. & Smithson, M. (2019). cdfquantreg: An R Package for CDF-Quantile Regression. *Journal of Statistical Software*, 88(1), 1-30. doi:10.18637/jss.v088.i01 URL https://doi.org/10.18637/jss.v088.i01.

Sing, T., Sander, O., Beerenwinkel, N., & Lengauer, T. (2005). ROCR: visualizing classifier performance in R. *Bioinformatics*, *21*(20), 3940-3941.

Smithson, M. & Shou, Y. (online 2023). Flexible cdf-quantile distributions on the closed unit interval, with software and applications. *Communications in Statistics – Theory and Methods*. https://www.tandfonline.com/doi/full/10.1080/03610926.2023.2166352

Yee, T.W. (2010). The VGAM Package for Categorical Data Analysis. *Journal of Statistical Software*, 32(10), 1-34. DOI: 10.18637/jss.v032.i10. URL https://www.jstatsoft.org/article/view/v032i10/.