Simulate Data for Micro Scenario-Analysis and Visualisation

Philipp Brauner

Abstract

This Quarto notebook is designed to generate synthetic data using the `faux` package, providing an illustration for the analysis and visualization of micro-scenarios. The generated dataset closely resembles datasets exported from Qualtrics and its Loop & Merge function. The data incorporates pre-defined characteristics, including well-defined means and correlations within and between the evaluation dimensions.

Introduction

This notebook generates synthetic data to demonstrate the analysis of a micro-scenario based study. The corresponding analysis notebook is located in the same folder. For detailed information on this approach and guidance on designing and analysing studies, please refer to the main article. You can find and cite the main article here:.

> Mapping Acceptance: Assessing Emerging Technologies and Concepts through Micro Scenarios, Philipp Brauner, http://arxiv.org/abs/2402.01551 (2024)

The general concept behind this simulated dataset is to mimic the data export from typical survey software systems, like Qualtrics. However, the data is clean, devoid of additional variables, speeders, or erroneous inputs requiring cleaning. Furthermore, the dataset is structured to exhibit the desired properties of a micro-scenario-based survey, showcasing specific patterns among topics and a defined correlation pattern for evaluation dimensions.

For generating synthetic data, we use the `faux` package, and guidance for this can be found in the package vignette.

Load libraries

In the analysis, we rely on the `tidyverse` and `ggplot` packages. Additionally, for generating synthetic data with specific properties (e.g., pre-defined means and correlations between variables), we utilize the `faux` package.

```r
library(faux)    # create simulated data based on given properties
```

```

************
Welcome to faux. For support and examples visit:
https://debruine.github.io/faux/
- Get and set global package options with: faux_options()
************
```

```r
library(matrixcalc)
library(tidyverse)
```

```
Warning: package 'lubridate' was built under R version 4.3.3
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.2
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
library(scales)  # format_percent
```

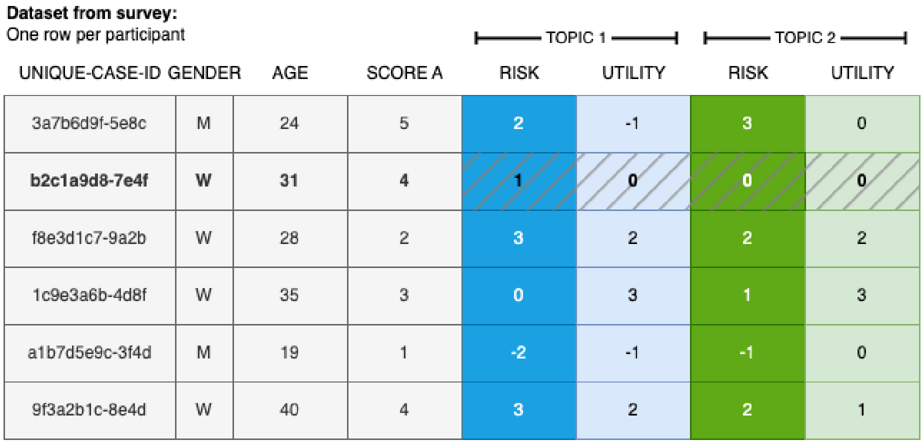
```

Attaching package: 'scales'

The following object is masked from 'package:purrr':
    discard

The following object is masked from 'package:readr':

    col_factor
```

```
library(ggplot2) # graphics
library(ggrepel) # label placement in the scatter plot
library(knitr)   # Tables
```

```
Warning: package 'knitr' was built under R version 4.3.3
```

Create Synthetic Data

In this section, we generate synthetic data that simulates properties akin to real survey data. Initially, we create a dataset resembling data from the survey tool Qualtrics. Figure Figure 1 illustrates the structure of a standard dataset from survey tools, where each row represents the responses from a single participant.



Dataset from survey:
One row per participant

				TOPIC 1		TOPIC 2	
UNIQUE-CASE-ID	GENDER	AGE	SCORE A	RISK	UTILITY	RISK	UTILITY
3a7b6d9f-5e8c	M	24	5	2	-1	3	0
b2c1a9d8-7e4f	**W**	**31**	**4**	**1**	**0**	**0**	**0**
f8e3d1c7-9a2b	W	28	2	3	2	2	2
1c9e3a6b-4d8f	W	35	3	0	3	1	3
a1b7d5e9c-3f4d	M	19	1	-2	-1	-1	0
9f3a2b1c-8e4d	W	40	4	3	2	2	1

Figure 1: Illustration of typical survey data utilizing the micro-scenario approach, incorporating user demographics, additional user factors, and topic evaluations.

The generated dataset will include several variables. Initially, a unique user identifier (id) is assigned, followed by a user variable (e.g., attitude towards a topic). Subsequently, N=*12* (adjustable) topic assessments are included, with two variables for each evaluation dimension for each topic. In this instance, *perceived risk* and *perceived utility* are used as examples for topic evaluations. However, one can utilize different or additional evaluation dimensions (as detailed in the article).

For the synthetic data, we specify N topics and two evaluation dimensions, defining that they should be inversely correlated with an intercept.

```
N <- 12  # number of topics to simulate
TOPICS <- read.csv2("matrixlabels.csv")

# Generate mean evaluations A and B, inversly correlated and with some noise, and an interce

evaluationA <- seq(-0.75, 0.50, length.out = N)
evaluationB <- seq(-0.75, 0.10, length.out = N)

# Add some random noise
for (i in 1:N) {
  evaluationA[i] <- evaluationA[i] + rnorm(1, mean=0, sd=0.05)
  evaluationB[i] <- evaluationB[i] - rnorm(1, mean=0, sd=0.05)
}

# Purposefully defined outlier
evaluationA[5] <- -1/3
evaluationB[5] <- +1/4
combinedMeans = c(evaluationA, evaluationB)
```

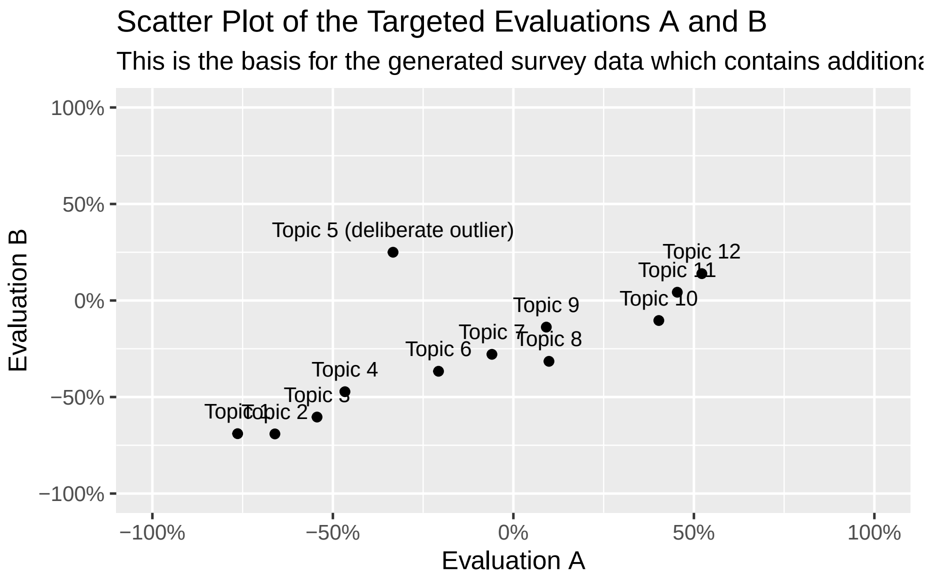


Figure 2: Target topic evaluations. Noise due to random sampling will be added in a later step.

The variables for the topic evaluations must adhere to a standardized naming scheme, i.e., `a01_matrix_02`, where `01` represents the ID of the queried topic, `02` stands for the queried evaluation dimension, and `matrix` denotes the name of the variable block in the survey tool. This is the naming scheme used by Quartics.

```r
varnames <- c(paste0("a", 1:N, "_matrix_1"), paste0("a", 1:N, "_matrix_2"))
```

Additionally, we specify that the evaluations of the first and second evaluation dimensions are strongly correlated within topics and negatively correlated between topics. The data is randomized to approximately, but not exactly, align with this scheme, simulating a scenario as if it were randomly drawn from a population parametrized in this manner.

```r
# Generate correlation matrix with random correlations in the specified range
generate_cor_matrix <- function(n, range = c(0.2, 0.6)) {
  corr_values <- matrix(runif(n^2, range[1], range[2]), nrow = n)
  for (i in 1:n) {
    for(j in 1:n) {
      corr_values[i,j] = max(-1, min(1, corr_values[i,j])) # check bounds
    }
  }
  for (i in 1:n) {
    for(j in 1:n) {
      corr_values[i,j] = corr_values[j,i] # symmetric please
    }
  }
  diag(corr_values) <- 1  # Set diagonal to 1
  corr_values
}

# Generating covariance matrices for variables A and B
cor_matrix_A <- generate_cor_matrix(N)
cor_matrix_B <- generate_cor_matrix(N, range = c(0.3, 0.5))

# Generating negative correlations between variables A and B
negative_corr <- -.25     # r between A and B
cor_matrix_AB <- matrix(negative_corr, nrow = N, ncol = N)

# Combining covariance matrices for A, A and B, and B
# Must be symmetric and positive-definite
cor_matrix <- matrix(0, nrow = 2 * N, ncol = 2 * N)
cor_matrix[1:N, 1:N] <- cor_matrix_A # upper-left
cor_matrix[(N + 1):(2 * N), (N + 1):(2 * N)] <- cor_matrix_B # lower-right
cor_matrix[1:N, (N + 1):(2 * N)] <- cor_matrix_AB
cor_matrix[(N + 1):(2 * N), 1:N] <- t(cor_matrix_AB) #transposed for neg. covariances
```

Check the validity of the data

Before proceeding, let's perform a check. Why? We require a symmetric and positive-definite matrix to generate random data based on the defined parameters. A correlation matrix is, by definition, symmetric and positive semi-definite. If we encounter a correlation matrix that is not positive semi-definite, it could be due to numerical precision issues or errors in computation, particularly when dealing with very large matrices or near-linear dependencies between variables. If this occurs, repeat the steps above—it should work out after a few tries.

Request to the readers: Any suggestions on how to address this issue are appreciated. However, this step is not necessary if you already have real survey data, as it specifically pertains to the creation of synthetic data.

Now, let's create the synthetic data based on the previously defined means and correlation parameters. Name the variables following the required schema. We provide population parameters and draw a random sample based on these.

```r
data <- rnorm_multi(
  varnames = varnames,
  n = 100, # sample size, i.e., number of participants in the survey
  mu = combinedMeans,
  sd = 0.25,
  r = cor_matrix,
  empirical = FALSE
)
```

Recode data

Finally, we record the data to resemble typical survey data. Firstly, ensure that the bounds are met (e.g., values between −1 and +1; due to random sampling, this may not always be the case). Next, convert the percent scores to the typical survey data domain, i.e., discrete values from 1 to 7.

```
data <- as.data.frame(data) %>%
  dplyr::mutate(across(everything(), ~ ifelse(. < -1, -1, ifelse(. > 1, 1, .)))) %>%
  dplyr::mutate(across(everything(), ~ (.+1)/2)) %>%
  dplyr::mutate(across(everything(), ~ (6*.))) %>%
  dplyr::mutate(across(everything(), ~ 1 + round(.)))
```

Ultimately, let's add a participant ID and a simulated user variable, calculated from the already simulated data and therefore strongly correlated.

```
data <- data %>%
  dplyr::mutate(id = paste0("fakeparticipantid-", row_number())) %>%
  dplyr::mutate(uservariable = rnorm_pre(a1_matrix_1+a2_matrix_1, mu = 10, sd = 2, r = 0.5))
  dplyr::relocate(id, uservariable)
```

Save data

Finally, we save the generated data to an RDS file.

```
saveRDS(data, "syntheticdata.rds")
```

That file can be used for the actual analysis and visualization (this main notebook utilizes this data).