0. PROGRAMMING WITH R
1. ANALYTIC QUESTIONS
2. EXPLORATORY DATA ANALYSIS
3. DATA REPRESENTATION
4. VECTORS
5. DATA FRAMES
6. LOAD DATA
7. TIDY DATA
8. STRINGS
9. TRANSFORM DATA
10. UNSTRUCTURED DATA
11. MACHINE LEARNING
12. VISUALIZE DATA
13. COMMUNICATE FINDINGS
14. PYTHON

PROGRAMMING WITH R

variables

control structures

loops

functions

libraries

ANALYTIC QUESTIONS

BACK

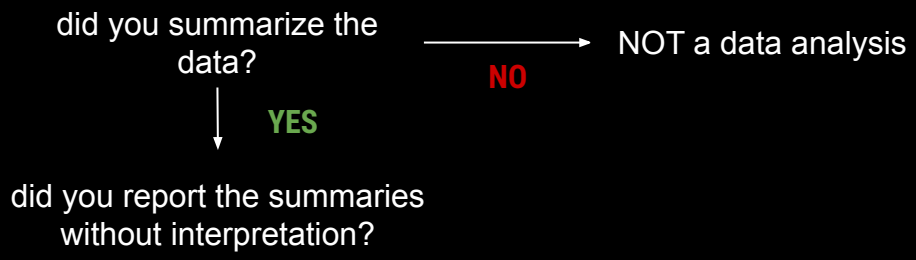
did you summarize the
data?

Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.

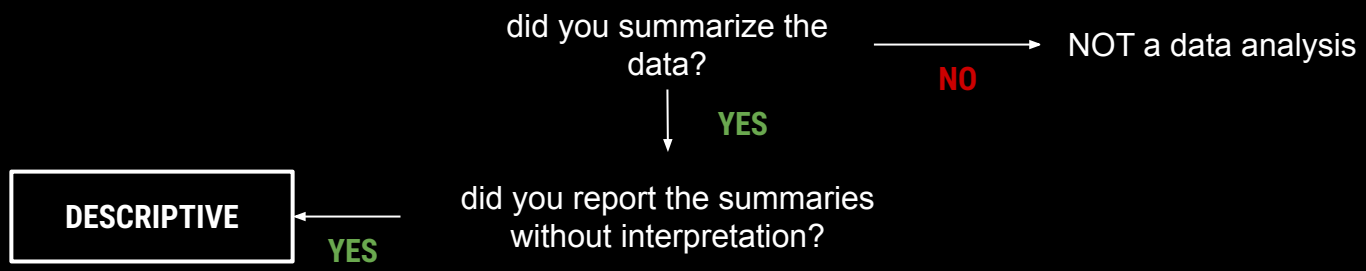
did you summarize the data? → NOT a data analysis

NO

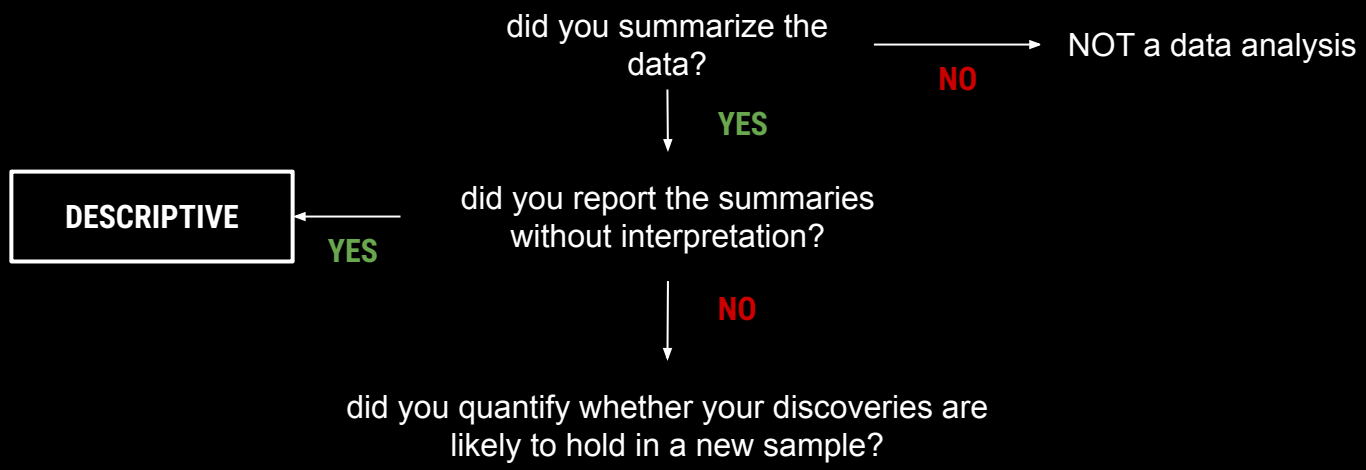
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



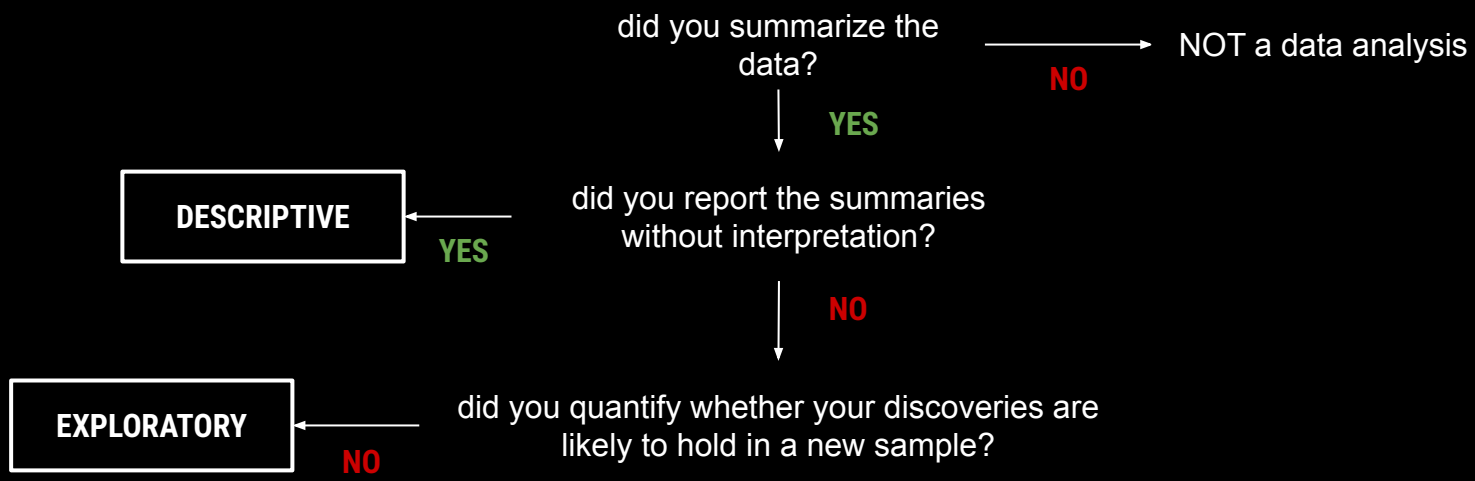
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



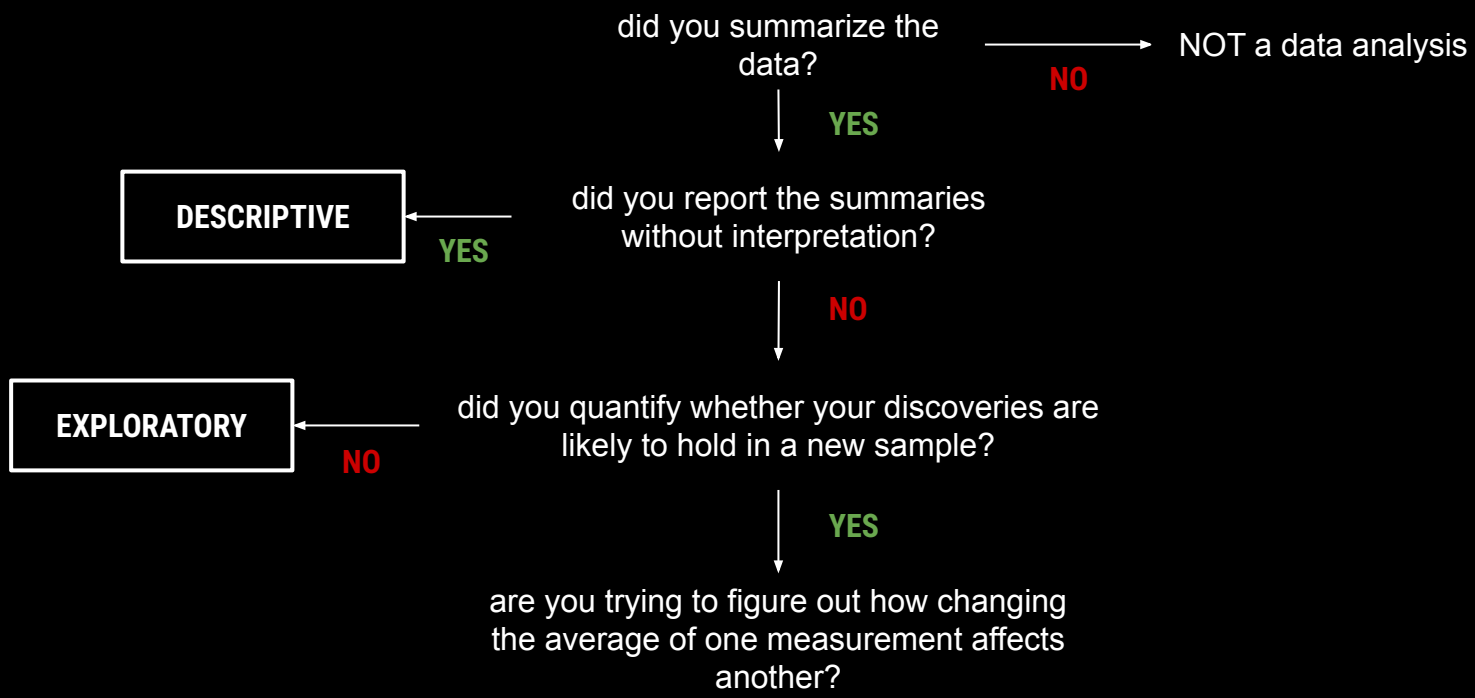
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



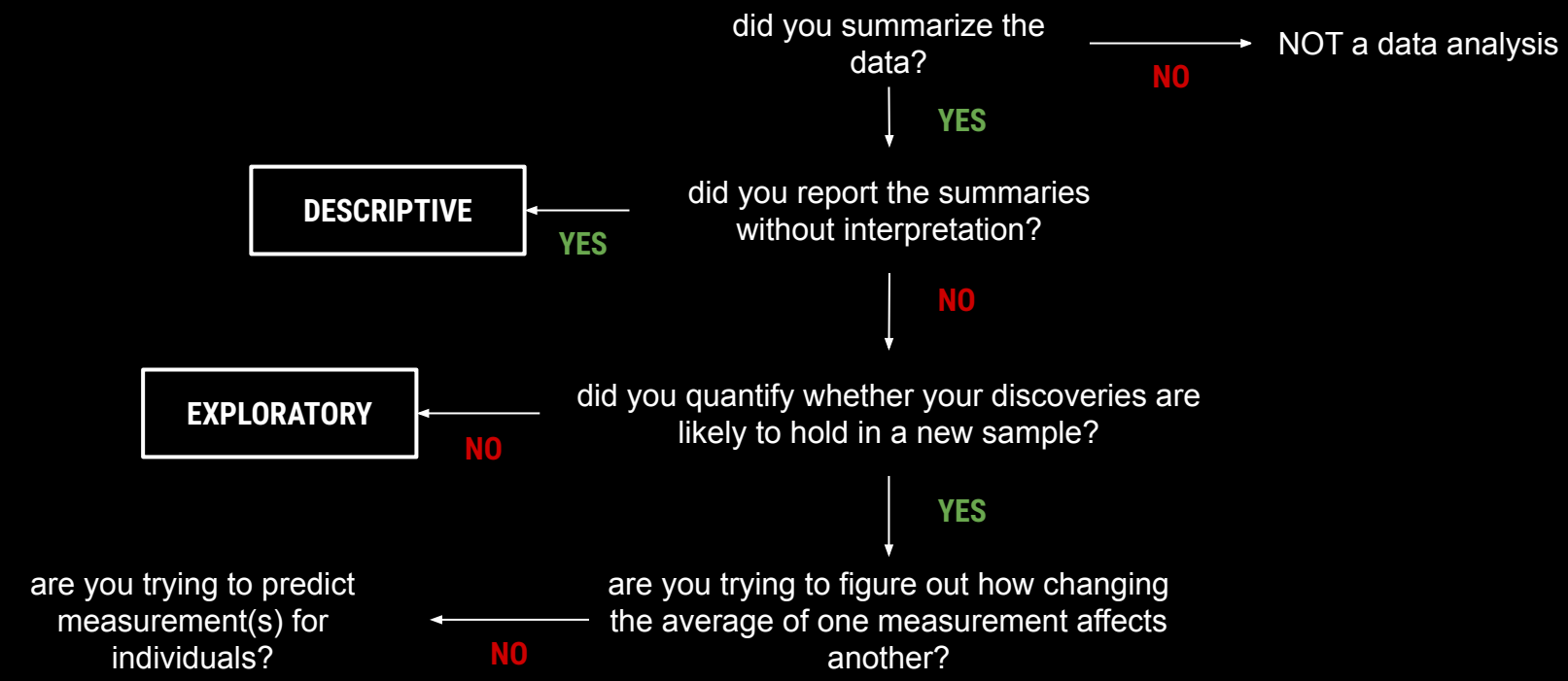
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



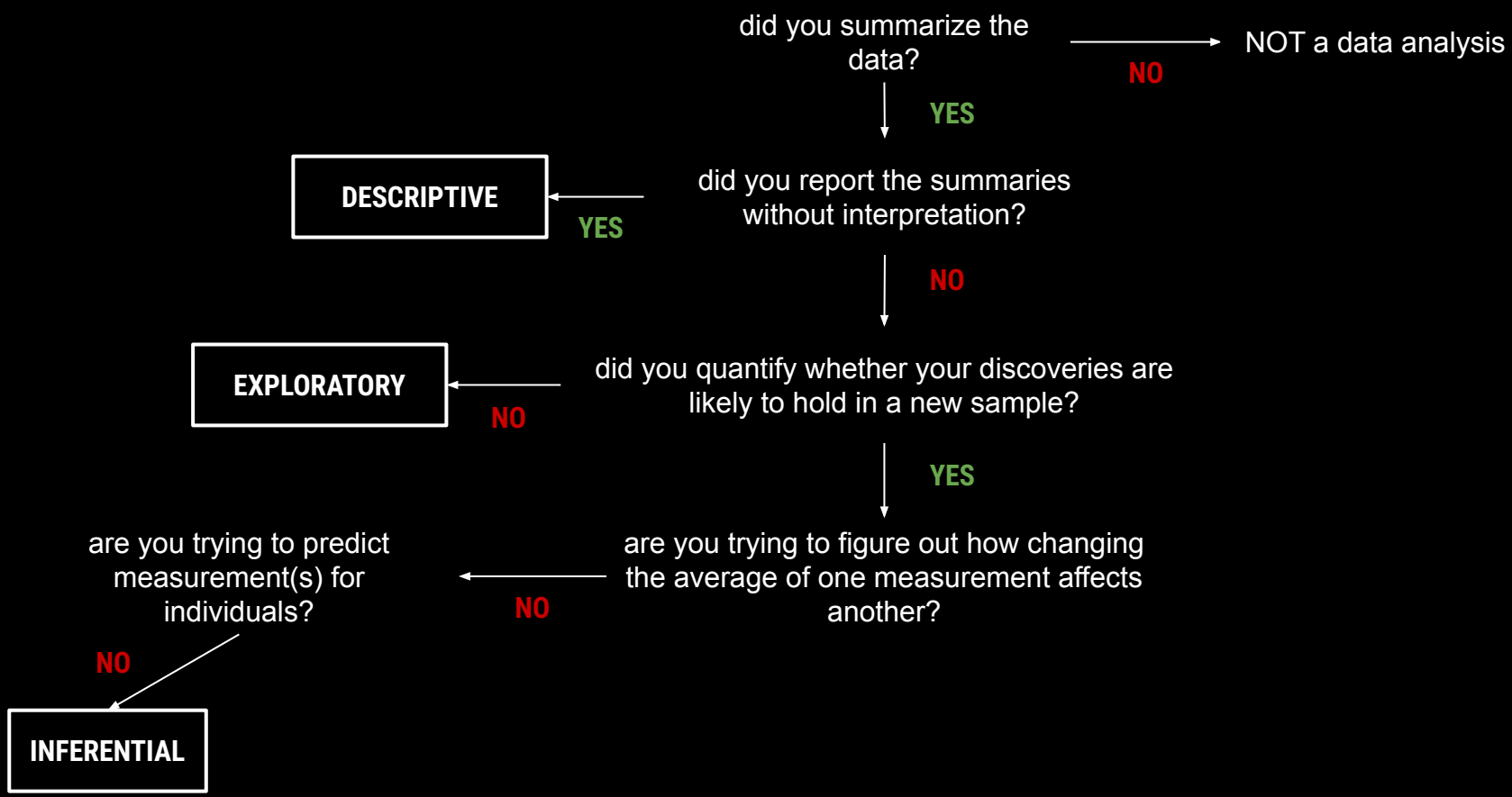
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



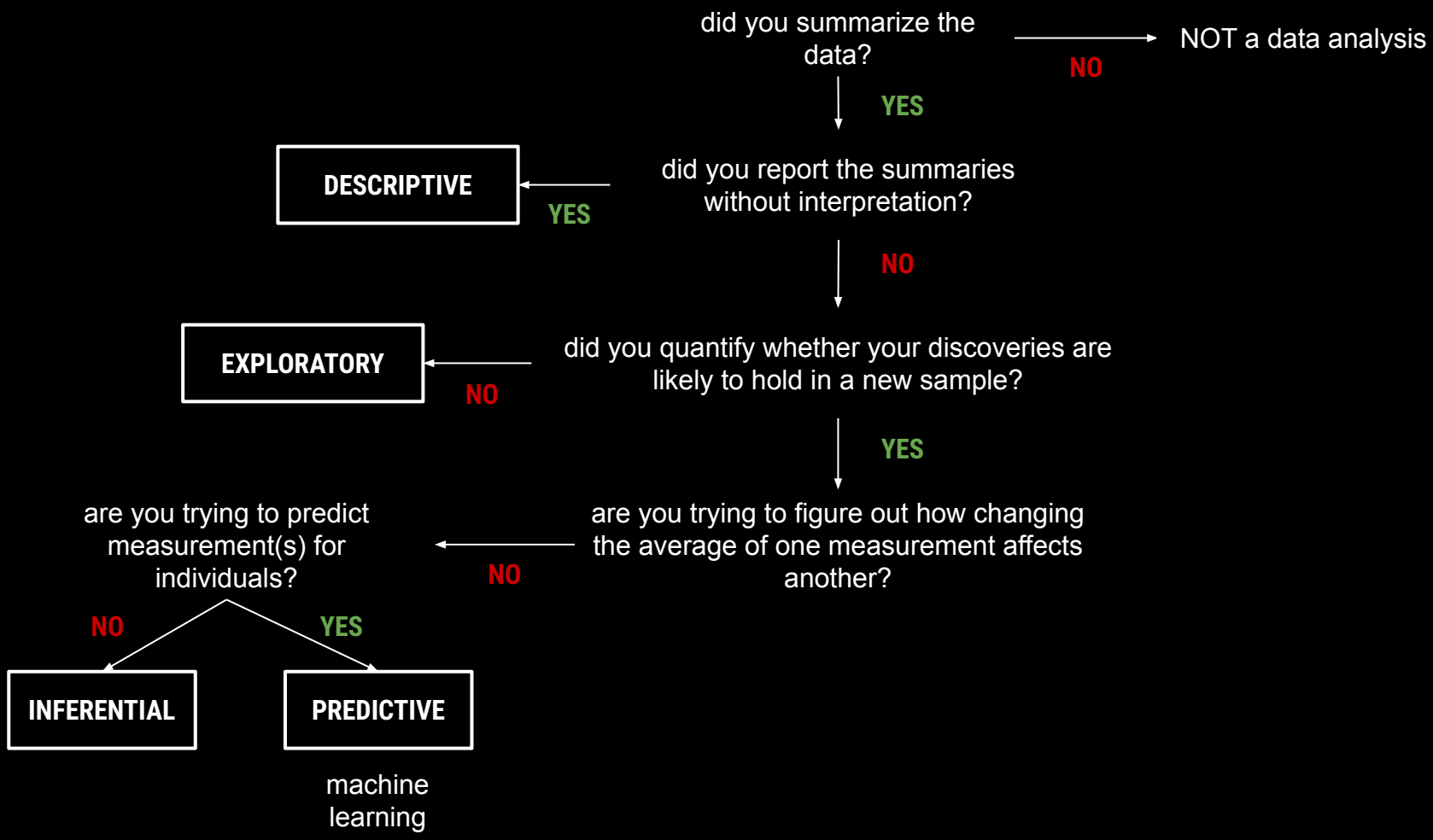
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



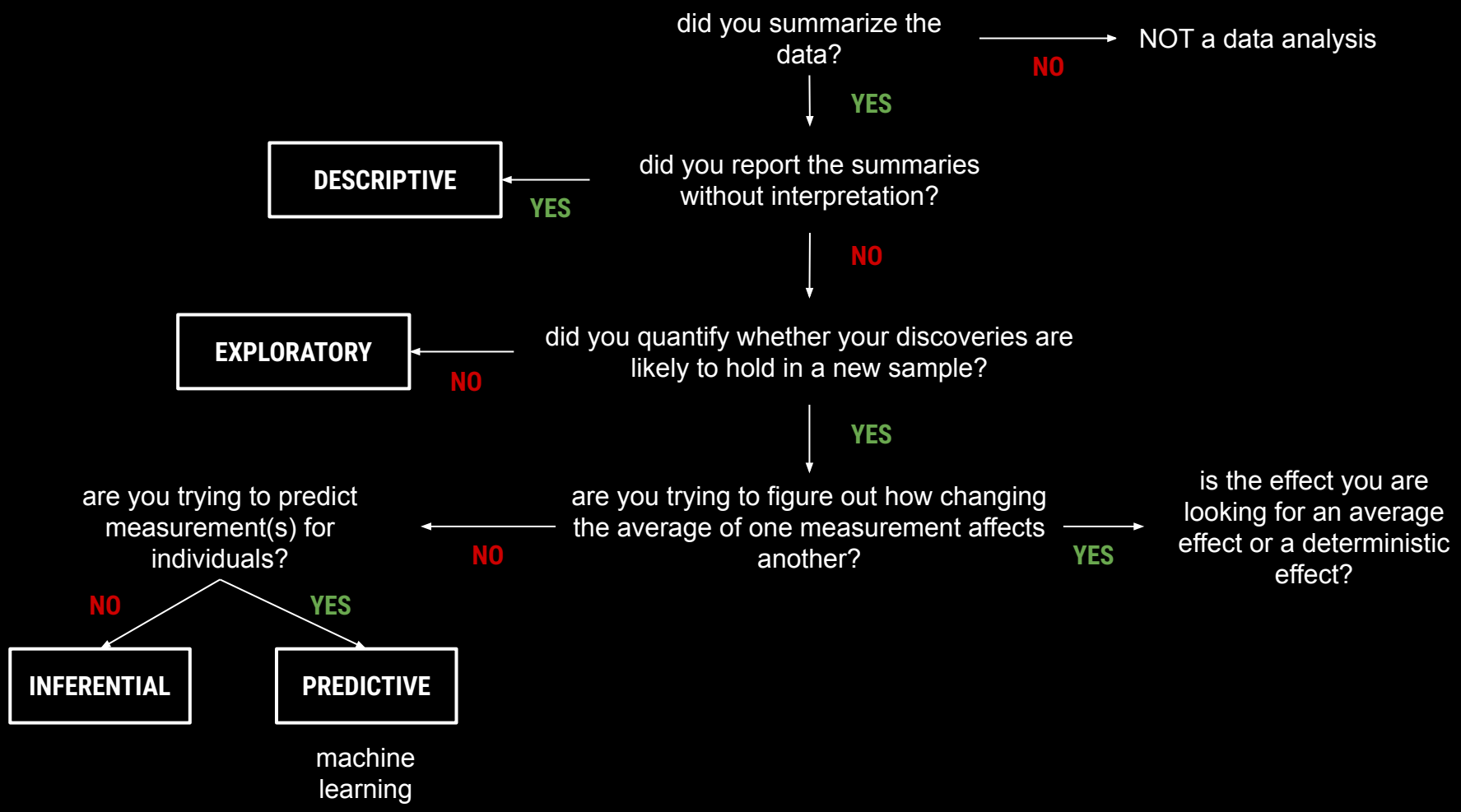
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



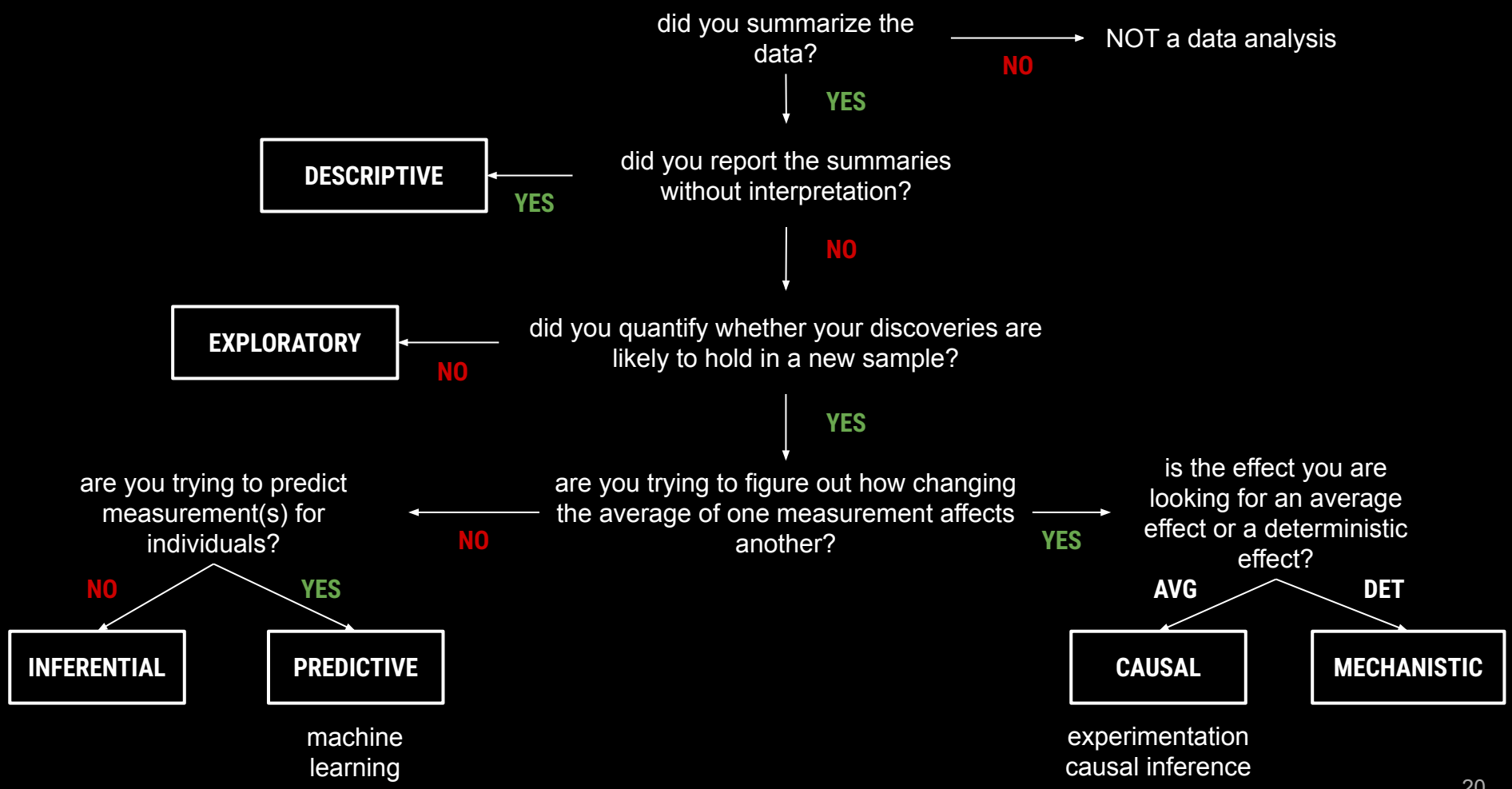
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



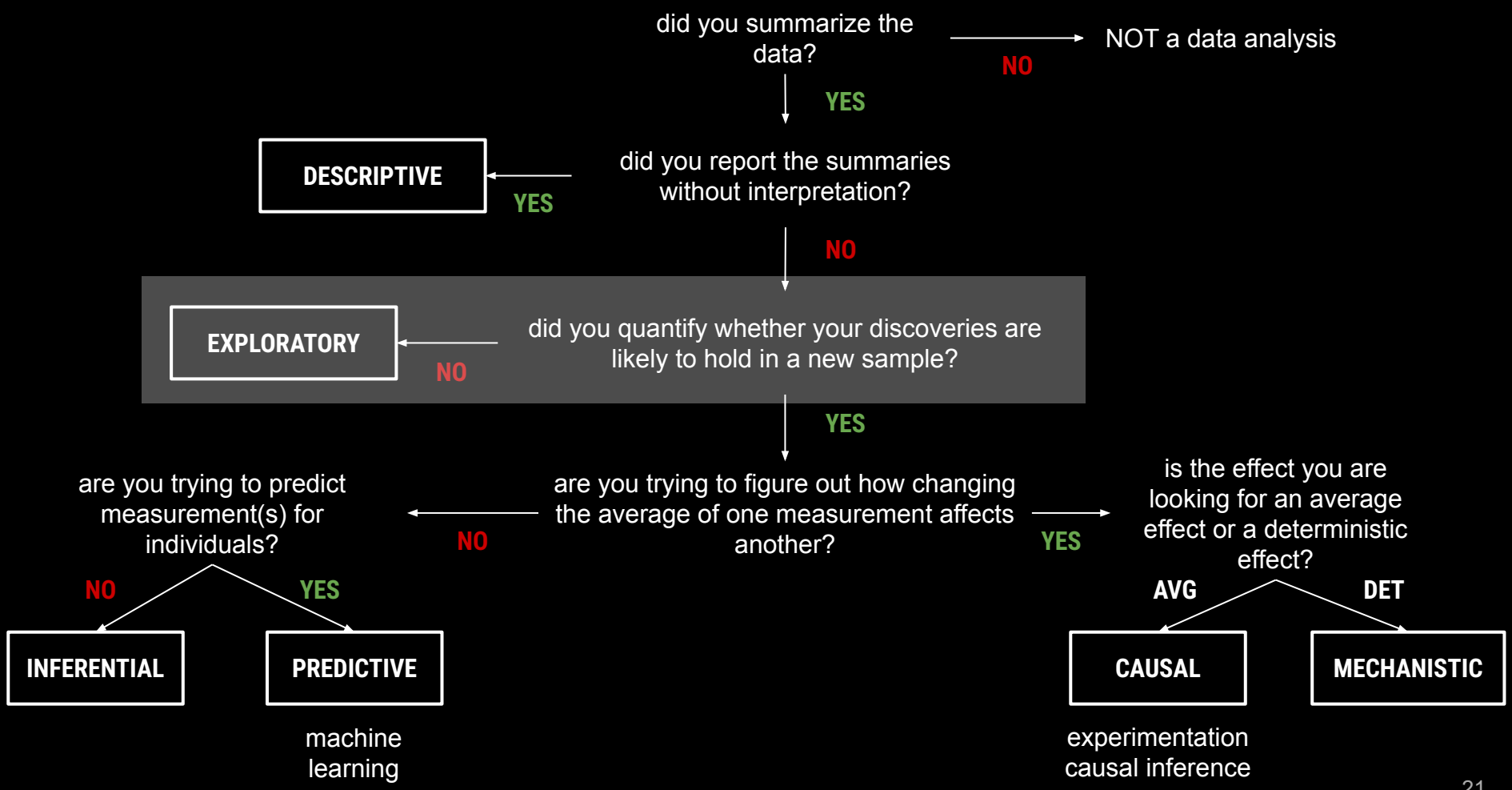
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



Source: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.



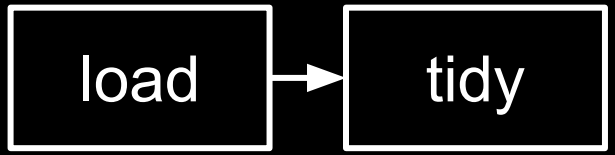
Quelle: Jeffery T. Leek, Roger D. Peng: What is the question? Science 20.03.2015: 1314-1315.

EXPLORATORY DATA ANALYSIS

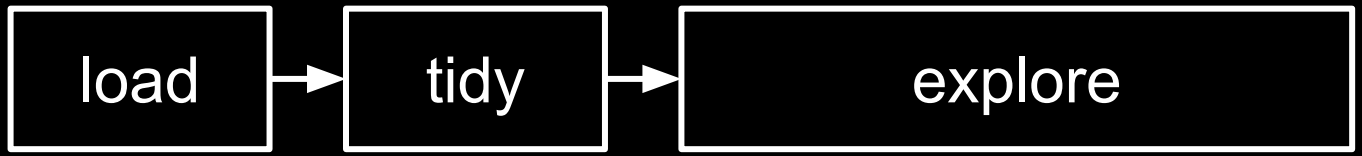
BACK

load

Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.



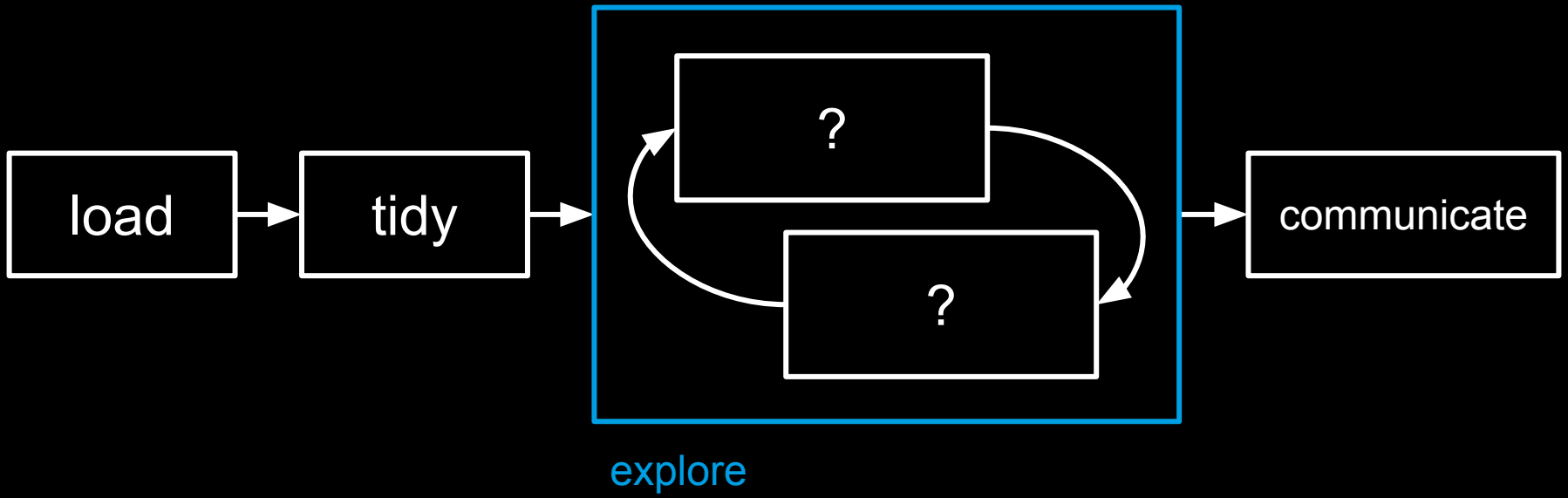
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.



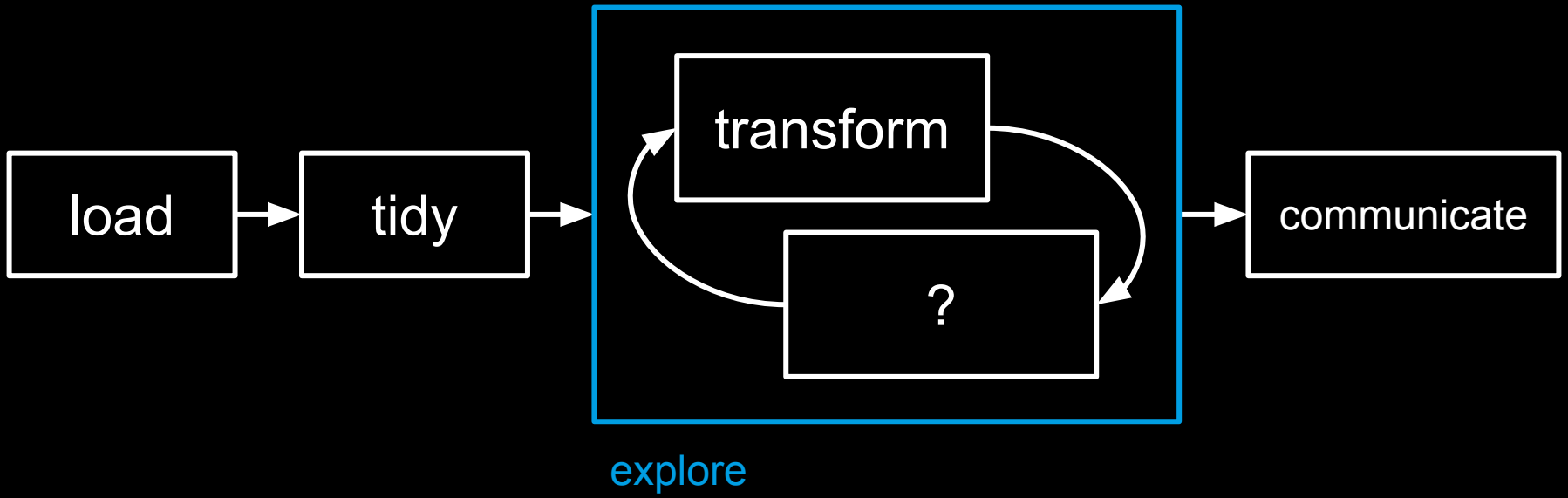
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.

Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.

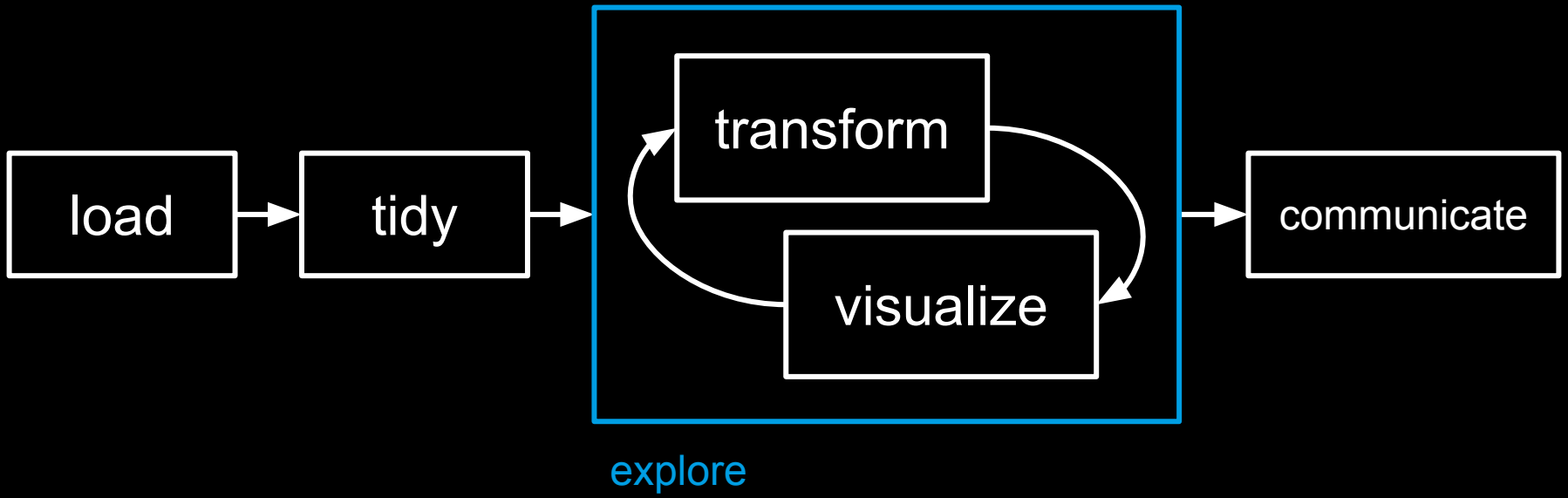
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.



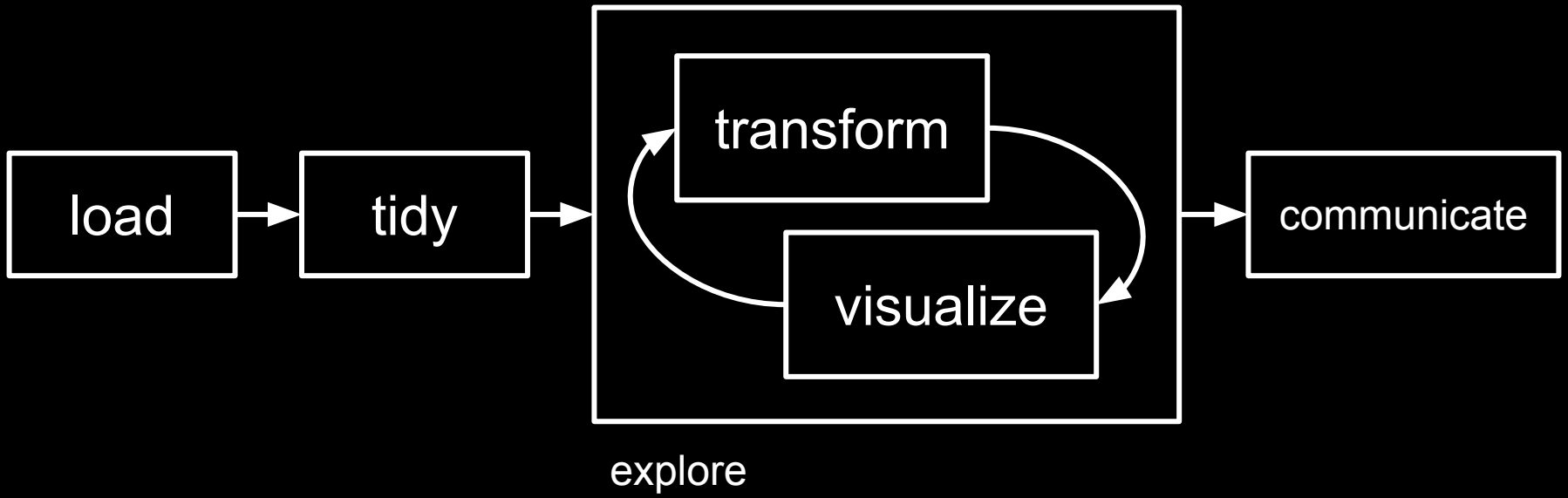
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.



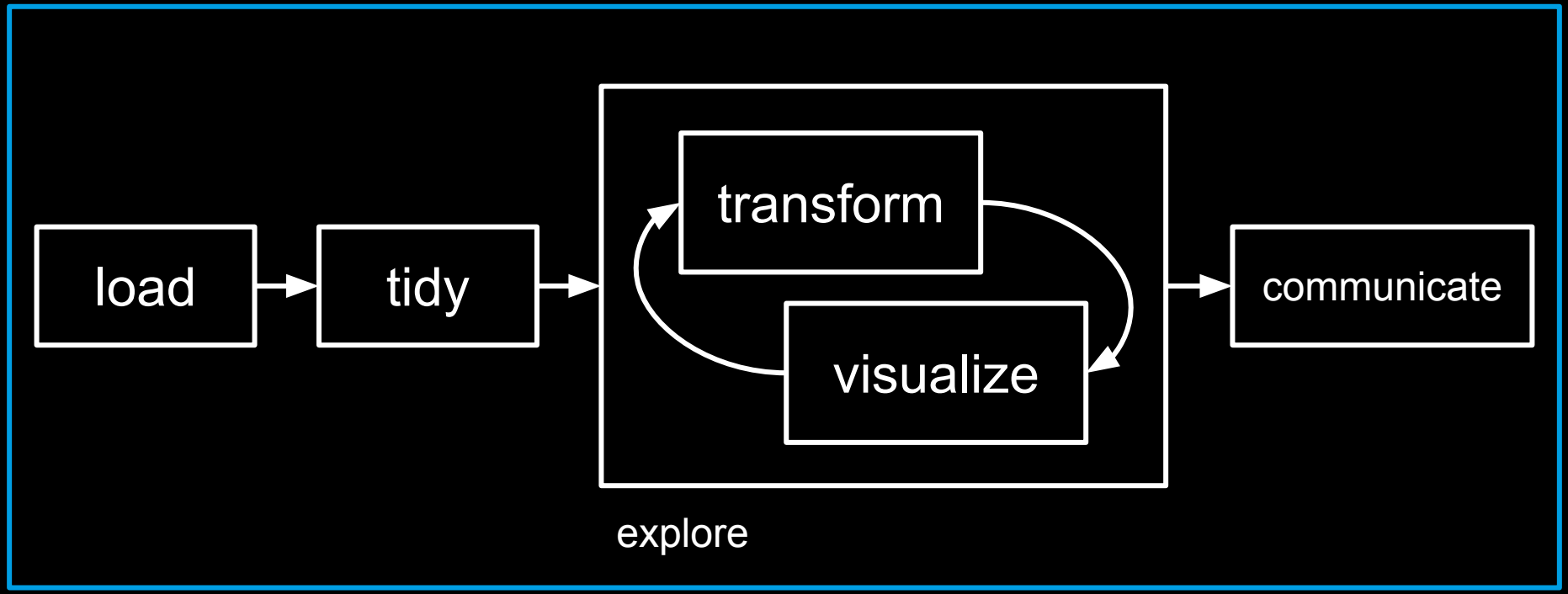
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.



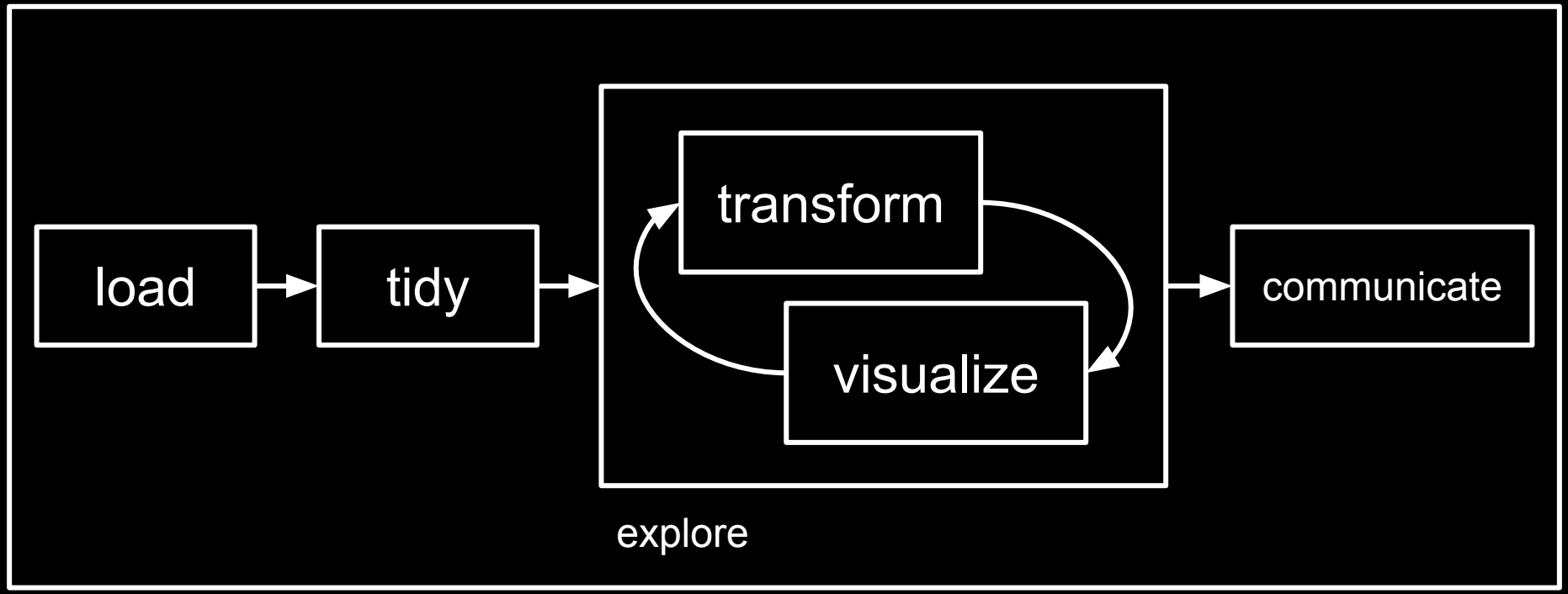
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.



Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.



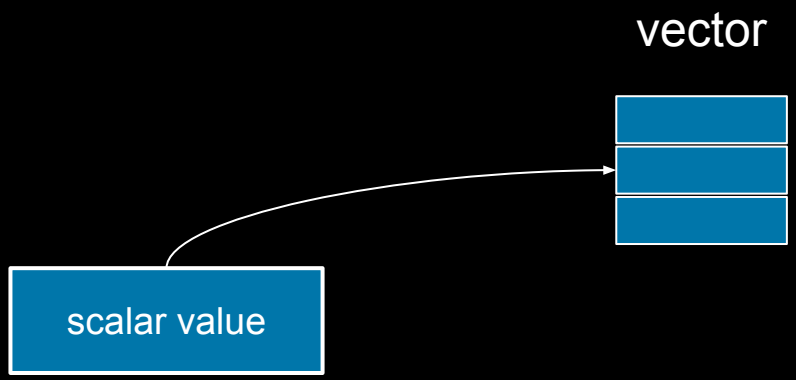
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.

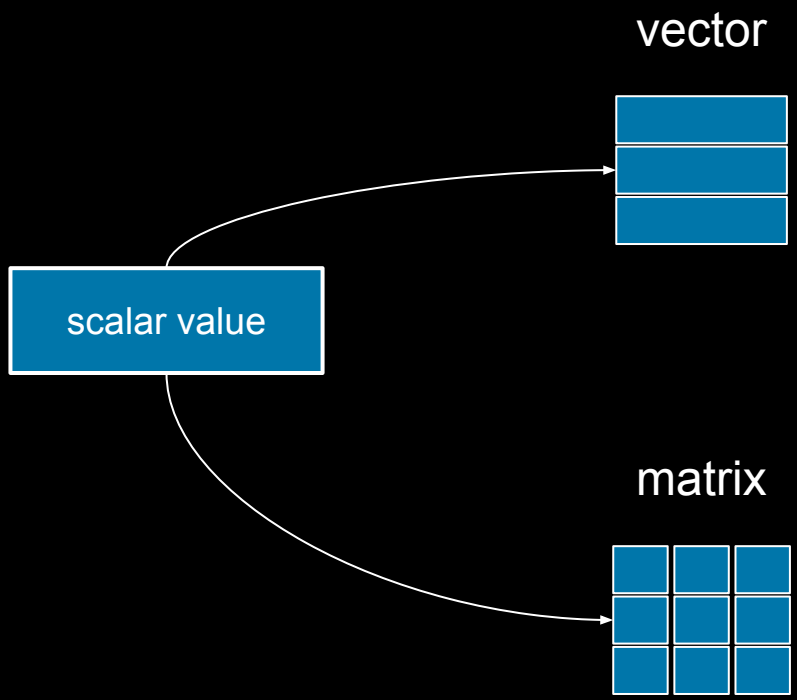


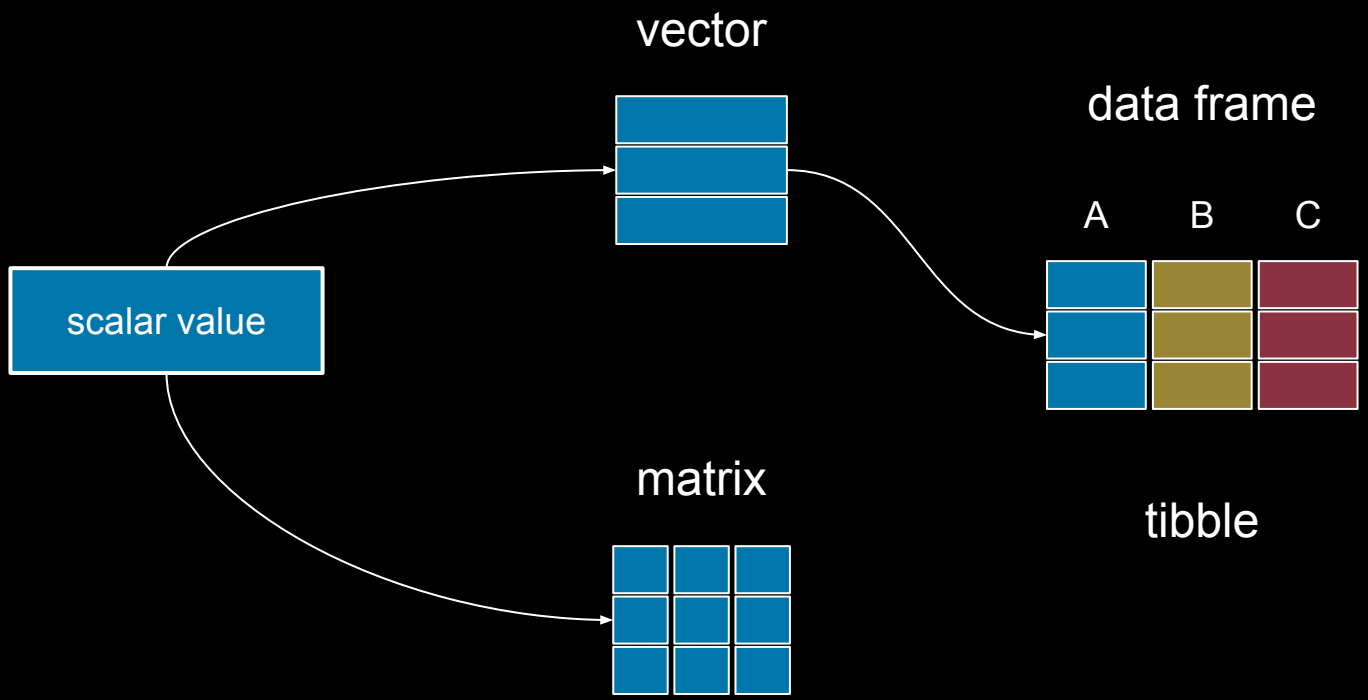
Following Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023.

DATA REPRESENTATION

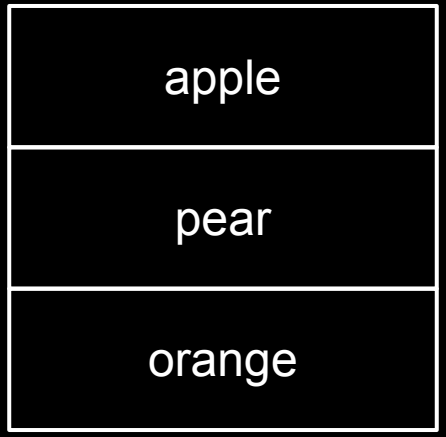

scalar value


vector
scalar value


vector
scalar value
matrix


vector
data frame
A
B
C
scalar value
matrix
tibble

VECTORS


apple
pear
orange

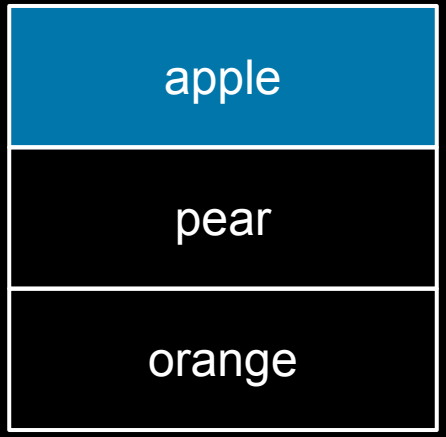
list of values with
the same storage mode

list of values with
the same storage mode

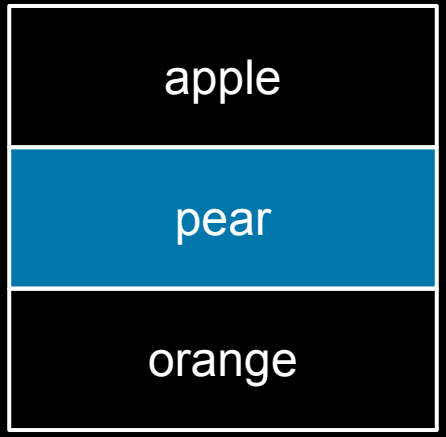
character
double
integer
logical

```
v <- c("apple", "pear", "orange")
```

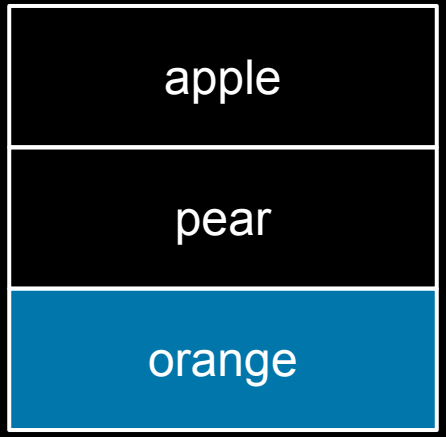
`v[1]`



v[2]



`v[3]`

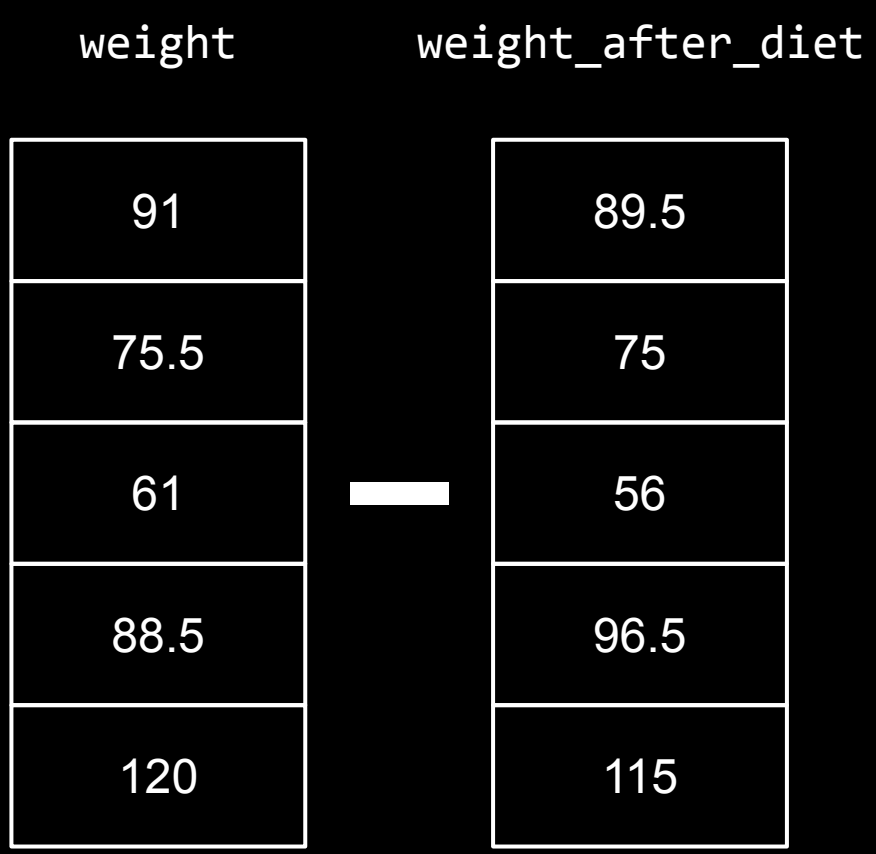


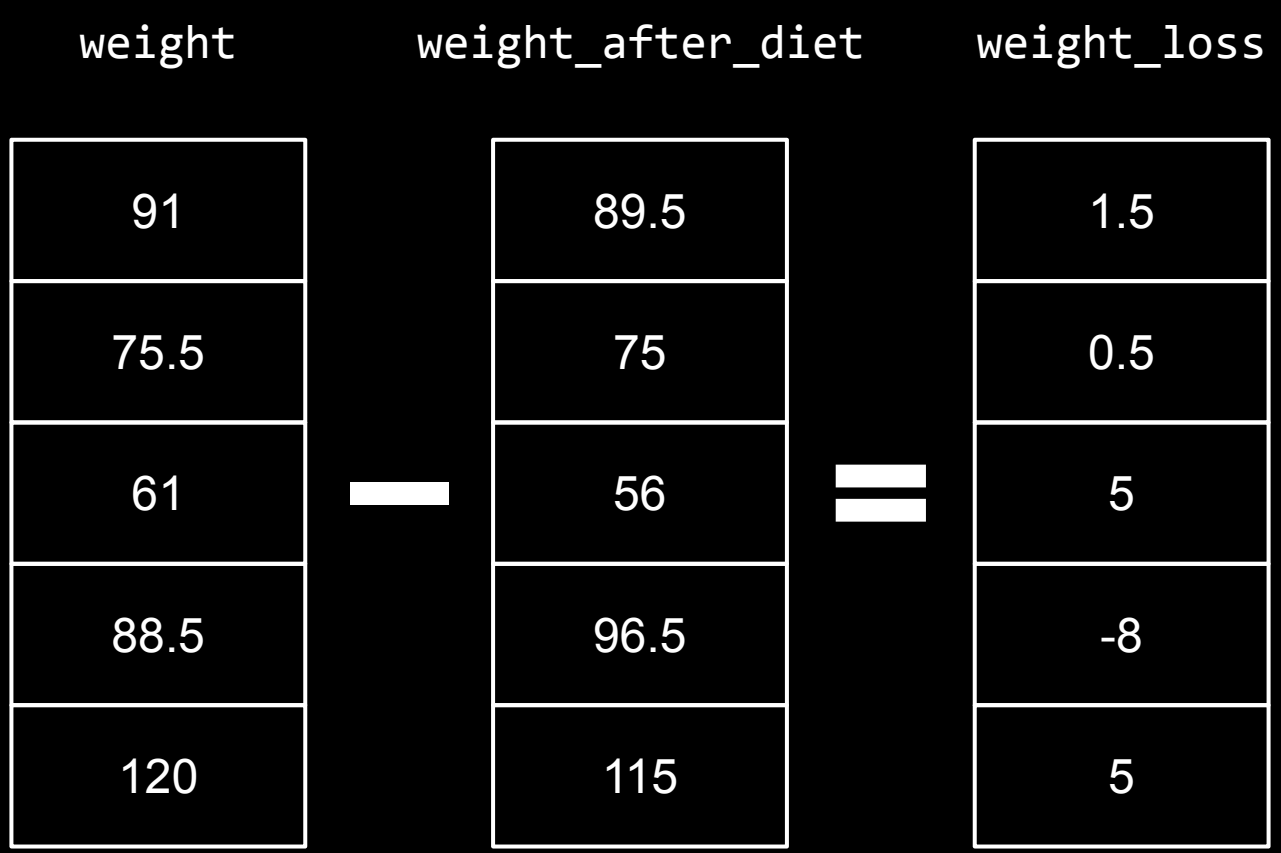
```
weight <- c(91, 75.5, 61, 88.5, 120)
```

```
weight <- c(91, 75.5, 61, 88.5, 120)
mean(weight)
```

```
sum         length
mean        sort
median      cumsum
sd          prod
var         quantile
min         abs
max         range
```

```
weight_after_diet <-
   c(89.5, 75, 56, 96.5, 115)
```


weight
weight_after_diet
91
75.5
61
88.5
120
89.5
75
56
96.5
115


weight
weight_after_diet
weight_loss
91
75.5
61
88.5
120
89.5
75
56
96.5
115
1.5
0.5
5
-8
5

```
weight_loss <-
   weight - weight_after_diet
```

subsetting vectors

weight[1]

```
weight[1]
weight[-1]
```

```
weight[1]
weight[-1]
weight[2:5]
```

```
weight[1]

weight[-1]

weight[2:5]

weight[1:length(weight)-1]
```

```
weight[1]
weight[-1]
weight[2:5]
weight[1:length(weight)-1]
weight[c(TRUE, FALSE, TRUE, TRUE, FALSE)]
```

```
weight[1]

weight[-1]

weight[2:5]

weight[1:length(weight)-1]

weight[c(TRUE, FALSE, TRUE, TRUE, FALSE)]

weight[weight > 80]
```

```
weight[1]

weight[-1]

weight[2:5]

weight[1:length(weight)-1]

weight[c(TRUE, FALSE, TRUE, TRUE, FALSE)]

weight[weight > 80]

weight[weight > 80 & weight < 100]
```

special values

NA
NULL
NaN
Inf
-Inf

factors

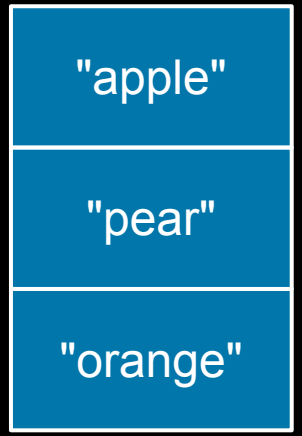
```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))
```

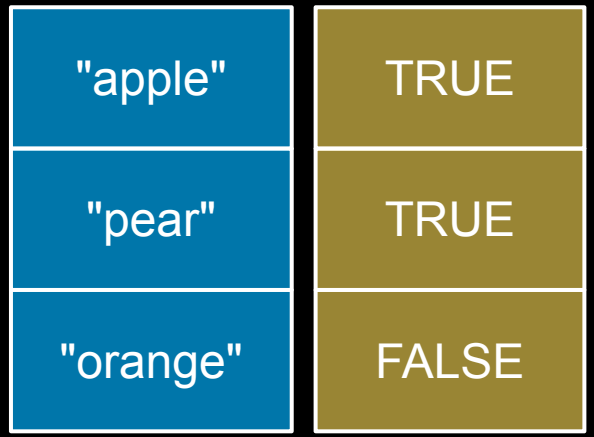
```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))

levels(weight_category)
```

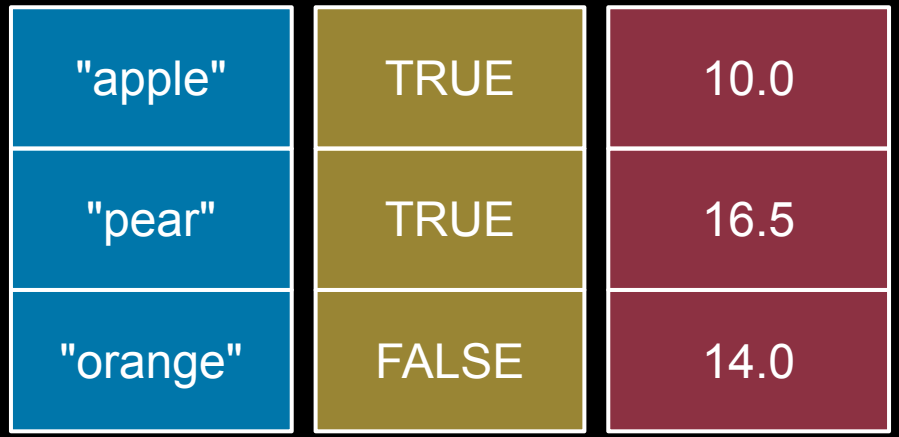
```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))

levels(weight_category)

category_reordered <- factor(category,
                             levels = c("light", "medium", "heavy"))
```

```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))

levels(weight_category)

category_reordered <- factor(category,
                             levels = c("light", "medium", "heavy"))

category_ordered <- factor(category,
                           levels = c("light", "medium", "heavy"),
                           ordered = TRUE)
```

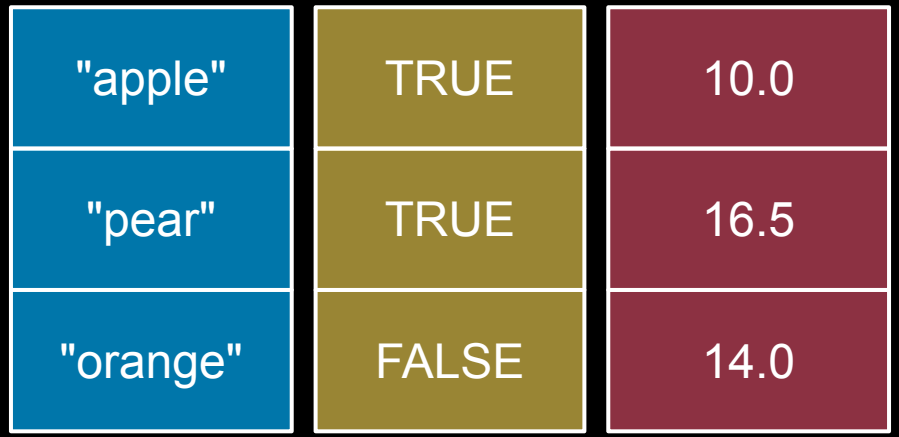
DATA FRAMES

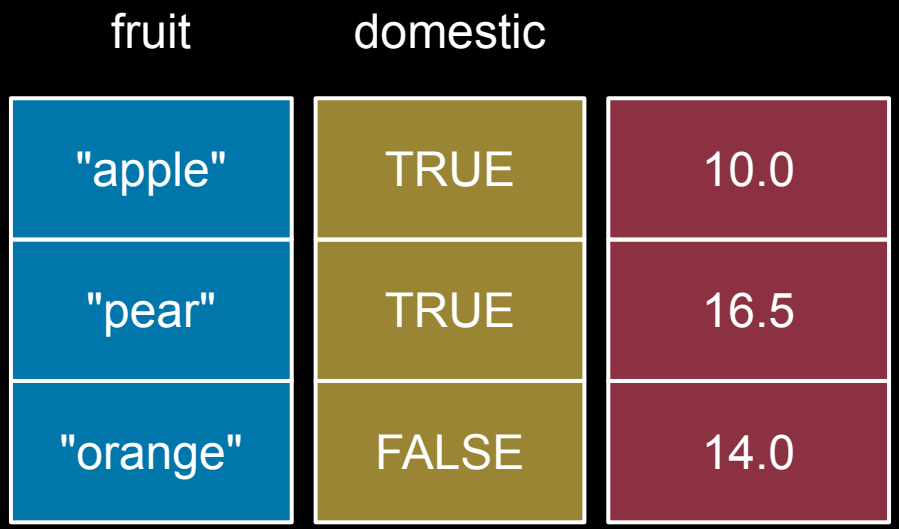

"apple"
"pear"
"orange"

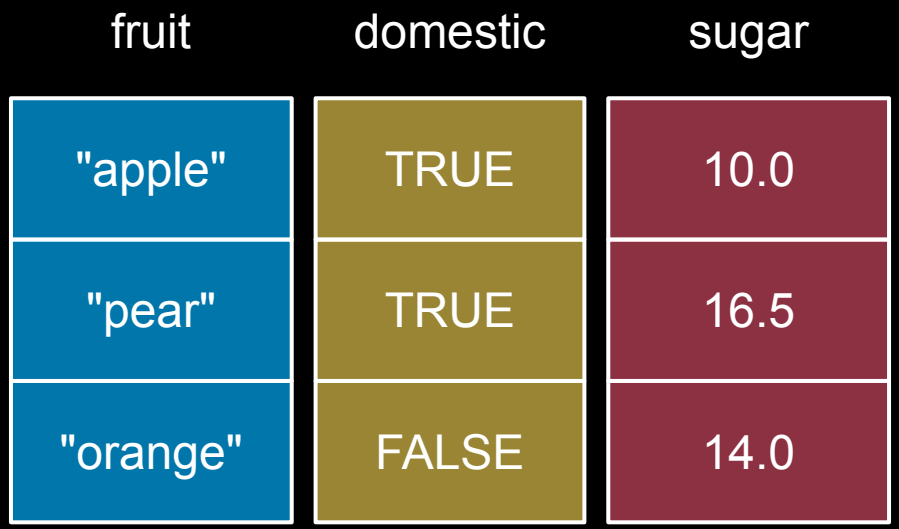

"apple"
TRUE
"pear"
TRUE
"orange"
FALSE

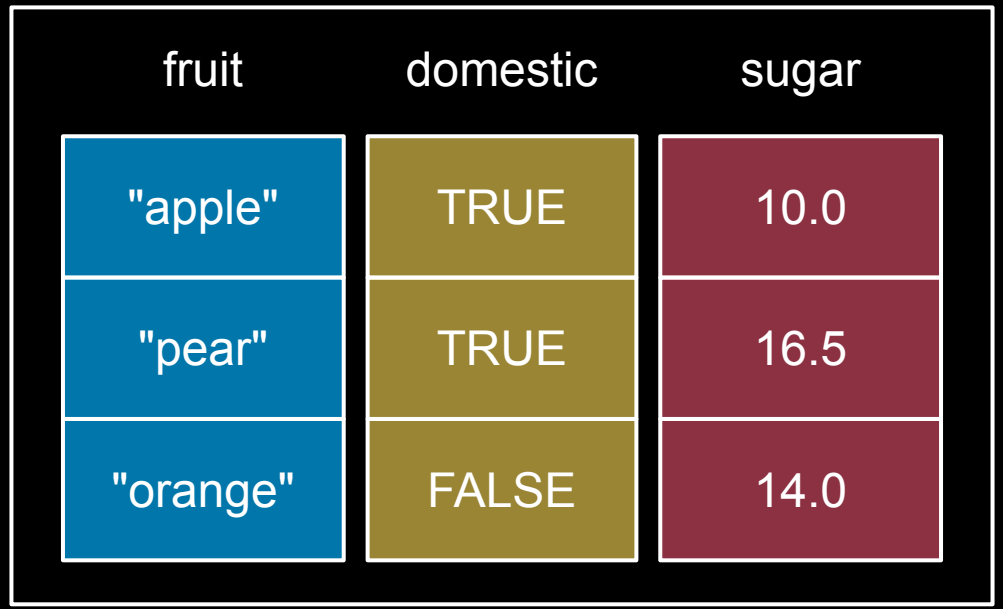

"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0

fruit

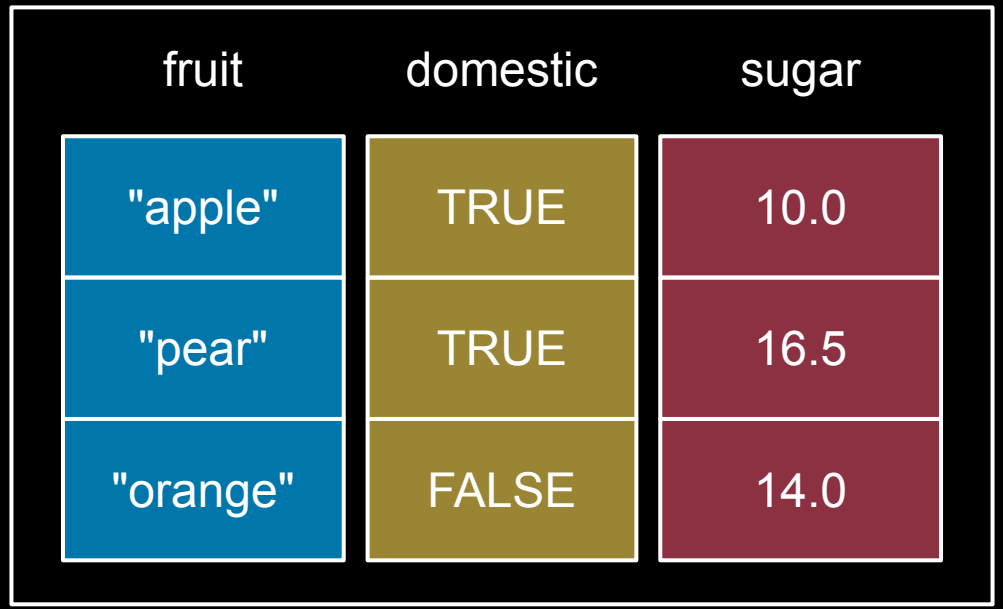



fruit
domestic
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0


fruit
domestic
sugar
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0


fruit
domestic
sugar
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0

data frame "fruits"



creating data frames

```
data.frame()
read.csv()
```

comma separated values (CSV)

data frame meta data

```
ncol()
nrow()
dim()
colnames()
```

accessing data frames

accessing data frames
accessing columns

```
monty$prize_door
monty$contestant_choice
monty$decision
```

```
monty$prize_door
monty$contestant_choice
monty$decision
```

return a vector

```
monty["prize_door"]
monty["contestant_choice"]
monty["decision"]
```

```
monty["prize_door"]
monty["contestant_choice"]
monty["decision"]
```

return a data frame

```
# multiple columns by name
monty(c("prize_door", "contestant_choice"))
```

```
monty[, 1]               # first column
monty[, 1:2]             # first two columns
monty[, ncol(monty)]     # last column
```

accessing data frames

accessing rows

```
monty[1,]                # first row
monty[1:10,]             # first 10 rows
monty[nrow(monty),]      # last row
```

changing columns

```
monty$decision <- as.factor(monty$decision)
```

adding columns

```
monty$correct_guess <-
   monty$contestant_choice == monty$prize_door
```

rename columns

```
colnames(monty)[2] <- "choice"
```

subsetting data frames

```
switched <-
   monty[monty_hall$decision == "switch", ]
```

```
switched <-
   monty[
      monty_hall$decision == "switch &
      monty$won == TRUE, ]
```

subset()

```
subset(monty, decision == "switch")
```

```
subset(
      monty,
      decision == "switch" & won == TRUE
)
```

sorting rows

```
monty[order(monty$prize_door),]
```

```
monty[order(monty$prize_door),]

monty[order(
         monty$prize_door,
         decreasing = TRUE
         ),]
```

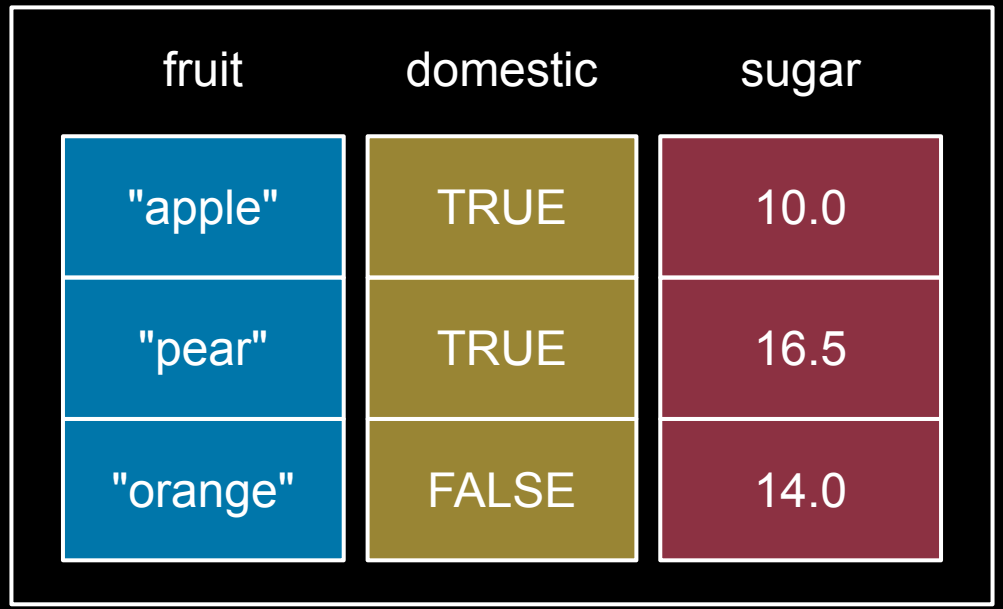
saving data frames

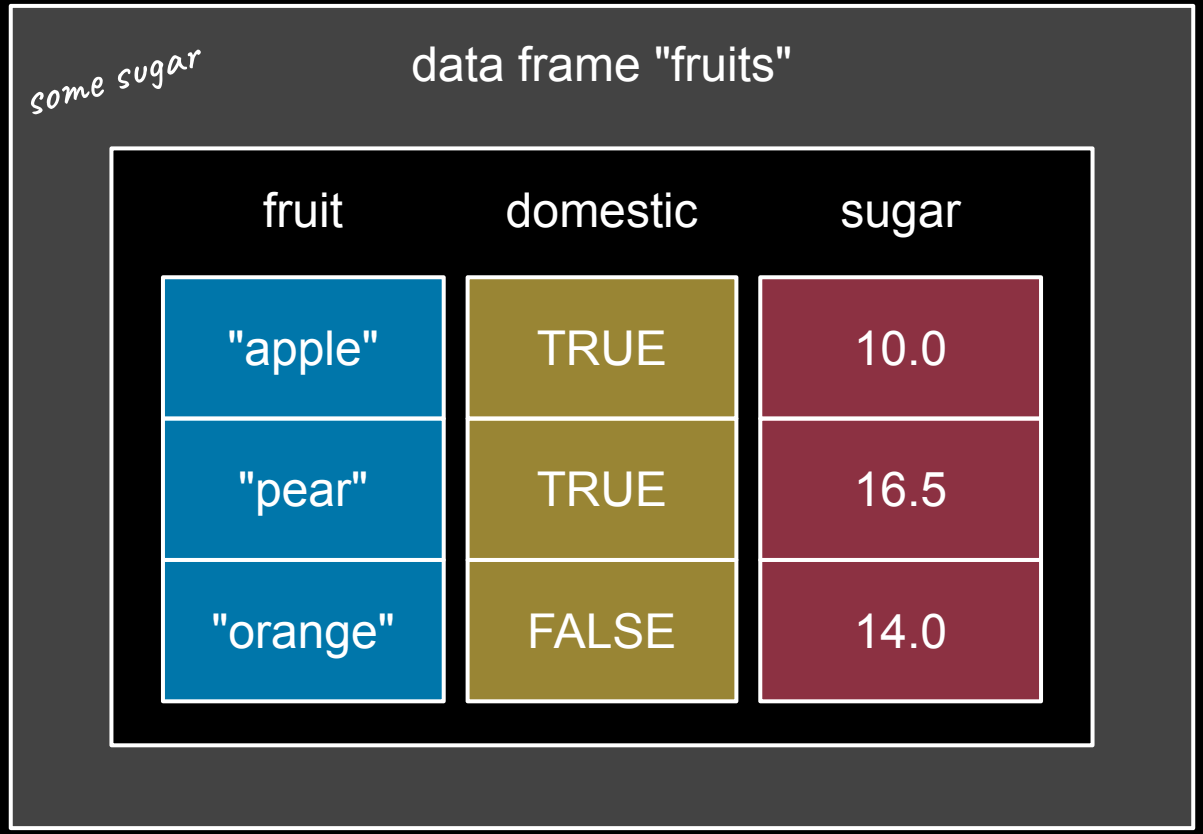
```
write.csv()
```

tibbles

{{ tibble }}

data frame "fruits"




some sugar
data frame "fruits"
fruit
domestic
sugar
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0
tibble "tbl_fruits"

```
as.tibble()
```

some sugar

better printing
subsets stay tibbles
better data type guessing
support for extended data types
…

LOAD DATA

{{ readr }}

```
read_csv()
read_delim()
```

{{ readxl }}

read_excel()

TIDY DATA

tidy data

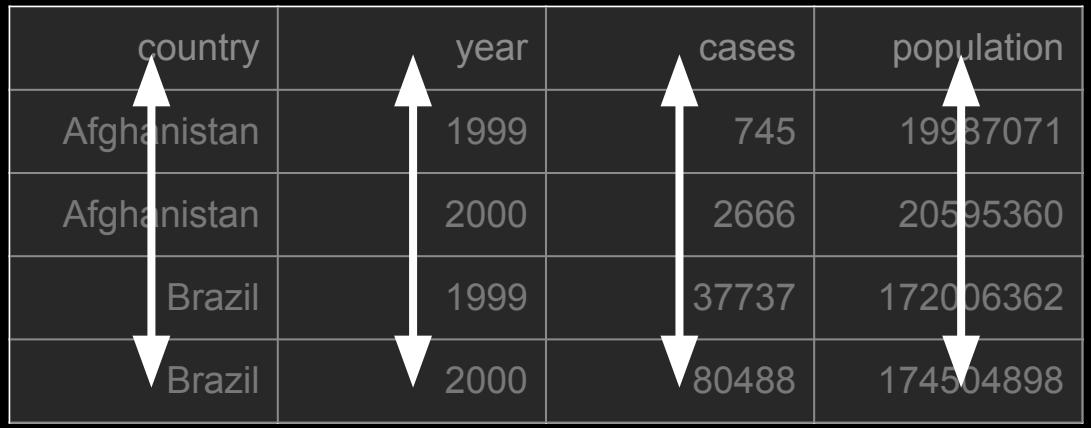
each variable is a column;
each column is a variable.

each observation is a row;
each row is an observation.

each value is a cell;
each cell is a single value.

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898

Example from: Wickham, Hadley, et al. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition, O'Reilly Media, Inc, 2023. Chapter 5.


country
year
cases
population
Afghanistan
1999
745
19987071
Afghanistan
2000
2666
20595360
Brazil
1999
37737
172006362
Brazil
2000
80488
174504898

variables

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898

observations

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898

values

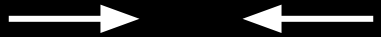
country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898

wider

country	cases_1999	cases_2000	pop_1999	pop_2000
Afghanistan	745	2666	19987071	20595360
Brazil	37737	172006362	80488	174504898

compressed



country	year	rate
Afghanistan	1999	745 / 19987071
Afghanistan	2000	2666 / 20595360
Brazil	1999	37737 / 172006362
Brazil	2000	80488 / 174504898

tidy

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898

tidy

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898

vector

{{ tidyr }}

pivot_wider()

pivot_longer()

STRINGS

{{ stringr }}

```
str_trim()
str_squish()
```

```
str_starts()
str_ends()
str_detect()
```

“Annabel Miller”

“Annabel Miller”

```
str_starts(txt, "Anna")
```

“Annabel Miller”

```
str_ends(txt, "Miller")
```

“Annabel Miller”

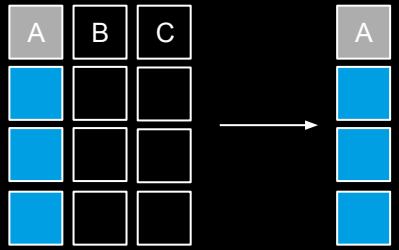
```
str_detect(txt, "Mill")
```

TRANSFORM DATA

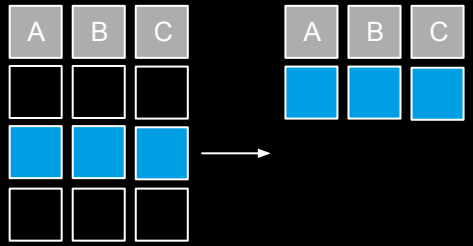
{{ dplyr }}

types of transformations

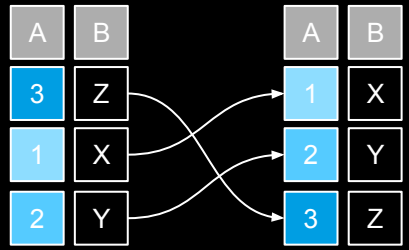
select()



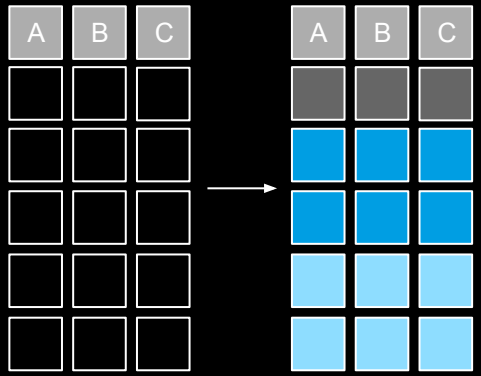
filter()



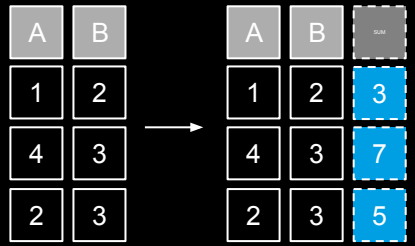
arrange()



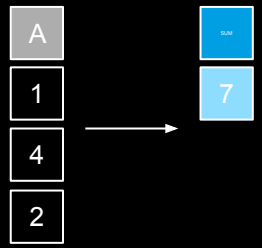
group_by()



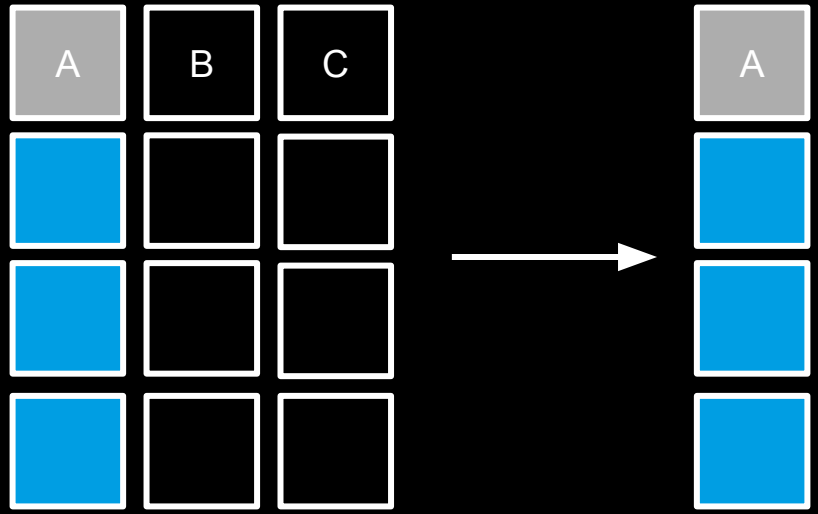
mutate()



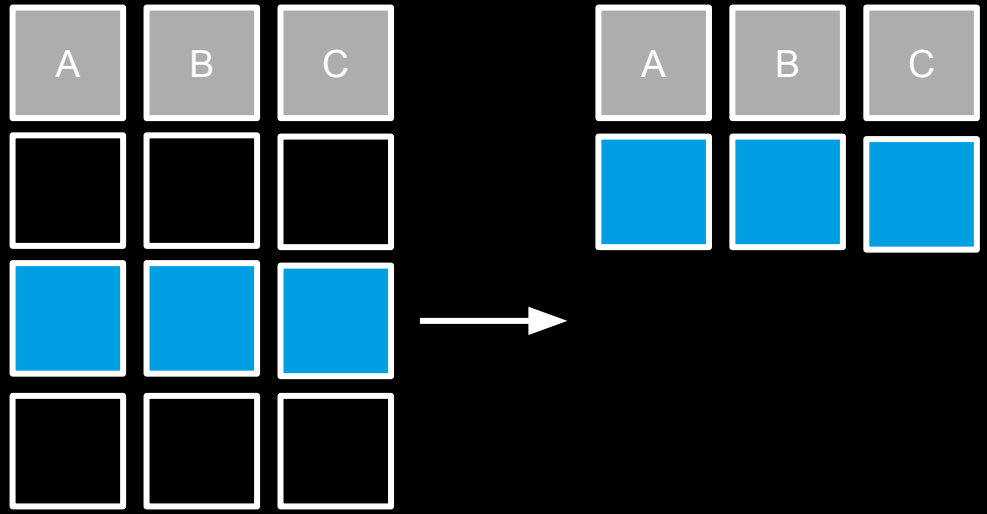
summarize()



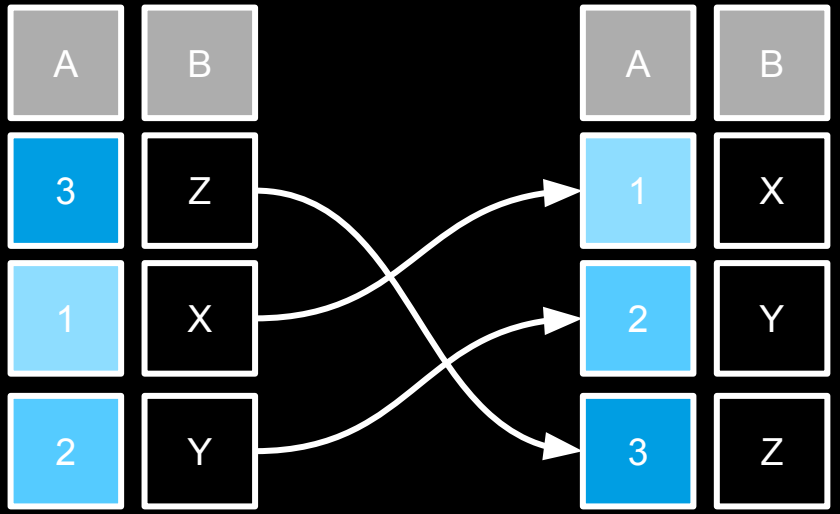
select()



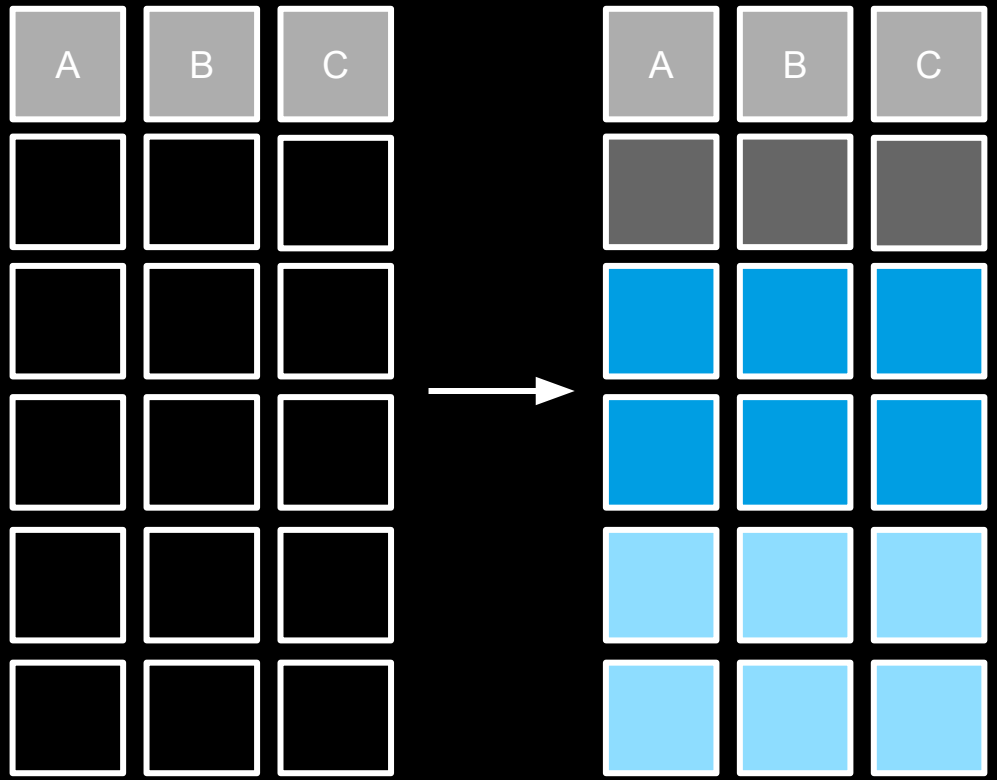
filter()



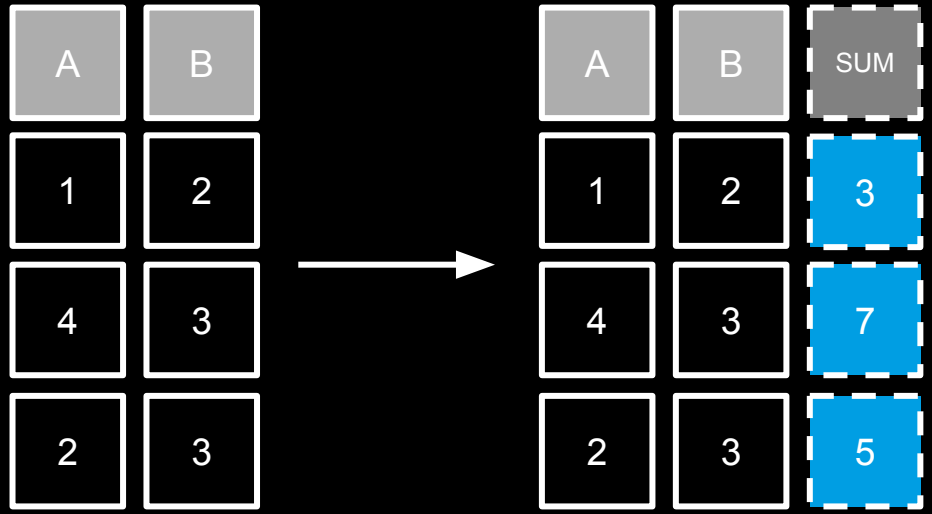
arrange()



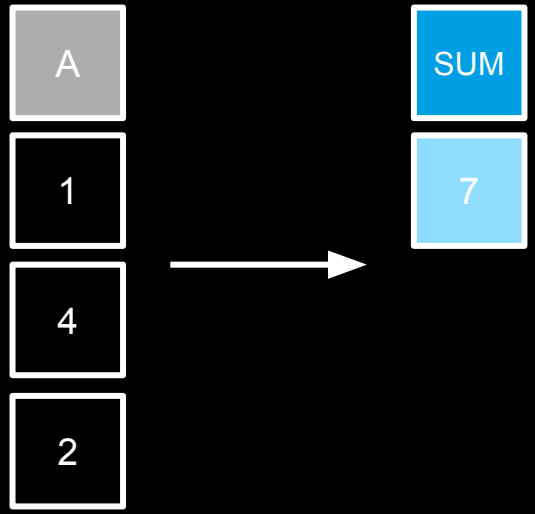
group_by()



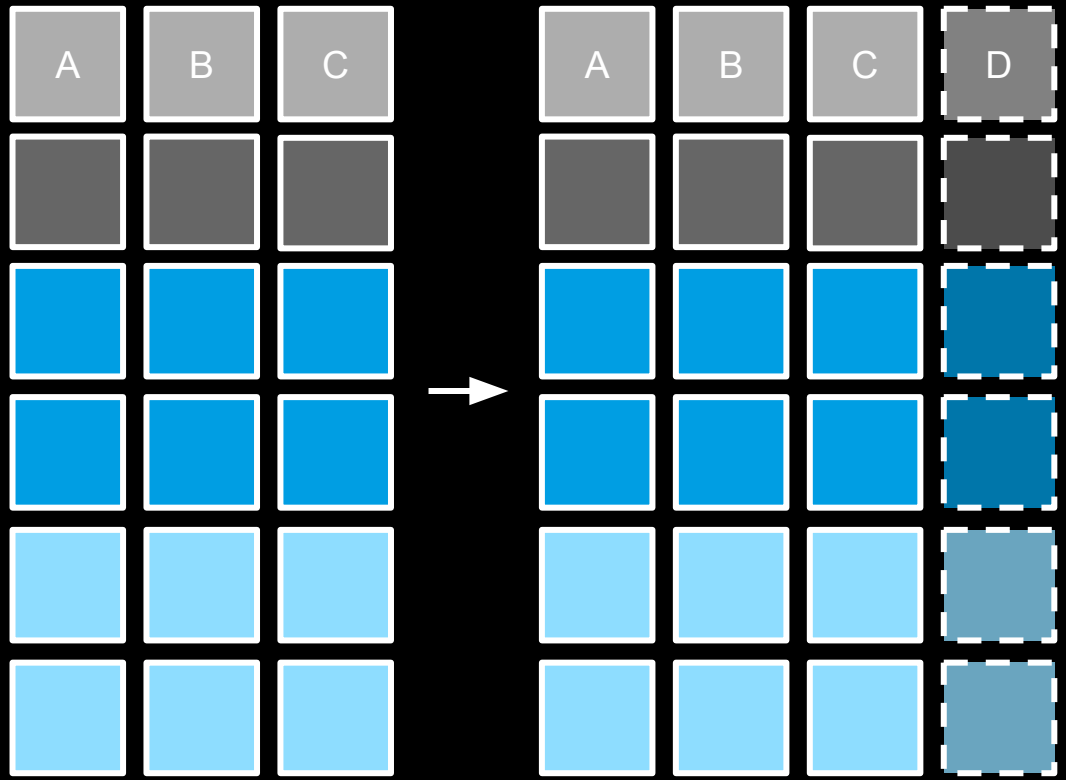
mutate()



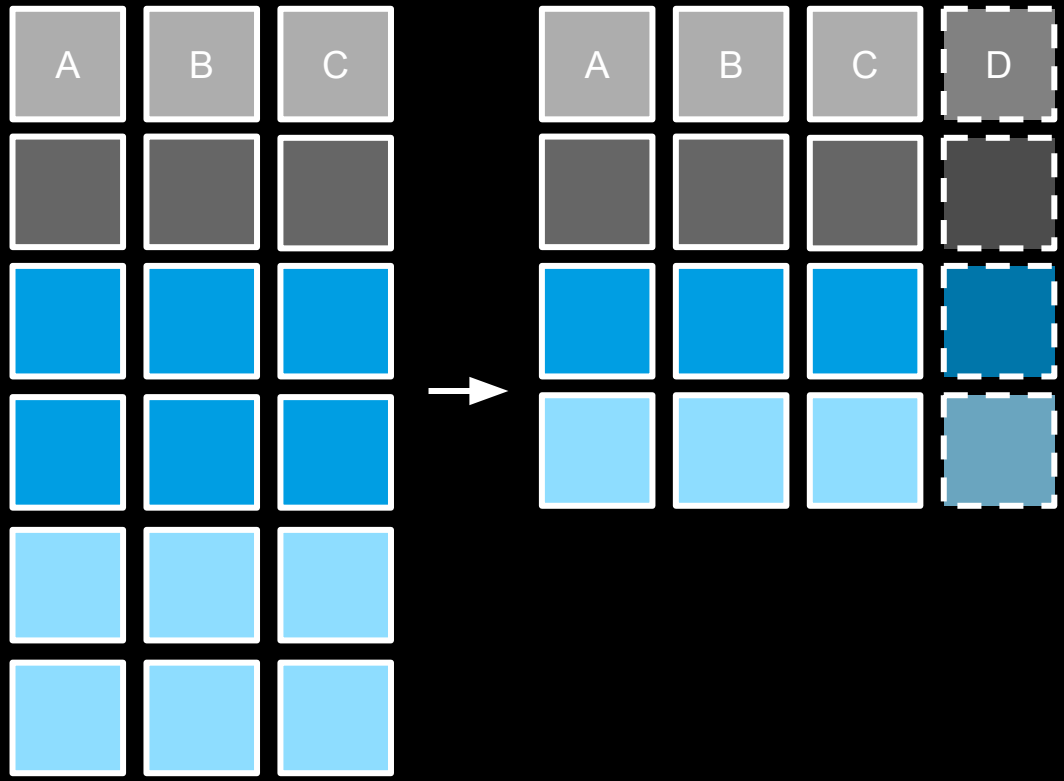
summarize()



```
group_by()
+
mutate()
```

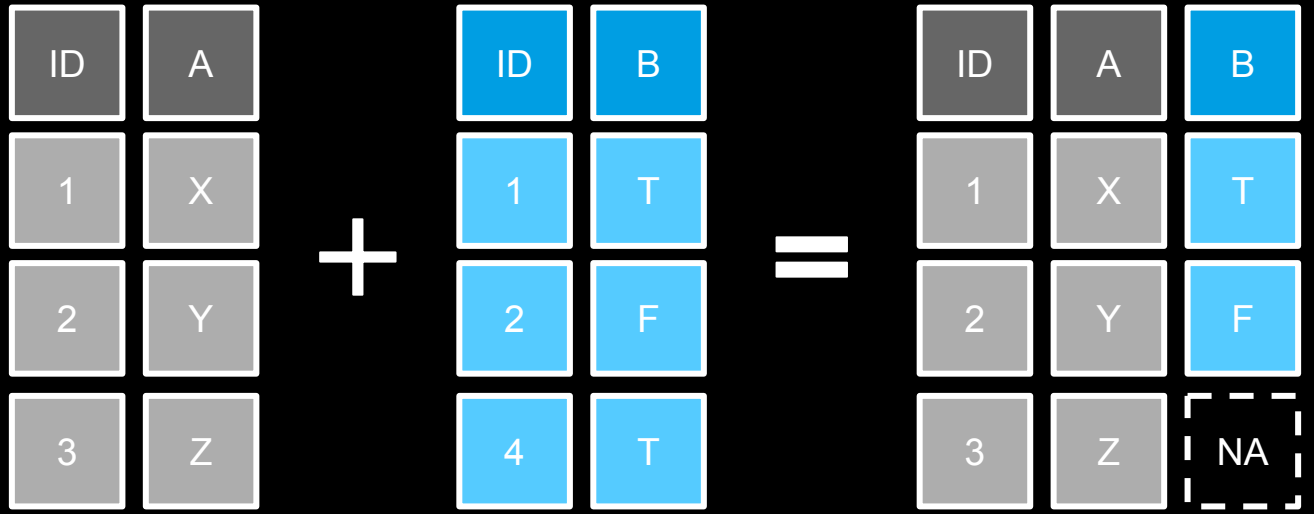


group_by()
+
summarize()

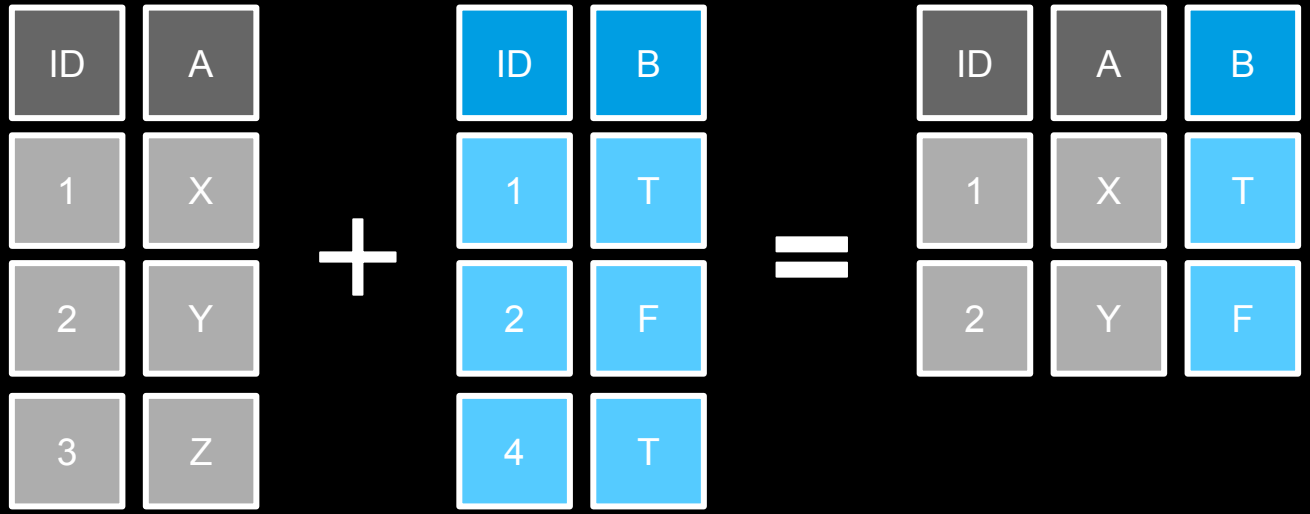


joining data

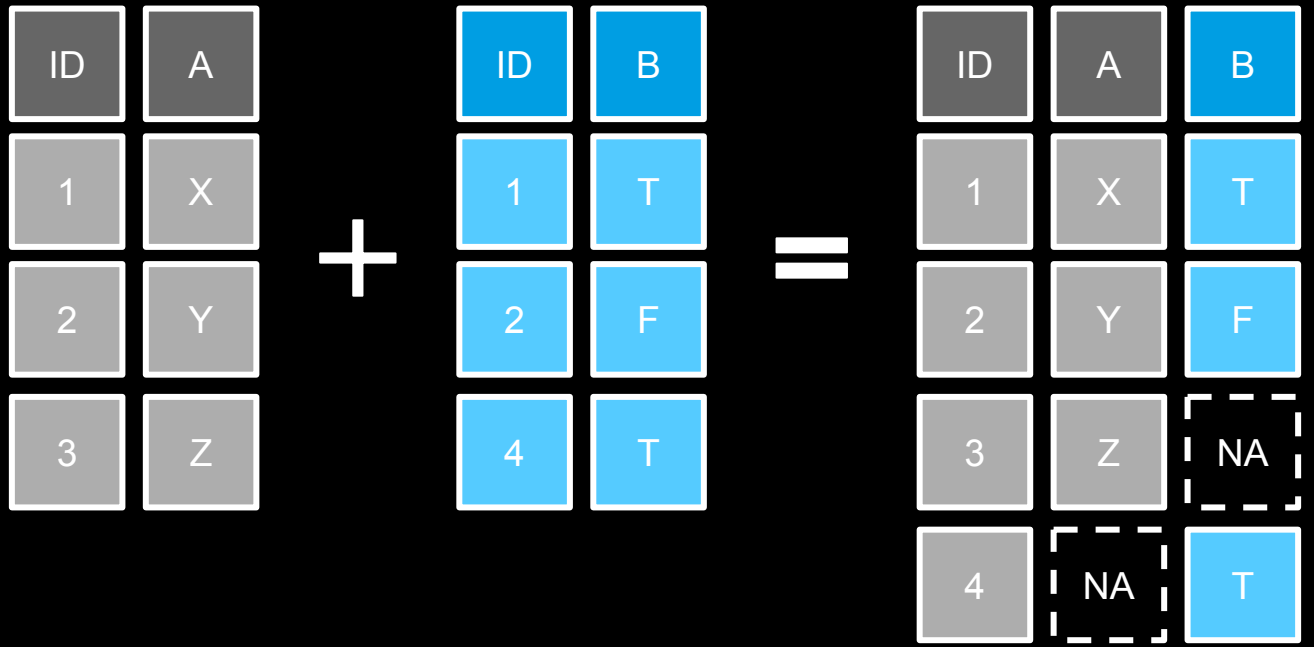
left_join()


ID
A
1
X
2
Y
3
Z
+
ID
B
1
T
2
F
4
T
=
ID
A
B
1
X
T
2
Y
F
3
Z
NA

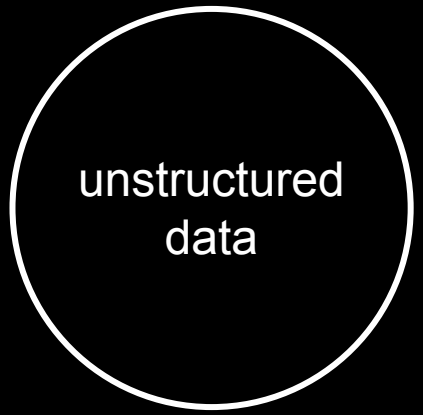
inner_join()


ID
A
1
X
2
Y
3
Z
+
ID
B
1
T
2
F
4
T
=
ID
A
B
1
X
T
2
Y
F

full_join()


ID
A
1
X
2
Y
3
Z
+
ID
B
1
T
2
F
4
T
=
ID
A
B
1
X
T
2
Y
F
3
Z
NA
4
NA
T

UNSTRUCTURED DATA


unstructured
data

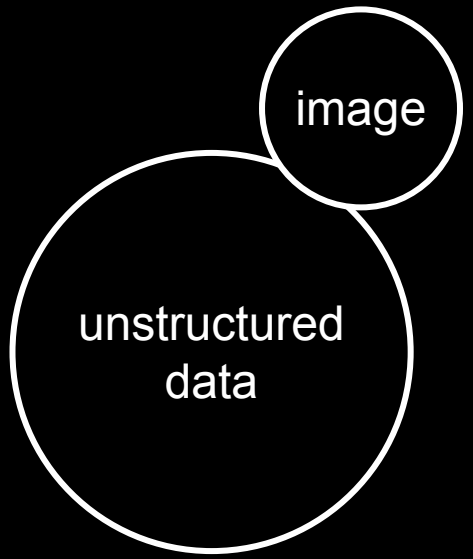


image
unstructured
data

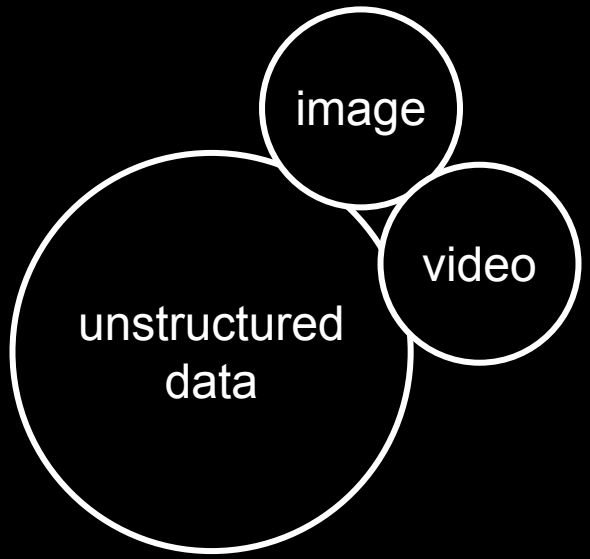


image
video
unstructured
data

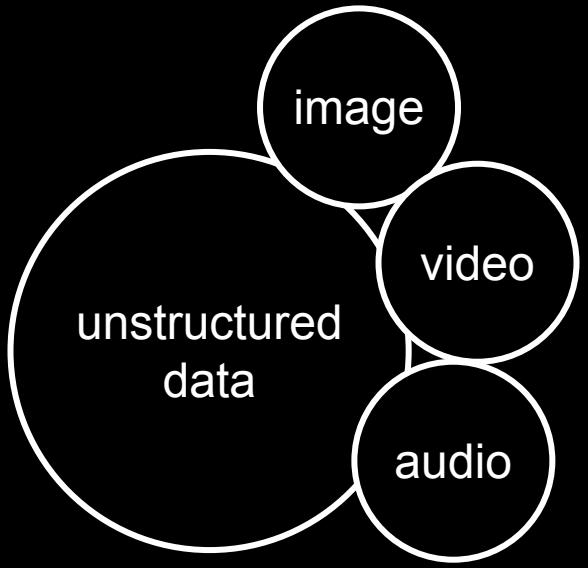


image
video
unstructured
data
audio

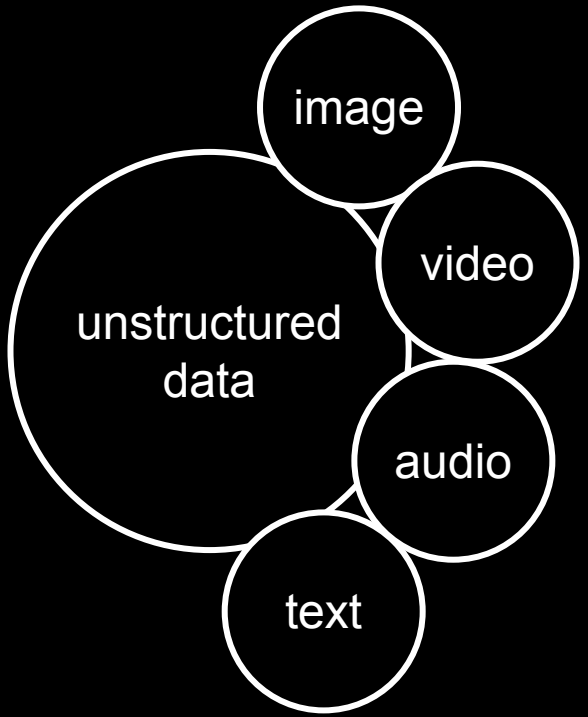
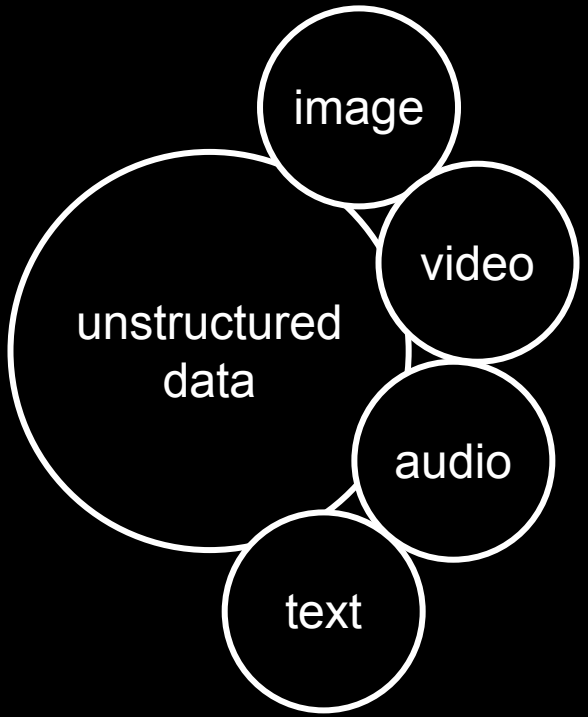
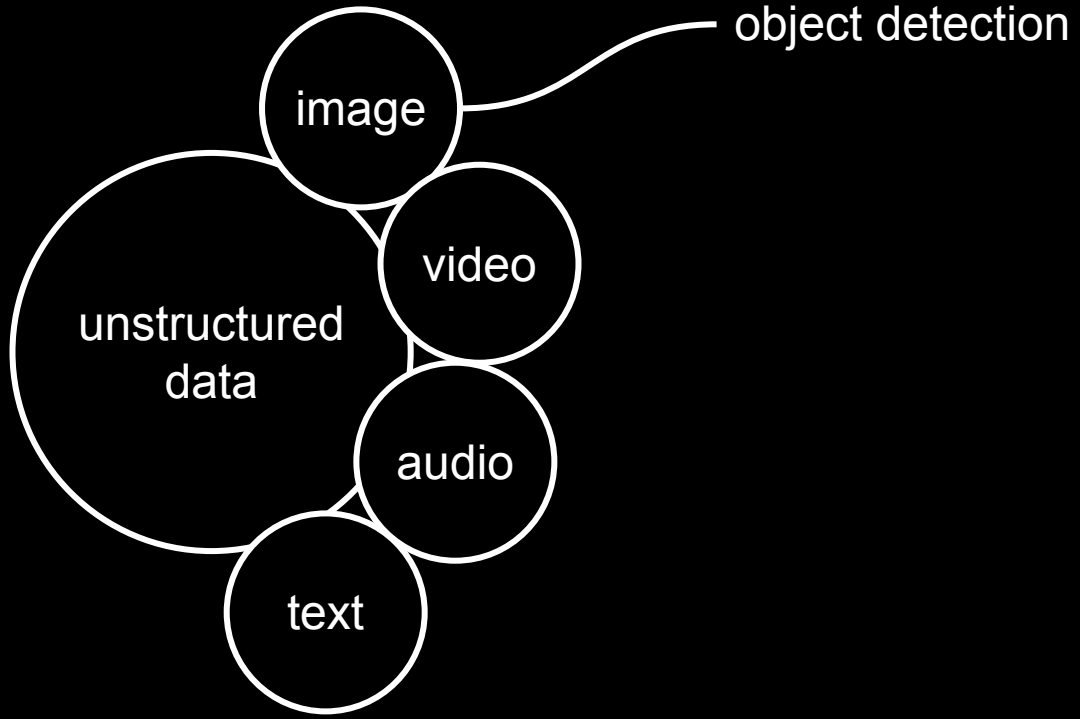


image
video
unstructured
data
audio
text

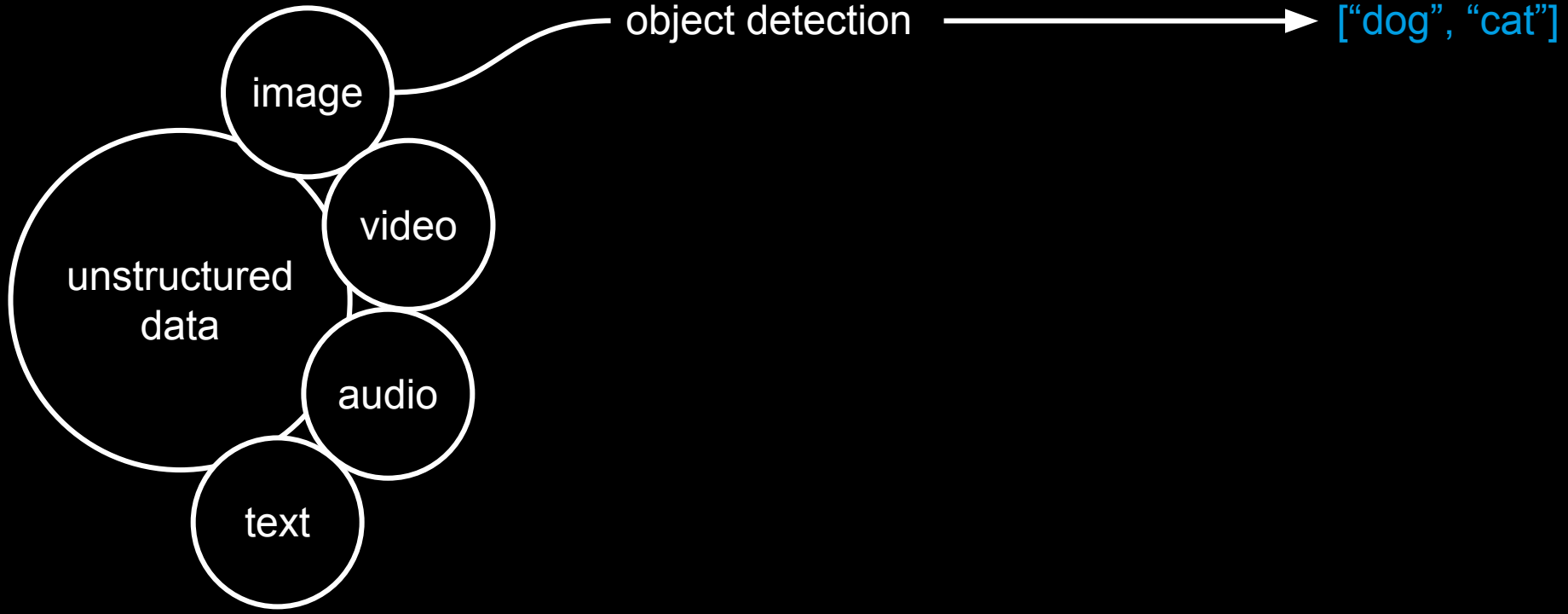
no handles to grab



no handles to grab



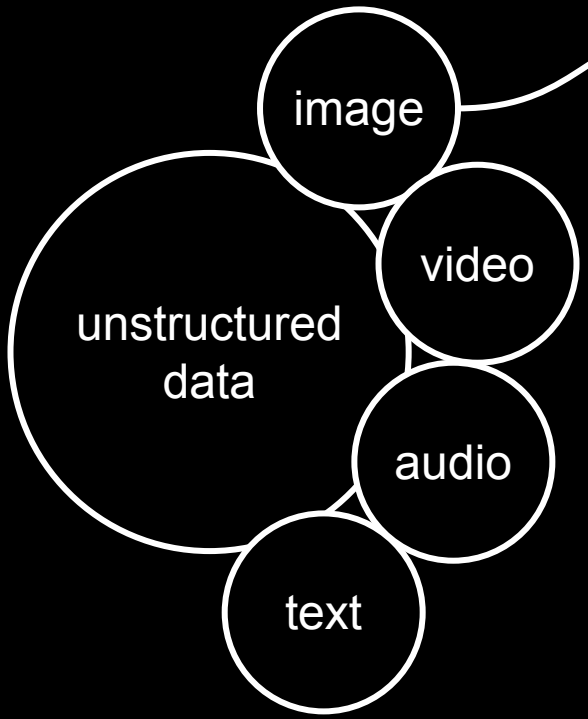
no handles to grab



no handles to grab

algorithm

extracted, structured information

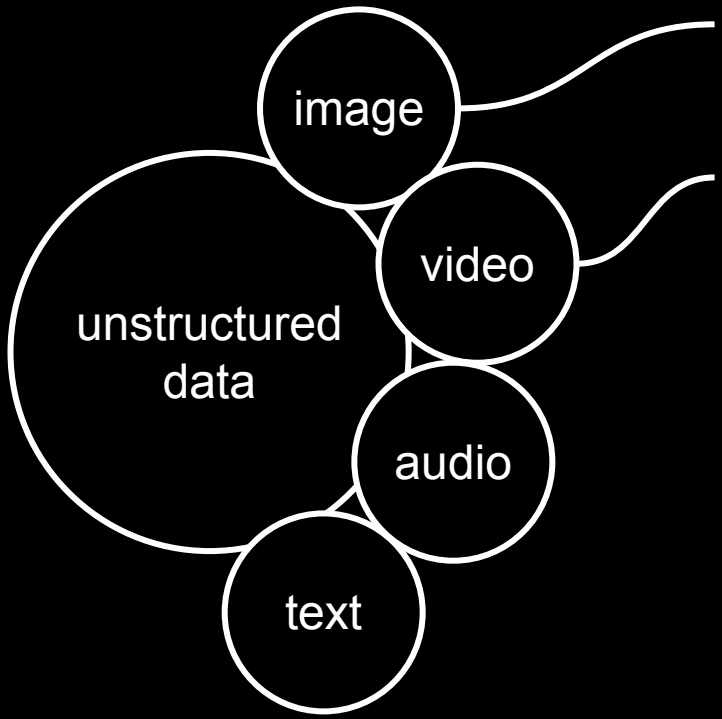


object detection → [“dog”, “cat”]

no handles to grab

algorithm

extracted, structured information



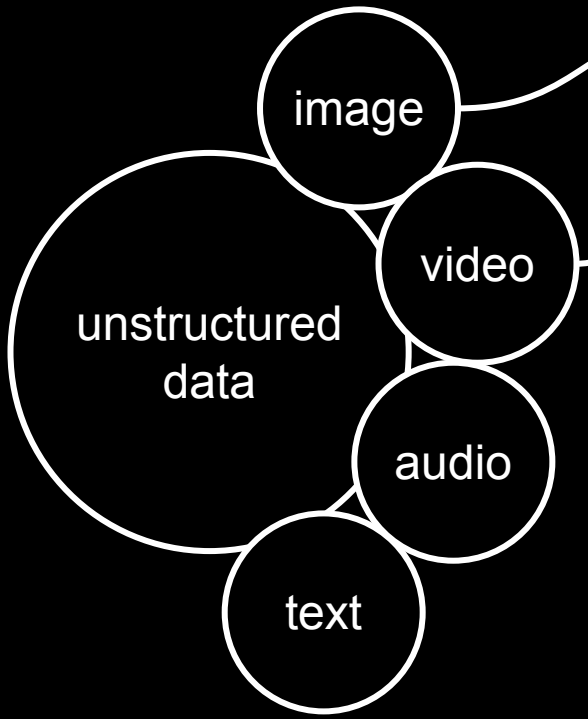
object detection → ["dog", "cat"]

object tracking

no handles to grab

algorithm

extracted, structured information



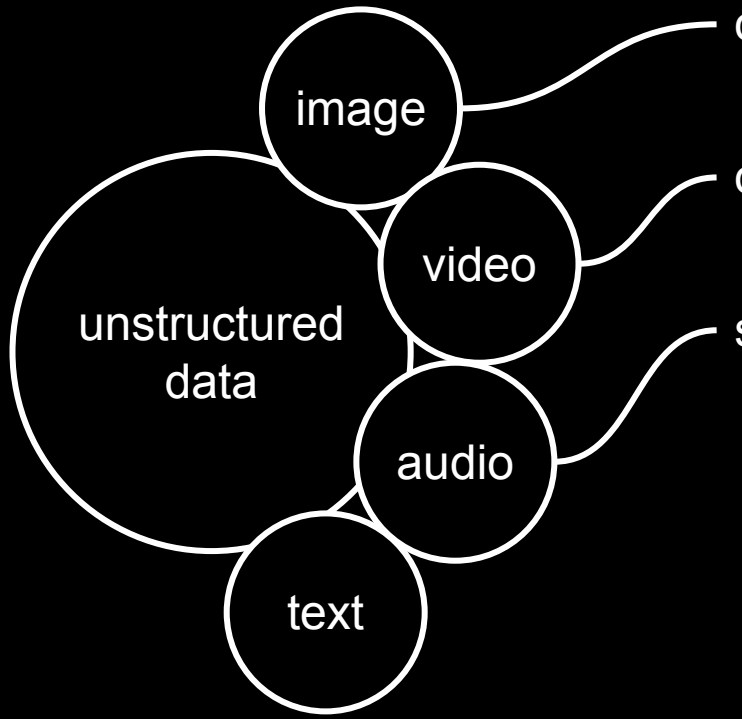
object detection → [“dog”, “cat”]

object tracking → { "t": 2.41, "x": 34, "y": 55 }

no handles to grab

algorithm

extracted, structured information



object detection → ["dog", "cat"]

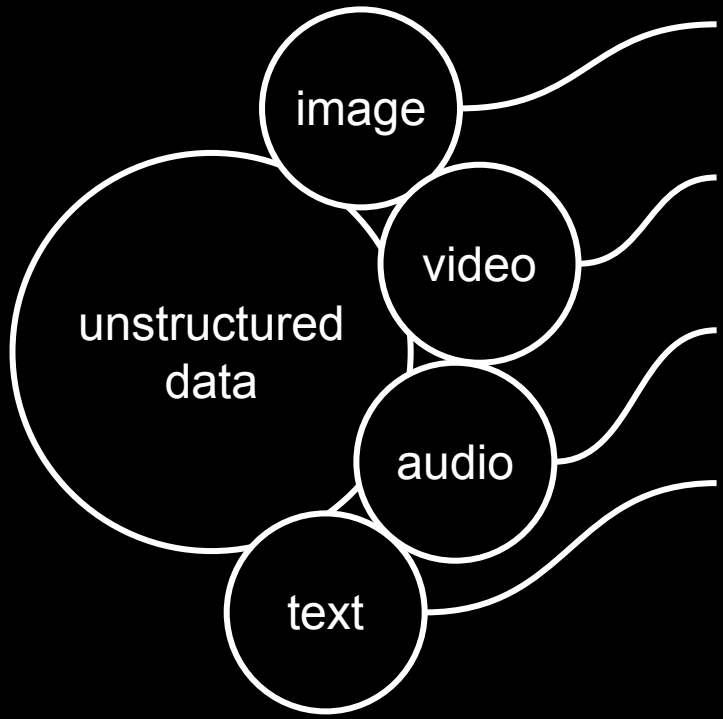
object tracking → { "t": 2.41, "x": 34, "y": 55 }

speech-to-text → { "t": 0.3, "text": "Guten Abend" }

no handles to grab

algorithm

extracted, structured information

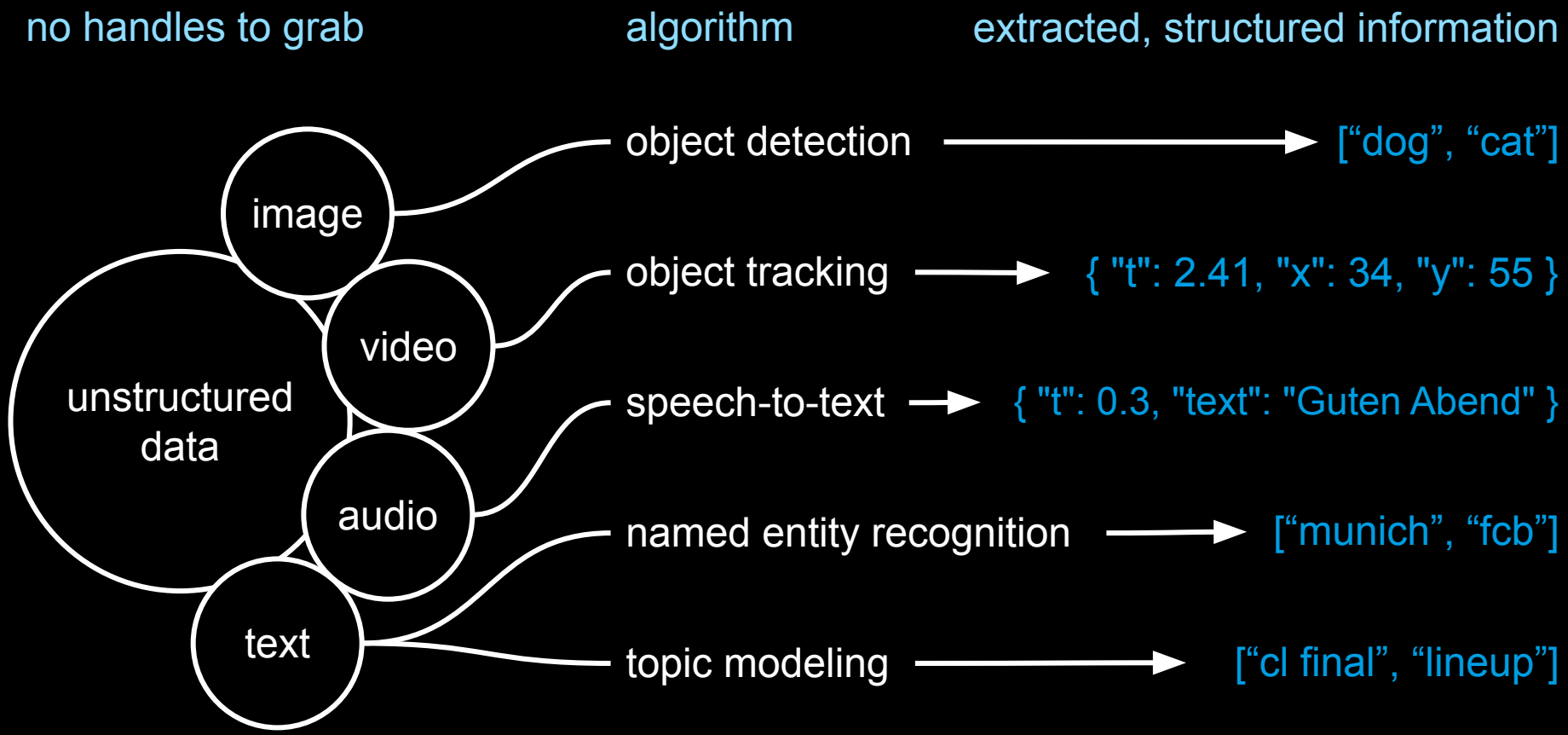


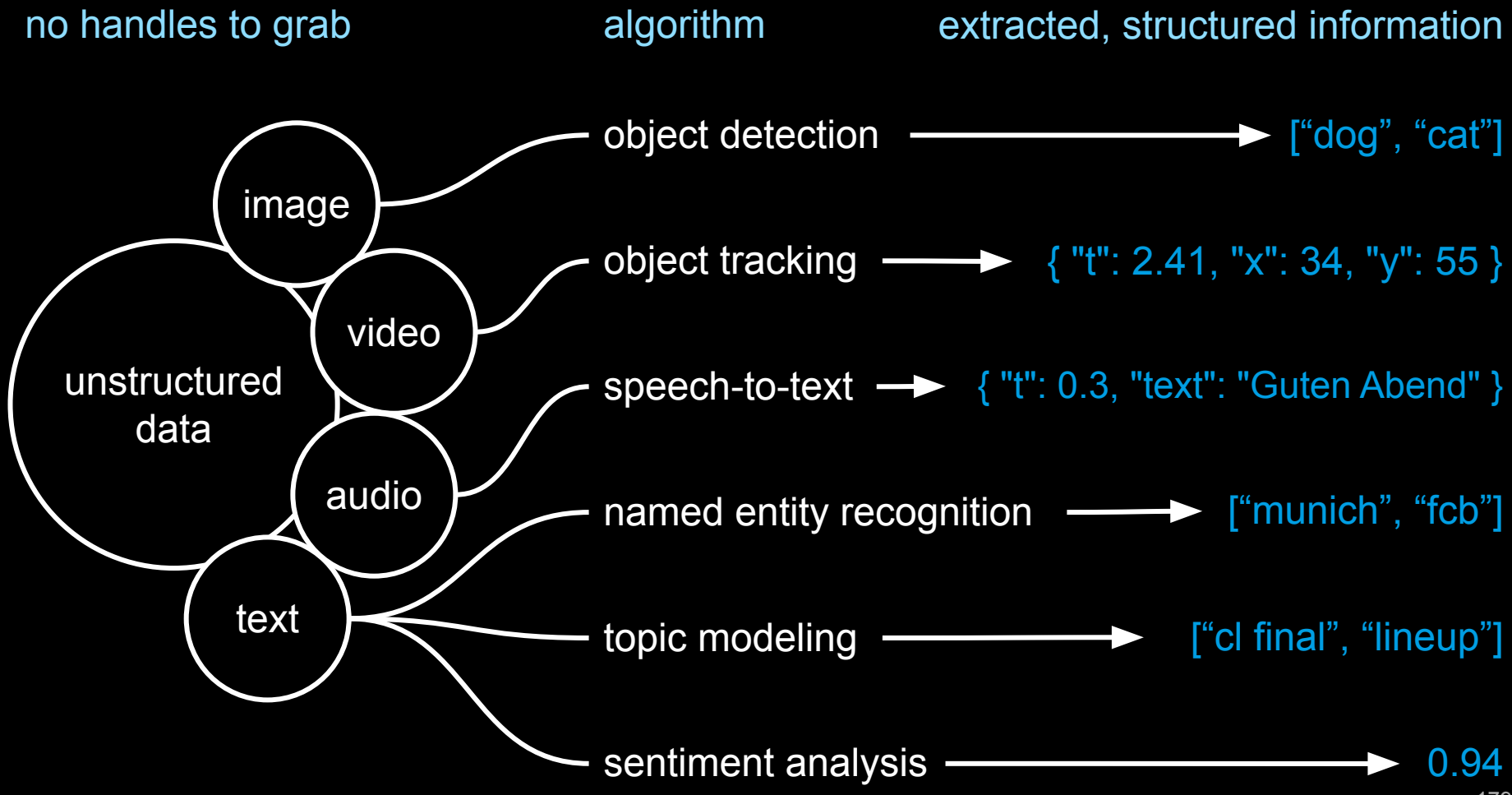
object detection → ["dog", "cat"]

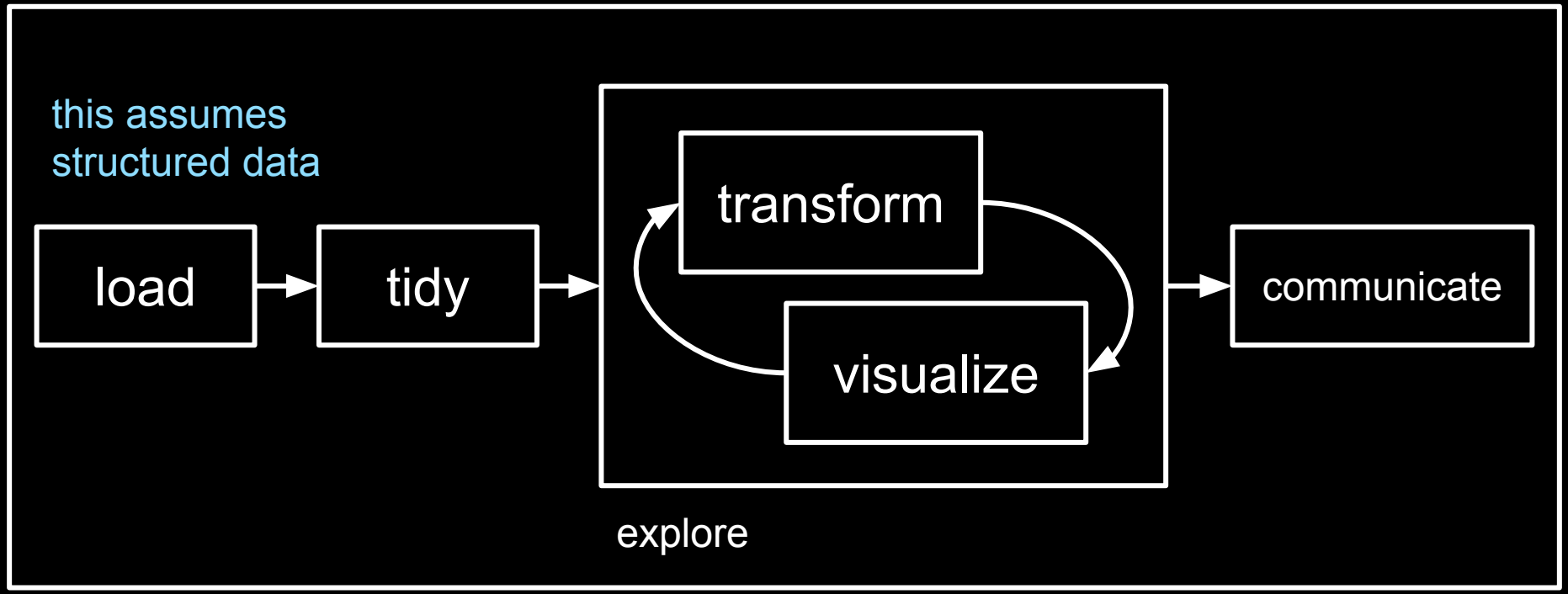
object tracking → { "t": 2.41, "x": 34, "y": 55 }

speech-to-text → { "t": 0.3, "text": "Guten Abend" }

named entity recognition → ["munich", "fcb"]

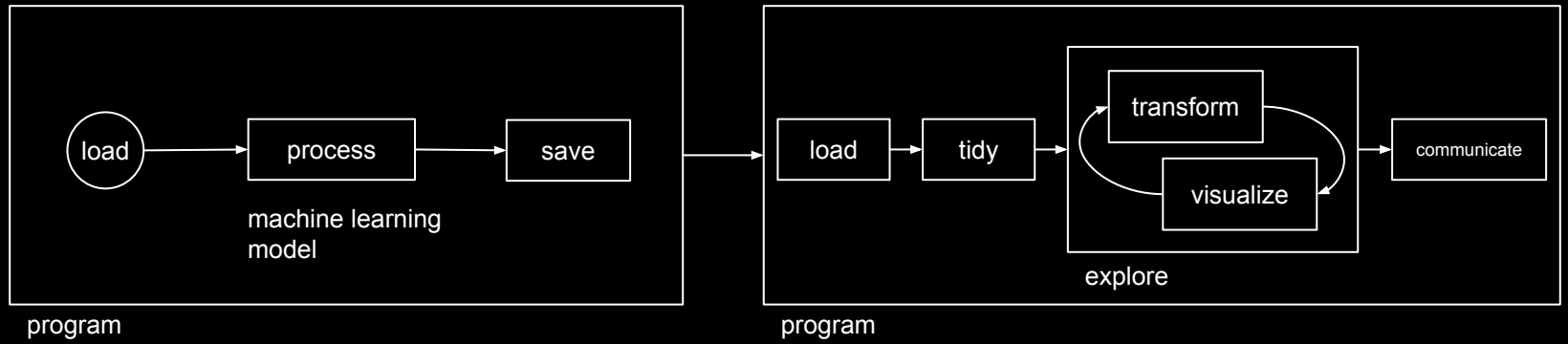

no handles to grab
algorithm
extracted, structured information
image
video
audio
text
unstructured data
object detection
["dog", "cat"]
object tracking
{ "t": 2.41, "x": 34, "y": 55 }
speech-to-text
{ "t": 0.3, "text": "Guten Abend" }
named entity recognition
["munich", "fcb"]
topic modeling
["cl final", "lineup"]


no handles to grab
algorithm
extracted, structured information
image
video
audio
text
unstructured data
object detection
["dog", "cat"]
object tracking
{ "t": 2.41, "x": 34, "y": 55 }
speech-to-text
{ "t": 0.3, "text": "Guten Abend" }
named entity recognition
["munich", "fcb"]
topic modeling
["cl final", "lineup"]
sentiment analysis
0.94


this assumes
structured data
load
tidy
transform
visualize
explore
communicate
program

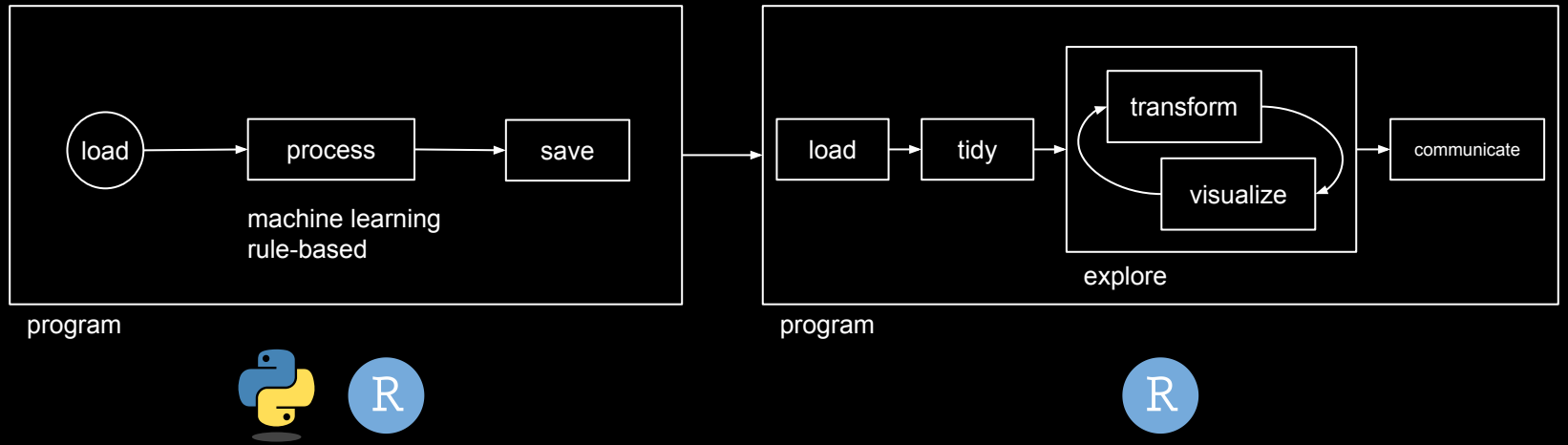
pre-process
unstructured data

exploratory data
analysis



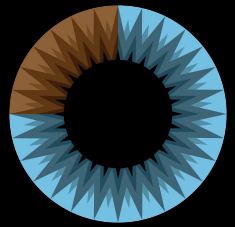
pre-process
unstructured data

exploratory data
analysis



MACHINE LEARNING

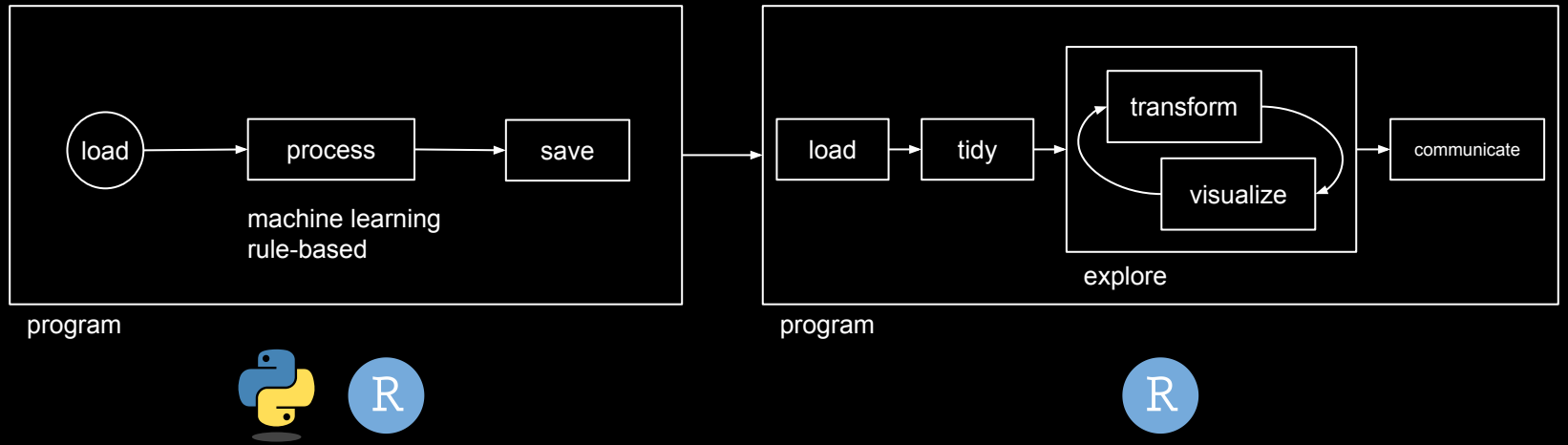
Highly recommended for background information



3Blue1Brown's YouTube Course on Neural Networks and Deep Learning

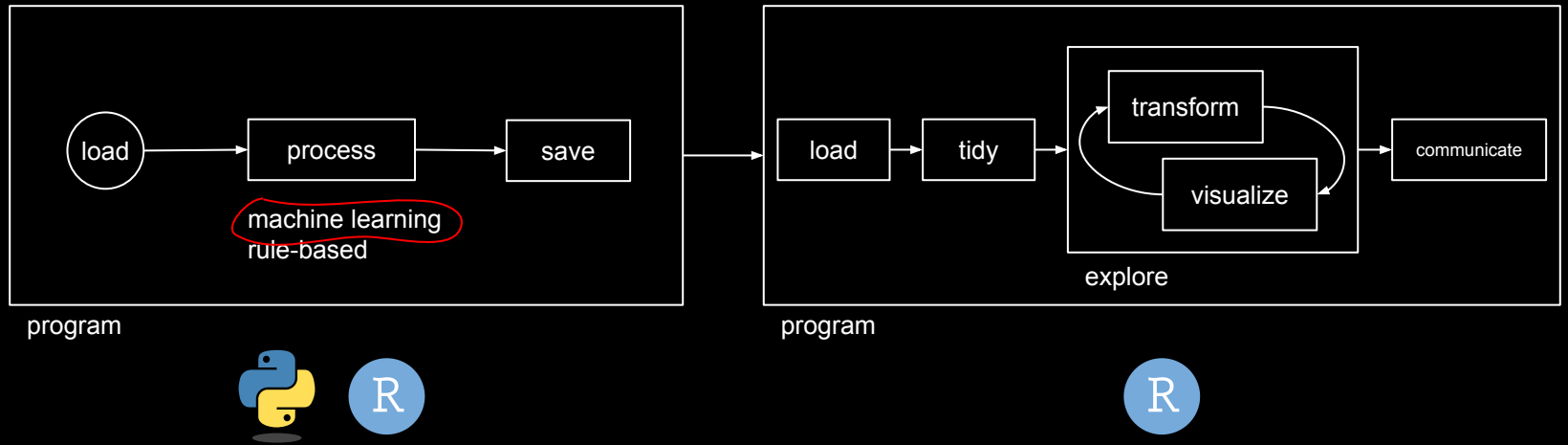
pre-process
unstructured data

exploratory data
analysis

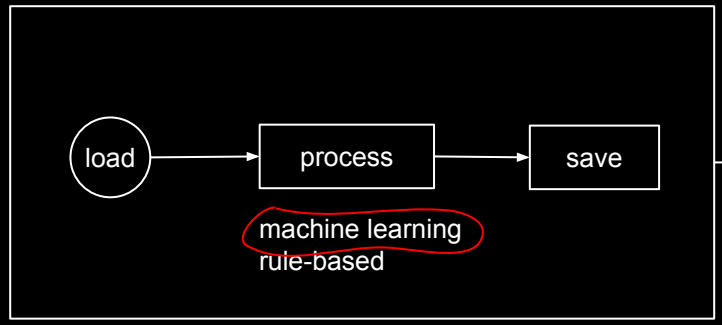


pre-process
unstructured data

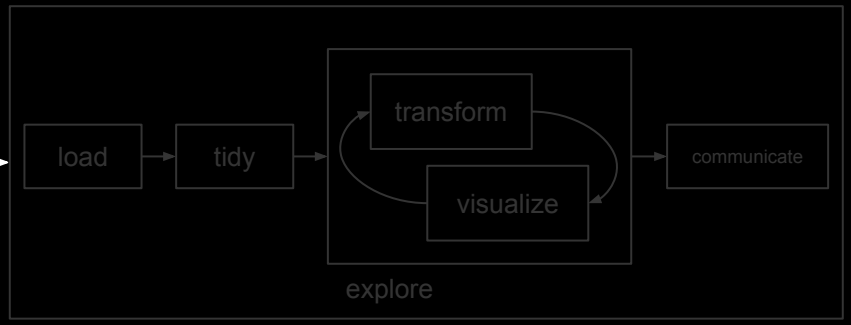
exploratory data
analysis

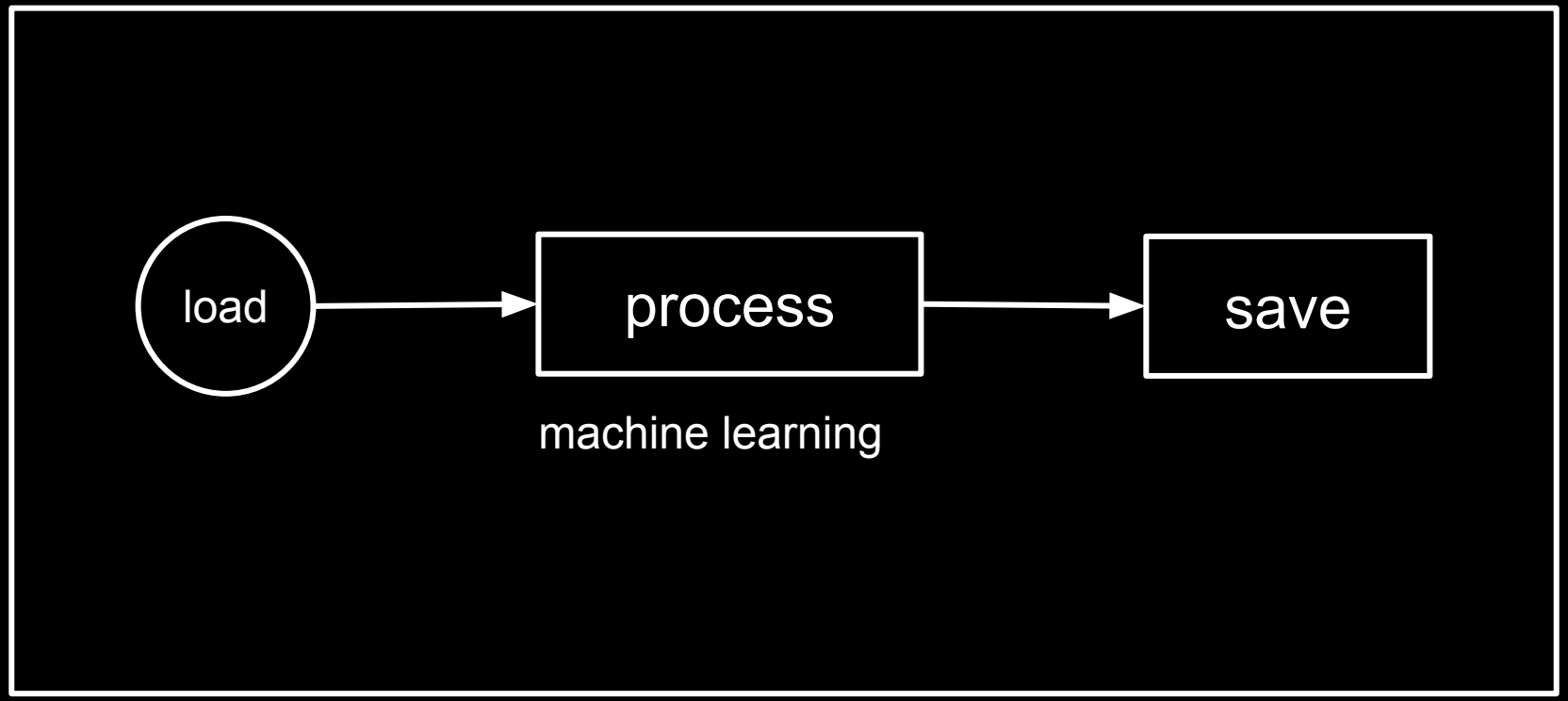


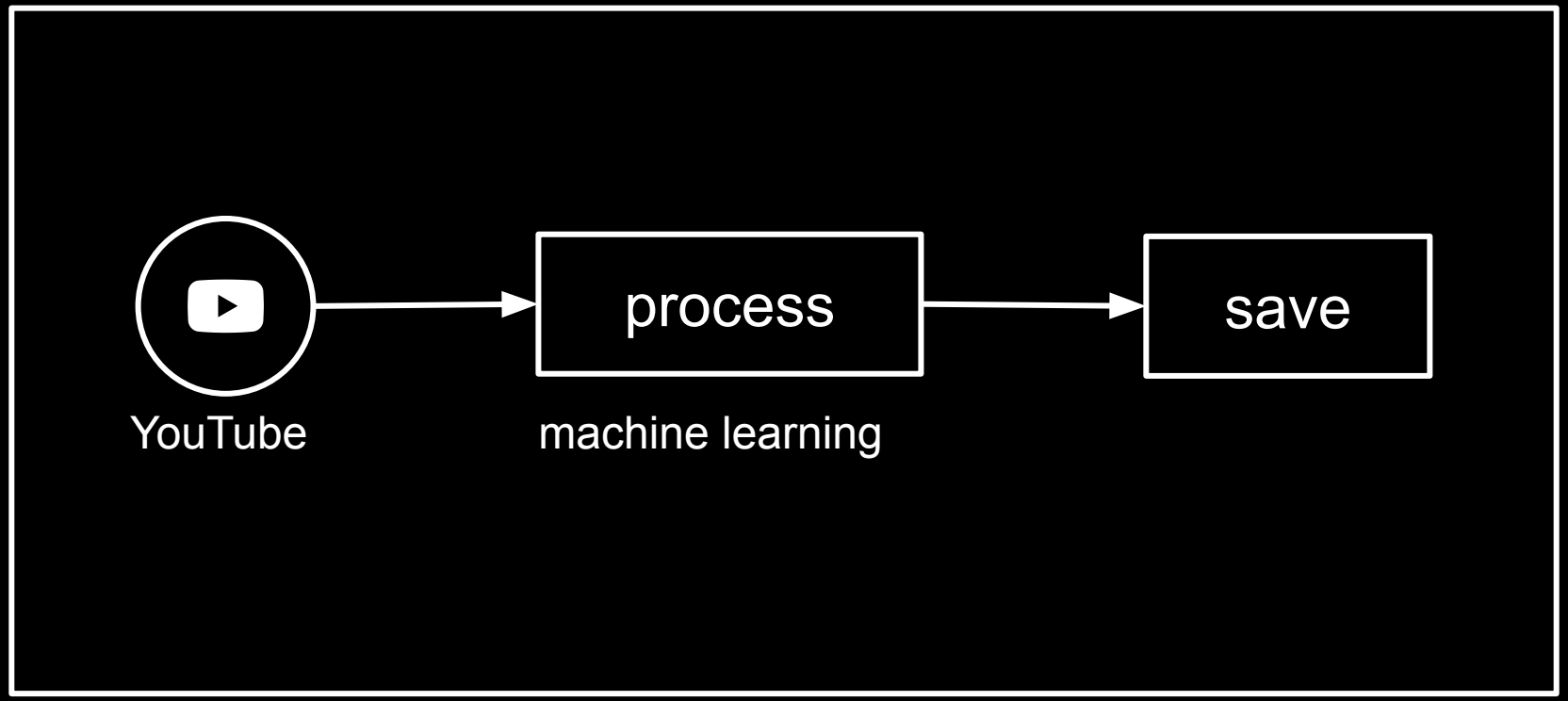
pre-process unstructured data

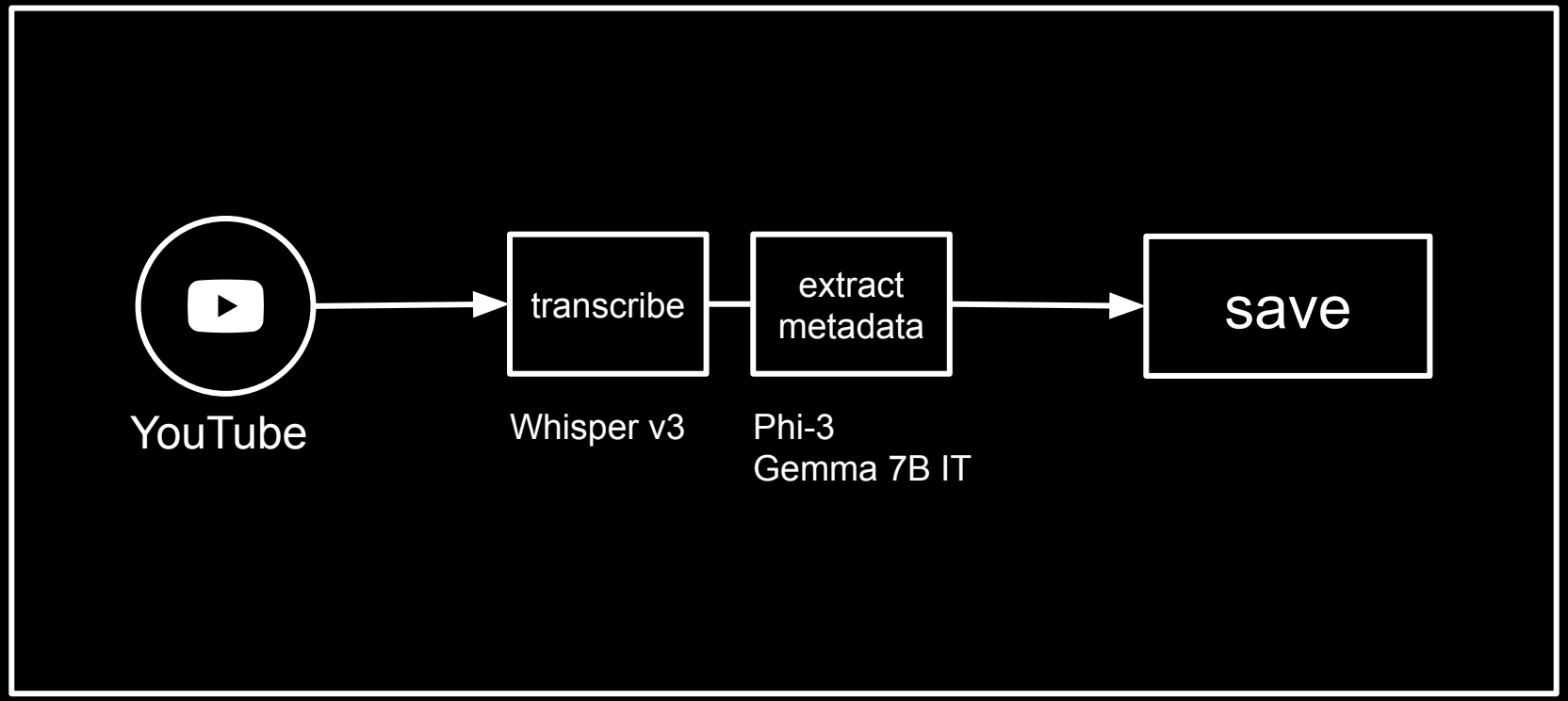


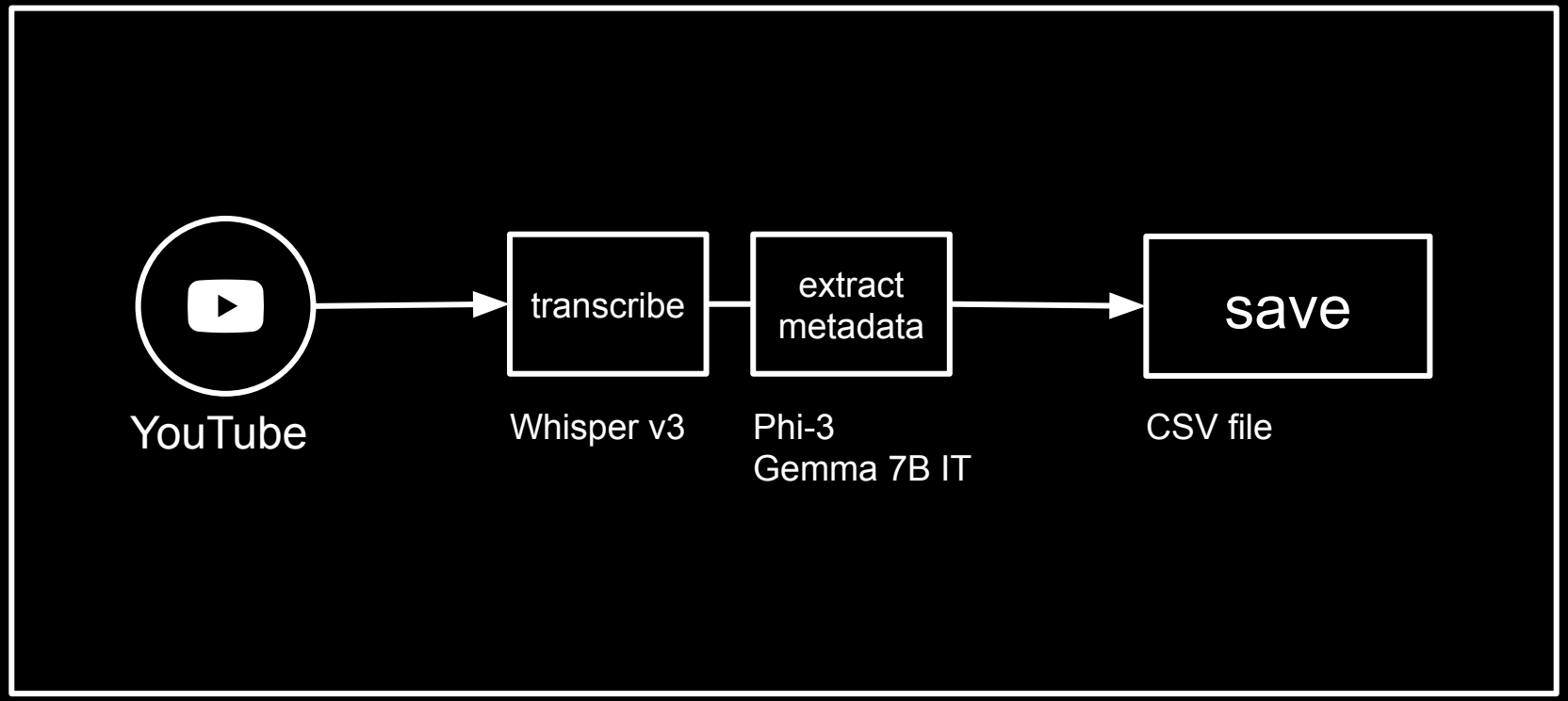
exploratory data analysis




load
process
save
machine learning
program


process
save
YouTube
machine learning
program


transcribe
extract metadata
save
YouTube
Whisper v3
Phi-3
Gemma 7B IT
program


transcribe
extract metadata
save
YouTube
Whisper v3
Phi-3
Gemma 7B IT
CSV file
program

YouTube API

Whisper v3

https://arxiv.org/abs/2212.04356

https://huggingface.co/openai/whisper-large-v3


audio file
whisper v3
transcript

Large Language Models (LLM)

what has been said so far?

(*prompt / context*)

what has been said so far?
(*prompt / context*)

prediction of next token based on
learnt probability distribution

what has been said so far?
(*prompt / context*)

prediction of next token based on learnt probability distribution

+

(randomness)

what has been said so far?
(*prompt / context*)

prediction of next token based on learnt probability distribution

+

(randomness)

+

(filter)

(discriminating, insulting content)

what has been said so far?
(*prompt / context*)

prediction of next token based on
learnt probability distribution

+

(randomness)

+

(filter)

(discriminating, insulting content)

next word (*token*)

what has been said so far?
(*prompt / context*)

prediction of next token based on
learnt probability distribution

+

(randomness)

+

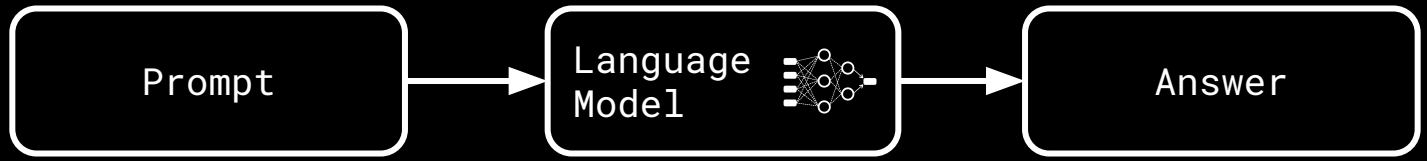
(filter)

(discriminating, insulting content)

next word (*token*)

PROMPTING

https://www.promptingguide.ai/


Prompt
Language
Model
Answer

elements of a prompt

```
<instruction>
<context>
<input data>
<output indicator>
```

elements of a prompt

```
<instruction>
<context>
<input data>
<output indicator>
```

example prompt

```
Explain the binary number system.
```

elements of a prompt

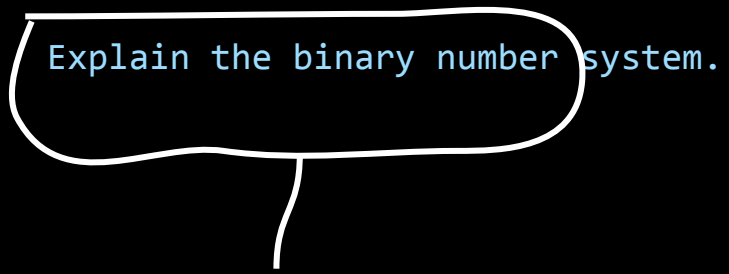
<instruction>

<context>

<input data>

<output indicator>

example prompt



start simple

elements of a prompt

<instruction>

<context>

<input data>

<output indicator>

example prompt

```
You are a friendly tutor and your
task is to explain complex
concepts as simple as possible.

Explain the binary number system.
```

elements of a prompt

<instruction>

<context>

<input data>

<output indicator>

example prompt

```
You are a friendly tutor and your
task is to explain complex
concepts as simple as possible.

Your answers are never longer
than 10 sentences.

Explain the binary number system.
```

ZERO-SHOT PROMPTING

elements of a prompt	example prompt
<instruction>	Classify the text into neutral, negative or positive.
<context>	
<input data>	Text: "What a great dinner!"
<output indicator>	Sentiment:

elements of a prompt

<instruction>

<context>

<input data>

<output indicator>

example prompt

```
Classify the text into neutral,
negative or positive.

Text: "What a great dinner!"

Sentiment:
```

this will be replaced with data later…

FEW-SHOT PROMPTING

IN-CONTEXT LEARNING

examples in the context to learn from

```
Extract all references to countries and their continent in the following text
using the format from the examples below.

Example 1: "They played the team called 'Die Mannschaft' in the world cup final"
Correct answer: Germany, Europe

Example 2: "The Three Lions once again lost to Germany in a semi final"
Correct answer: England, Europe, Germany, Europe

Text: "The Selecao was destroyed 1:7 by the DFB selection in their home stadium."
Answer:
```

examples in the context to learn from

Extract all references to countries and their continent in the following text using the format from the examples below.

Example 1: "They played the team called 'Die Mannschaft' in the world cup final"
Correct answer: Germany, Europe

Example 2: "The Three Lions once again lost to Germany in a semi final"
Correct answer: England, Europe, Germany, Europe

Text: "The Selecao was destroyed 1:7 by the DFB selection in their home stadium."
Answer:

more prompting strategies

chain-of-thought (CoT)
self-consistency
generate knowledge prompting
prompt chaining (subtasks)
tree-of-thoughts (ToT)
retrieval-augmented-generation (RAG)
…

Phi-3

https://arxiv.org/abs/2404.14219

~~https://huggingface.co/microsoft/Phi-3-mini-128k-instruct~~

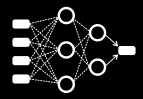
https://huggingface.co/microsoft/Phi-3-medium-128k-instruct

Gemma 2B / 7B Instruct

https://arxiv.org/abs/2403.08295

https://huggingface.co/google/gemma-2b-it

https://huggingface.co/google/gemma-7b-it



prompt +
context

phi-3
gemma-2b-it
gemma-7b-it

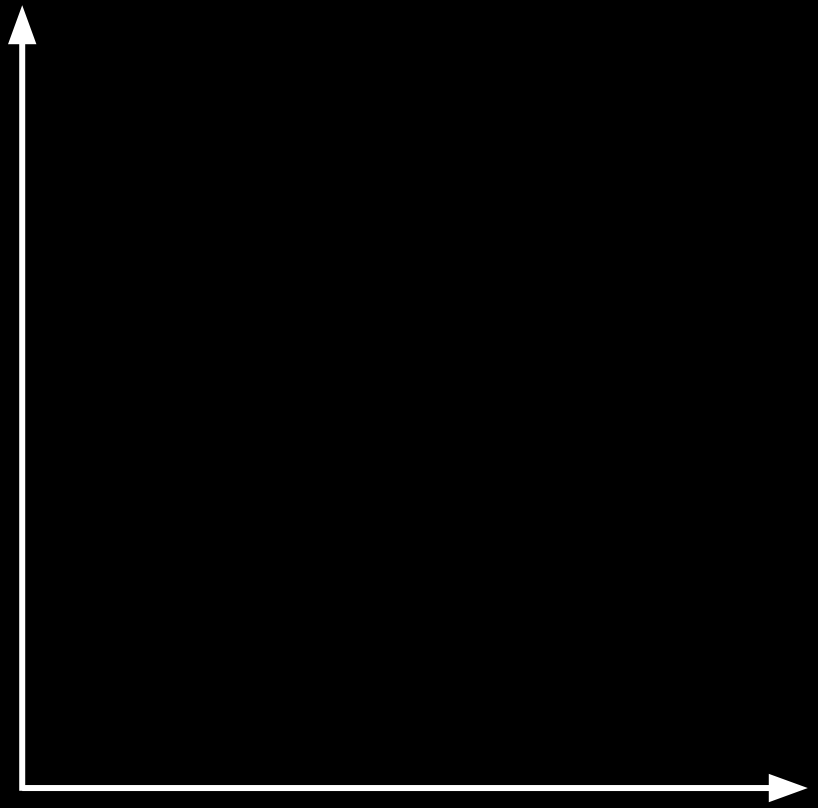
answer

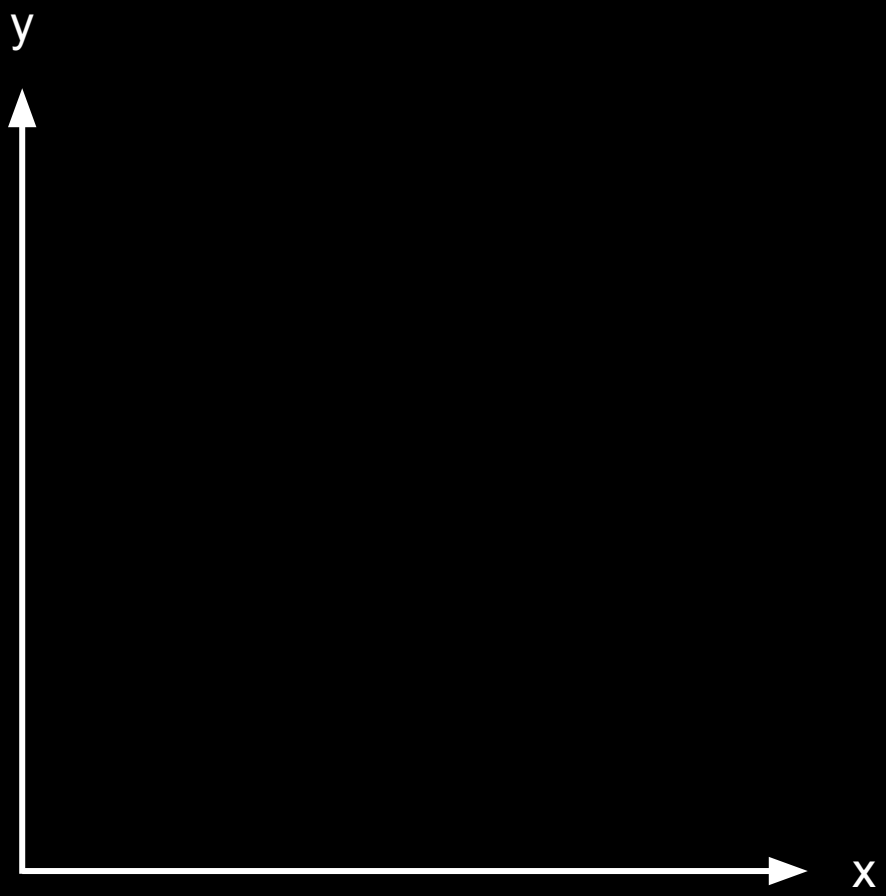
OpenAI GPT-4o

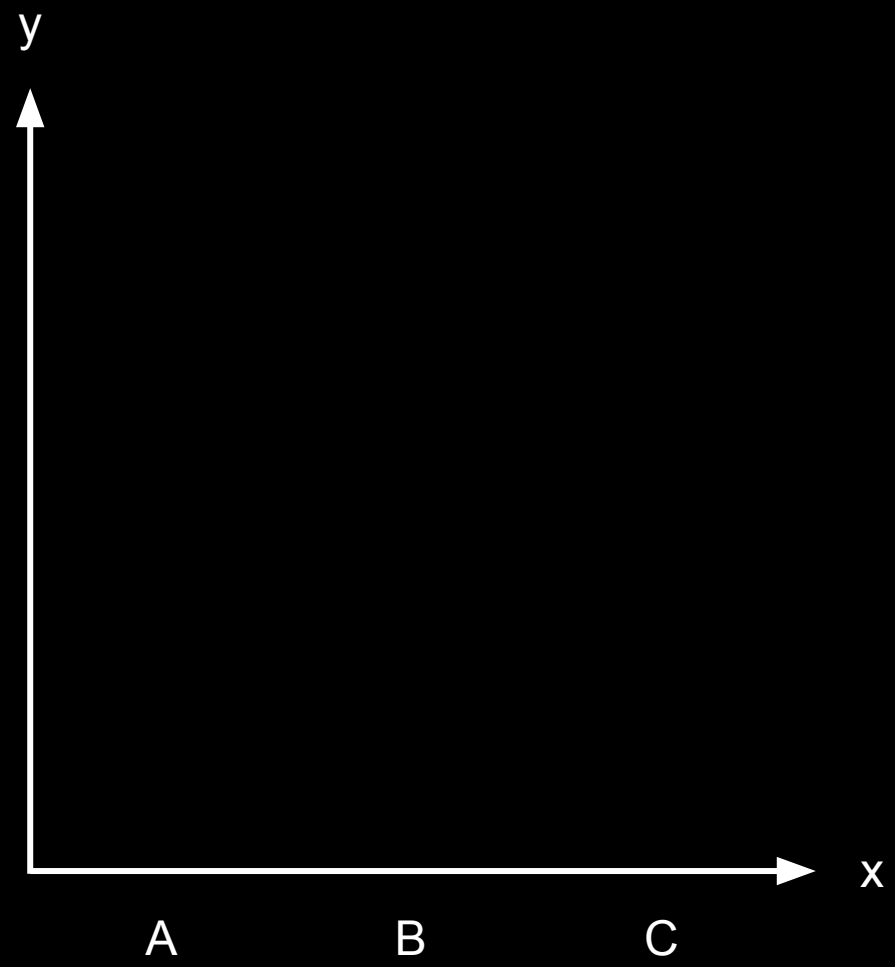
VISUALIZE DATA

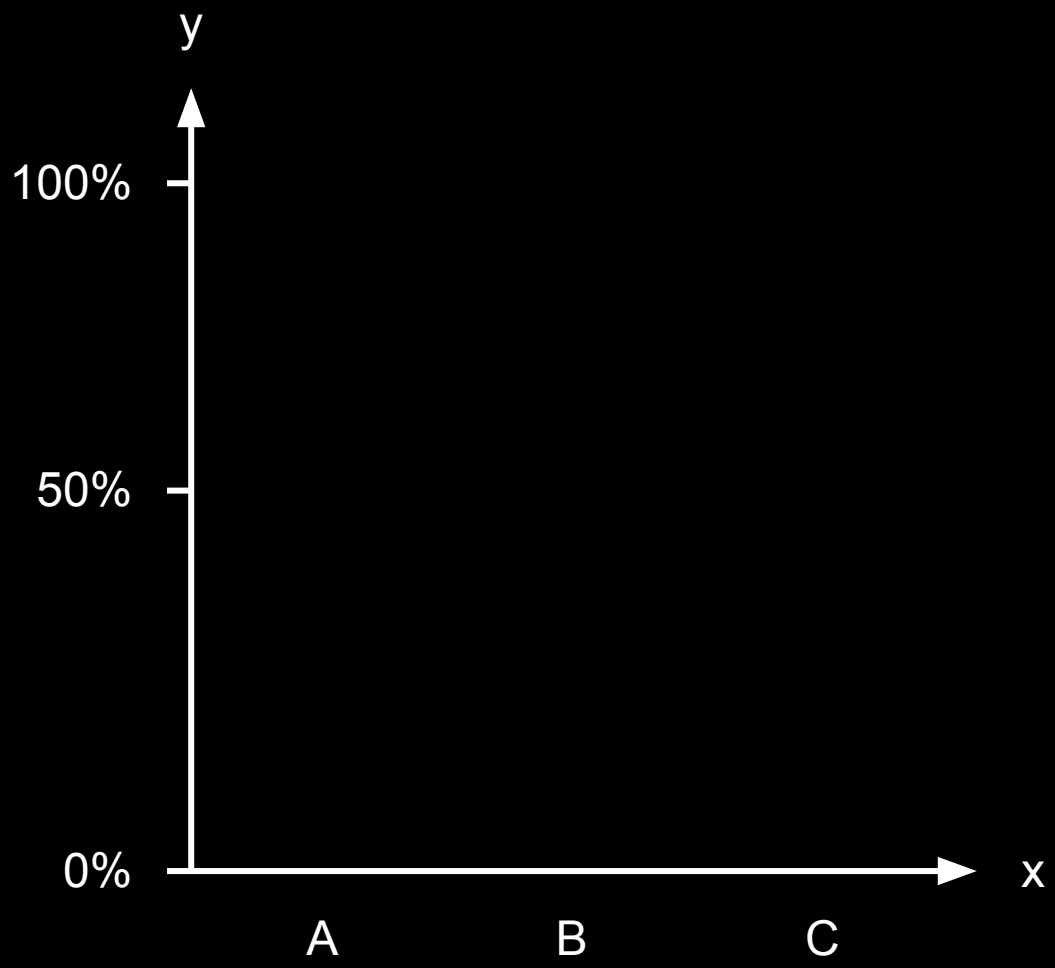
data

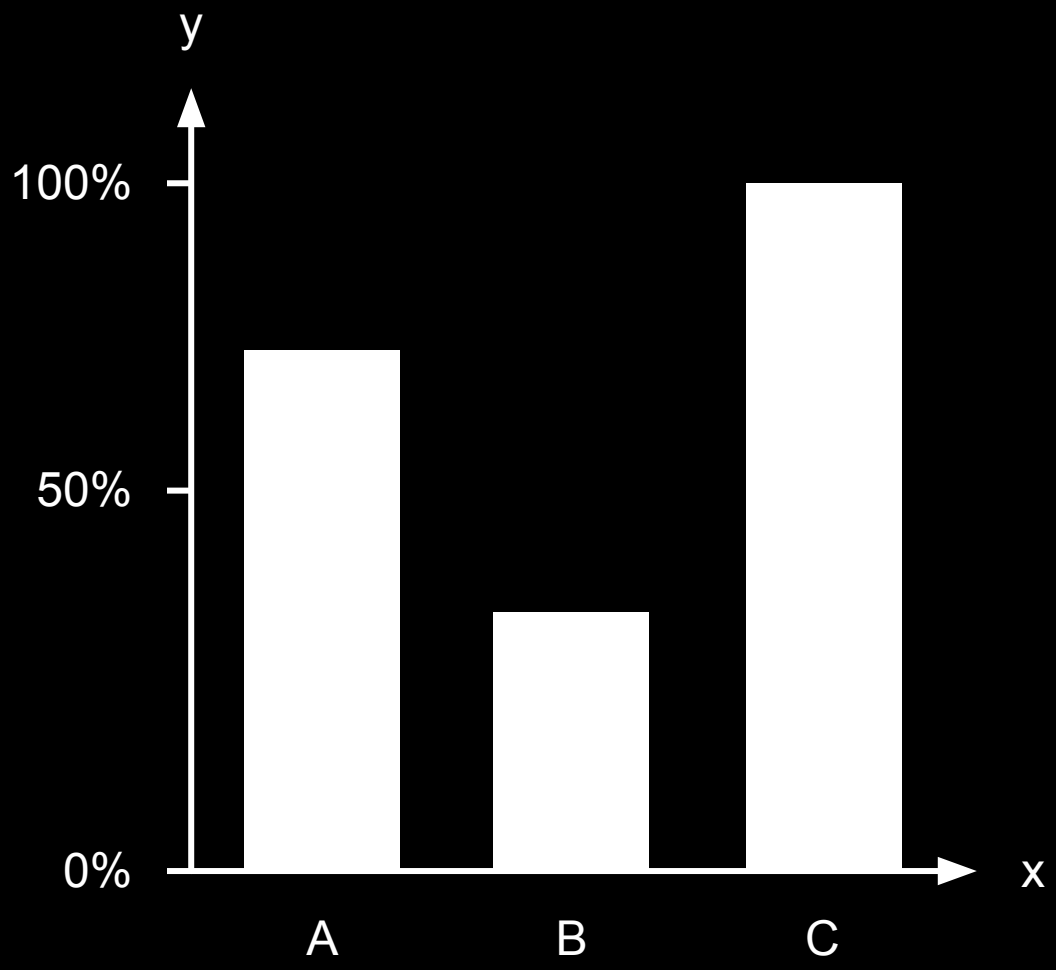
category	pct
A	75
B	33
C	100



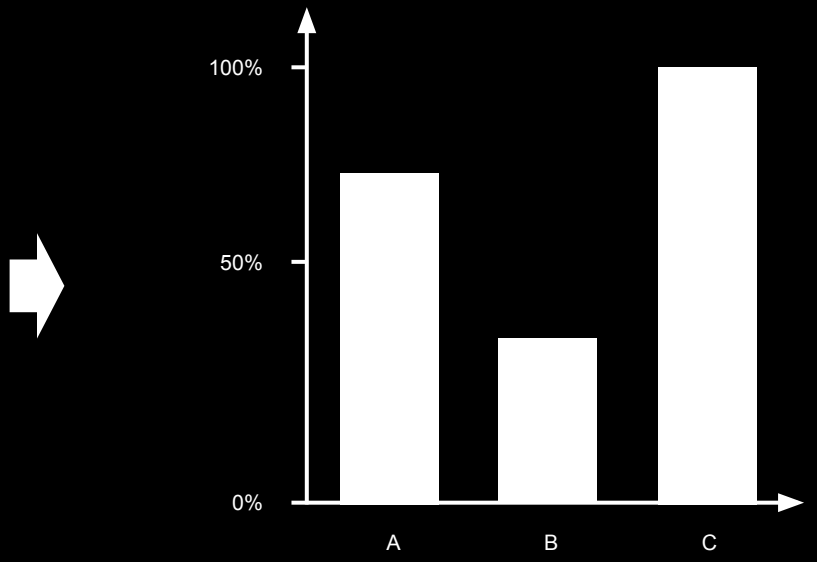

y
x


y
x
A
B
C


y
100%
50%
0%
x
A
B
C


y
100%
50%
0%
x
A
B
C

category	pct
A	75
B	33
C	100



{{ ggplot2 }}

why visualize?

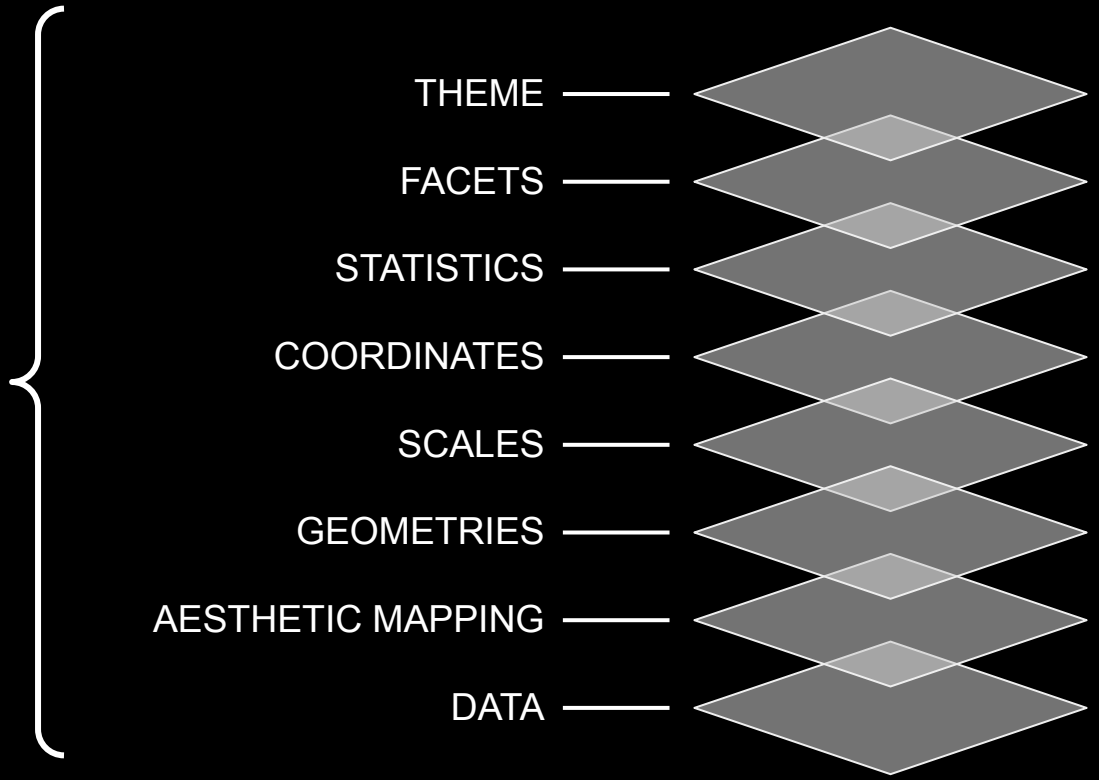
{{ ggplot2 }}

grammar of graphics

any

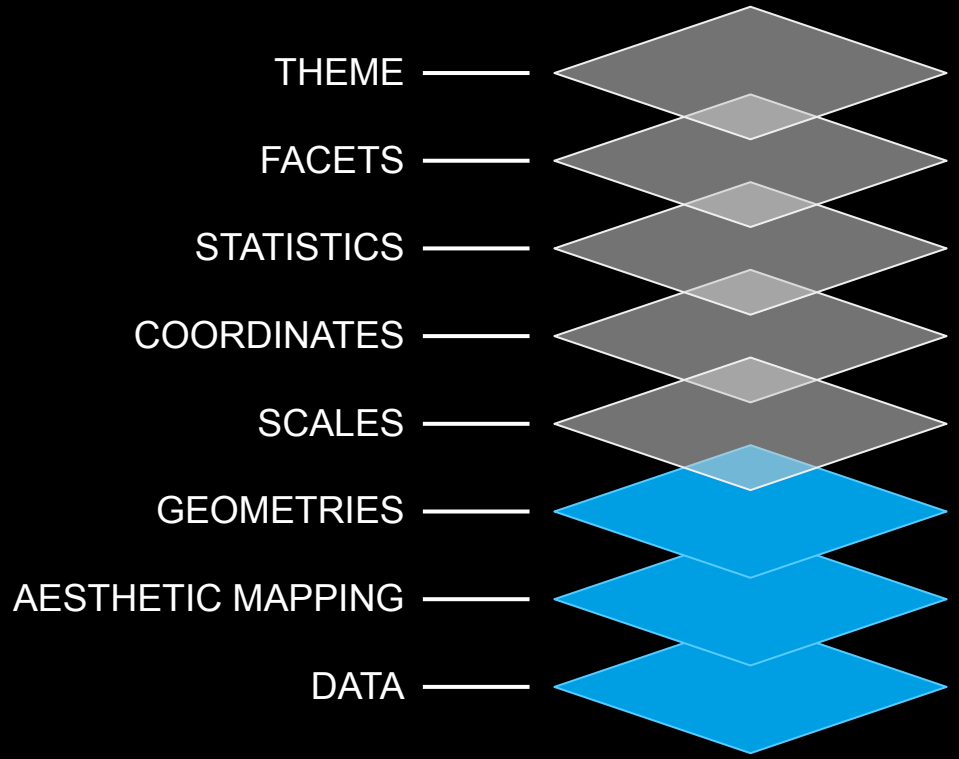
data

visualization



has useful defaults

mandatory

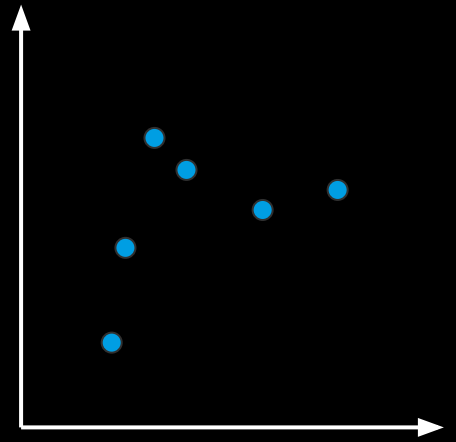


ggplot()

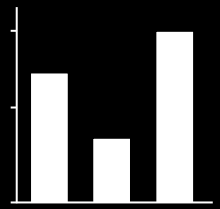
```
ggplot() +
  aes()
```

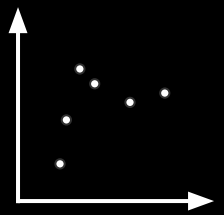
```
ggplot() +
  aes() +
  geom_point()
```

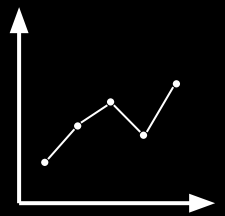
```
ggplot() +
  aes() +
  geom_point()
```

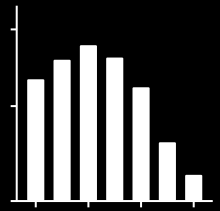


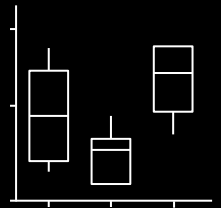
basic plots

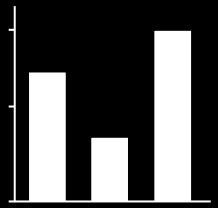




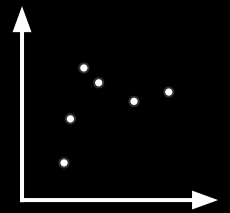




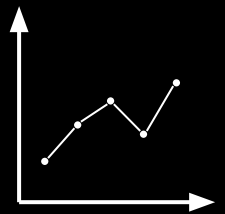




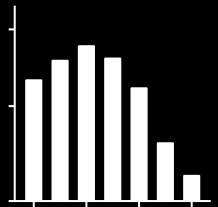
bar chart



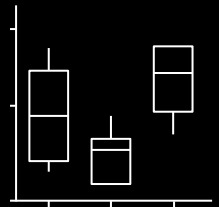
scatter plot



line chart

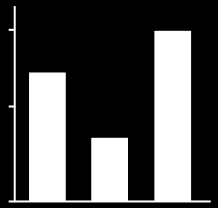


histogram



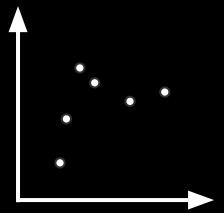
box plot

amounts
proportions
distributions (discrete)



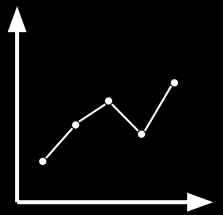
bar chart

associations
patterns



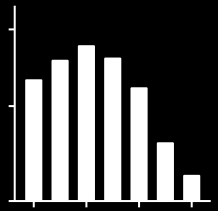
scatter plot

trends
developments



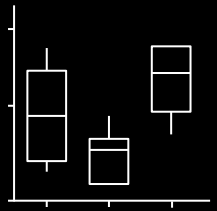
line chart

distributions (continuous)



histogram

compare distributions (continuous)



box plot

COMMUNICATE FINDINGS

Quarto

PYTHON

{{ reticulate }}