GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Dr. Christian Czymara

FORSCHUNGSPRAKTIKUM I UND II: LÄNGSSCHNITTDATENANALYSE IN R

Preparing panel data
session iv

AGENDA

- Today we will get to know the nature of panel data
- Different ways to organize it
- And various ways to combine different data sets in R

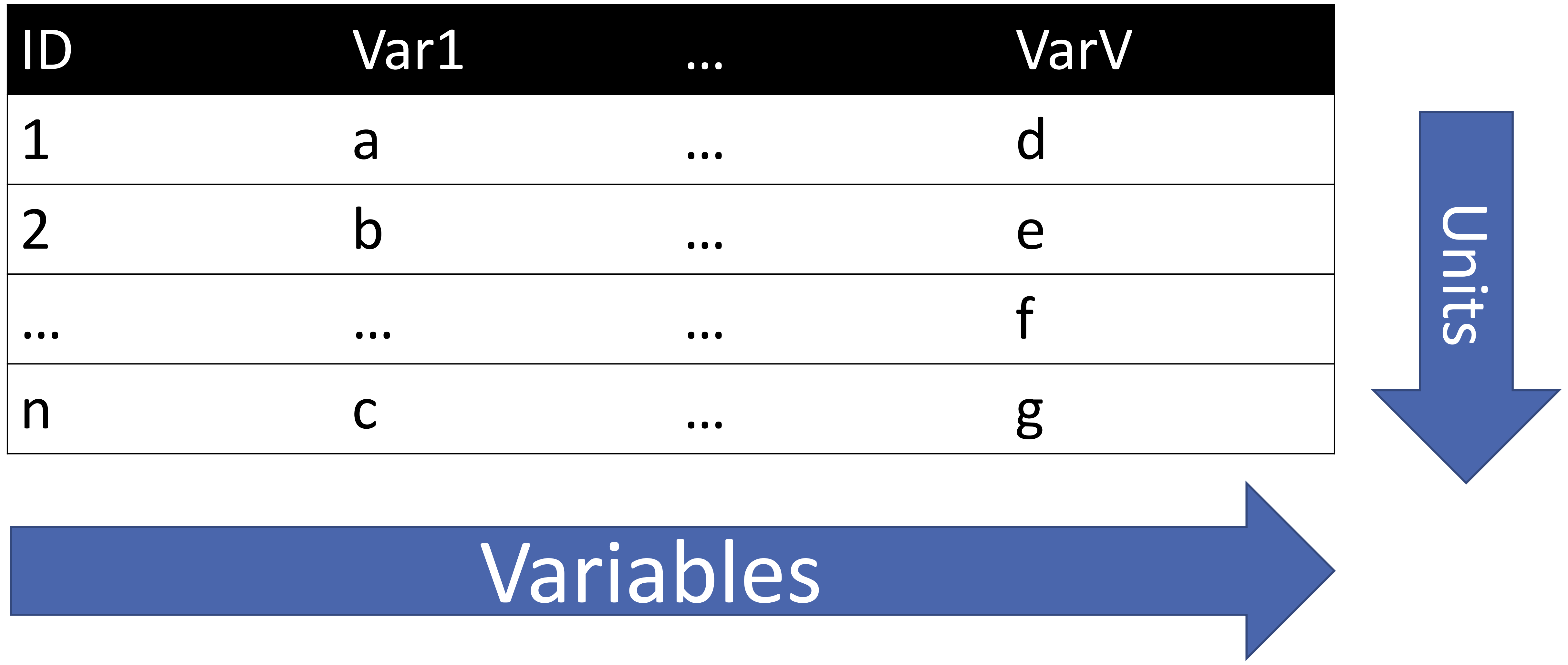
THE NATURE OF PANEL DATA

PANEL DATA

- Panel data have a three-dimensional structure
  - Units ($i = 1, \dots, n$): E. g. persons
  - Variables ($v = 1, \dots, V$): E. g. poverty status
  - Time-points or *waves* ($t = 1, \dots, T$): E. g. 2020

PANEL DATA

- How can you organize three-dimensional data space in a two-dimensional dataset?
- Cross-sectional dataset with n units and v variables:



ID	Var1	…	VarV
1	a	…	d
2	b	…	e
…	…	…	f
n	c	…	g

PANEL DATA

- Two panel waves; each with n units and v variables:

ID	Var1	…	VarV
1	a	…	d
2	b	…	e
…	…	…	f
n	c	…	g

ID	Var1	…	VarV
1	a	…	d
2	b	…	e
…	…	…	f
n	c	…	g

PANEL DATA

- Time is a relevant information

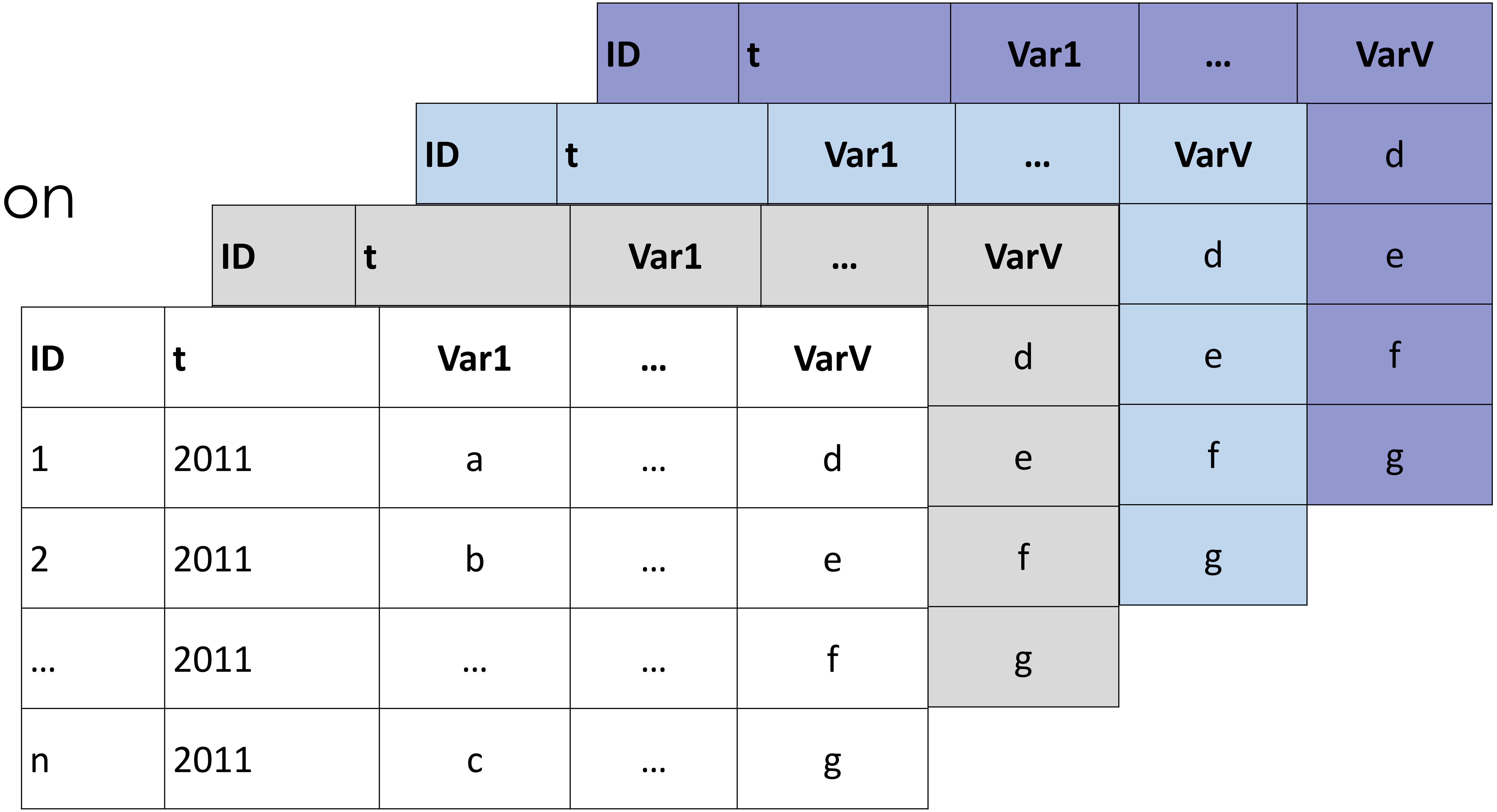
ID	t	Var1	…	VarV
1	2011	a	…	d
2	2011	b	…	e
…	2011	…	…	f
n	2011	c	…	g

ID	t	Var1	…	VarV
1	2012	a	…	d
2	2012	b	…	e
…	2012	…	…	f
n	2012	c	…	g

THE PANEL DATA CUBE

- Time adds a third dimension

→Panel data are cubic



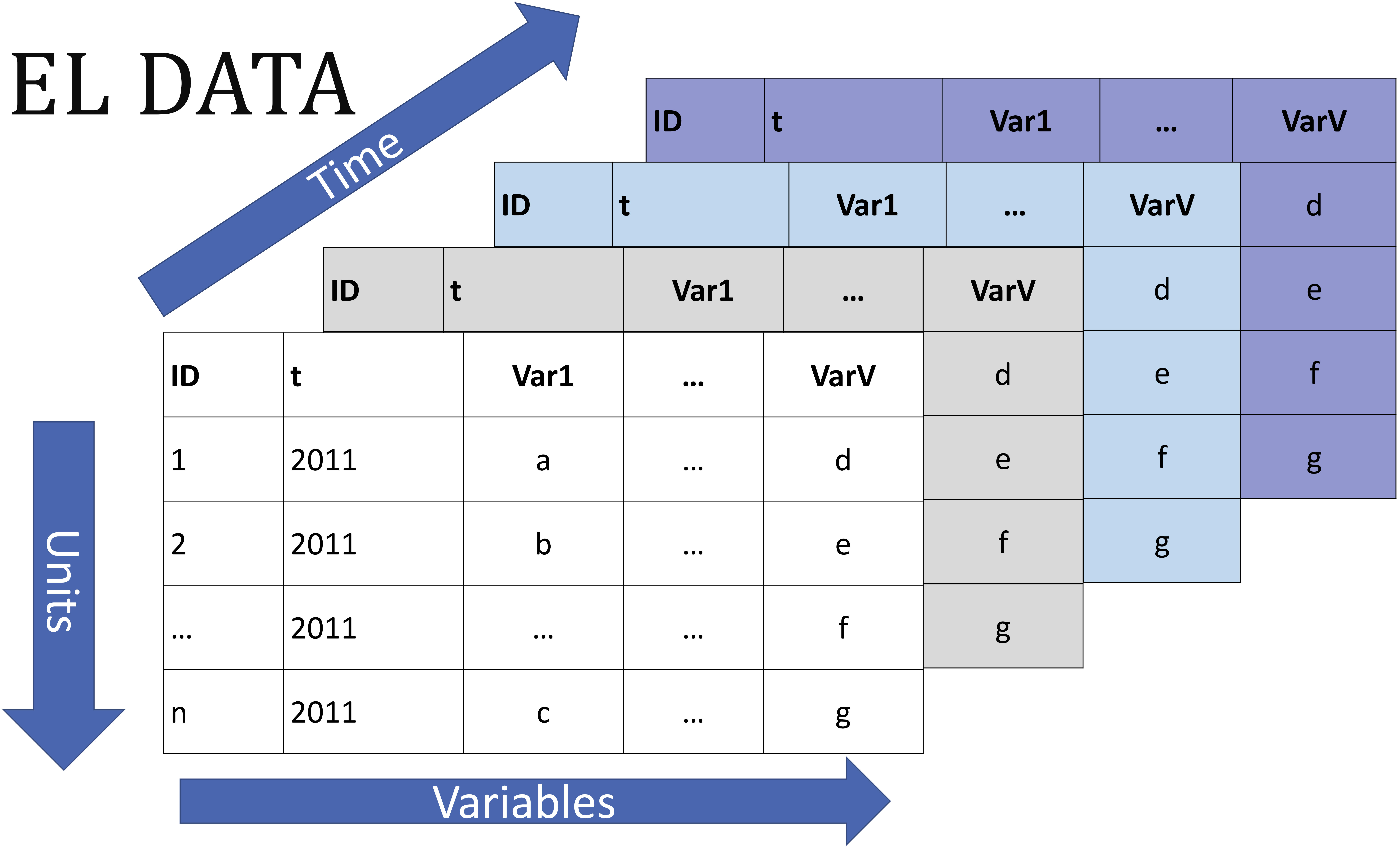
ID	t	Var1	...	VarV
1	2011	a	...	d
2	2011	b	...	e
...	2011	...	...	f
n	2011	c	...	g

ID	t	Var1	...	VarV
				d
				e
				f
				g

ID	t	Var1	...	VarV
				d
				e
				f
				g

ID	t	Var1	...	VarV
				d
				e
				f
				g

PANEL DATA



ID	t	Var1	…	VarV
1	2011	a	…	d
2	2011	b	…	e
…	2011	…	…	f
n	2011	c	…	g

ID	t	Var1	…	VarV
				d
				e
				f
				g

ID	t	Var1	…	VarV
				d
				e
				f
				g

ID	t	Var1	…	VarV
				d
				e
				f
				g

WIDE OR LONG?

WIDE AND LONG FORMAT

- Three dimensional panel data can be organized in a two-dimensional matrix in two ways
- Wide format
  - Rows ≙ units
  - Repeated measurements as separate variables
  - n rows and $t * v$ columns
- Long format
  - Rows ≙ single measurements
  - $n * t$ rows and v columns

WIDE FORMAT

ID	Gender	Poor_2012	Poor_2014	Poor_2016
1	0	0	0	1
2	1	1	0	0
…	…	…	…	…
999	1	1	1	1
1000	0	0	0	0

- Time dimension integrated in columns
- Variable names need to indicate time-point of measurement

LONG FORMAT

ID	Year	Poor
1	2012	0
1	2014	0
1	2016	1
…	…	…
1000	2012	0
1000	2014	0
1000	2016	0

- Time dimension integrated in rows
- Dataset needs a variables indicating time point at which information has been recorded

WIDE VS LONG FORMAT

ID	Poor_2012	Poor_2014	Poor_2016
1	0	0	1
2	1	0	0
…	…	…	…
999	1	1	1
1000	0	0	0

ID	Year	Poor
1	2012	0
1	2014	0
1	2016	1
…	…	
1000	2012	0
1000	2014	0
1000	2016	0

WIDE VS LONG FORMAT

- Most methods require long format
- Wide format better for analyzing associations of repeated measurements
- Wide format also demonstrates that measurements are not independent
- Hierarchical data structure; repeated measurements nested in units (e. g. person-years)

WIDE VS LONG IN R

- One way to wide and long transform data is provided by the `tidyr` package
  - From wide to long: `gather()`
  - From long to wide: `spread()`
- In the context of panel data, however, working with the `panelr` package is easier
  - First, declare the panel structure of the data using the `panel_data()` function, e.g.: `panel_data(pcspoverty, id = ID, wave = year)`
  - From wide to long: `long_panel()`
  - From long to wide: `widen_panel()`

LONG_PANEL()

- `long_panel(wide_data, prefix = "_", periods = c(2012, 2014, 2016), label_location = "end")`

ID	Poor_2012	Poor_2014	Poor_2016
1	0	0	1
2	1	0	0
…	…	…	…
999	1	1	1
1000	0	0	0

ID	year	Poor
1	2012	0
1	2014	0
1	2016	1
…	…	
1000	2012	0
1000	2014	0
1000	2016	0

WIDEN_PANEL()

- `widen_panel(long_data, separator = "_")`
- Both commands only work when information on the person and time identifiers was already provided with `panel_data()`

ID	year	Poor
1	2012	0
1	2014	0
1	2016	1
…	…	
1000	2012	0
1000	2014	0
1000	2016	0

ID	Poor_2012	Poor_2014	Poor_2016
1	0	0	1
2	1	0	0
…	…	…	…
999	1	1	1
1000	0	0	0

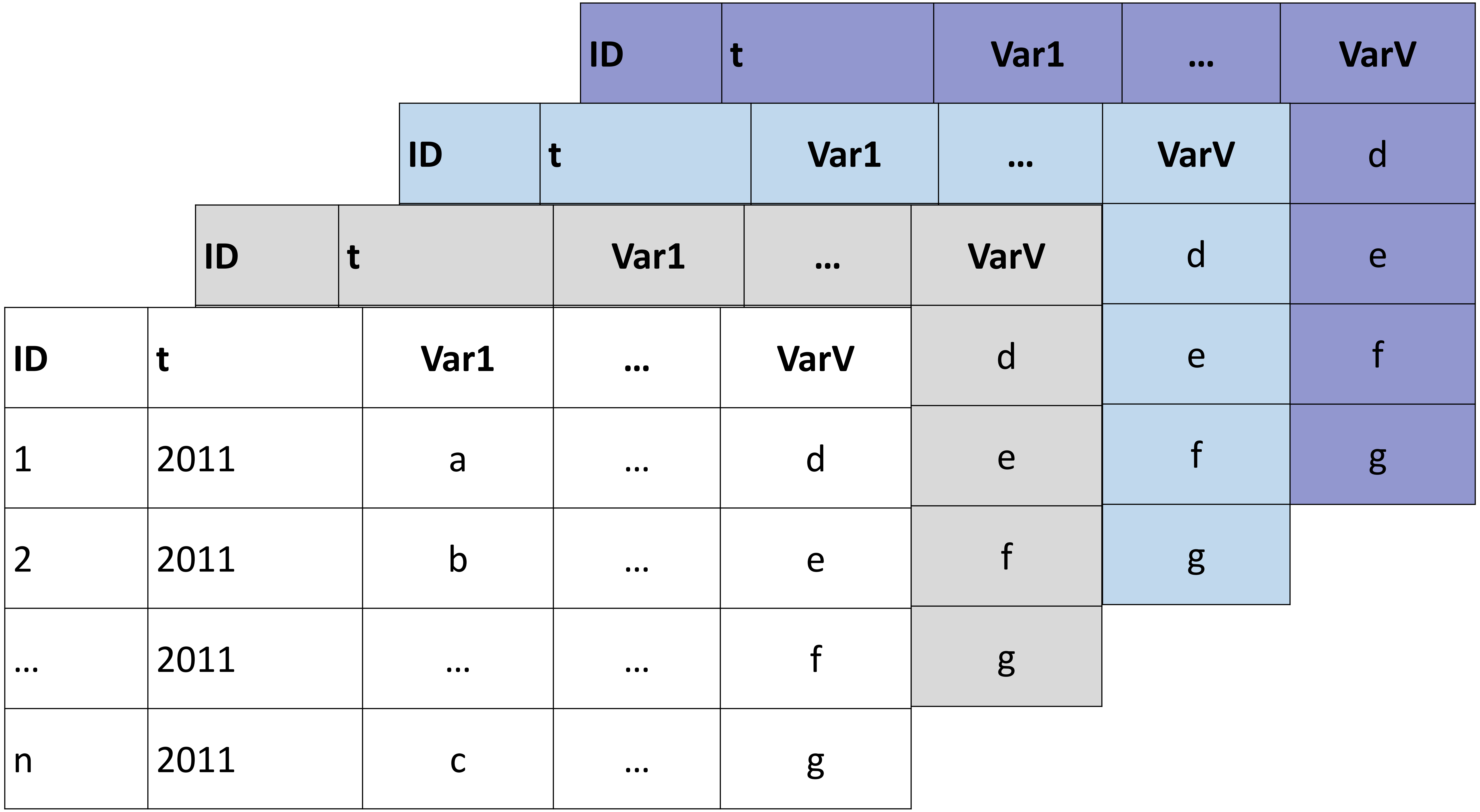
PREPARING PANEL DATA IN R

IMPORTING DIFFERENT FILE TYPES

- There are numerous ways to store data, each needs a different import function in R
- Stata's dta files: `read_dta()` (haven package)
- Excel xlsx files: `read_excel()` (readxl package)
- CSV files: `read.csv()` (base R)
- (Rdate files: `load()` (base R)
- And a lot more…

PANEL DATA MANAGEMENT

- Raw data typically provides units nested in time points
- Each new wave adds a new dataset



ID	t	Var1	…	VarV
1	2011	a	…	d
2	2011	b	…	e
…	2011	…	…	f
n	2011	c	…	g

PANEL DATA MANAGEMENT

- Which period should be analyzed? (determine t)
- Which variables are relevant? (determine v)
- What is target population? (determine n)

→Identify which datasets provide necessary information

PANEL DATA MANAGEMENT

- Moreover, data from *one* wave may be provided in several files
- For example GSOEP: individual and household questionnaires

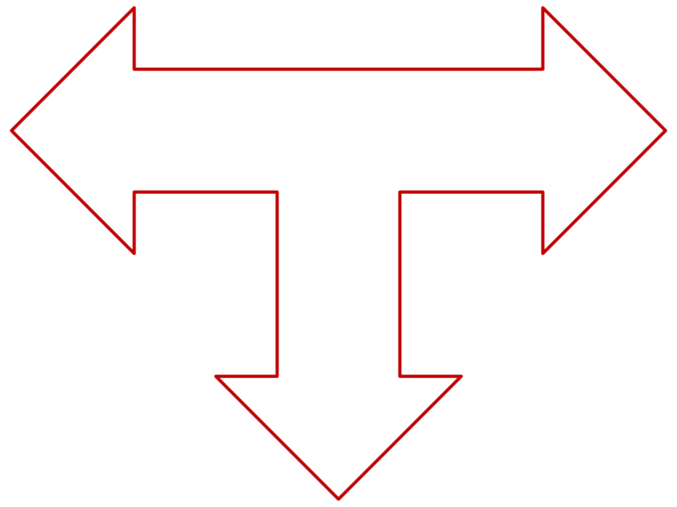
ID	HHID	t	Age	Gender
1	100	2011	36	0
2	101	2011	42	1
3	101	2011	40	0
4	102	2011	19	1

HHID	t	Income	Rent
100	2011	2200	900
101	2011	4100	1300
102	2011	1390	450

BINDING DATA

- Binding means combining rows (`rbind()`) or columns (`cbind()`) of two tables

ID	HHID	t	Age	Gender
1	100	2011	36	0
2	101	2011	42	1
3	101	2011	40	0



ID	HHID	t	Age	Gender
4	100	2011	8	1
5	101	2011	6	1

ID	HHID	t	Age	Gender
1	100	2011	36	0
2	101	2011	42	1
3	101	2011	40	0
4	100	2011	8	1
5	101	2011	6	1

BINDING ROWS

- Binding waves (in long format) means adding *rows* to an existing dataset → `rbind()`

ID	HHID	t	Age	Income
1	100	2011	36	2200
2	101	2011	42	3100
3	101	2011	40	1600

ID	HHID	t	Age	Income
1	100	2012	37	2400
2	101	2012	43	3100
3	101	2012	41	1900

ID	HHID	t	Age	Income
1	100	2011	36	2200
2	101	2011	42	3100
3	101	2011	40	1600
1	100	2012	37	2400
2	101	2012	43	3100
3	101	2012	41	1900

BINDING ROWS

ID	HHID	t	Age	Income
1	100	2011	36	2200
2	101	2011	42	3100
3	101	2011	40	1600

ID	HHID	t	Age	Income
1	100	2012	37	2400
2	101	2012	43	3100
3	101	2012	41	1900

ID	HHID	t	Age	Income
1	100	2011	36	2200
1	100	2012	37	2400
2	101	2011	42	3100
2	101	2012	43	3100
3	101	2011	40	1600
3	101	2012	41	1900

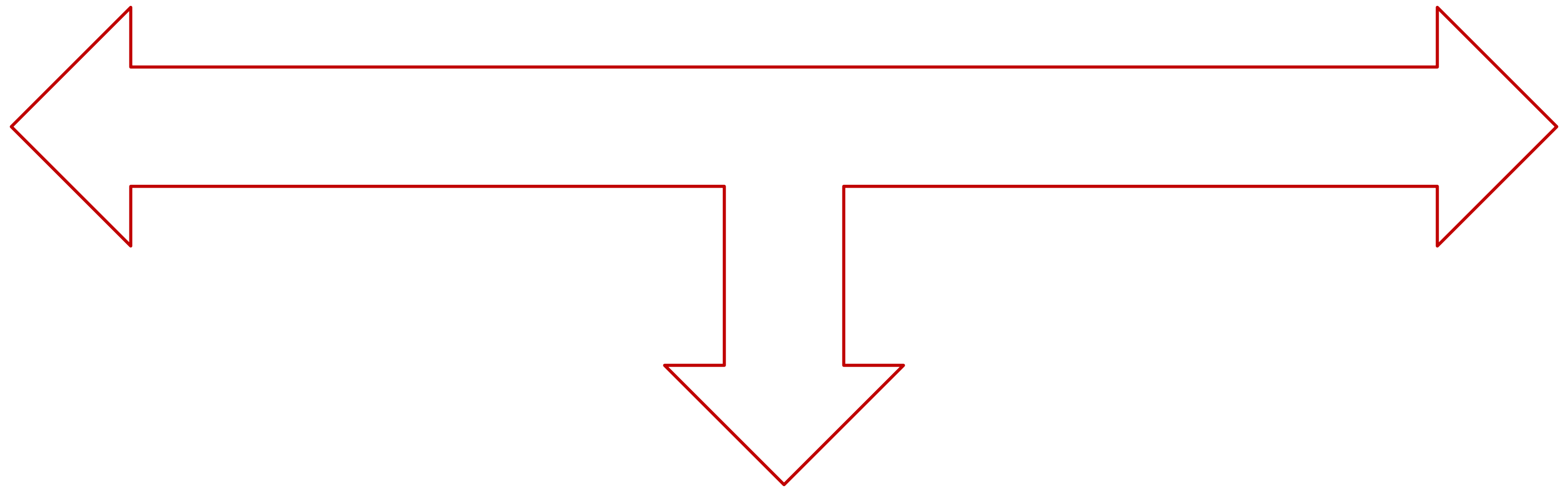
Sorted by ID (and t)

BINDING COLUMNS

- Binding variables means adding *columns* → `cbind()`

ID	HHID	t	Age
1	100	2011	36
2	101	2011	42
3	101	2011	40
1	100	2012	37
2	101	2012	43
3	101	2012	41

ID	HHID	t	Income
1	100	2011	2200
2	101	2011	3100
3	101	2011	1600
1	100	2012	2400
2	101	2012	3100
3	101	2012	1900



ID	HHID	t	Age	Income
1	100	2011	36	2200
2	101	2011	42	3100
3	101	2011	40	1600
1	100	2012	37	2400
2	101	2012	43	3100
3	101	2012	41	1900

BINDING DATA

- A drawback of `rbind()` is that it will only work when both tables have the same number of columns
- … and `cbind()` only when both data sets have the same number of rows
- Hence, `rbind()` will only work when both data sets have the exact same variables (as in the example)
- … and `cbind()` is useful when you have the exact same observations in two datasets (hardly the case)

JOIN()

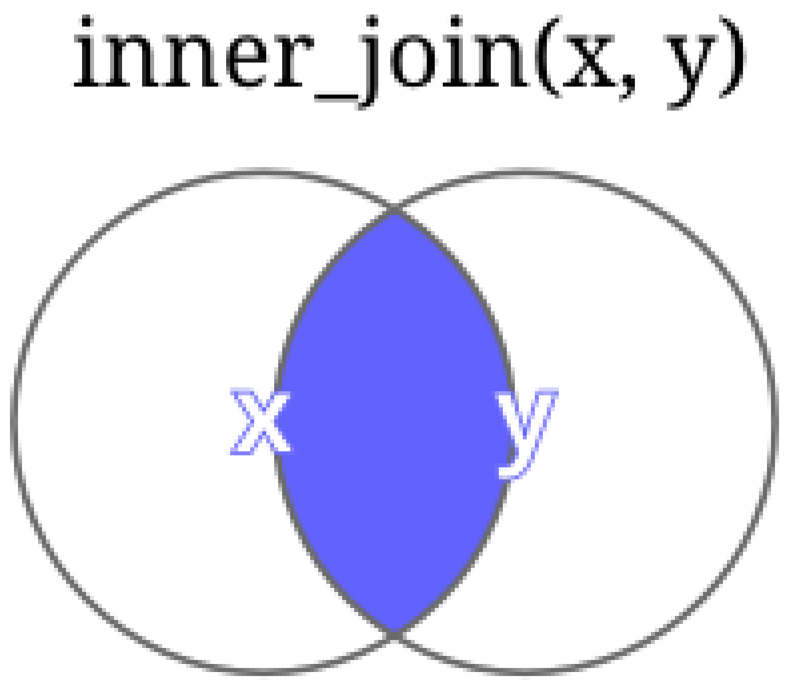
- The functions of the `join` family of the dplyr package combine two (or more) tables / data sets
- Let us call table 1 *master data*. It is the one to which we add other data (e. g.: individual-level GSOEP data)
- Table 2 should be added to data set 1, let us call it *using data* (e. g.: additional household-level GSOEP data)
- Finally, we need to know based on which column(s) we want to merge both data sets, let us call this the *key variable*
- The general syntax is: `join_type(masterData, usingData, by = keyVariable)`
- For example: `innerJoinDf <- inner_join(soep_ind, soep_hh, by = c("hid","welle"))`

`DPLYER`'S JOIN TYPES

- Inner Join (`inner_join()`): Combines observations of data 1 and 2 that are available in *both* data sets
- Left Join (`left_join()`): Adds data 2 to data 1
- Right Join (`right_join()`): Adds data 1 to data 2
- Full Join (`full_join()`): Combines observations of data 1 and 2 that are available in *either* data set
- Semi Join (`semi_join()`): Similar to `inner_join()`
- Anti Join (`anti_join()`): Only keeps observations of data 1 that are *not* available in data 2

INNER_JOIN()

- Adds master data to using data based on key variable
- Only includes observations that exist in *both data*
- E. g.: `inner_join(master, using, by = “ID”)`

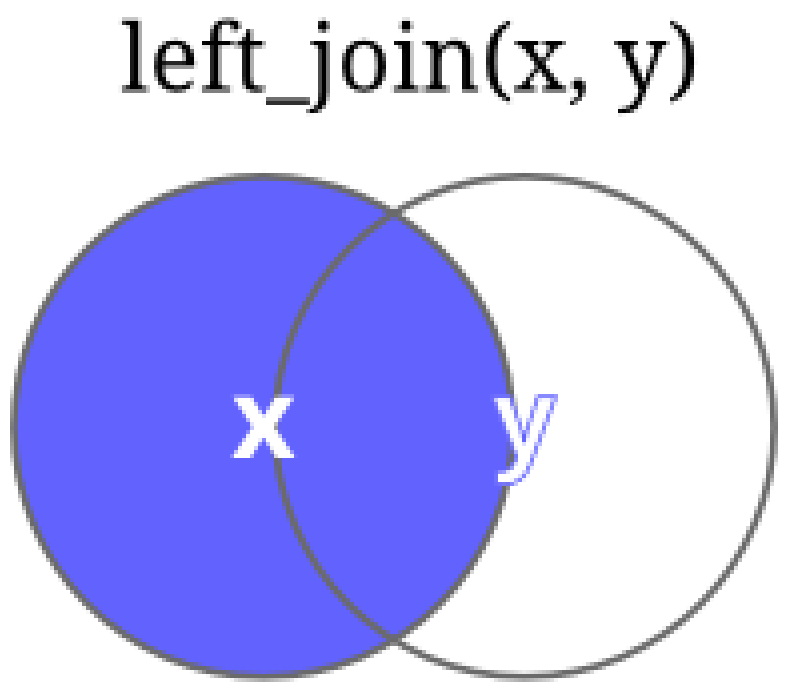


ID	Age	Gender
1	36	0
2	42	1
3	23	0

ID	Income	Rent
1	2200	900
2	4100	1300
4	3600	1200

ID	Age	Gender	Income	Rent
1	36	0	2200	900
2	42	1	4100	1300

LEFT_JOIN()


left_join(x, y)

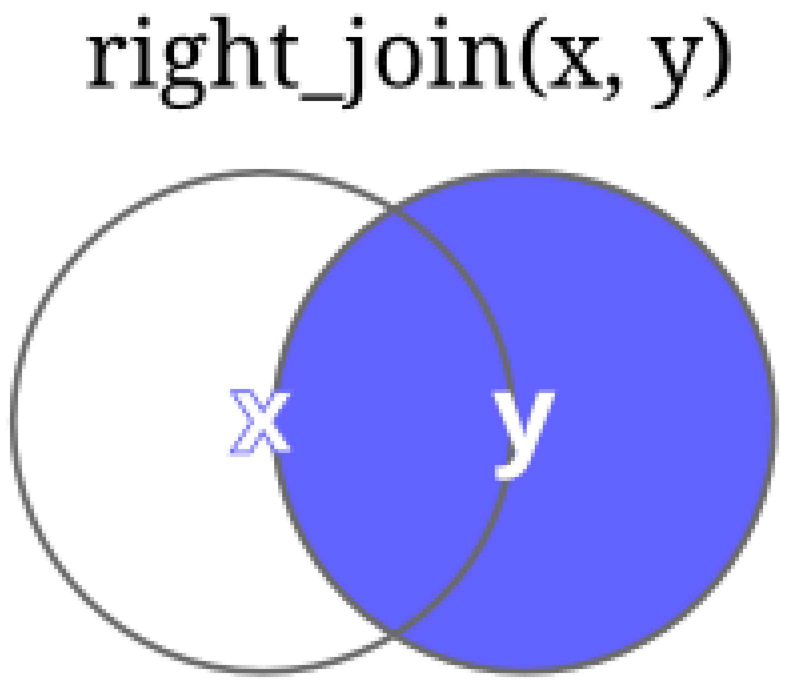
- Adds using data to master data based on key variable
- Only includes observations that are included in the *master data*
- Generates *NA* if observation missing in using data
- E. g.: `left_join(master, using, by = “ID”)`

ID	Age	Gender
1	36	0
2	42	1
3	23	0

ID	Income	Rent
1	2200	900
2	4100	1300
4	3600	1200

ID	Age	Gender	Income	Rent
1	36	0	2200	900
2	42	1	4100	1300
3	23	0	*NA*	*NA*

RIGHT_JOIN()



- Adds master data to using data based on key variable
- Only includes observations that are included in the *using data*
- Generates NA if observation missing in master data
- E. g.: `right_join(master, using, by = “ID”)`

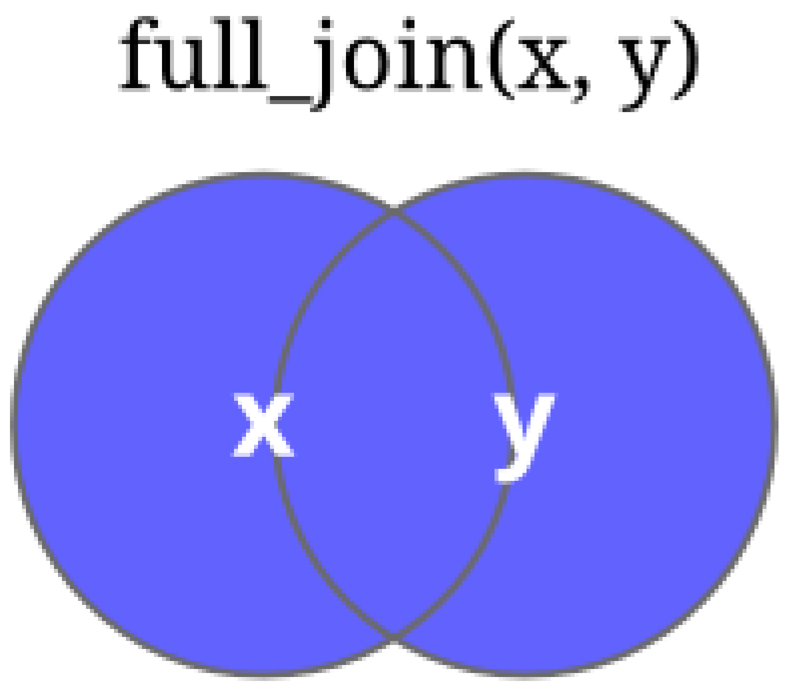
ID	Age	Gender
1	36	0
2	42	1
3	23	0

ID	Income	Rent
1	2200	900
2	4100	1300
4	3600	1200

ID	Age	Gender	Income	Rent
1	36	0	2200	900
2	42	1	4100	1300
4	*NA*	*NA*	3600	1200

FULL_JOIN()

- Adds master data to using data based on key variabl
- Includes all observations that exist in *either data*
- E. g.: `full_join(master, using, by = “ID”)`



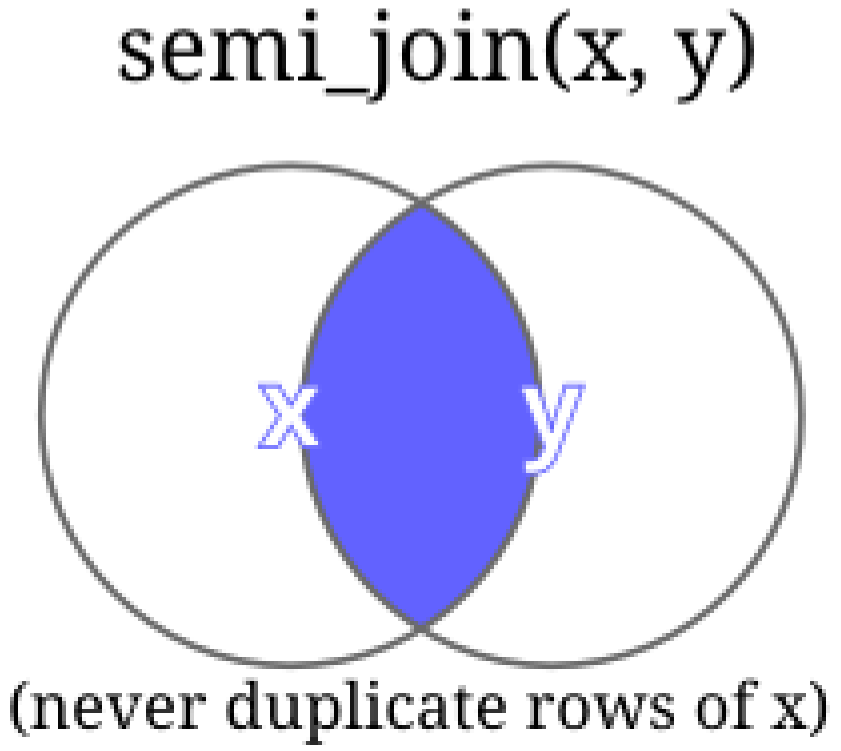
ID	Age	Gender
1	36	0
2	42	1
3	23	0

ID	Income	Rent
1	2200	900
2	4100	1300
4	3600	1200

ID	Age	Gender	Income	Rent
1	36	0	2200	900
2	42	1	4100	1300
3	23	0	*NA*	*NA*
4	*NA*	*NA*	3600	1200

SEMI_JOIN()

- Adds master data to using data based on key variable
- Only includes observations that exist in *both data*
- … but only keeps variables that exist in the master data
- E. g.: `semi_join(master, using, by = "ID")`


semi_join(x, y)

(never duplicate rows of x)

ID	Age	Gender
1	36	0
2	42	1
3	23	0

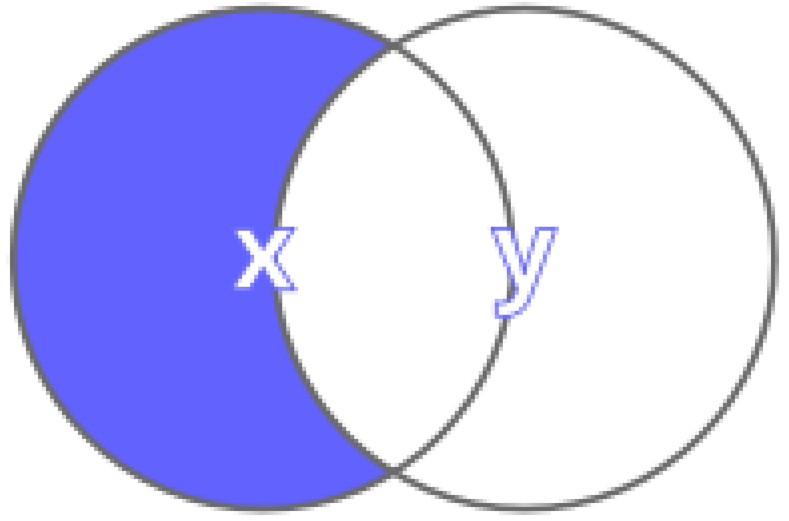
ID	Income	Rent
1	2200	900
2	4100	1300
4	3600	1200

ID	Age	Gender
1	36	0
2	42	1

ANTI_JOIN()

- Keeps observations of the master data that do not match the using data
- Generates NA if missing in master data
- E. g.: `anti_join(master, using, by = “ID”)`

anti_join(x, y)



ID	Age	Gender
1	36	0
2	42	1
3	23	0

ID	Income	Rent
1	2200	900
2	4100	1300
4	3600	1200

ID	Age	Gender
3	23	0

JOINING CLUSTERED DATA

- The logic of each join function also applies when we have several observations per key variable value (e. g.: multiple interviews per individual)
- In this case, each person-year in data 1 will get the (time constant) person value of the respective person in data 2

ID	Year	income
1	2021	980
1	2022	1000
2	2021	2600
2	2022	2600
3	2021	2300
3	2022	2400

ID	Birth year
1	1980
2	2002
3	1967

ID	Year	income	Birth year
1	2021	980	1980
1	2022	1000	1980
2	2021	2600	2002
2	2022	2600	2002
3	2021	2300	1967
3	2022	2400	1967

JOINING CLUSTERED DATA

- The same logic also applies for multiple members per household
- In this case, each respondent of the household in data 1 will get the household's value in data 2

ID	HHID	age
1	100	34
2	100	57
3	101	35
4	102	64
5	102	24
6	102	36

HHID	rent
100	900
101	1300
102	1700

ID	HHID	age	rent
1	100	34	900
2	100	57	900
3	101	35	1300
4	102	64	1700
5	102	24	1700
6	102	36	1700

MULTIPLE DATA SETS OR MULTIPLE KEY VARIABLES

- More than two data sets can also easily be combined stepwise:

```
→left_join(data1, data2, by = "id") %>%
      left_join(., data3, by = "id") %>%
      left_join(., data4, by = "id")
```

- With panel data, we will often have to combine data sets based on multiple key variables because we have variation by *person* and by *wave* (so person ID and year):

```
left_join(data1, data2, by=c("id", "year"),
match="all")
```

- Of couse, don't forget to assign these operations to an object

MULTIPLE KEY VARIABLES

- What if you want to add household-level panel data to individual-level panel data (Like the GSOEP)?

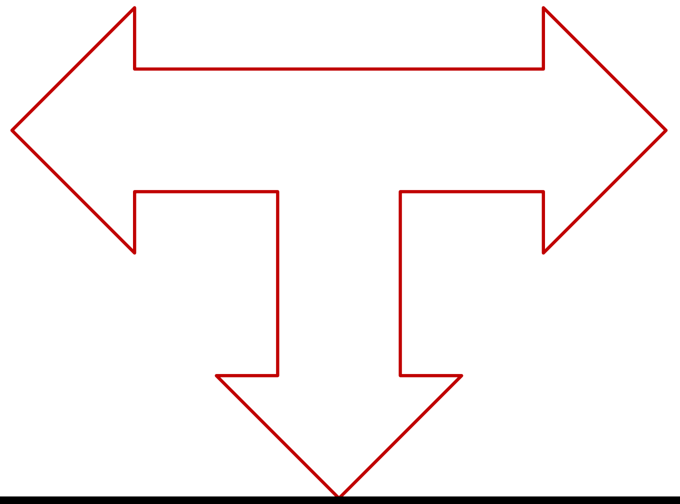
ID	HHID	t	Age	Gender
1	100	2011	36	0
1	100	2012	37	0
2	101	2011	40	1
2	101	2012	41	1
3	101	2011	37	0
3	101	2012	38	0

HHID	t	Income	Rent
100	2011	4500	1400
100	2012	4800	1400
101	2011	2200	800
101	2012	2000	820

MULTIPLE KEY VARIABLES

ID	HHID	t	Age	Gender
1	100	2011	36	0
1	100	2012	37	0
2	101	2011	40	1
2	101	2012	41	1
3	101	2011	37	0
3	101	2012	38	0

HHID	t	Income	Rent
100	2011	4500	1400
100	2012	4800	1400
101	2011	2200	800
101	2012	2000	820



ID	HHID	t	Age	Gender	Income	Rent
1	100	2011	36	0	4500	1400
1	100	2012	37	0	4800	1400
2	101	2011	40	1	2200	800
2	101	2012	41	1	2000	820
3	101	2011	37	0	2200	800
3	101	2012	38	0	2000	820

→Combination of HHID and t uniquely identifies observations

SUMMING UP

- Simple combination of data sets can be achieved with `rbind()` or `cbind()`
- However, in many instances this is not sufficient (e.g., missing data in one data set, clustered data, …)
- The `join()` family, which merges data based on key variables, helps in these cases
- This is especially relevant in the case of panel data, where we have multiple observations per unit
- I.e.: each observation (person-year) can only be identified by the (time-constant) person ID and the wave *simultaneously*
- *ALWAYS CHECK YOUR DATA MANUALLY AFTER COMBINING*

LITERATURE

- Chapter 2 (pages 15 - 48) in: Andreß, Golsch, & Schmidt (2014). Applied panel data analysis for economic and social surveys. Springer Science & Business Media
- More on joining with R: http://rstudio-pubs-static.s3.amazonaws.com/227171_618ebdce0b9d44f3af65700e833593db.html