

Preparing your Data for SEM Estimation

Basic Steps

Lilian Kojan and André Calero Valdez

updated: 2021-07-12

Data preparation

1. Data requirements
2. Recoding variables
3. Treating missing values
4. Renaming variables

Data preparation steps

Basic steps:

- Recoding variables
- Treating missing data
- Renaming variables

Advanced steps:

- Examining data distribution
- Removing low quality responses
- Treating outliers

Example data

Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment
7	rather agree	5	6	NA
10	strongly agree	2	10	10
7	rather agree	4	8	7
7	strongly agree	6	10	NA
8	rather agree	1	10	8
10	agree	4	8	NA

Recoding variables: Numerical

Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment
7	rather agree	5	6	NA
10	strongly agree	2	10	10
7	rather agree	4	8	7
7	strongly agree	6	10	NA
8	rather agree	1	10	8
10	agree	4	8	NA

Recoding variables: Data type

Data should be

- numerical

But also...

Recoding variables: Data type

Data should be

- numerical

But also...

... approximately equidistant



Recoding variables: Data type

Data should be

- numerical

But also...

... approximately equidistant (i.e., not scaled like this)



Recoding variables: Data type

Data should be

- numerical
- ordinal scaled
- and the scale should be approximately equidistant

Recoding variables: Data type

Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment
7	rather agree	5	6	NA
10	strongly agree	2	10	10
7	rather agree	4	8	7
7	strongly agree	6	10	NA
8	rather agree	1	10	8
10	agree	4	8	NA

Recoding variables: Data type

```
df <- df %>%  
  dplyr::mutate(  
    `Expectation Products` =  
      dplyr::recode(  
        `Expectation Products`,  
        "rather agree" = 7,  
        "agree" = 9,  
        "strongly agree" = 10  
      )  
  )  
  
# because we gave the new variable the same name,  
# it replaces the old variable  
  
# use mutate(across(v1:v3), fnc) to recode variables v1 to v3 using fnc
```

Recoding variables: Data type

Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment
7	7	5	6	NA
10	10	2	10	10
7	7	4	8	7
7	10	6	10	NA
8	7	1	10	8
10	9	4	8	NA

Recoding variables: Direction

Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment
7	7	5	6	NA
10	10	2	10	10
7	7	4	8	7
7	10	6	10	NA
8	7	1	10	8
10	9	4	8	NA

Recoding variables: Direction

Expectation			Satisfaction		
Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment	
7	7	5	6	NA	
10	10	2	10	10	
7	7	4	8	7	
7	10	6	10	NA	
8	7	1	10	8	
10	9	4	8	NA	

Recoding variables: Direction

```
df <- df %>%  
  mutate(  
    `Expectation Products` = dplyr::recode(  
      `1` = 10,  
      `2` = 9,  
      `3` = 8,  
      `4` = 7,  
      `5` = 6,  
      `6` = 5,  
      `7` = 4,  
      `8` = 3,  
      `9` = 2,  
      `10` = 1  
    )  
)
```

assign changes to existing data frame
add new variable based on existing variables
name for the new variable
replace values
old value = new value

Recoding variables: Direction

Quicker option:

```
# Reverse scale using mutate() with subtraction:  
# For a scale ranging from 1 to x: x + 1 - scale  
# For a scale ranging from 0 to x: x - scale  
df <- df %>%  
  mutate(  
    `Problem Expectation` = 11 - `Problem Expectation`  
  )
```

Recoding variables: Direction

Expectation			Satisfaction		
Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment	
7	7	6	6	NA	
10	10	9	10	10	
7	7	7	8	7	
7	10	5	10	NA	
8	7	10	10	8	
10	9	7	8	NA	

Treating missing values

Expectation			Satisfaction	
Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment
7	7	6	6	NA
10	10	9	10	10
7	7	7	8	7
7	10	5	10	NA
8	7	10	10	8
10	9	7	8	NA

Treating missing values

Expectation			Satisfaction	
Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment
7	7	6	6	NA
10	10	9	10	10
7	7	7	8	7
7	10	5	10	NA
8	7	10	10	8
10	9	7	8	NA

Treating missing values

Expectation			Satisfaction		
Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment	
7	7	6	6	8	
10	10	9	10	10	
7	7	7	8	7	
7	10	5	10	8	
8	7	10	10	8	
10	9	7	8	8	

Treating missing values

Expectation			Satisfaction		
Quality Expectation	Expectation Products	Problem Expectation	Satisfaction Overall	Expectation Fulfillment	
7	7	6	6	NA	
10	10	9	10	10	
7	7	7	8	7	
7	10	5	10	NA	
8	7	10	10	8	
10	9	7	8	NA	

Treating missing values

- Impute missing data
- Remove variables containing missing data (Hair et al., 2017)
- Ignore missing data

Treating missing values

- Impute missing data
- Remove variables containing missing data (Hair et al., 2017)
- Ignore missing data

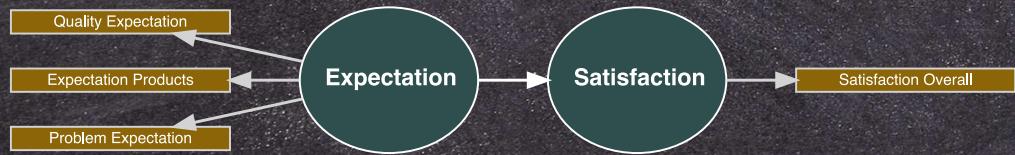
```
# remove variable with missing data
df <- df %>%
  select(!'Expectation Fulfillment')
```

Renaming variables

Expectation			Satisfaction	
Quality Expectation	Expectation Products	Problem Expectation	Satisfaction	Overall
7		7	6	6
10		10	9	10
7		7	7	8
7		10	5	10
8		7	10	10
10		9	7	8

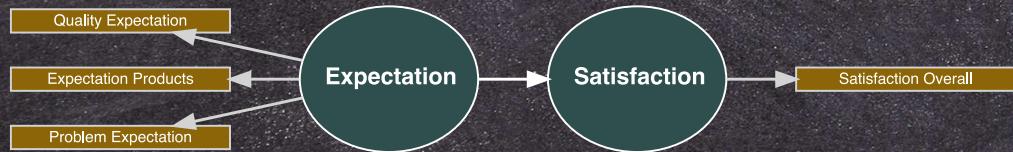
Renaming variables

Long variable names...

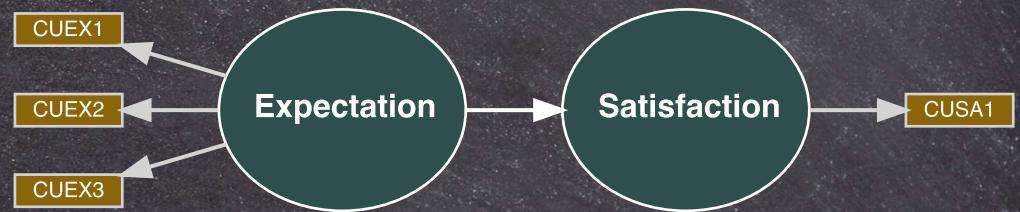


Renaming variables

Long variable names...



... vs. abbreviated names



Renaming variables

Long variable names...

```
measurement_model <- constructs(      # define measurement model
  reflective(                      # define a reflective construct
    construct_name = "Expectation", # construct name
    item_names = c(                 # item names = df variable names
      "Quality Expectation",
      "Expectation Products",
      "Problem Expectation"
    )
  ),
  reflective(
    construct_name = "Satisfaction",
    item_names = c("Expectation Fulfillment")
  )
)
```

Renaming variables

... vs. abbreviated names

```
measurement_model <- constructs(  
  reflective(  
    construct_name = "Expectation",  
    item_names = multi_items("CUEX", 1:3)      # calls variables with same pre  
,  
  reflective(construct_name = "Satisfaction",  
    item_names = "CUSA1")  
)
```

Renaming variables

... vs. abbreviated names

```
measurement_model <- constructs(  
  reflective(  
    construct_name = "Expectation",  
    item_names = multi_items("CUEX", 1:3)  
  ),  
  reflective(construct_name = "Satisfaction",  
    item_names = "CUSA1")  
)
```

Renaming variables

Rename variables associated with the same construct with the same prefix, e.g. for *CUSA* for *Customer Satisfaction*

```
df <- df %>%
  rename("CUSA1" = "Satisfaction Overall") # new name = old name
```

Renaming variables

Rename variables associated with the same construct with the same prefix, e.g. for *CUSA* for *Customer Satisfaction*

```
df <- df %>%  
  rename("CUSA1" = "Satisfaction Overall") # new name = old name
```

```
df <- df %>%  
  rename_with(~ paste0("CUEX", 1:3), # function to generate new name  
             .cols = c(1:3))          # apply to columns 1 to 3
```

Summary

CUEX1	CUEX2	CUEX3	CUSA1
7	7	6	6
10	10	9	10
7	7	7	8
7	10	5	10
8	7	10	10
10	9	7	8

- Data is numerical and unidirectional
- There are no missing values
- Variables are named for use in SEMinR

Sources for this video

Hair, J. F., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2017). A primer on partial least squares structural equation modeling (PLS-SEM) (Second edition). Sage.