…Let's say you've got data of many paired cases of two ordinal variables, like you might when you ask a large number of people the same two Likert scale questions (e.g. "poor", "fair", "good", "very good", "excellent").

What could you learn from the data from those two questions?

Here's a few common approaches:

1) Compare the means of each variable by abusing a t-test.

2) Compare the distribution of each variable with a chi-squared goodness-of-fit test.

3) Check for a relationship between responses of each variable with a chi-squared independence test.

4) Estimate the strength of such a relationship with a Spearman correlation.

Spearman correlation works because it's non-parametric; it doesn't care about the distribution of the variables, but it leaves a lot of information unused. There's a better alternative for ordinal-to-ordinal data: Polychoric correlation.

Polychoric correlation is designed for to find correlations between ordinal variables (ordered categorical data) like Likert scales and binned data.

Great, but how does it work?

Polychoric correlation assumes that each ordinal data point represents a binned continuous value from a normal distribution, and then tries to estimate the correlation coefficient on that assumption.

For actual binned data like responses to "How many movies did you watch this year: 0-4, 5-9, 10-19, 20 or more?", then that's nothing new to say that a "10-19" response could represent someone who has watched 12 movies.

With Likert data like responses to "How was your last movie: poor, fair, good, very good, or excellent?", we treat the responses like they're binned values for some abstract variable we didn't ask for, like a quality scale from 0-100, so "very good" could be a 73 or a 64.

The actual number of movies watched, or the actual 0-100 quality score that someone would

give their last movie aren't recorded in the data, so we would call them ***latent*** variables.

In polychoric correlation, we don't need to know or specify where the boundary between "good" and "very good" is, just that it exists. The distribution of the ordinal responses, along with the assumption that the latent values follow a normal distribution, is enough that the polychor()function in the polycor R package can do that for us. In most practical cases, you don't even need to know where the cutoffs are, but they are useful for demonstration that the method works.

Polychoric correlation estimates the correlation between such latent variables as if you actually knew what those values were. In the examples given, we start with the latent variables and use cutoffs to set them into bins, and then use polychoric on the artificially binned data. In any practical use case, the latent data would be invisible to you, and the cutoffs would be determined by whoever designed the survey.

The only reason the latent variables are visible in these examples is to demonstrate how effective polychoric correlation is under different conditions. Ideally, polychoric correlation on the (realistic) binned / ordinal data will closely match the Pearson correlation on the latent data.

To start, here's the modified example code given in the help file for polychor().

This code first checks that you have mvtnorm installed, which is used to generate 1000 values from a standard multivariate normal distribution with covariance 0.5.

```
library(polycor)

library(mvtnorm)

set.seed(12345)

data <- rmvnorm(1000, c(0, 0), matrix(c(1, .5, .5, 1), 2, 2))

# Then it takes the Pearson correlation of those data points.

cor(data) # 0.5264

# And the Spearman correlation

cor(data, method="spearman") #0.5043
```

```
# Now let’s try with polychoric correlation.

# First we need some cutoffs to break the data into cells

# ideally with breakpoints that avoid nearly empty cells.


x <- cut(dat[,1], c(-Inf, .75, Inf))

y <- cut(dat[,2], c(-Inf, -1, .5, 1.5, Inf))


# Then take the polychoric correlation.

# The default method uses the faster 2-step method and only returns the correlation.

polychor(x, y)

# [1] 0.5230474
```

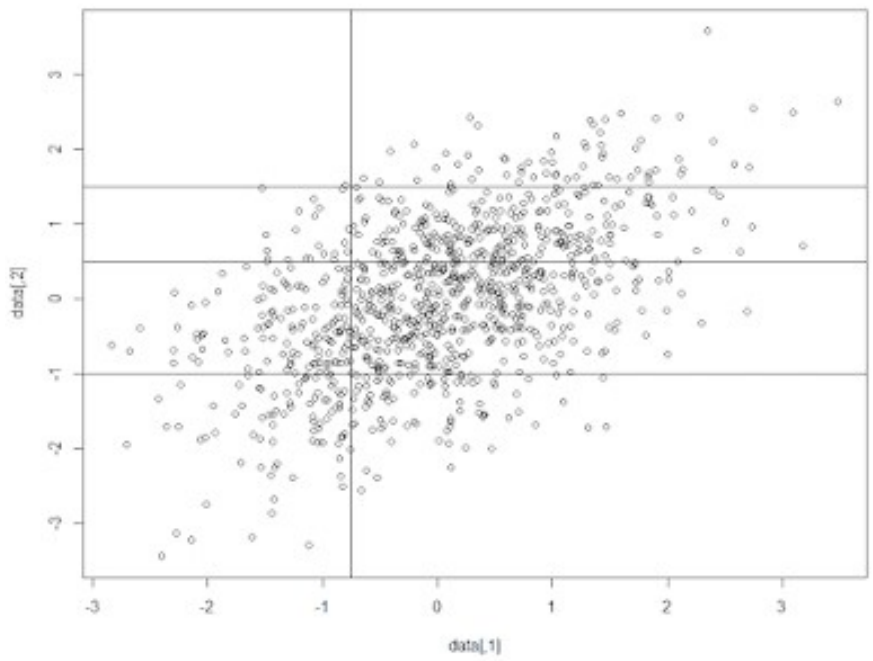
Setting standard error to TRUE gives us a standard error of the correlation and a chi-squared test for bivariate normality. If the data is approximately normal, the normality test should yield a moderate p-value (between .025 and .975).

```
polychor(x, y, std.err=TRUE)

# Polychoric Correlation, 2-step est. = 0.5231 (0.03729)

# Test of bivariate normality: Chisquare = 2.758, df = 2, p = 0.2519
```

Setting ML to TRUE changes the method the slower Maximum Likelihood method but it also allows for estimates of the cutoffs between cells of the latent, unseen normally distributed values that have been binned into cells. Compare the estimates to the true cutoffs of .75 and c(-1.00, 0.50, 1.50)

```
polychor(x, y, ML=TRUE, std.err=TRUE)


#Polychoric Correlation, ML est. = 0.5231 (0.03819)

#Test of bivariate normality: Chisquare = 2.739, df = 2, p = 0.2543
```

```
#  Row Threshold
#  Threshold Std.Err.
#     0.7537  0.04403


# Column Thresholds
#  Threshold Std.Err.
#1   -0.9842  0.04746
#2    0.4841  0.04127
#3    1.5010  0.06118


# Let’s plot this out with dividing lines
plot(data)
abline(h=c(-1, .5, 1.5))
abline(v=c(-.75))
```



You don’t need all the multivariate random normal overhead, polychor()works on any crosstab. We can check that this with the dataset we already made.

table(x,y)

155	437	162	21
7	88	84	46

polychor(tab)

0.5230474

Let’s try the same data with some different cutoffs and see how it does

x2 <- cut(data[,1], c(-Inf, -2, -1, 0, 1, 2, Inf))

y2 <- cut(data[,2], c(-Inf, -2, -1, 0, 1, 2, Inf))

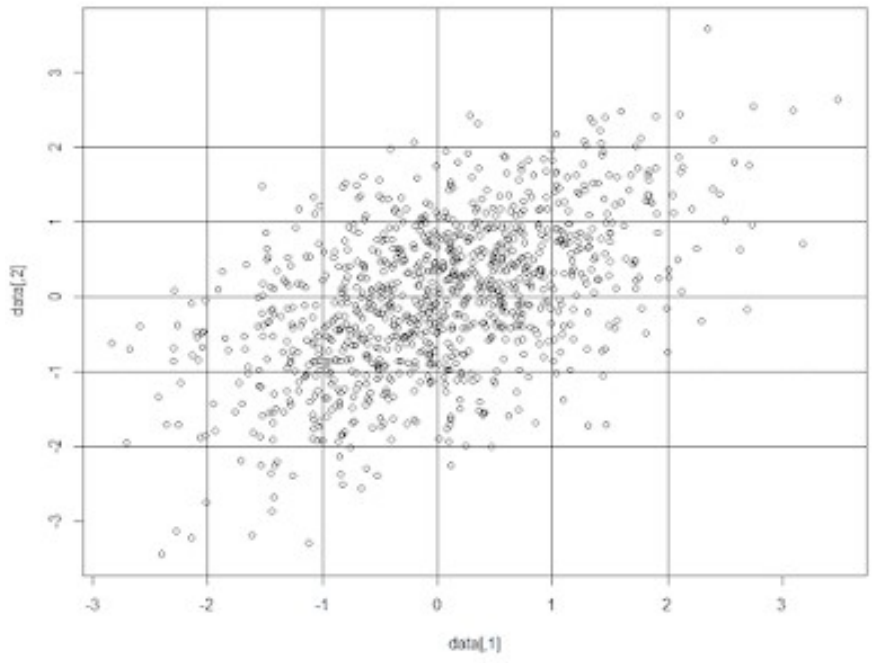
table(x2,y2)

4	7	15	1	0	0
10	38	49	25	5	0
7	58	147	102	28	1
2	31	92	158	50	2
0	5	22	62	39	12
0	0	3	8	11	6

plot(data)

abline(h=c(-2, -1, 0, 1, 2))

abline(v=c(-2, -1, 0, 1, 2))



polychor(x2,y2,ML=TRUE,std.err=TRUE) # takes 5-10 seconds

#Polychoric Correlation, ML est. = 0.5069 (0.02623)

#Test of bivariate normality: Chisquare = 22.89, df = 24, p = 0.5266

Row Thresholds

Threshold Std.Err.

#1 -1.915000 0.08103

#2 -1.016000 0.04787

#3 -0.009191 0.03940

#4 0.962900 0.04710

#5 1.928000 0.08108

Column Thresholds

Threshold Std.Err.

#1 -1.97800 0.08591

#2 -0.98300 0.04712

#3 -0.02738 0.03947

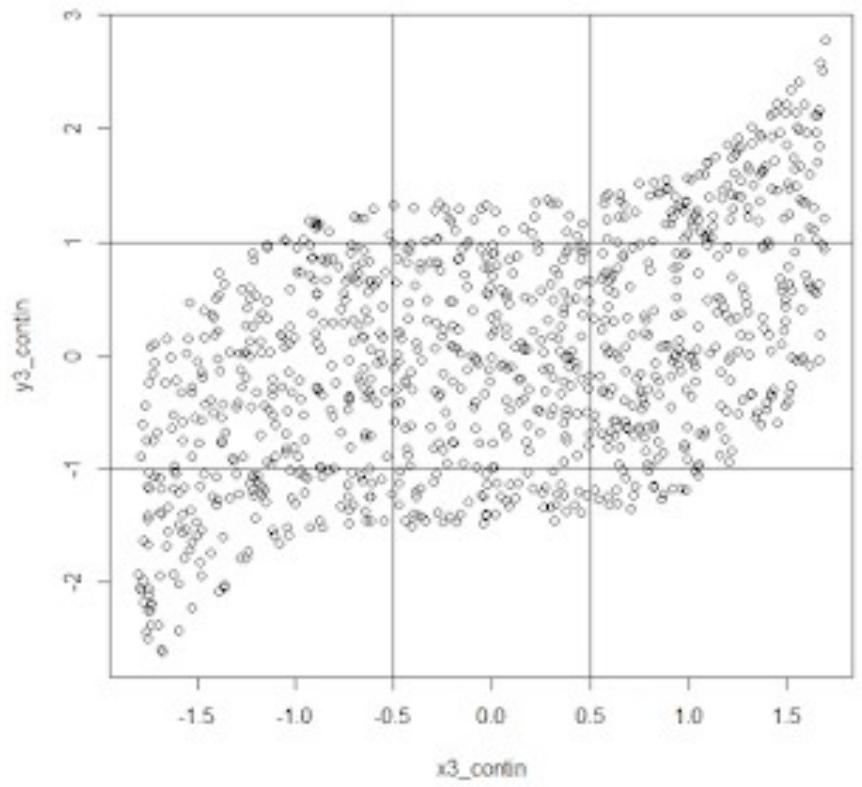
#4   1.01800 0.04797

#5   2.05600 0.09097

The polychor()function estimated the thresholds well, but with a little less accuracy and certainty at the edges than the middle. It estimated the correlation near the true value of 0.5 pretty well. Finally, the chi-squared test yielded a moderate p-value, indicating that there wasn't any significant difference between our binned data and typical data binned from a normal distribution. In short, it worked great.

Finally, let's try it with some non-linear data, but with evenly-spaced cutoffs.

```
# Make random data in a wide s-curve and force it to mean 0, sd 1.

set.seed(12345)

x3_contin = runif(1000)

y3_contin = 4*(x3_contin -0.5)^3 + runif(1000) – 0.5


x3_contin = (x3_contin – mean(x3_contin)) / sd(x3_contin)

y3_contin = (y3_contin – mean(y3_contin)) / sd(y3_contin)


plot(x3_contin,y3_contin)

abline(h=c(-1,1))

abline(v=c(-0.5,0.5))
```



Pearson

cor(x3_contin,y3_contin)

[1] 0.5021618

Spearman

cor(x3_contin,y3_contin,method=”spearman”)

#[1] 0.4696544

Polychoric

x3_ord = cut(x3_contin, c(-Inf, -0.5, 0.5, Inf))

y3_ord = cut(y3_contin, c(-Inf, -1, 1, Inf))

table(x3_ord, y3_ord)

121	211	16

52	208	28
21	211	132

polychor(x3_ord,y3_ord,ML=TRUE,std.err=TRUE)

#Polychoric Correlation, ML est. = 0.5429 (0.03126)

#Test of bivariate normality: Chisquare = 6.296, df = 3, p = 0.09808

Row Thresholds

Threshold Std.Err.

#1 -0.3957 0.04054

#2 0.3415 0.04056

Column Thresholds

Threshold Std.Err.

#1 -0.8624 0.04531

#2 0.9315 0.04675

Compare to Pearson, Spearman correlation on binned data

cor(as.numeric(x3_ord), as.numeric(y3_ord)) # 0.4217

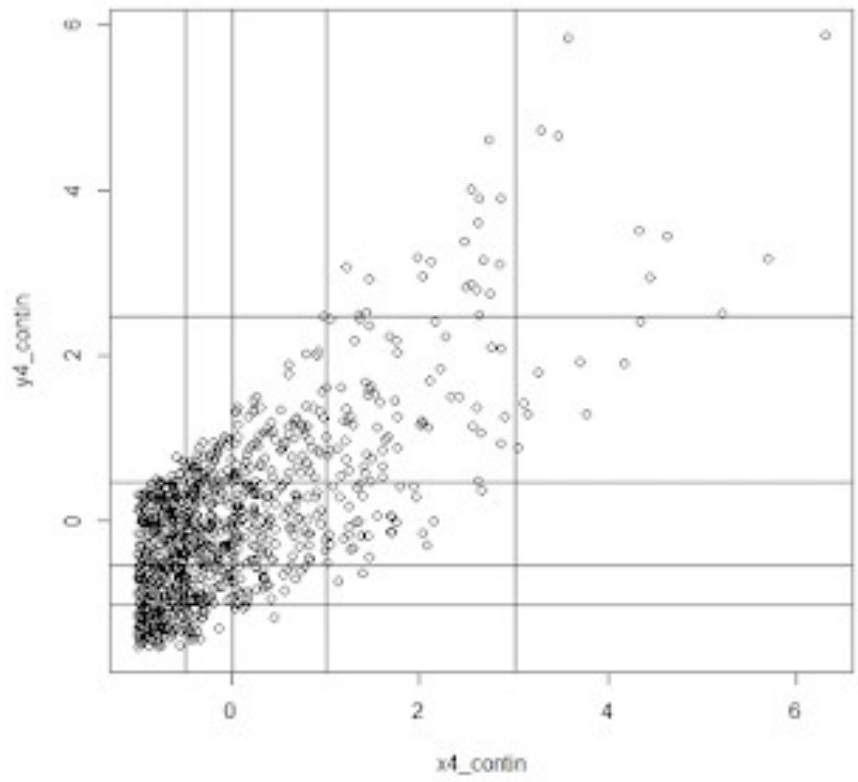
cor(as.numeric(x3_ord), as.numeric(y3_ord), method=”spearman”) # 0.4217

The correlation coefficients on the latent, unbinned data are 0.4696 and 0.5021. That’s a good, informal target range for the ‘true’ correlation. Here, polychor()overestimated the correlation as 0.5429, but it did a little better than the classic correlation measures on the binned data, which underestimated the correlation as 0.4217.

The cutoffs in both dimensions were too close to the middle, but only by 2.5 standard errors, so it’s not egregiously wrong. The polychoric correlation is close the other correlation measures on the latent data, not the test of bivariate normality failed to detect that the data was non-normal.

Let’s try one more example with exponential bins.

```
set.seed(12345)

x4_contin = rexp(1000)

y4_contin = x4_contin * exp(runif(1000, min=log(0.5), max=log(2))) + 3*runif(1000)


x4_contin = (x4_contin – mean(x4_contin)) / sd(x4_contin)

y4_contin = (y4_contin – mean(y4_contin)) / sd(y4_contin)

x_cut = min(x4_contin) + c(0.5,1,2,4)

y_cut = min(y4_contin) + c(0.5,1,2,4)


round(x_cut,2)

# [1] -0.48  0.02  1.02  3.02

round(y_cut,2)

# [1] -1.03 -0.53  0.47  2.47



plot(x4_contin, y4_contin)

abline(v=x_cut)

abline(h=y_cut)
```



Latent correlations

cor(x4_contin,y4_contin) # 0.7152

cor(x4_contin,y4_contin,method=”spearman”) # 0.5781

Polychoric

x4_ord = cut(x4_contin, c(-Inf, x_cut, Inf))

y4_ord = cut(y4_contin, c(-Inf, y_cut, Inf))

table(x4_ord, y4_ord)

99	105	183	11	0
21	56	123	45	0
3	33	104	85	1
0	2	33	59	20
0	0	0	8	9

polychor(x4_ord,y4_ord,ML=TRUE,std.err=TRUE)

#Polychoric Correlation, ML est. = 0.6688 (0.02182)

#Test of bivariate normality: Chisquare = 48.03, df = 15, p = 2.519e-05

Row Thresholds

Threshold Std.Err.

#1 -0.2707 0.03957

#2 0.3434 0.04026

#3 1.1220 0.05072

#4 2.1960 0.09929

Column Thresholds

Threshold Std.Err.

#1 -1.1360 0.05041

#2 -0.4750 0.04056

#3 0.6973 0.04375

#4 1.9430 0.08171

Compare to Pearson, Spearman correlation on binned data

cor(as.numeric(x4_ord), as.numeric(y4_ord)) # 0.5747

cor(as.numeric(x4_ord), as.numeric(y4_ord), method="spearman") # 0.5600

This was by far the most challenging case for the `polychor()` function because a row and column are nearly empty, and several cells are completely empty. Some of the cutoff estimates are off, especially in the tail where there are fewer data points. The test of bivariate normality has also detected the non-normality.

However what matters most is that the estimated correlation coefficient is a reasonable number, and it's near the middle of our target range of 0.5781 to 0.7182, which we got from the unbinned correlations. By comparison, classic correlation methods missed the range by a little bit. that's the measurement that matters. So ultimately, `polychor()` did the job, even in far from ideal conditions, so it's worth considering the next time you're dealing with ordinal data.