

Handling experimental data with R

Day 2

SALOS 2025

Alena Witzlack-Makarevich

Overview

- Today
 - Expanding your GECO notebook
 - By inspecting a dataset in R
 - Doing a bunch of data manipulation and summaries with `dplyr`
 - Getting ready for rocking it with visualization tomorrow
- In the next days
 - Producing some visualization with `ggplot2`
 - More data manipulation and visualization
 - Some inferential statistics and modeling
 - And whatever else we can squeeze into the 5 days

L2 Lexical decision dataset (part of GECO)

Getting ready

- You have the GECO R notebook open
- You loaded tidyverse with `library("tidyverse")`
- You have the GECO dataset loaded into R
- And you really want to get the most out of this dataset

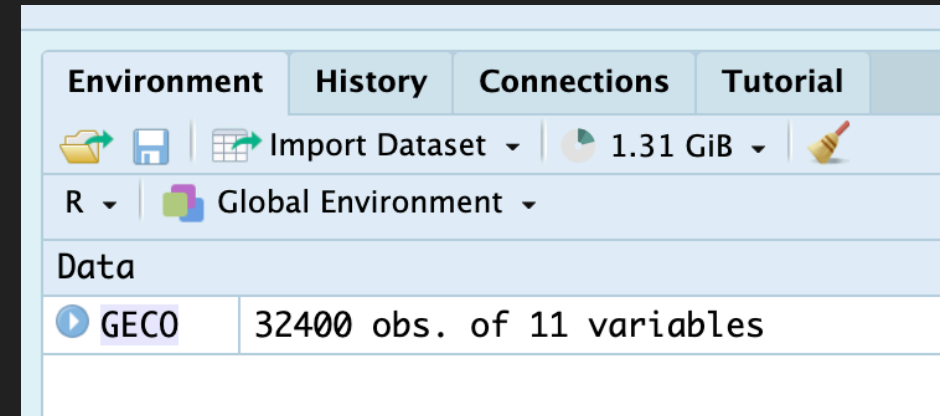
Inspecting the dataset in R

Inspecting the dataset in

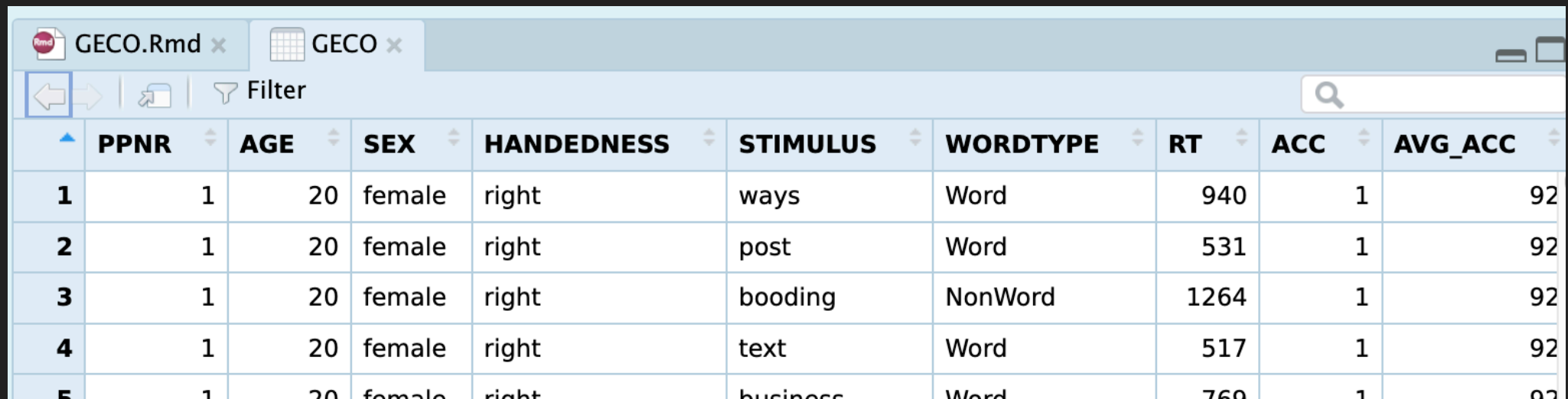
- Before starting any kind of analysis or further processing in R, always inspect your dataset:
 - Do you have all the variables?
 - Are they encoded the way they should?
 - What do the abbreviations stand for?
 - Do you have all the rows?
 - Does anything look funny?
- What functions and other options you remember of?

Inspecting the dataset in

- Check the Environment
- Click on the dataset there. What happens?
Is it useful?
- Notice what happened in the console



The screenshot shows the RStudio Environment pane. At the top, there are tabs for 'Environment', 'History', 'Connections', and 'Tutorial'. Below the tabs, there are icons for file operations and a status bar showing '1.31 GiB'. The 'Global Environment' is selected, and under the 'Data' section, a dataset named 'GECO' is listed with '32400 obs. of 11 variables'.



The screenshot shows the RStudio Data Viewer window. The title bar indicates the file 'GECO.Rmd' and the dataset 'GECO'. The viewer displays a table with the following columns: PPNR, AGE, SEX, HANDEDNESS, STIMULUS, WORDTYPE, RT, ACC, and AVG_ACC. The first five rows of data are visible.

	PPNR	AGE	SEX	HANDEDNESS	STIMULUS	WORDTYPE	RT	ACC	AVG_ACC
1	1	20	female	right	ways	Word	940	1	92
2	1	20	female	right	post	Word	531	1	92
3	1	20	female	right	booding	NonWord	1264	1	92
4	1	20	female	right	text	Word	517	1	92
5	1	20	female	right	business	Word	760	1	92

Inspecting the dataset in

- Try `head(GECO)`
- Could you guess what `tail(GECO)` does?
- Try `str(GECO)`. Can you interpret its output?
- What kind of R data structure is `GECO`?

R object classes

- There are several major object classes in R
- They are vectors, ... , ... , and ...
- What kind of object classes is **GECO**?

Dataframes vs. tibbles

- dataframe and tibble are a spreadsheet-like data structures
- a **tibble** is essentially a modern data frame
- or in other words, it is a data frame in tidyverse
- some silly etymology: tibble is a playful shortening of **tbl**, which is short for table

Dataframes vs. tibbles

- So tibbles are a kind of data frames, but slightly tweaked to work better in the tidyverse
- **Why** a new data structure?
- R is meanwhile an old language and some concepts, which made sense 15 or 25 years ago, are meanwhile obsolete or deserve an update
- One cannot change base R, major innovations take place in the individual packages (e.g. tidyverse)
- In most cases you don't need to know much about the differences, as you won't be working with the old-school base R

Inspecting the dataset in

- try `head(GECO)`

a convention: `head(df)`

`df` = any data frame,
replace with your variable name

- Could you guess what `tail(df)` does?
- Try `str(df)`. Can you interpret its output?
- What kind of R data structure is `GECO`?
- What variables types do we have?
Is it the type you would expect?

`str(df)`

- `str()` is another method to get a rapid overview of your data
- it shows you the **structure** of your dataset
- For a data frame/tibble it tells you:
 - the total number of observations
 - the total number of variables
 - a full list of the variables names
 - the data type of each variable
 - the first observations
- Ignore the bottom of the output for now

str(df)

- `str()` is another method to get a rapid overview of your data
- it shows you the **structure** of your dataset
- For a data frame/tibble it tells you:
 - the total number of observations
 - the total number of variables
 - a full list of the variables names
 - the data type of each variable
 - the first observations
- Ignore the bottom of the output for now

```
spc_tbl_ [32,400 × 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
$ PPNR      : num [1:32400] 1 1 1 1 1 1 1 1 1 1 ...
$ AGE       : num [1:32400] 20 20 20 20 20 20 20 20 20 ...
$ SEX       : chr [1:32400] "female" "female" "female" "female" ...
$ HANDEDNESS: chr [1:32400] "right" "right" "right" "right" ...
$ STIMULUS  : chr [1:32400] "ways" "post" "booding" "text" ...
$ WORDTYPE  : chr [1:32400] "Word" "Word" "NonWord" "Word" ...
$ RT        : num [1:32400] 940 531 1264 517 769 ...
$ ACC       : num [1:32400] 1 1 1 1 1 1 0 1 1 0 ...
$ AVG_ACC   : num [1:32400] 92 92 92 92 92 92 92 92 92 ...
$ LEXTALE_EN: chr [1:32400] "80" "80" "80" "80" ...
$ LEXTALE_DU: chr [1:32400] "85" "85" "85" "85" ...
- attr(*, "spec")=
.. cols(
..   PPNR = col_double(),
..   AGE = col_double(),
..   SEX = col_character(),
..   HANDEDNESS = col_character(),
..   STIMULUS = col_character(),
..   WORDTYPE = col_character(),
..   RT = col_double(),
..   ACC = col_double(),
..   AVG_ACC = col_double(),
..   LEXTALE_EN = col_character(),
..   LEXTALE_DU = col_character()
.. )
- attr(*, "problems")=<externalptr>
```

str(df)

```
spc_tbl_ [32,400 × 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ PPNR      : num [1:32400] 1 1 1 1 1 1 1 1 1 1 ...
 $ AGE       : num [1:32400] 20 20 20 20 20 20 20 20 20 20 ...
 $ SEX       : chr  [1:32400] "female" "female" "female" "female" ...
 $ HANDEDNESS: chr  [1:32400] "right" "right" "right" "right" ...
 $ STIMULUS  : chr  [1:32400] "ways" "post" "booding" "text" ...
 $ WORDTYPE  : chr  [1:32400] "Word" "Word" "NonWord" "Word" ...
 $ RT        : num [1:32400] 940 531 1264 517 769 ...
 $ ACC       : num [1:32400] 1 1 1 1 1 1 0 1 1 0 ...
 $ AVG_ACC   : num [1:32400] 92 92 92 92 92 92 92 92 92 92 ...
 $ LEXTALE_EN: chr  [1:32400] "80" "80" "80" "80" ...
 $ LEXTALE_DU: chr  [1:32400] "85" "85" "85" "85" ...
```

GECO

L2 English lexical proficiency dataset

What is the dataset?

- All participants completed a “battery of language proficiency tests”, including a vocabulary test, a lexical decision task, ...
- A dataset of a lexical decision task
- Performed in English (L2) by **81** Dutch L1 participants
- The dataset contains raw reaction times (**RT**) and accuracy (**ACC**) scores for **800** stimuli
- As well as some participant information
- In the larger eye-tracking study, the dataset is used to compare the L1 and L2 word-level age-of-acquisition effect between lexical decision and eye movement measures

What is our dataset? A tiny part of GECCO

	PPNR	AGE	SEX	HANDEDNESS	STIMULUS	WORDTYPE	RT	ACC	AVG_ACC	LEXTALE_EN	LEXTALE_DU
1	1	20	female	right	ways	Word	940	1	92	80	85
2	1	20	female	right	post	Word	531	1	92	80	85
3	1	20	female	right	booding	NonWord	1264	1	92	80	85
4	1	20	female	right	text	Word	517	1	92	80	85
5	1	20	female	right	business	Word	769	1	92	80	85
6	1	20	female	right	lale	NonWord	1130	1	92	80	85
7	1	20	female	right	snings	NonWord	746	0	92	80	85

Participants' details

Speeded lexical decision task:
reaction times (**RT**)
and accuracy (**ACC**) scores
for 800 stimuli

Un-speeded
lexical decision tasks

What is our dataset? A tiny part of GECCO

- We want to explore this dataset
- What variable(s) can be interesting to have a closer look?
- What kinds of relations can we explore?
- What kind of manipulations might we need for this?
- Discuss with your neighbor and share your suggestions!

PPNR	AGE	SEX	HANDEDN	STIMULUS	WORDTYPE	RT	ACC	AVG_ACC	LEXTALE_EN	LEXTALE_DU
41	18	male	right	bugtime	NonWord	675	1	93	68.75	83.75
41	18	male	right	poison	Word	480	1	93	68.75	83.75
41	18	male	right	marvel	Word	540	0	93	68.75	83.75
41	18	male	right	pame	NonWord	557	1	93	68.75	83.75
41	18	male	right	hole	Word	511	1	93	68.75	83.75

Data wrangling with `dplyr`

Why is it called `dplyr`?



dplyr



- `dplyr` is an R package
- its functions are designed to enable dataframe or a tibble (a spreadsheet-like data structure) manipulation in an intuitive, user-friendly way
- one of the core packages of the popular `tidyverse` set of packages
- data analysts typically use `dplyr` in order to **transform** existing datasets into a format better suited for some particular type of analysis, or data visualization

dplyr



- You rarely get the data in exactly the right form you need.

Results

Data reduction

First, we removed all trials on which participants responded faster than 250 ms or slower than 4,000 ms (fewer than 1% of the data). For the remaining trials, items were removed on the basis of the error rates for the particular age groups. The second graders made more than 50% errors on 26 items, which were subsequently removed from the data for this grade (see Appendix). The fourth graders had error rates higher than 50% on five items, which were subsequently removed from the data of this grade (see Appendix).

dplyr



- You rarely get the data in exactly the right form you need.
- Sometimes you need to create new variables or summaries,
- or maybe you just want to rename the variables or reorder the observations in order to make the data a little easier to work with
- And often you need just a subset of your dataset
- All this is possible with the package [dplyr](#)



dplyr

- the typical data management tasks are conceptualized as five verbs:
 - `select()`: pick variables/columns by their names
 - `filter()`: pick observations by their values
 - `mutate()`: create new variables with functions of existing variables
 - `summarize()`: collapse many values down to a single summary
 - `arrange()`: reorder the rows



dplyr

- All verbs work similarly:

```
verb(df, variable)
```

The first argument is a data frame

The subsequent arguments describe what variables one wants

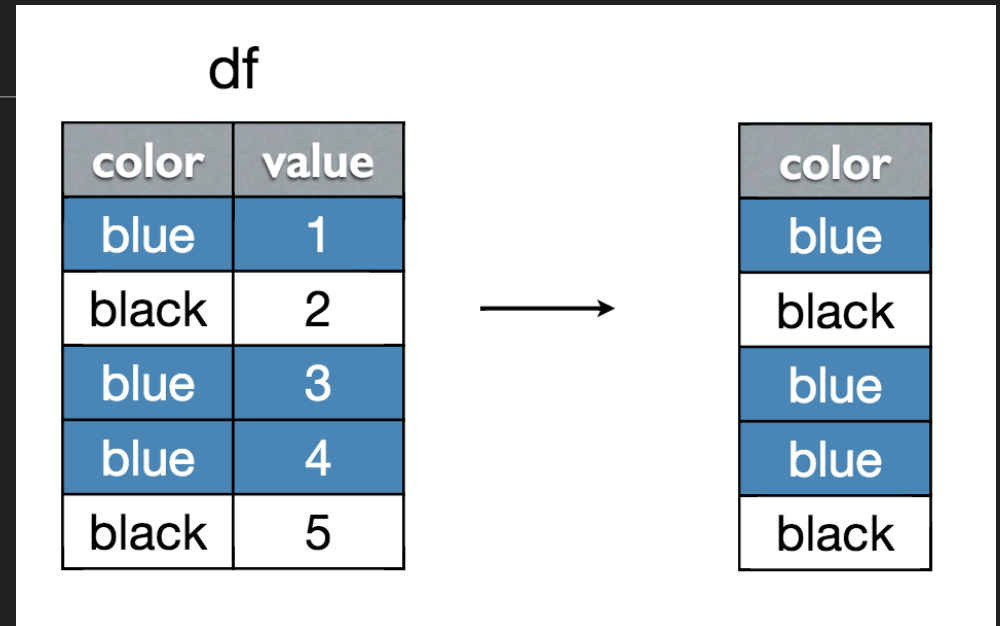
to consider (without quotes)

- The result is a new data frame/tibble, which needs to be assigned to a new variable

```
select()
```

select()

- pick variables/columns by their names:
`select(df, color)`



`select()`

- Try it with the GECO data:
 1. Select all variables describing the participants (`AGE`, `SEX`, etc.) and save them into a new data frame `participants`

`select()`

- Try it with the GECO data:
 1. Select all variables describing the participants (`AGE`, `SEX`, etc.) and save them into a new data frame `participants`
 2. Select all variables related to LEXTALE lexical proficiency testing and the participant number and save them as `LEXTALE`
 3. Guess first, then try: what will happen here
 - a. `select(GECO, -HANDEDNESS)`
 - b. `select(GECO, contains("LEXTALE"))`
 - c. `select(GECO, PPNR:HANDEDNESS)`

select()

```
45 # select()
46 This function **select** some variables:
47
48 ```{r message = FALSE}
49 participants <- select(GECO, PPNR, SEX, AGE, HANDEDNESS)
50 head(participants)
51 ```
```

A tibble: 6 × 4

PPNR <dbl>	SEX <chr>	AGE <dbl>	HANDEDNESS <chr>
1	female	20	right
1	female	20	right
1	female	20	right
1	female	20	right
1	female	20	right
1	female	20	right

6 rows

Coming next: `filter()`

- The five `dplyr` verbs:

- ✓ `select()`

- `filter()`

- `mutate()`

- `summarize()`

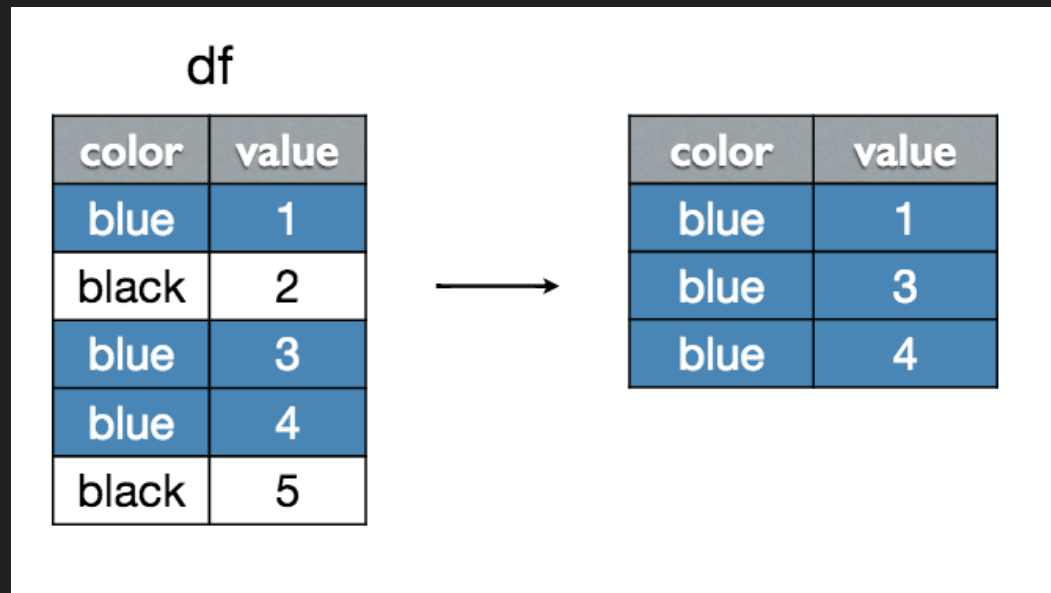
- `arrange()`

filter()

- Filter rows with filter(): pick observations by their values

E.g.

```
filter(df, color == "blue")
```



filter()

- Filter rows with filter(): pick observations by their values, e.g.
`filter(df, color == "blue")`
- Filter all the rows
 - for left-handed participants. How many rows do you get?
 - for the stimulus word "weather". How many rows do you get?
 - with reaction time $RT > 2400$. How many rows do you get?
 - with the average accuracy (`AVG_ACC`) above 96. How many rows?

filter()

- Filter all the rows for left-handed participants.
How many rows do you get? **4800**

```
57 ▾ ```{r message = FALSE}  
58 filter(GECO, HANDEDNESS == "left")  
59 ▲ ```
```

A tibble: 4,800 × 11

PPNR <dbl>	AGE <dbl>	SEX <chr>	HANDEDNESS <chr>	STIMULUS <chr>	WORDTYPE <chr>
57	18	female	left	business	Word
57	18	female	left	pig	Word
57	18	female	left	disnost	NonWord

filter()

- Filter all the rows for the stimulus word “weather”.
How many rows do you get? **21**

```
61  
62 ▾ ```{r message = FALSE}  
63 filter(GECO, STIMULUS == "weather")  
64 ▲ ```
```

A tibble: 20 × 11

PPNR <dbl>	AGE <dbl>	SEX <chr>	HANDEDNESS <chr>	STIMULUS <chr>	WORDTYPE <chr>
1	20	female	right	weather	Word
17	18	female	right	weather	Word
25	18	female	right	weather	Word

filter()

- Filter all responses (all rows) for the participant with the average accuracy above 96:
- How many rows do you get? **400**

```
65
66 Filter all responses (all rows) for the participant with the average accuracy ab
67
68 ▾ ```{r message = FALSE}
69 filter(GECO, AVG_ACC > 96)
70 ▸ |```
```

A tibble: 400 × 11

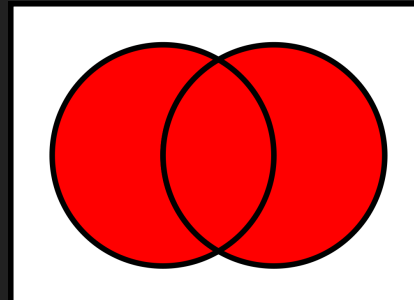
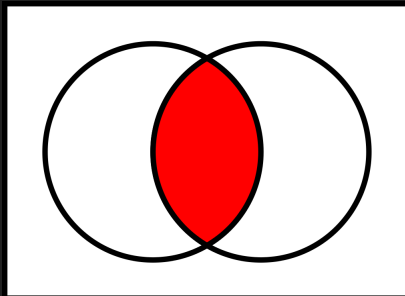
PPNR <dbl>	AGE <dbl>	SEX <chr>	HANDEDNESS <chr>	STIMULUS <chr>	WORDTYPE <chr>	RT <dbl>	ACC <dbl>
71	18	male	right	gaze	Word	868	1
71	18	male	right	windake	NonWord	583	1
71	18	male	right	reroof	NonWord	767	1
71	18	male	right	rilks	NonWord	557	1
71	18	male	right	nail	Word	513	1
71	18	male	right	body	Word	472	1
71	18	male	right	brain	Word	489	1

`filter()`

- Can you think of more complex filter operations?
- What do we need to know to execute them?

Logical operators for `filter()`

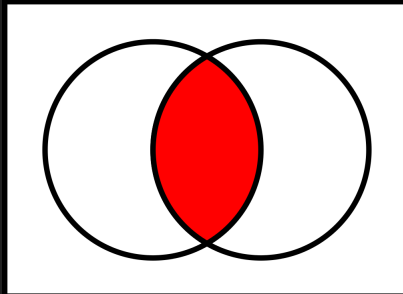
- What logical operators are these?



Logical operators for `filter()`

- What logical operators are these?

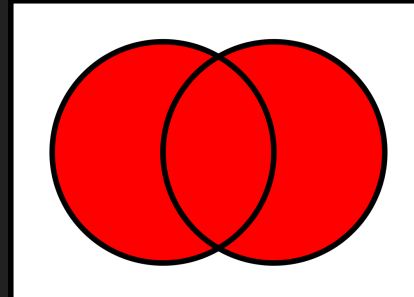
conjunction (AND)



female

18 y.o.

vs. disjunction (OR)



female

18 y.o.

Logical operators for `filter()`

- Multiple conditions to `filter()` are combined with `&` “and”
- Every part of the expression must be true in order for a row to be included in the output.
- What output will you get here?

```
filter(df, color == "blue" & value == 2)
```

df

color	value
blue	1
black	2
blue	3
blue	4
black	5

Logical operators for `filter()`

- Multiple conditions to `filter()` are combined with `&` “and”
- Every part of the expression must be true in order for a row to be included in the output.
- What output will you get here?

```
filter(df, color == "blue" & value == 2)
```

df

color	value
blue	1
black	2
blue	3
blue	4
black	5

→

color	value
blue	1
blue	3
blue	4

Logical operators for `filter()`

- Multiple conditions to `filter()` can be combined with `|` (pipe) “or”
- Every part of the expression must be true in order for a row to be included in the output.
- What output will you get here?

```
filter(df, color == "blue" | value == 3)
```

df

color	value
blue	1
black	2
blue	3
blue	4
black	5

Logical operators for `filter()`

- Multiple conditions to `filter()` can be combined with `|` (pipe) “or”
- Every part of the expression must be true in order for a row to be included in the output.
- What output will you get here?

```
filter(df, color == "blue" | value == 3)
```

df

color	value
blue	1
black	2
blue	3
blue	4
black	5



color	value
blue	1
blue	3
blue	4

Logical operators for `filter()`

Match		Combine	
<code>==</code>	equal to	<code>&</code>	<code>and</code>
<code>!=</code>	not equal to	<code> </code>	<code>or</code>
<code><</code>	less than	<code>!</code>	<code>not</code>
<code>></code>	greater than	<code>%in%</code>	member of a group
<code><=</code>	less than or equal to	<code>is.na</code>	is NA
<code>>=</code>	less than or equal to	<code>!is.na</code>	is not NA

Let's build some pipes

- How many times **female** participants had a **reaction time over 1400**?
- How many times the **stimulus word "brain"** got a **reaction time of over 1000**?
- How many times **non-words** were **inaccurately** identified? (i.e. identified as words)
- Think of a medium complex filtering task for this dataset. Try it out. Does it work correctly? Write the task down on a slip of paper? Write the expected answer on the backside and pass it around to be solved by your peers.

Logical operators for `filter()`

- How many times **female** participants had a **reaction time over 1400**?

Answer: **2**

```
66 How many times female participants had a reaction time over 1400?  
67 ```{r message = FALSE}  
68 filter(GECO, SEX == "female" & RT > 1400)  
69 ```
```

A tibble: 900 × 11

PPNR <dbl>	AGE <dbl>	SEX <chr>	HANDEDNESS <chr>	STIMULUS <chr>	WORDTYPE <chr>	RT <dbl>
1	20	female	right	grease	Word	1645
1	20	female	right	vutses	NonWord	1612
1	20	female	right	visit	Word	1484
1	20	female	right	pade	NonWord	1636
1	20	female	right	pooges	NonWord	1615
1	20	female	right	wint	NonWord	1698

Logical operators for `filter()`

- How many times the stimulus word “brain” got a reaction time of over? Answer: 2

```
74 How many times the word "brain" got the reaction time of over 1000?  
75  
76 ```{r message = FALSE}  
77 filter(GECO, STIMULUS == "brain" & RT > 1000)  
78 ```
```

A tibble: 2 × 11

PPNR <dbl>	AGE <dbl>	SEX <chr>	HANDEDNESS <chr>	STIMULUS <chr>	WORDTYPE <chr>	RT <dbl>	A <chr>
27	18	female	right	brain	Word	2380	
47	18	female	right	brain	Word	1117	

2 rows | 1-9 of 11 columns

Logical operators for `filter()`

- How many times **non-words** were **inaccurately** identified? (i.e. identified as words)? Answer: **1944**

```

70
71 How many times non-words were inaccurately identified (i.e. identified as words)?
72
73 ```{r message = FALSE}
74 filter(GECO, WORDTYPE == "NonWord" & ACC == 0)
75 ```

```

A tibble: 1,944 × 11

PPNR	AGE	SEX	HANDEDNESS	STIMULUS	WORDTYPE	RT	ACC	AVG.
<dbl>	<dbl>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	
1	20	female	right	snings	NonWord	746	0	
1	20	female	right	ludge	NonWord	1317	0	
1	20	female	right	ey	NonWord	565	0	

Interlude: building pipes





Build a pipe with %>%

- All verbs work similarly:
`verb(df, variable)`
- Often you want to do several things with a data frame in one go:
a more elegant solution is to build a **pipe** with `%>%`
- `df %>% verb(variable) %>% verb(variable) -> newdf`



Build a pipe with `%>%`

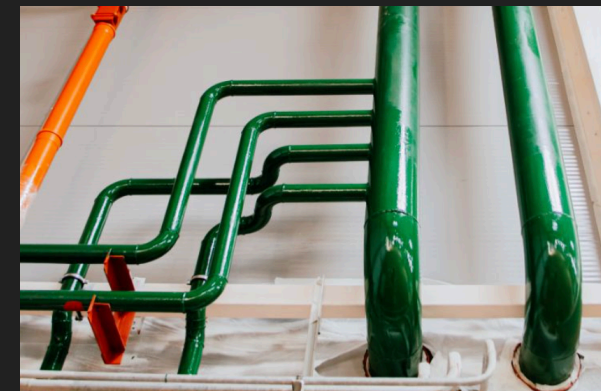
- All verbs work similarly:
`verb(df, variable)`
- Often you want to do several things with a data frame in one go:
a more elegant solution is to build a **pipe** with `%>%`
- `df %>% verb(variable) %>% verb(variable) -> newdf`

Potential for confusion:
the symbol `|` “OR” is
also called “pipe”



Build a pipe with %>%

- All verbs work similarly:
`verb(df, variable)`
- Often you want to do several things with a data frame in one go:
a more elegant solution is to build a **pipe** with `%>%`
- `df %>% verb(variable) %>% verb(variable) -> newdf`
- `%>%` can be read “then”
- a pipe enables a flow of data through a series of operations, just like water flowing through pipes
- Pipes belong to tidy code





Build a pipe with %>%

- `df %>% verb1(var) %>% verb2(var) %>% verb3(var)`
- It is common to do some nice indentation to read pipes easier:
 - `df %>%
 verb1(variable) %>%
 verb2(variable) %>%
 verb3(variable)`



Build a pipe with %>%

- And you often need to save the outcome to a new dataframe

```
newdf <- df %>%  
  verb1(variable) %>%  
  verb2(variable) %>%  
  verb3(variable)
```



Build a pipe with %>%

- And you often need to save the outcome to a new data frame:

```
newdf <- df %>%  
  verb1(var) %>%  
  verb2(var) %>%  
  verb3(var)
```

- Or non-idiomatically at the end of the pipe:

```
df %>%  
  verb1(var) %>%  
  verb2(var) %>%  
  verb3(var) -> newdf
```

Let's build some pipes

- Build a **pipe** which produces a data frame for LEXTALE results for male participants only, keep **PPNR** and **SEX**
- Think of another **sensible** combination of select and filter for this dataset, build a pipe, and show it to your neighbor.
- Before we proceed:
Spend a few minutes cleaning up your R notebook, adding notes, and knitting it. Does it produce a report (html or pdf)?

Coming next: `summrize()`

Learn on your own:

- The five `dplyr` verbs:

- ✓ `select()`

- ✓ `filter()`

- `mutate()`

- `summarize()`

- `arrange()`

Derive new variables with `mutate()`

- Besides selecting sets of existing columns, it's often useful to add new columns that are functions of existing columns → `mutate()`
- `mutate()` always adds new columns at the end of your dataset
- In RStudio, the easiest way to see all the columns of a wide dataset in environment/`View()`

df

color	value
blue	1
black	2
blue	3
blue	4
black	5

→

color	value	double
blue	1	2
black	2	4
blue	3	6
blue	4	8
black	5	10

```
mutate(df, double = 2 * value)
```

```
summarize() / summarise()  
group_by()
```



Aggregate values with mutate()

- `summarize()` aggregate many values down to a single summary

df

color	value
blue	1
black	2
blue	3
blue	4
black	5

→

total
15

```
df %>% summarize(total = sum(value))
```

Aggregate values with `summarize()`

- `summarize()` aggregate many values down to a single summary

df

color	value
blue	1
black	2
blue	3
blue	4
black	5

→

total
15

```
df %>% summarize(total = sum(value))
```

Aggregate values with `summarize()`

- `summarize()` the `RT` variable to calculate the mean reaction time
- name the new variable `mean_RT`, use the function `mean()`
- Check on your neighbor whether they know what “mean” means and how is it different (or not) from average

A

- The dataset contains 32,400 responses.
- Use `summarize()` to add up (`sum()`) all the correct answers (`ACC` of 1), name the new variable `sum_ACC`

B

- Run a quick sanity check with `filter()` for `ACC==1`.
You should get the same number. Do you?

Aggregate values with `summarize()`

- `summarize()` the `RT` variable to calculate the mean reaction time
- name the new variable `mean_RT`, use the function `mean()`
- Check on your neighbor whether they know what “mean” means and how is it different (or not) from average

A

```
92 # summarize()
93
94 Summarize() the RT variable to calculate the mean reaction time
95
96 ```{r message = FALSE}
97 GECO %>% summarize(mean_RT = mean(RT))
98 ```
```

A tibble: 1 × 1

mean_RT
<dbl>
715.867

1 row

no need to save
this as a new variable

Aggregate values with `summarize()`

B

- The dataset contains 32,400 responses.
- Use `summarize()` to add up (`sum()`) all the correct answers (`ACC` of 1), name the new variable `sum_ACC`
- Run a quick sanity check with `filter()` for `ACC==1`.
You should get the same number. Do you?

```

```{r message = FALSE}
GECO %>% summarize(sum_ACC = sum(ACC))
```

```

A tibble: 1 × 1

| sum_ACC
<dbl> |
|------------------|
| 29161 |

1 row

```

```{r message = FALSE}
GECO %>% filter(ACC == 1)
```

```

A tibble: 29,161 × 11

| PPNR
<dbl> | AGE
<dbl> | SEX
<chr> | HANDEDNESS
<chr> |
|---------------|--------------|--------------|---------------------|
| 1 | 20 | female | right |
| 1 | 20 | female | right |

group_by() + summarize()

- `summarize()` is not very useful on its own, pair it with `group_by()`
- This changes the unit of analysis from the complete dataset to individual groups
- `summarize()` is then apply to the dataset “by group”:

| df | |
|-------|-------|
| color | value |
| blue | 1 |
| black | 2 |
| blue | 3 |
| blue | 4 |
| black | 5 |

→

| color | total |
|-------|-------|
| blue | 8 |
| black | 7 |

```
df %>%  
  group_by(color) %>%  
  summarize(total = sum(value))
```

group_by() + summarize()

- What kind of `group_by()` and `summarize()` operations would be useful for the GECO dataset?

df

| color | value |
|-------|-------|
| blue | 1 |
| black | 2 |
| blue | 3 |
| blue | 4 |
| black | 5 |

→

| color | total |
|-------|-------|
| blue | 8 |
| black | 7 |

```
df %>%  
  group_by(color) %>%  
  summarize(total = sum(value))
```

`group_by()` + `summarize()`

```
df %>%  
  group_by(color) %>%  
  summarize(total = sum(value))
```

- Group the dataset by **SEX** and calculate the mean reaction time **RT**.

A

Which group (**male** or **female**) is faster?

- What about accuracy? Which group is more accurate?
Calculate mean accuracy in the basis of the variable **ACC**.

- Are the participants slower in reacting to **NonWord** than to **Word**?

Group by **WORDTYPE** and calculate the mean reaction time for each group?

B

- Are there sex differences? Group by **WORDTYPE** and **SEX** and compare the average reaction time. Discuss the results with your neighbor.

`group_by()` + `summarize()`

```
df %>%  
  group_by(color) %>%  
  summarize(total = sum(value))
```

- Group the dataset by **SEX** and calculate the mean reaction time **RT**.

A

Which group (**male** or **female**) is faster?

```
114 ▾ ````{r message = FALSE, echo = FALSE}  
115 GECO %>% group_by(SEX) %>% summarize(mean_AGE = mean(AGE))  
116 ▲ ````
```

A tibble: 2 × 2

| SEX
<chr> | mean_AGE
<dbl> |
|---------------------|--------------------------|
| female | 18.36923 |
| male | 18.75000 |

2 rows

`group_by()` + `summarize()`

```
df %>%  
  group_by(color) %>%  
  summarize(total = sum(value))
```

- What about accuracy? Which group is more accurate?
Calculate mean accuracy in the basis of the variable **ACC**.

A

```
117 ▾ ```{r message = FALSE, echo = FALSE}  
118 GECO %>% group_by(SEX) %>% summarize(mean_ACC = mean(ACC))  
119 ▲ ```
```

A tibble: 2 × 2

| SEX
<chr> | mean_ACC
<dbl> |
|---------------------|--------------------------|
| female | 0.8931538 |
| male | 0.9279687 |

2 rows

group_by() + summarize()

- What could we do next with these results?

```
114 ▾ ```{r message = FALSE, echo = FALSE}
115 GECO %>% group_by(SEX) %>% summarize(mean_AGE = mean(AGE))
116 ▲ ```
```

A tibble: 2 × 2

| SEX
<chr> | mean_AGE
<dbl> |
|--------------|-------------------|
| female | 18.36923 |
| male | 18.75000 |

2 rows

```
117 ▾ ```{r message = FALSE, echo = FALSE}
118 GECO %>% group_by(SEX) %>% summarize(mean_ACC = mean(A))
119 ▲ ```
```

A tibble: 2 × 2

| SEX
<chr> | mean_ACC
<dbl> |
|--------------|-------------------|
| female | 0.8931538 |
| male | 0.9279687 |

2 rows

group_by() + summarize()

```
df %>%  
  group_by(color) %>%  
  summarize(total = sum(value))
```

- Group the dataset by **SEX** and calculate the mean reaction time **RT**.

A

Which group (**male** or **female**) is faster?

- What about accuracy? Which group is more accurate?
Calculate mean accuracy in the basis of the variable **ACC**.

- Are the participants slower in reacting to **NonWord** than to **Word**?

Group by **WORDTYPE** and calculate the mean reaction time for each group?

B

- Are there sex differences? Group by **WORDTYPE** and **SEX** and compare the average reaction time. Discuss the results with your neighbor.

**Getting ready for tomorrow:
What is worth visualizing?**

What is worth visualizing tomorrow?

- Stand up and join someone you haven't had a chance to talk to
- In groups of three look at the GECO dataset
- What variables and what relations should we inspect visually?
- What kind of plots would make sense?
- Draft on a piece of paper what those plots should look.
- Come up with at least two different plot types and try to use different variables

Getting ready for tomorrow: Chapter 2 of Introduction to Tidyverse

INTERACTIVE COURSE

Introduction to the Tidyverse

[Start](#) [Bookmark](#) [...](#)

Basic 4 hr 16 videos 50 Exercises 360,229 participants 4150 XP Updated: May 2025

Description

This is an introduction to the programming language R, focused on a powerful set of tools known as the Tidyverse. You'll learn the intertwined processes of data manipulation and visualization using the tools dplyr and ggplot2. You'll learn to manipulate data by filtering, sorting, and summarizing a real dataset of historical country data in order to answer exploratory questions. You'll then learn to turn this processed data into informative line plots, bar plots, histograms, and more with the ggplot2 package. You'll get a

[Read More](#) ▾

1 Data wrangling 0%

In this chapter, you'll learn to do three things with a table: filter for particular observations, arrange the observations in a desired order, and mutate to add or change a column. You'll see how each of these steps allows you to answer questions about your data.

[View Chapter Details](#) ▾ [Start Chapter](#)

2 Data visualization 0%

Often a better way to understand and present data as a graph. In this chapter, you'll learn the essential skills of data visualization using the ggplot2 package, and you'll see how the dplyr and ggplot2 packages work closely together to create informative graphs.

[View Chapter Details](#) ▾ [Start Chapter](#)