IDENTIFYING MULTIVARIATE OUTLIERS USING MAHALANOBIS DISTANCE

2024-01-01

#Identifying multivariate outliers in your data can be helpful #if you want to determine whether the correlations between multiple #values or variables are unusually strong for individual cases/customers/participants

STEP 1

#Identify what variables are in linear combination. #This could be, for example, a group of independent #variables used in a multiple linear regression or #a group of dependent variables used in a MANOVA. #Usually, this will include your predictor variable, #any outcome variables, and any mediators or moderators.

```
data  <- read_csv("Cues to Infidelity - MEN ONLY 7.12.23.csv")
```

```
## Rows: 239 Columns: 554
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (6): StartDate, EndDate, RecordedDate, M_SC_MRSI_1, M_SC_MRSI_4_2, M_S...
## dbl (462): USE, CUES, SEX_CUES, EMO_CUES, MISC_CUES, REACT, SUS, SOI, VAI_A_...
## lgl  (86): Q566_1, Q566_2, W_PDIS_A_1, W_PDIS_A_2, W_PDIS_A_3, W_PDIS_A_4, W...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

STEP 2

#Create a dataframe containing all of the variables you identified in step 1.

```
df <- data.frame(data$REACT, data$SUS, data$SOI, data$SMIRB)
```

```
head(df, 10)
```

```
##    data.REACT data.SUS  data.SOI data.SMIRB
## 1       5.375 1.066667 0.8888889   1.000000
## 2       4.625 1.133333 2.4444444   1.857143
## 3       4.625 1.066667 1.8888889   2.428571
## 4       5.125 1.533333 1.7777778   1.000000
## 5       5.500 1.666667 1.5555556   4.142857
## 6       3.750 1.400000 1.1111111   1.000000
## 7       1.750 1.400000 1.0000000   1.285714
## 8       5.125 1.600000 2.1111111   2.000000
## 9       6.375 2.400000 3.4444444   2.428571
## 10      4.875 3.133333 4.5555556   3.000000
```

STEP 3

#Use the mahalanobis() function in R to calculate the distance for each observation:

```r
df$mah <- mahalanobis(df, colMeans(df), cov(df))
```

```r
head(df, 10)
```

```
##    data.REACT data.SUS  data.SOI data.SMIRB        mah
## 1       5.375 1.066667 0.8888889   1.000000  2.4467709
## 2       4.625 1.133333 2.4444444   1.857143  1.1614183
## 3       4.625 1.066667 1.8888889   2.428571  2.0668241
## 4       5.125 1.533333 1.7777778   1.000000  1.5541001
## 5       5.500 1.666667 1.5555556   4.142857  8.5213431
## 6       3.750 1.400000 1.1111111   1.000000  2.9889809
## 7       1.750 1.400000 1.0000000   1.285714 11.0081866
## 8       5.125 1.600000 2.1111111   2.000000  0.1157724
## 9       6.375 2.400000 3.4444444   2.428571  5.6029036
## 10      4.875 3.133333 4.5555556   3.000000 10.9619960
```

STEP 4

#Calculate p-values for each distance using chi-square

```r
df$pvalue <- pchisq(df$mah, df=3, lower.tail = FALSE)
```

```r
head(df, 10)
```

```
##    data.REACT data.SUS  data.SOI data.SMIRB        mah     pvalue
## 1       5.375 1.066667 0.8888889   1.000000  2.4467709 0.48498762
## 2       4.625 1.133333 2.4444444   1.857143  1.1614183 0.76227187
## 3       4.625 1.066667 1.8888889   2.428571  2.0668241 0.55865358
## 4       5.125 1.533333 1.7777778   1.000000  1.5541001 0.66984151
## 5       5.500 1.666667 1.5555556   4.142857  8.5213431 0.03638068
## 6       3.750 1.400000 1.1111111   1.000000  2.9889809 0.39332723
## 7       1.750 1.400000 1.0000000   1.285714 11.0081866 0.01168169
## 8       5.125 1.600000 2.1111111   2.000000  0.1157724 0.98987971
## 9       6.375 2.400000 3.4444444   2.428571  5.6029036 0.13261177
## 10      4.875 3.133333 4.5555556   3.000000 10.9619960 0.01193316
```

STEP 5

#Identify cases where $p < .001$ and consider removing these from your data.

```r
head(df[order(df$pvalue),], 10)
```

```
##     data.REACT data.SUS  data.SOI data.SMIRB      mah       pvalue
## 43       4.875 1.600000 1.1111111   5.285714 20.34492 0.0001439733
## 95       4.500 3.933333 4.5555556   3.285714 19.20449 0.0002480300
## 238      1.000 1.066667 1.5555556   2.714286 17.49418 0.0005591832
## 16       1.000 1.066667 0.7777778   1.857143 16.72444 0.0008052121
## 157      5.375 3.866667 1.8888889   4.166667 16.51441 0.0008893148
## 151      1.000 1.200000 1.0000000   2.000000 16.15714 0.0010528885
## 210      1.000 1.200000 1.2222222   1.000000 15.89539 0.0011913789
## 79       1.000 1.066667 2.1111111   1.000000 15.12760 0.0017107888
## 163      5.500 3.800000 2.5555556   4.142857 14.77238 0.0020218705
## 204      4.500 3.333333 1.0000000   4.142857 14.05326 0.0028335507
```

```r
df_no_multi_outliers <- df[-(which(df$pvalue < .001)),]
```

```r
head(df_no_multi_outliers, 10)
```

```
##    data.REACT data.SUS  data.SOI data.SMIRB        mah     pvalue
## 1       5.375 1.066667 0.8888889   1.000000  2.4467709 0.48498762
## 2       4.625 1.133333 2.4444444   1.857143  1.1614183 0.76227187
## 3       4.625 1.066667 1.8888889   2.428571  2.0668241 0.55865358
## 4       5.125 1.533333 1.7777778   1.000000  1.5541001 0.66984151
## 5       5.500 1.666667 1.5555556   4.142857  8.5213431 0.03638068
## 6       3.750 1.400000 1.1111111   1.000000  2.9889809 0.39332723
## 7       1.750 1.400000 1.0000000   1.285714 11.0081866 0.01168169
## 8       5.125 1.600000 2.1111111   2.000000  0.1157724 0.98987971
## 9       6.375 2.400000 3.4444444   2.428571  5.6029036 0.13261177
## 10      4.875 3.133333 4.5555556   3.000000 10.9619960 0.01193316
```