

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



did you summarize the  
data?

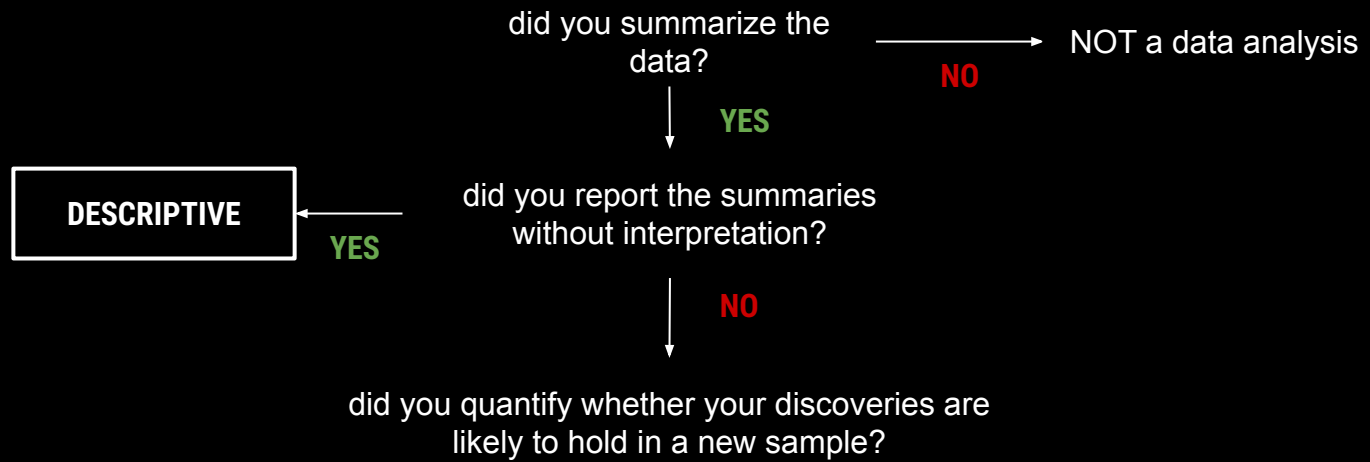
did you summarize the  
data?

NO

→ NOT a data analysis

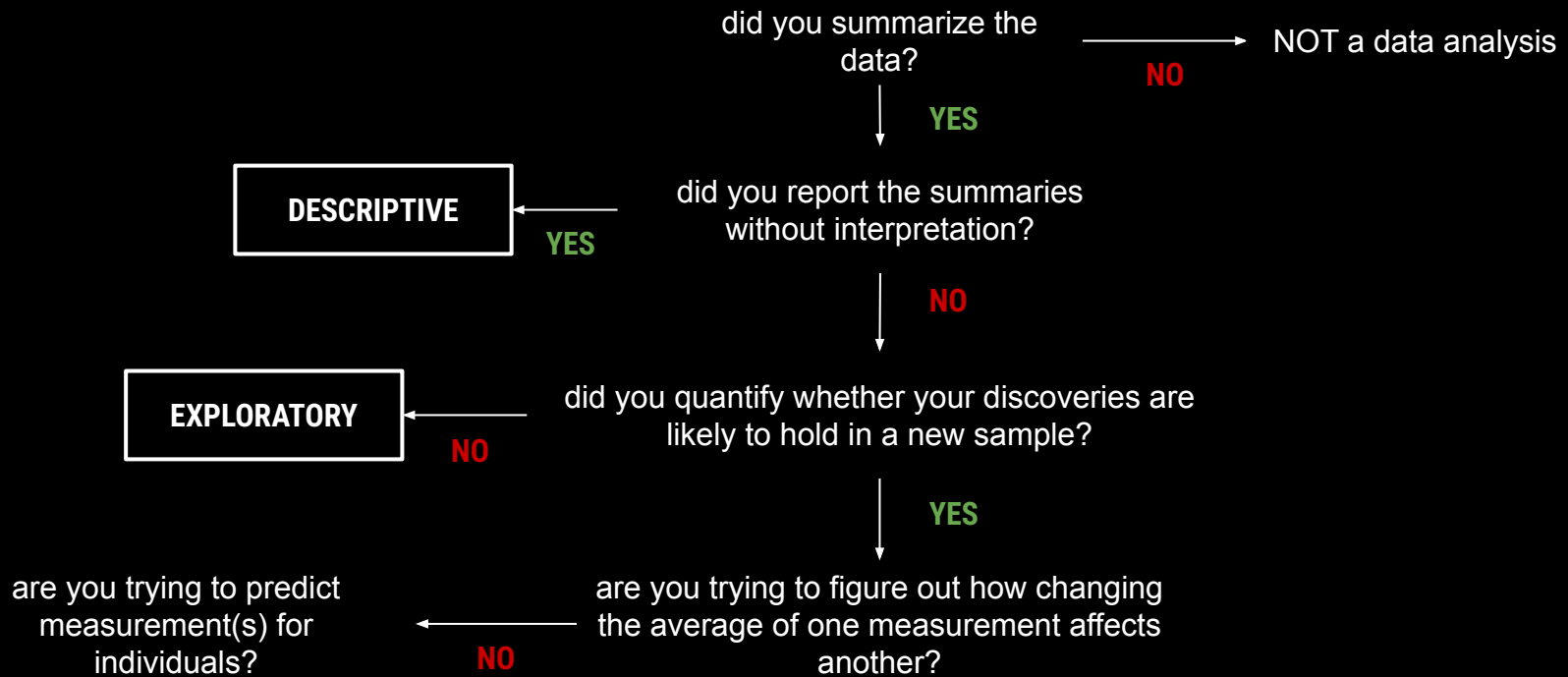












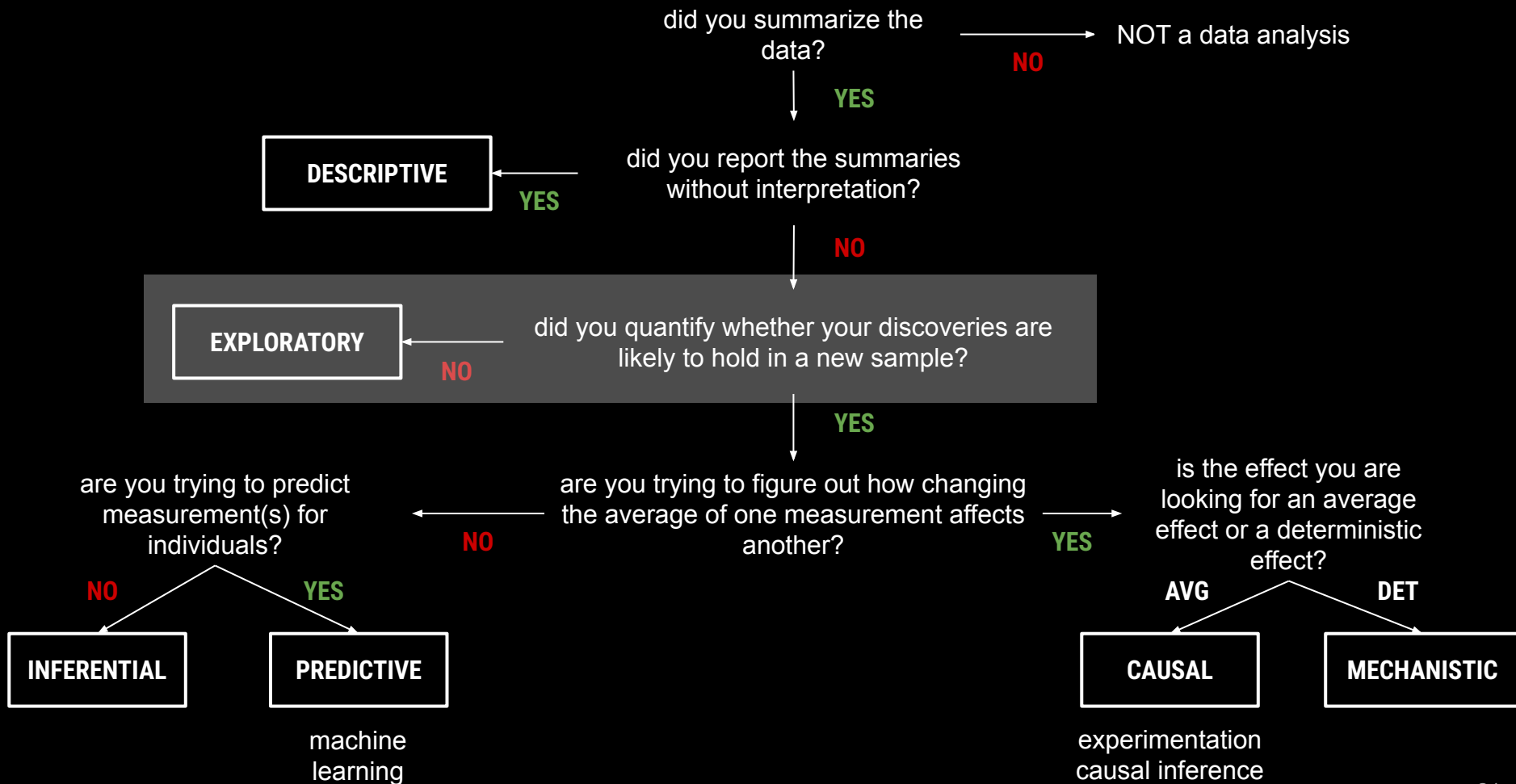






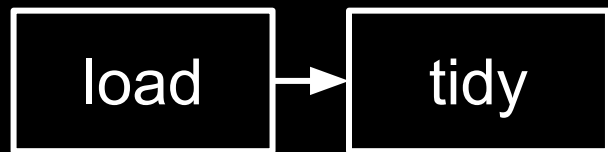




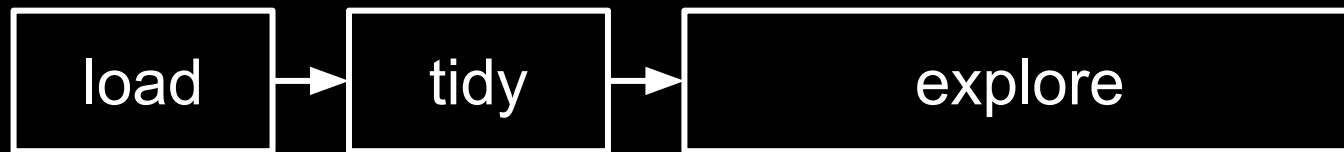


# EXPLORATORY DATA ANALYSIS

load

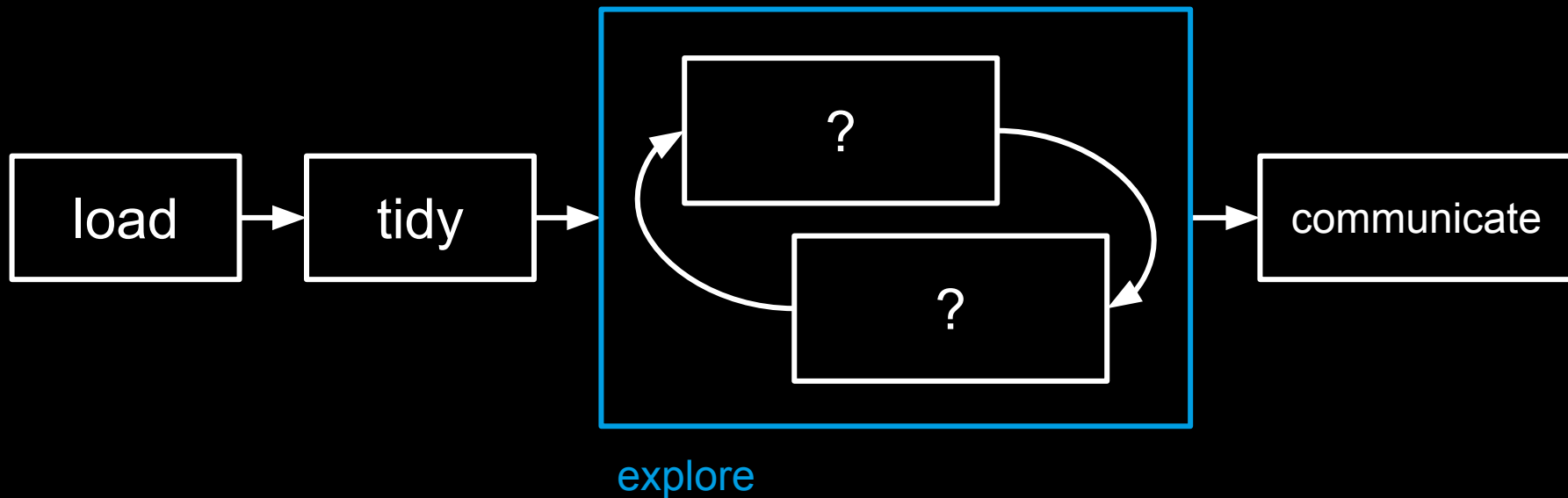


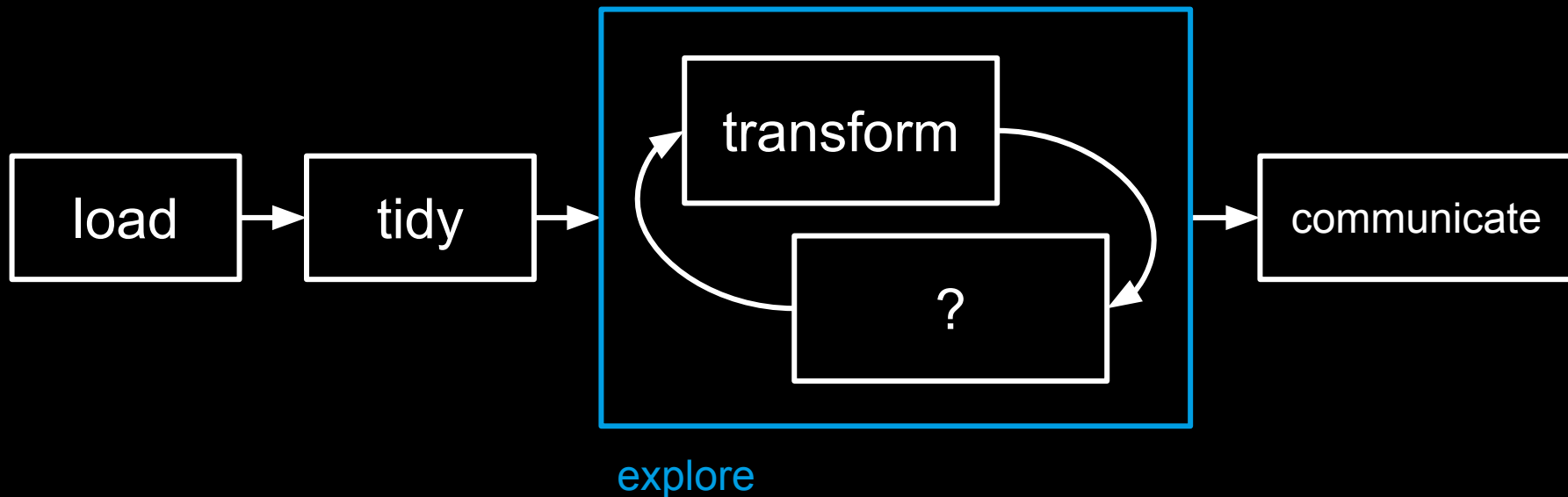


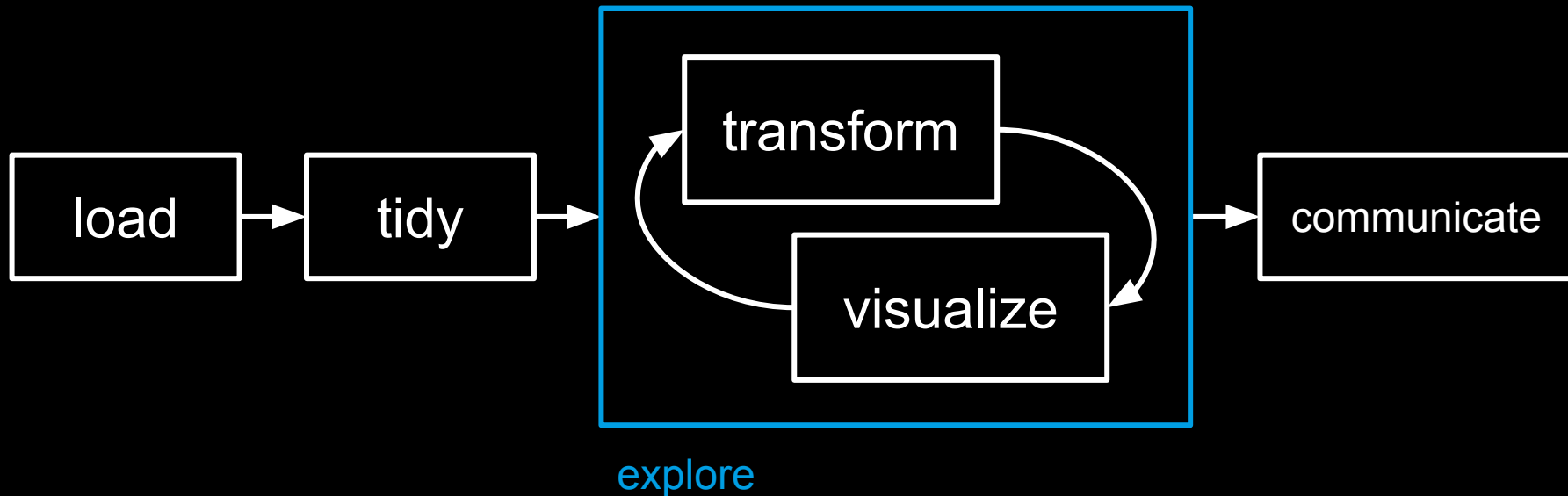


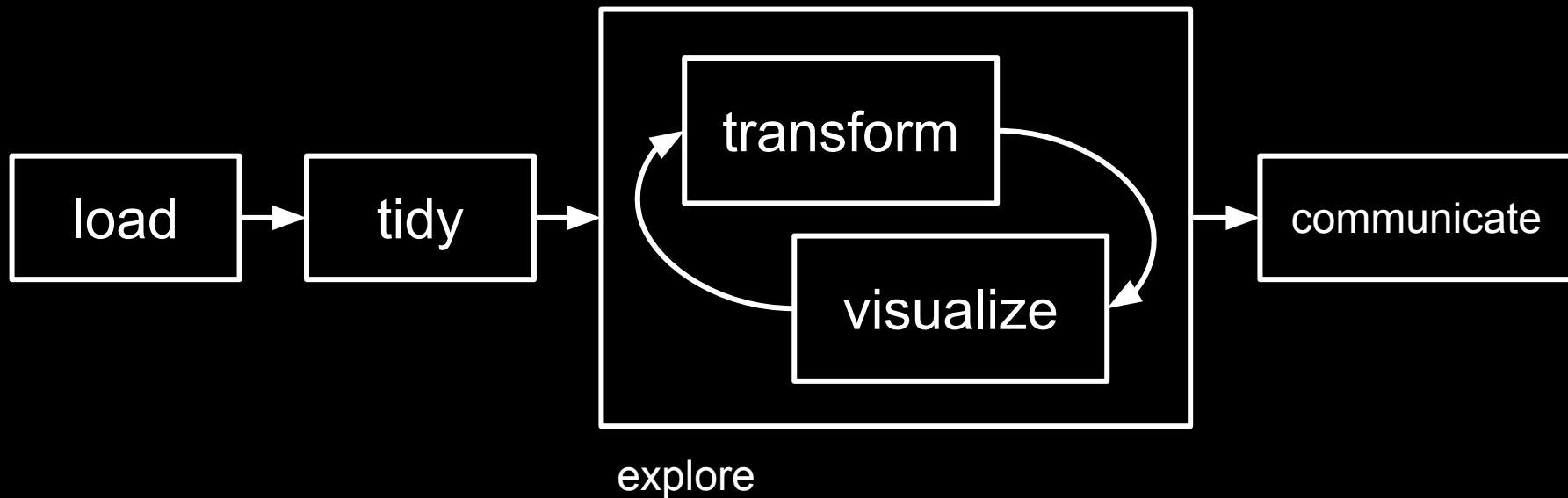


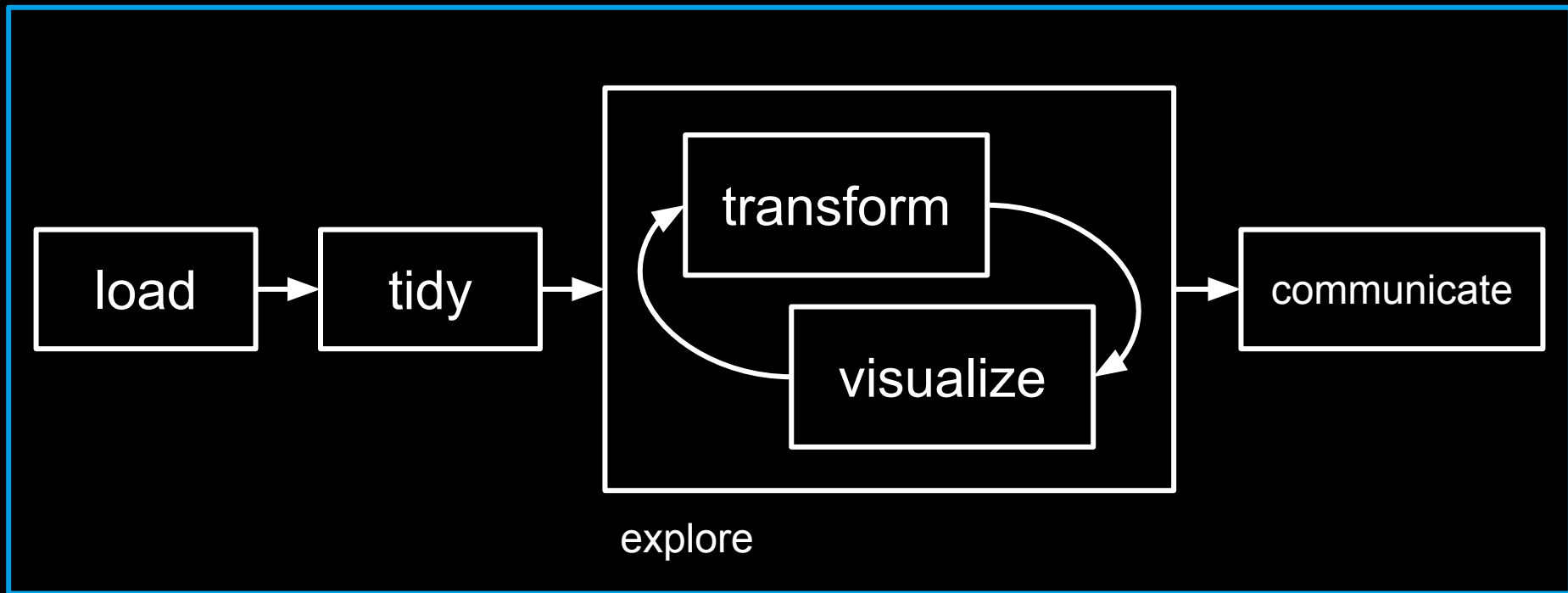






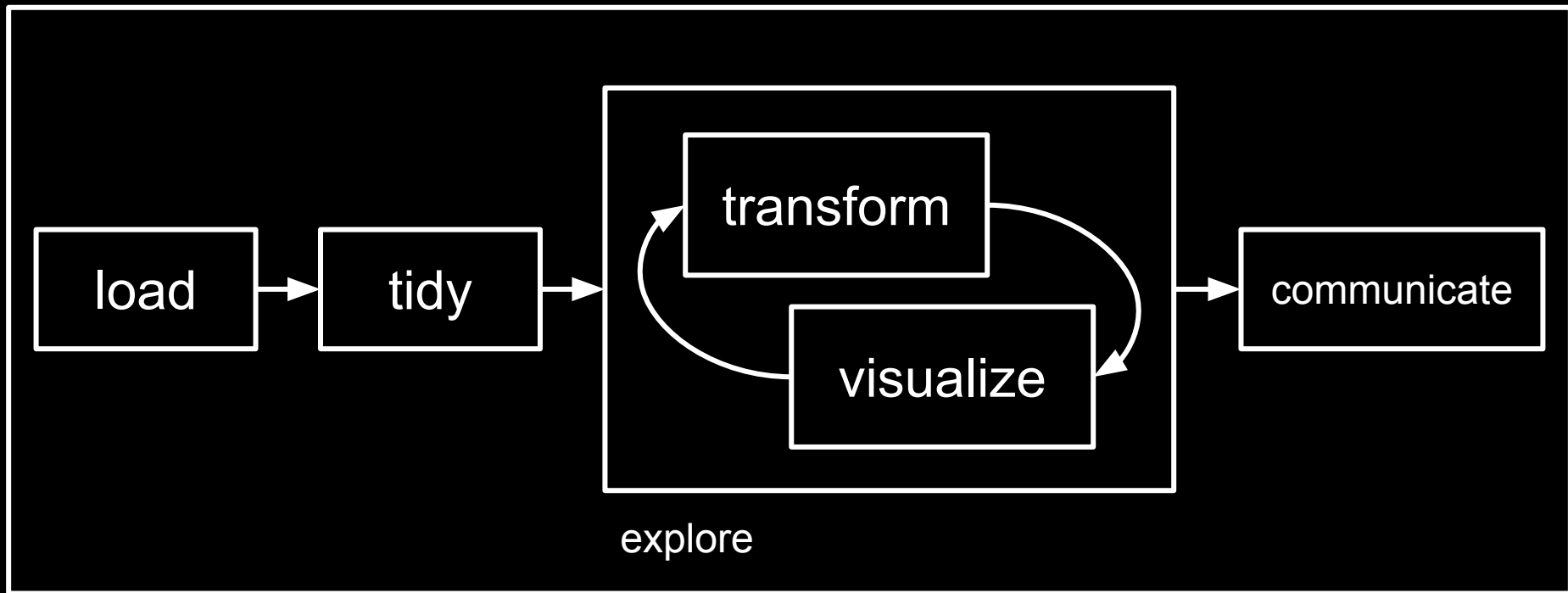






program

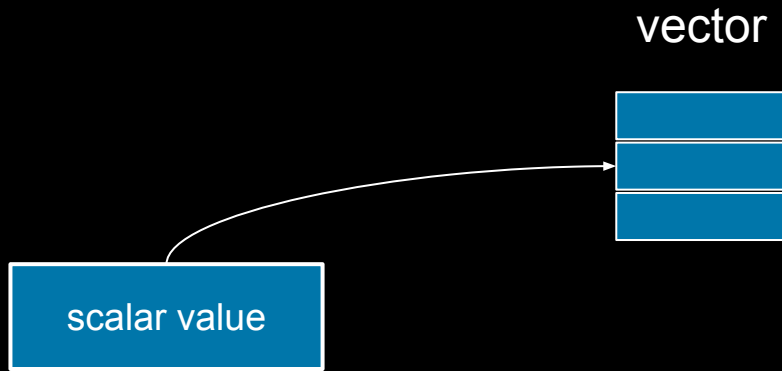


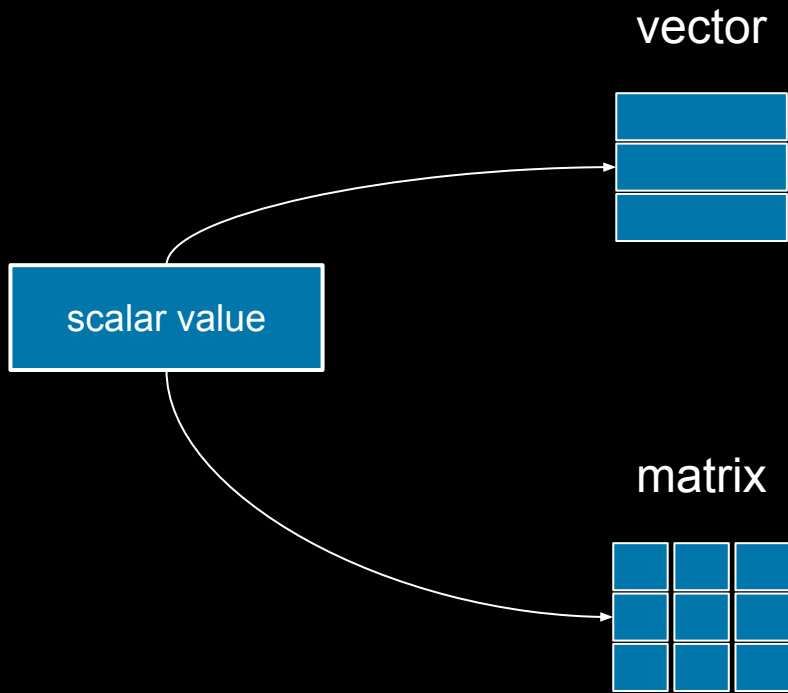


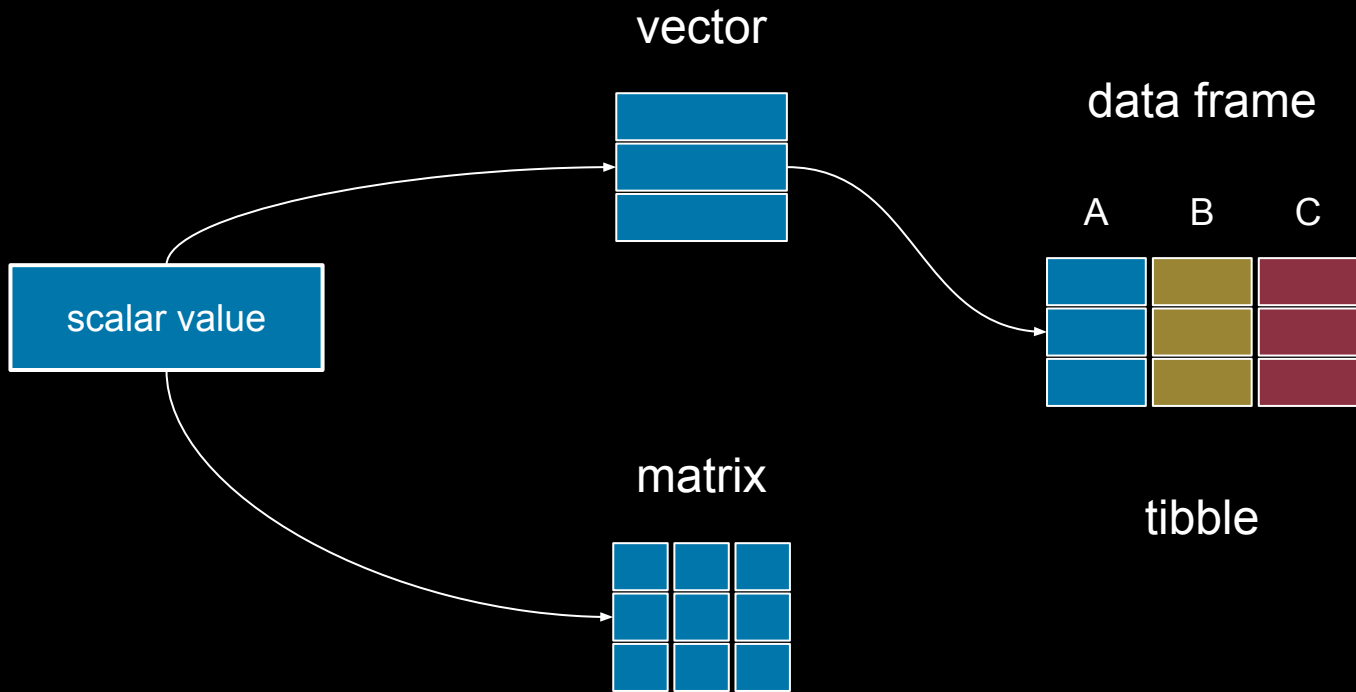
program

# DATA REPRESENTATION

scalar value







# 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]

apple
pear
orange

v[2]

apple
pear
orange

v[3]

apple
pear
orange

```
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

91
75.5
61
88.5
120

```
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

91
75.5
61
88.5
120

weight\_after\_diet

89.5
75
56
96.5
115

weight\_loss

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

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

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

```
levels(weight_category)
```

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

```
levels(weight_category)
```

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

```
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

{{ tibble }}

# 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

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	1997071
Afghanistan	2000	2666	2005360
Brazil	1999	37737	17296362
Brazil	2000	60488	17494898

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

longer



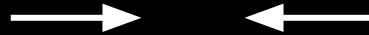
country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898

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



{{ tidyrr }}

```
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”

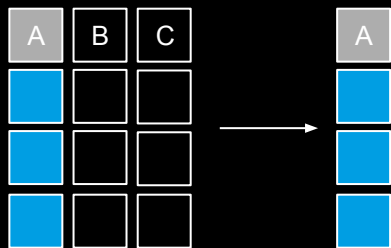
```
str_detect(txt, "Mill")
```

# TRANSFORM DATA

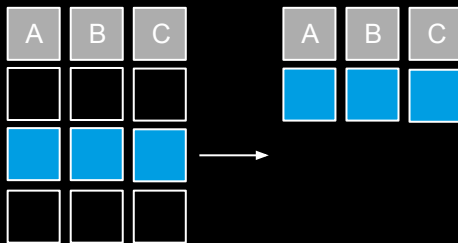
{{ dplyr }}

types of transformations

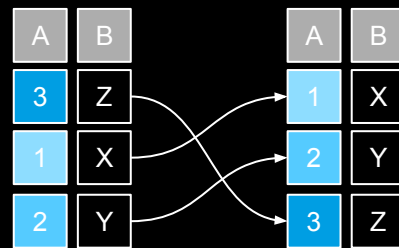
`select()`



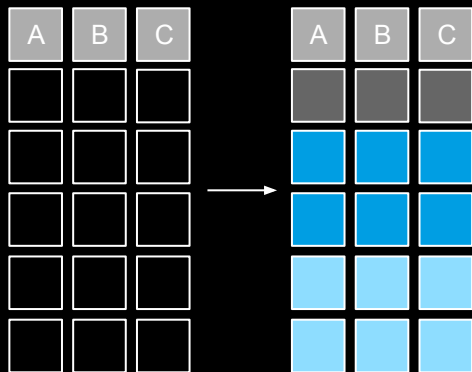
`filter()`



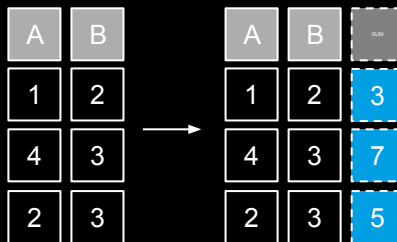
`arrange()`



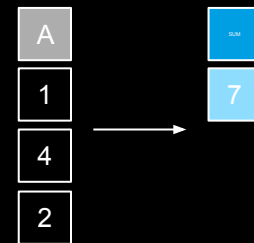
`group_by()`



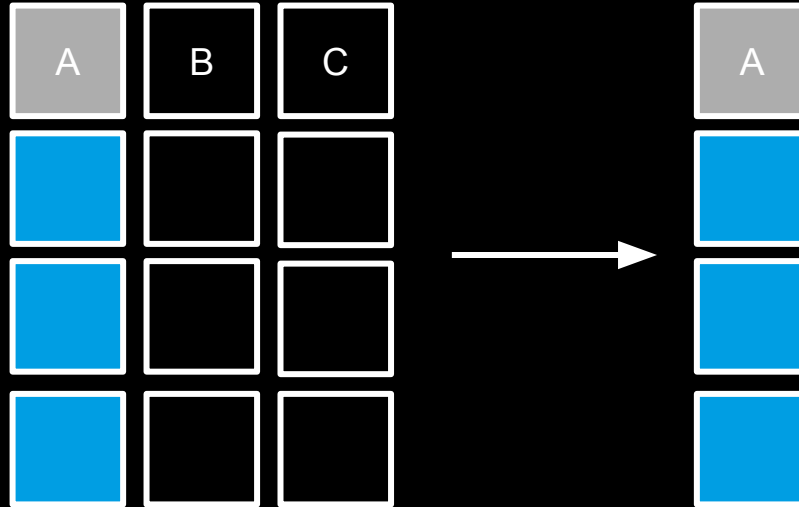
`mutate()`



`summarize()`

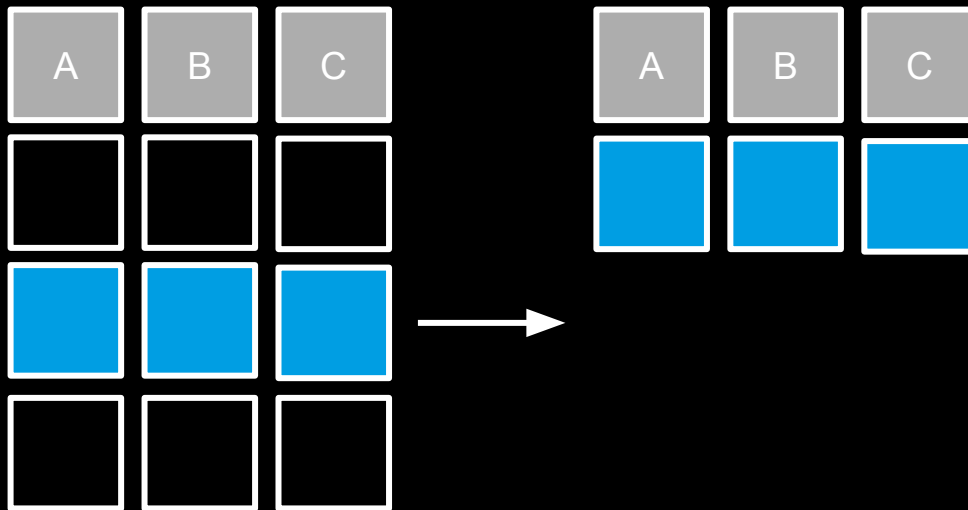


`select()`

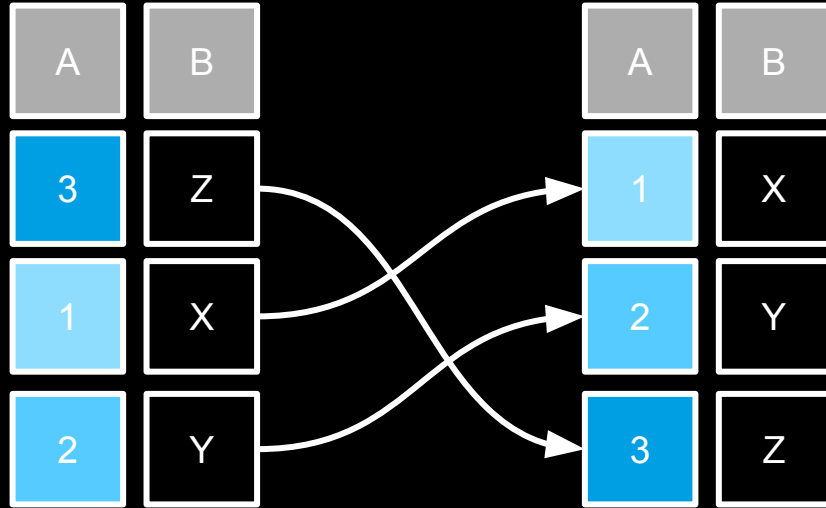




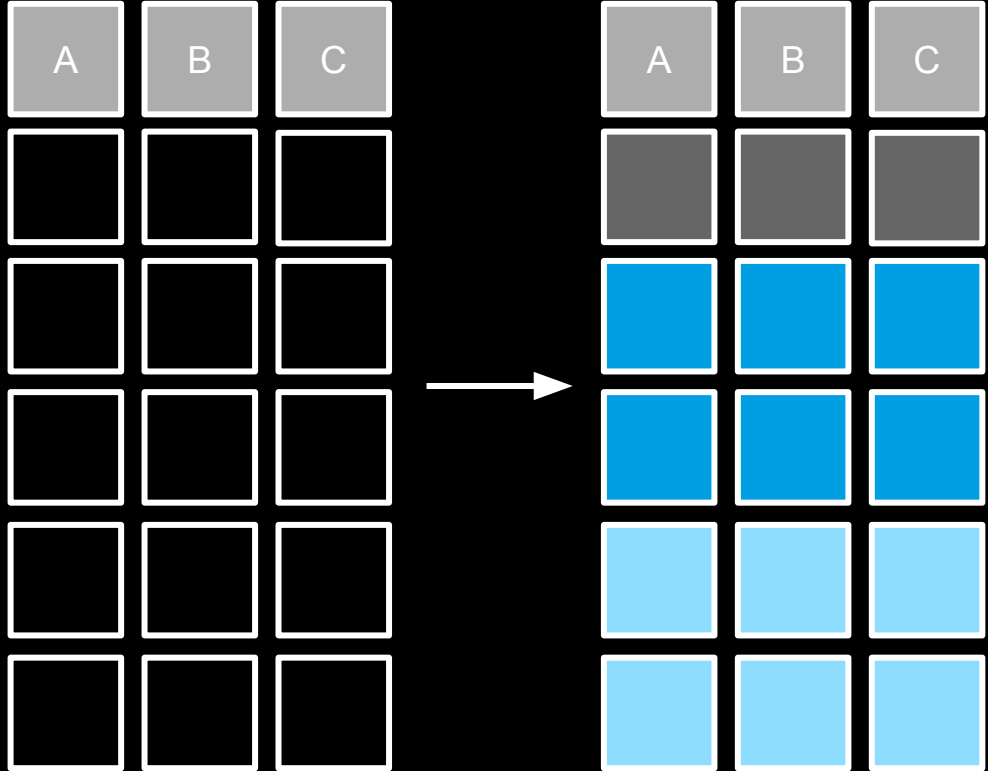
`filter()`



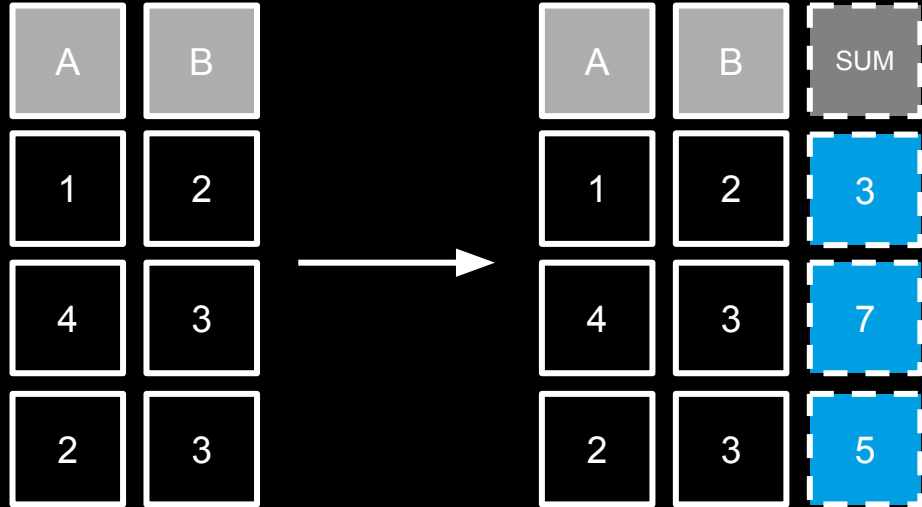
arrange()



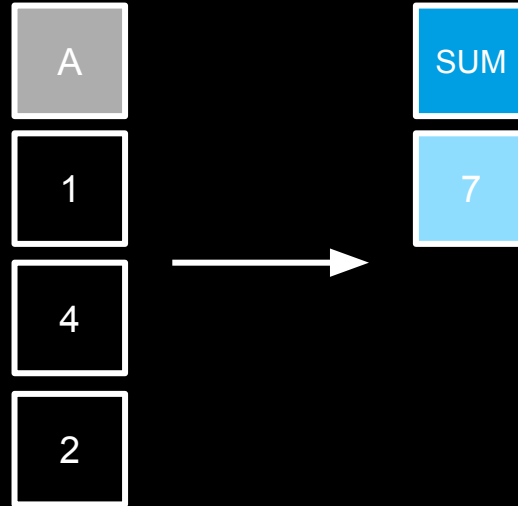
`group_by()`



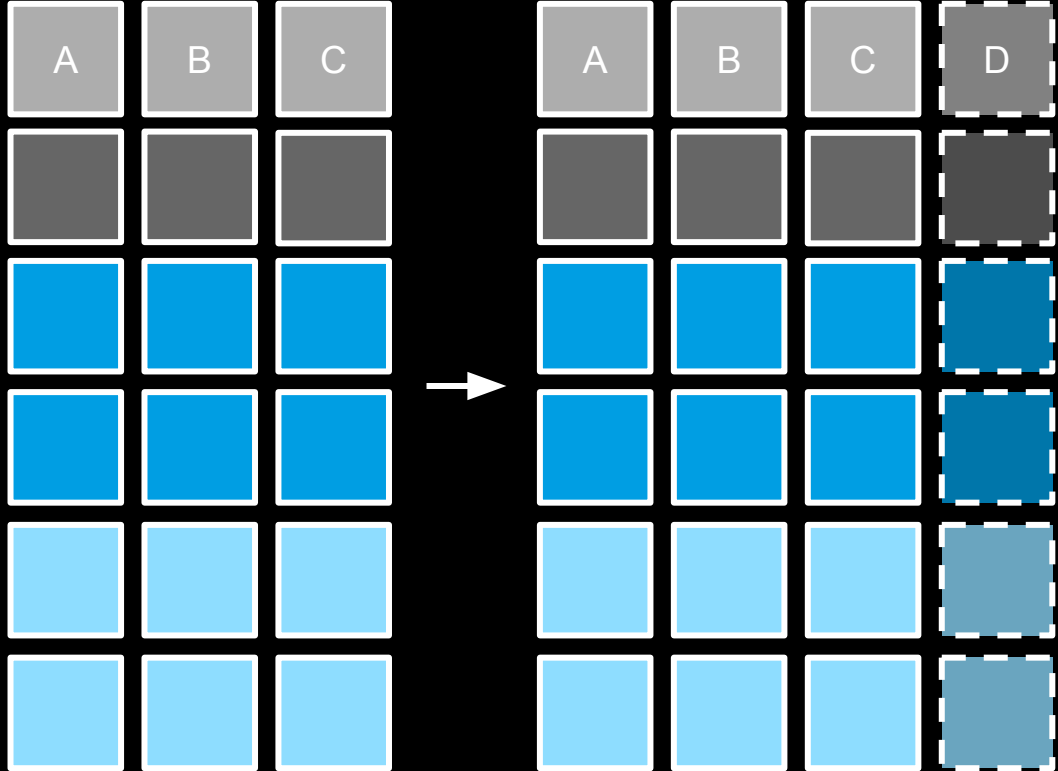
mutate()



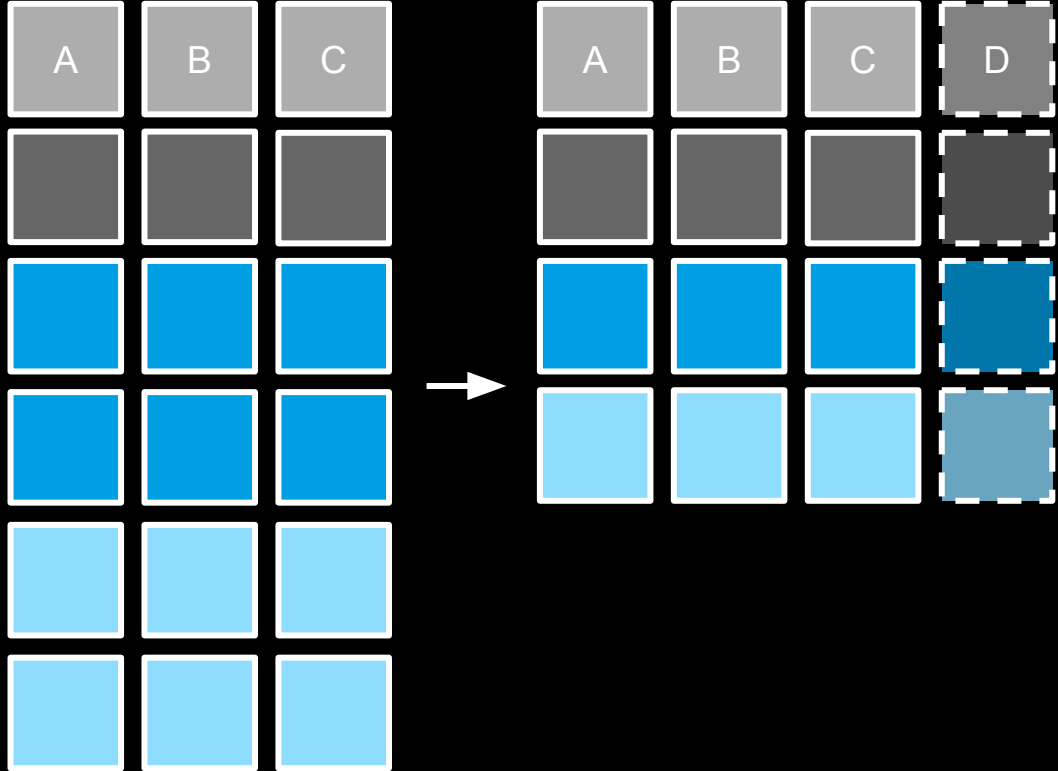
`summarize()`



`group_by()`  
+  
`mutate()`



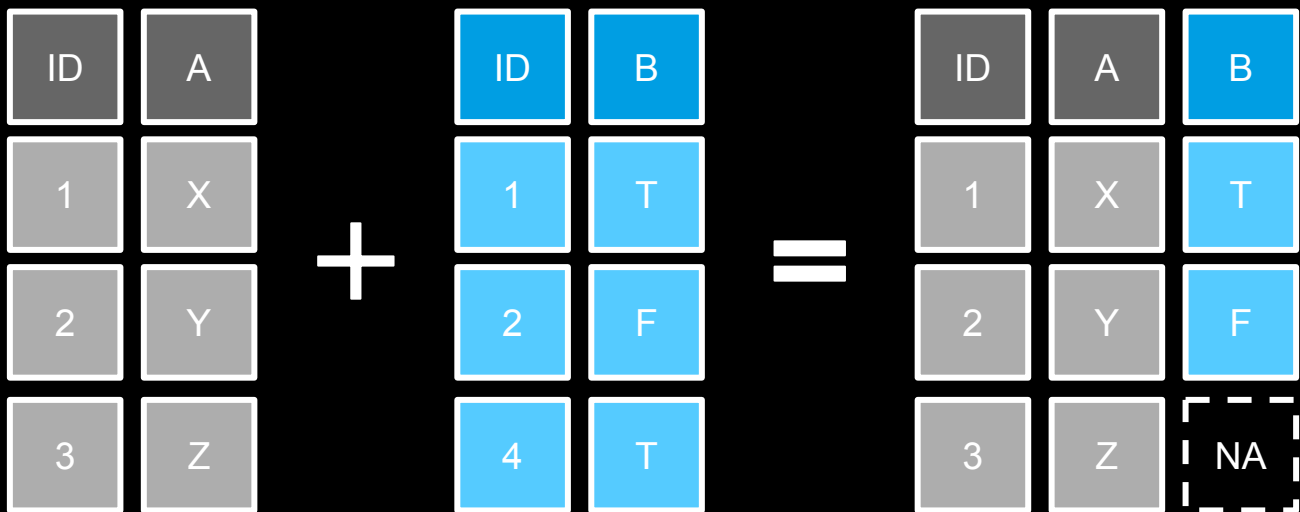
`group_by()`  
+  
`summarize()`



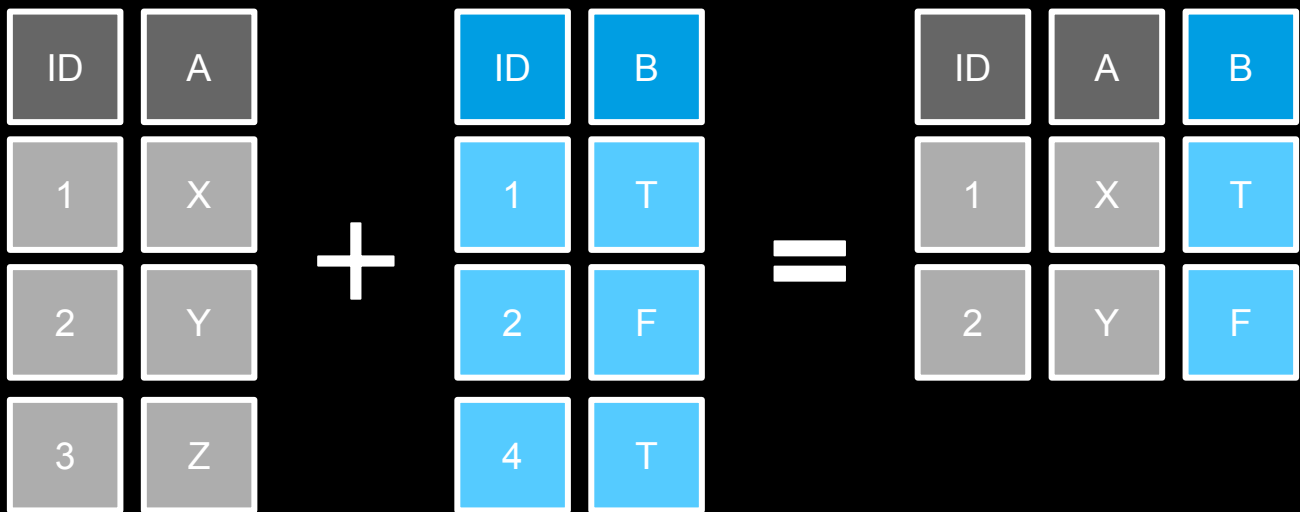
joining data



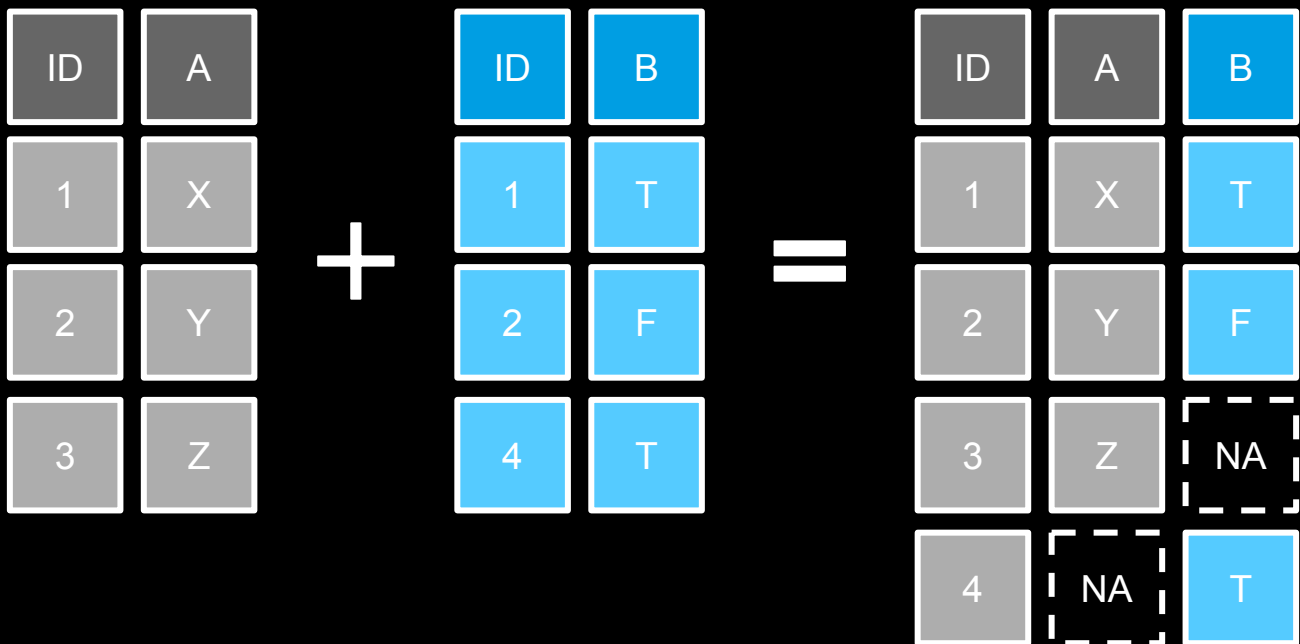
`left_join()`



`inner_join()`



```
full_join()
```

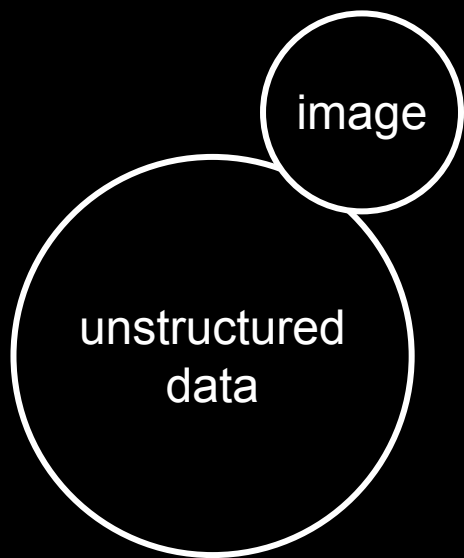


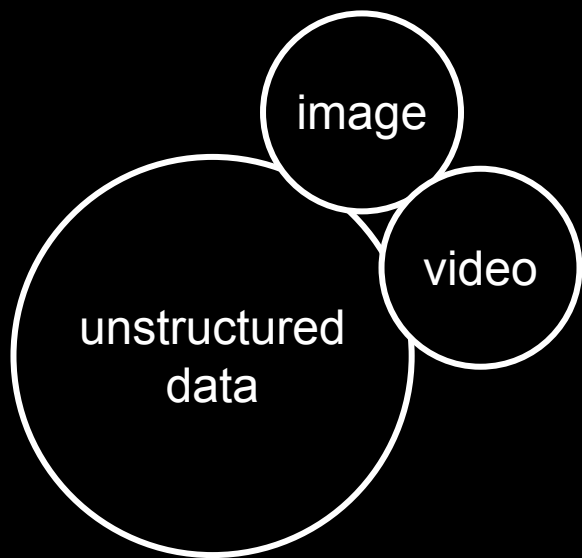
# UNSTRUCTURED DATA

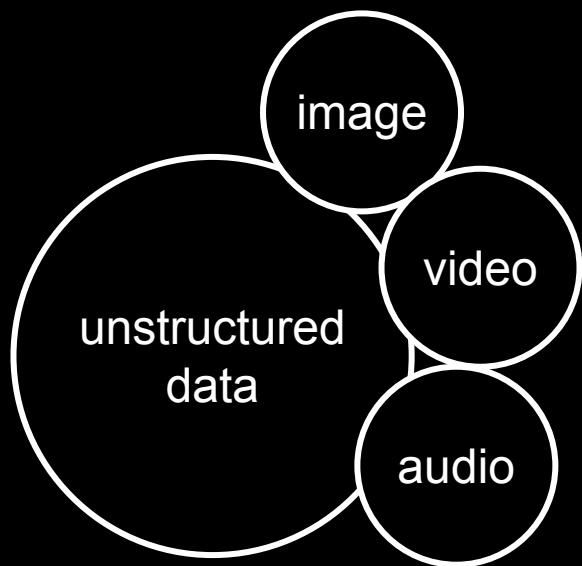


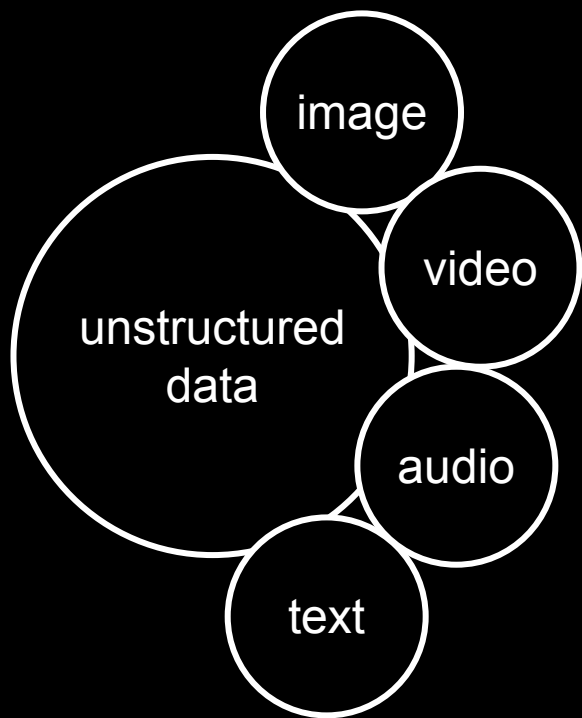
unstructured  
data



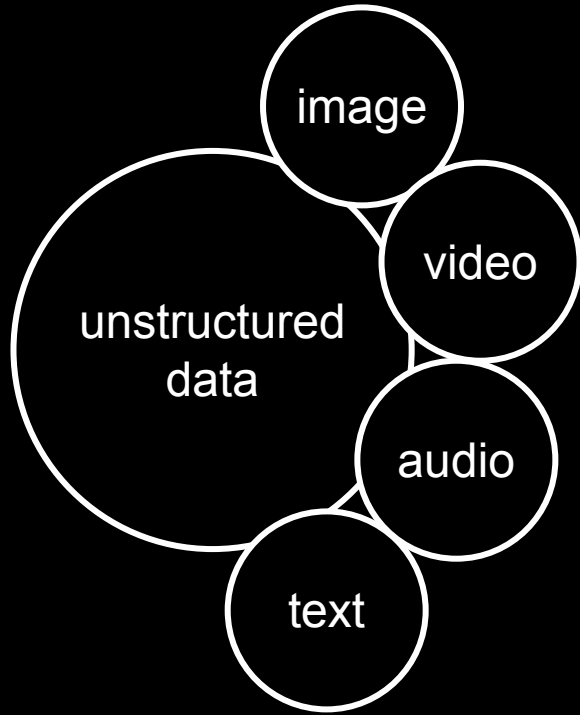




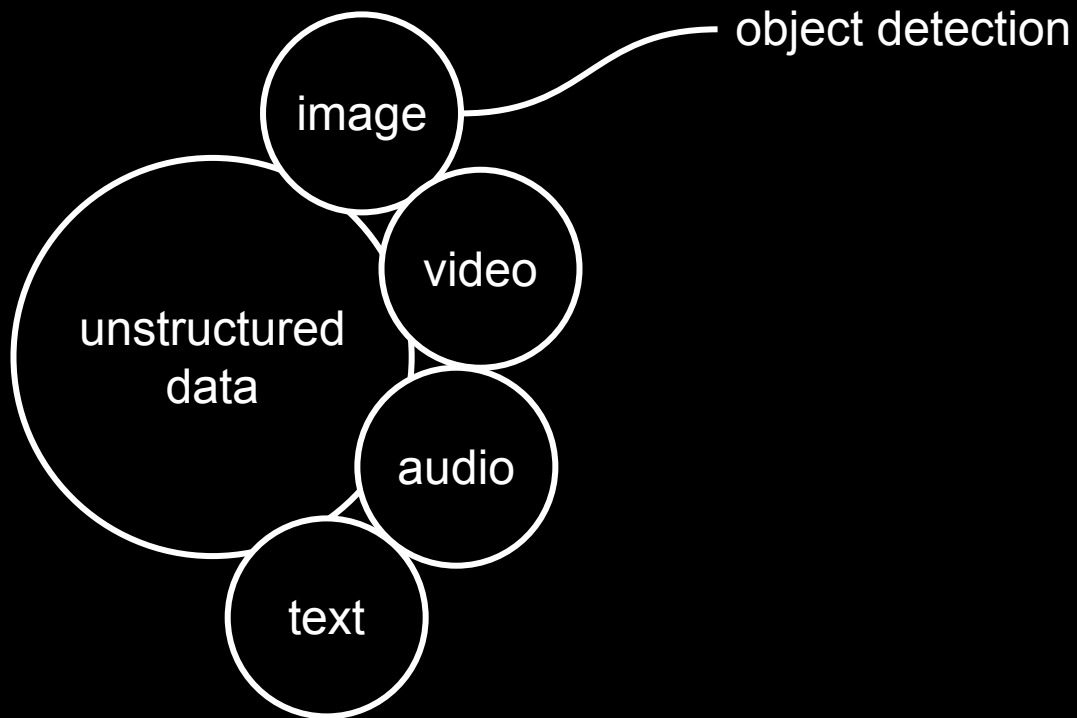




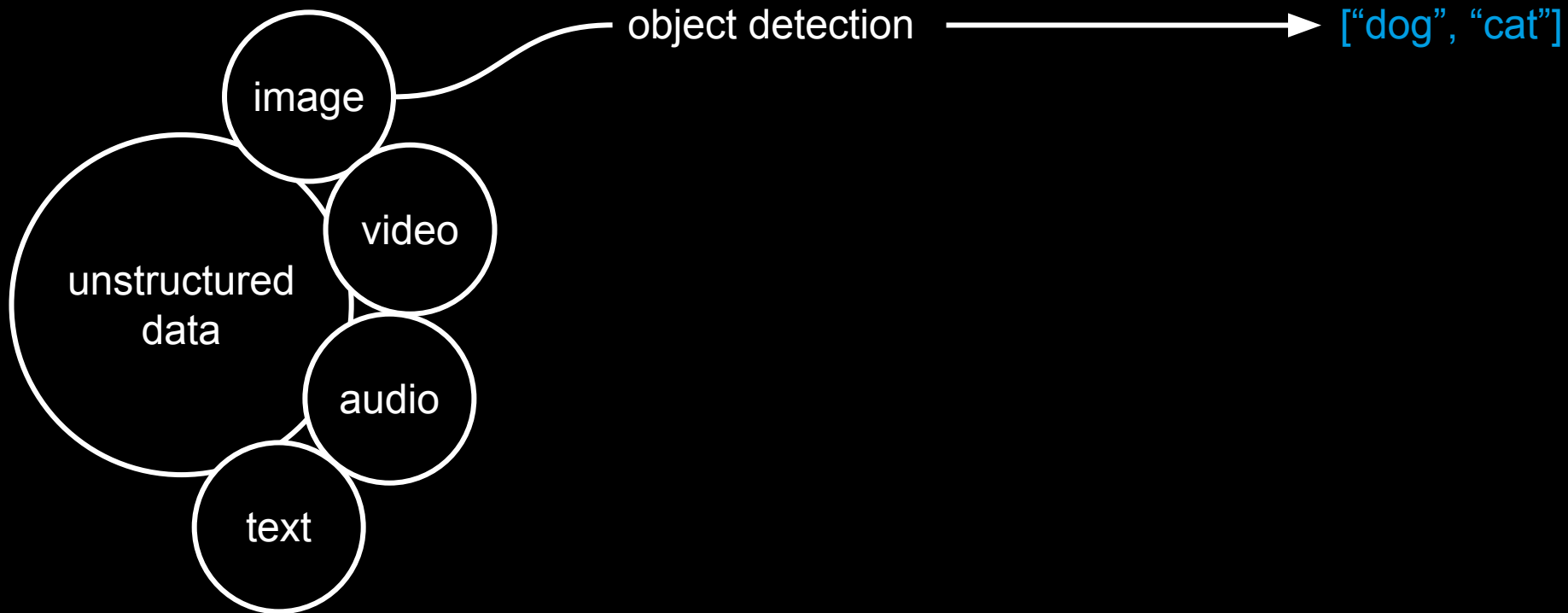
no handles to grab



no handles to grab



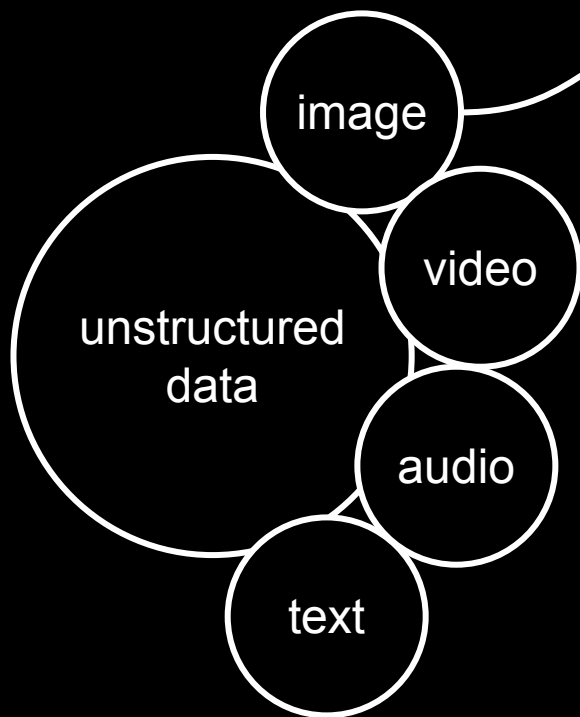
no handles to grab



no handles to grab

algorithm

extracted, structured information



object detection



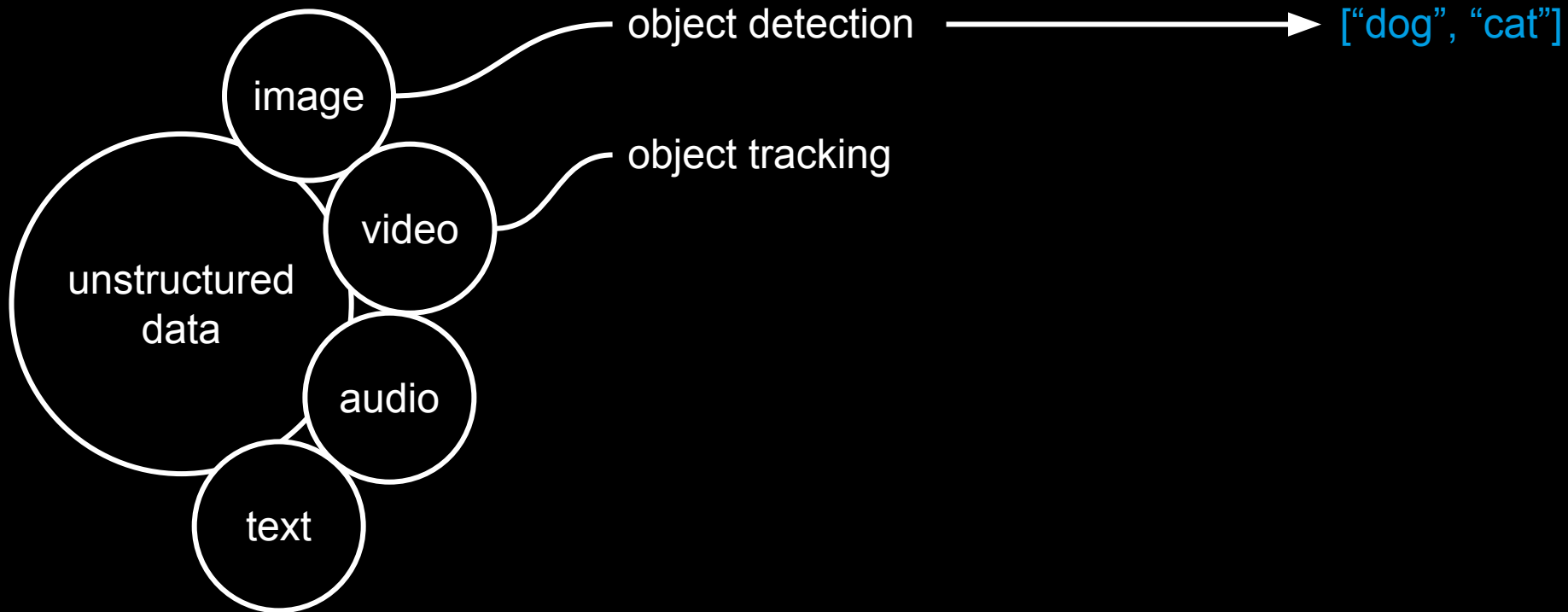
`["dog", "cat"]`



no handles to grab

algorithm

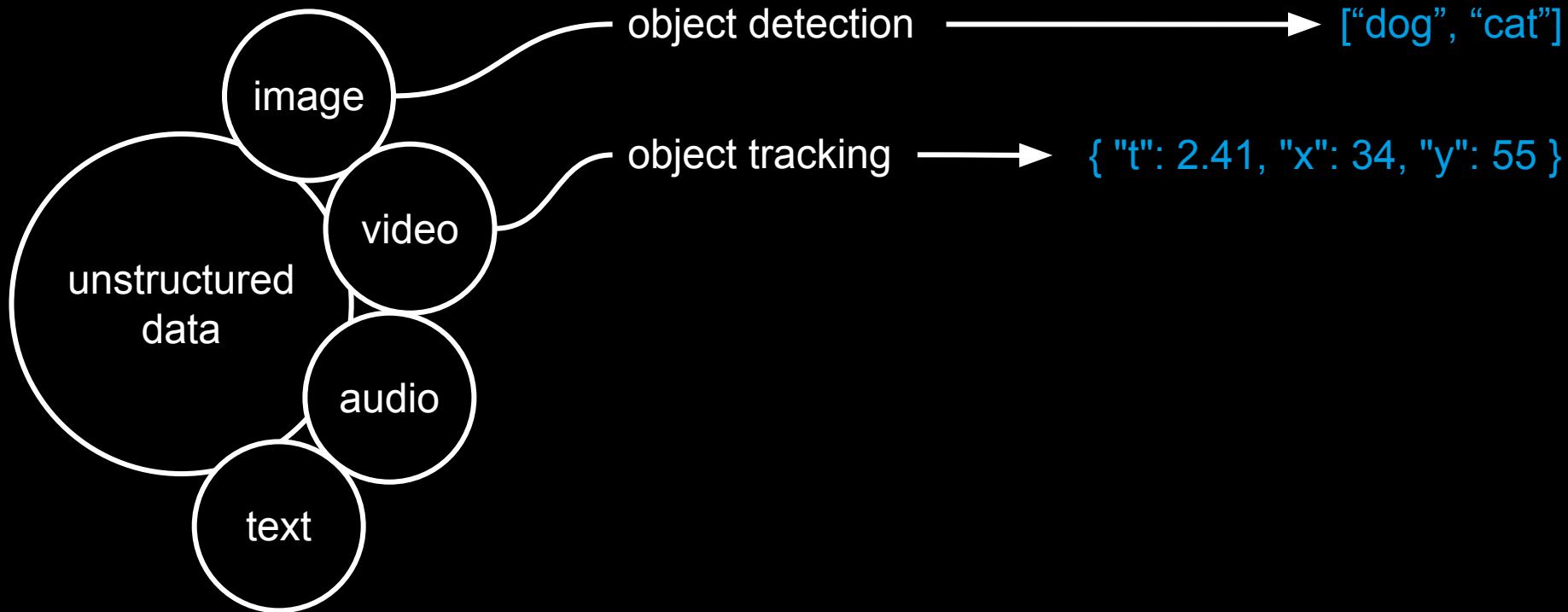
extracted, structured information



no handles to grab

algorithm

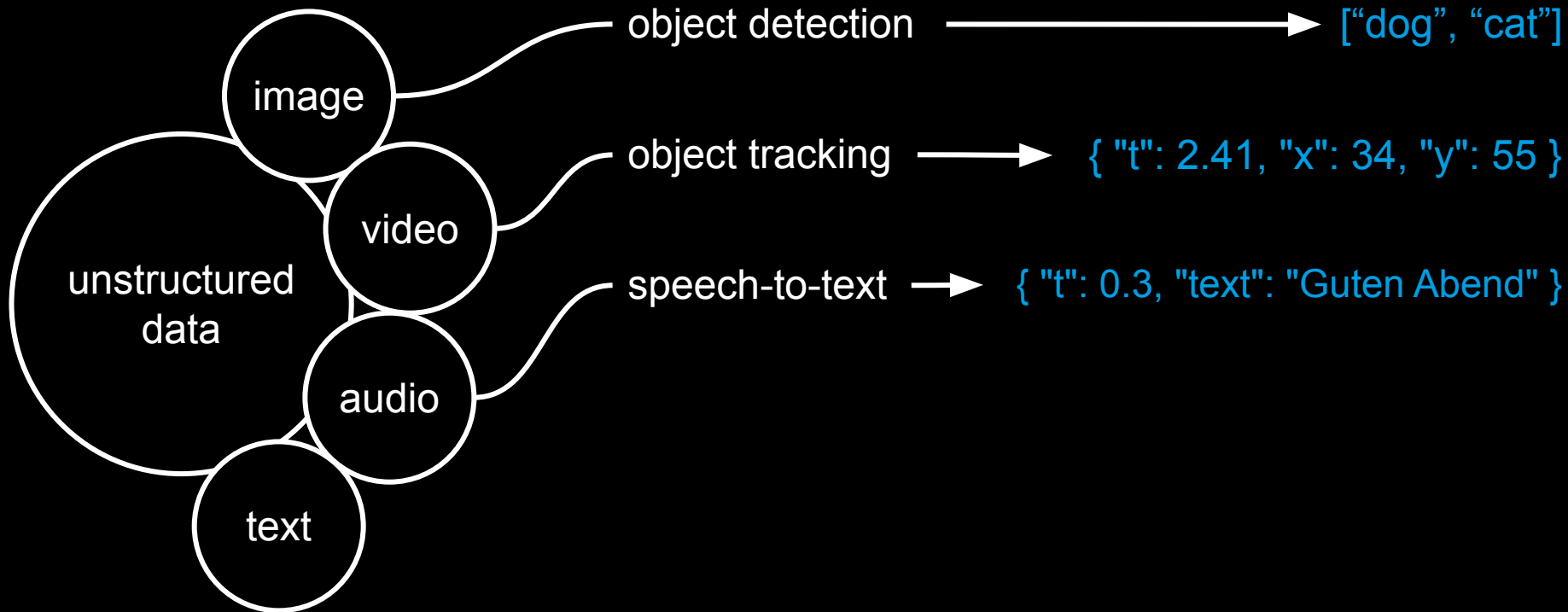
extracted, structured information



no handles to grab

algorithm

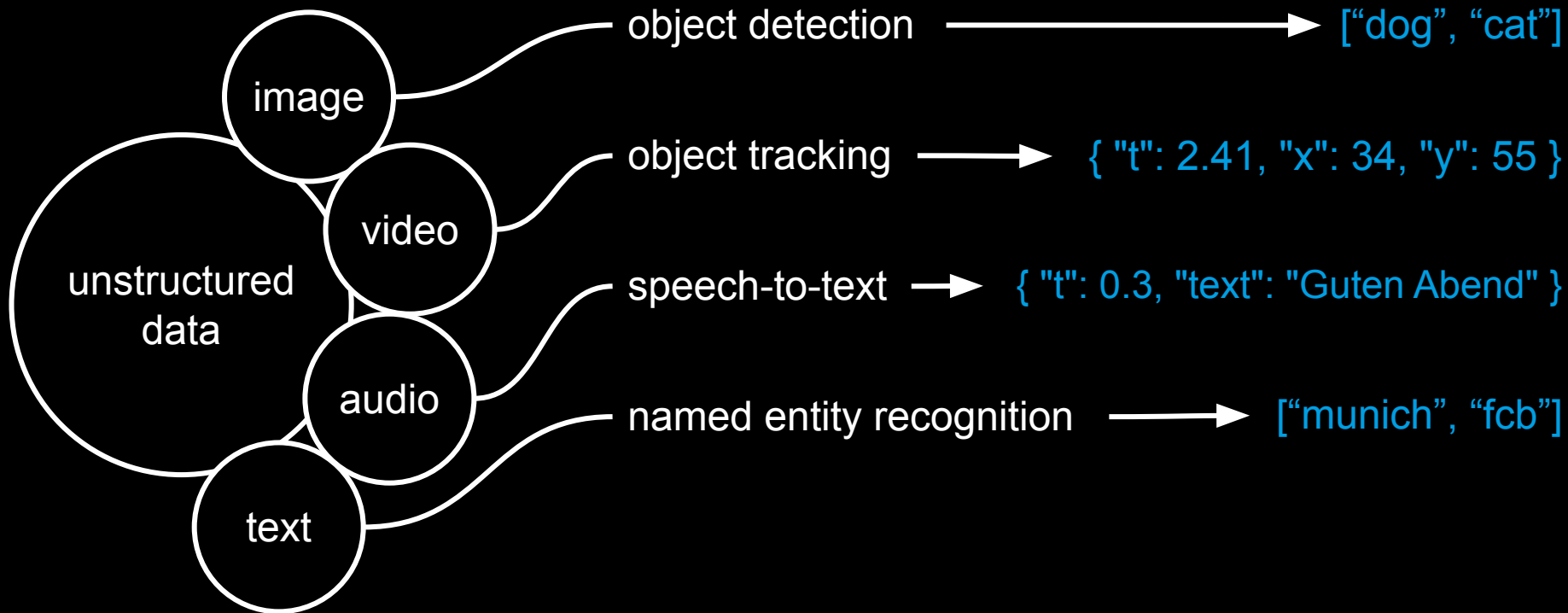
extracted, structured information



no handles to grab

algorithm

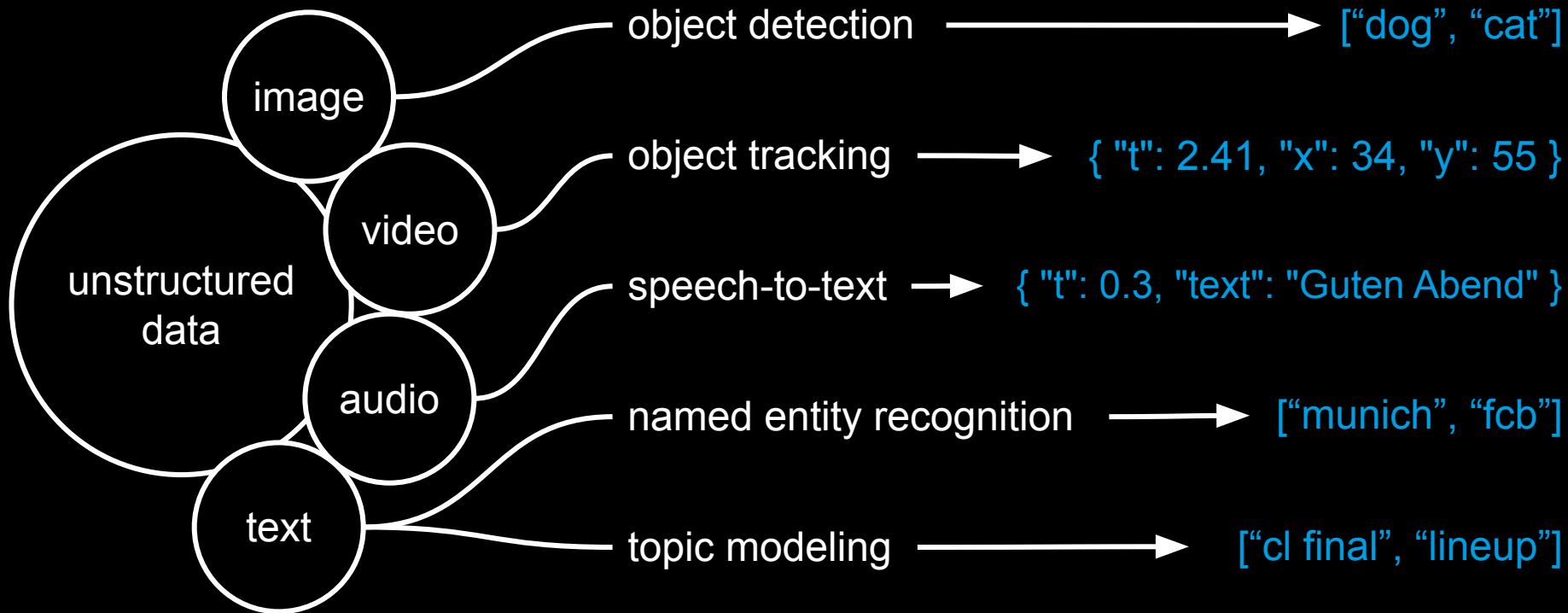
extracted, structured information



no handles to grab

algorithm

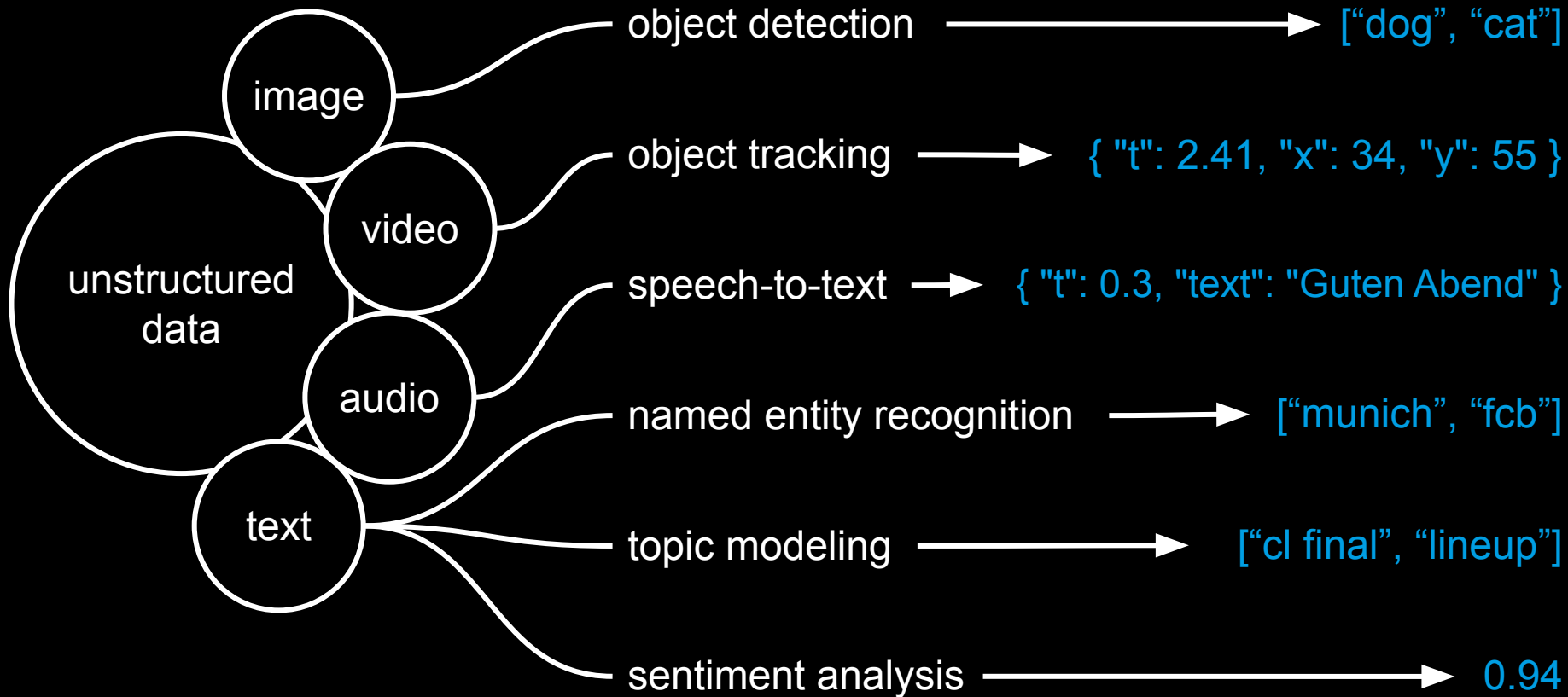
extracted, structured information



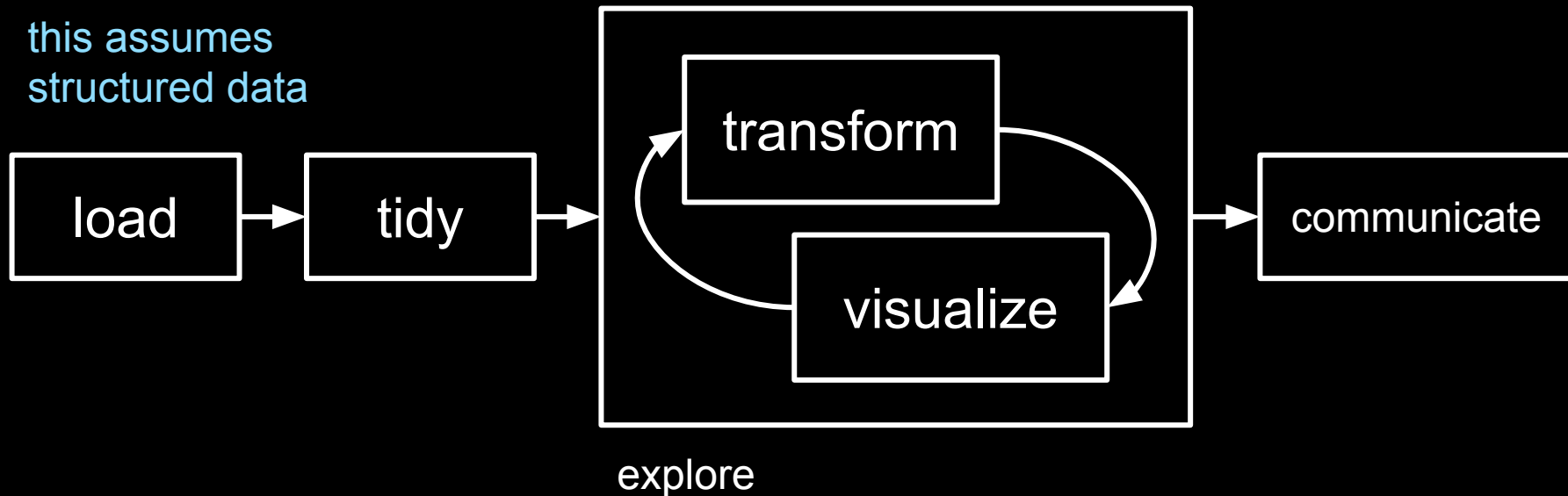
no handles to grab

algorithm

extracted, structured information

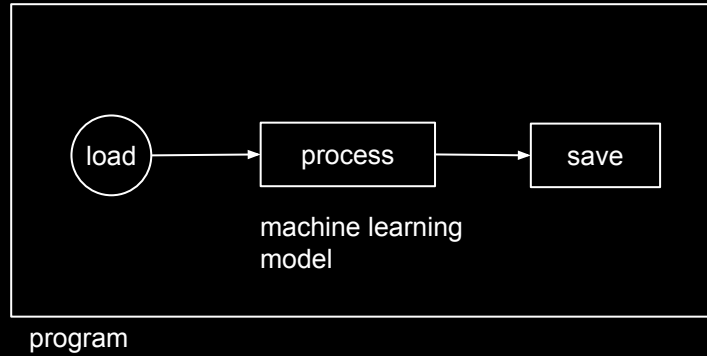


this assumes  
structured data

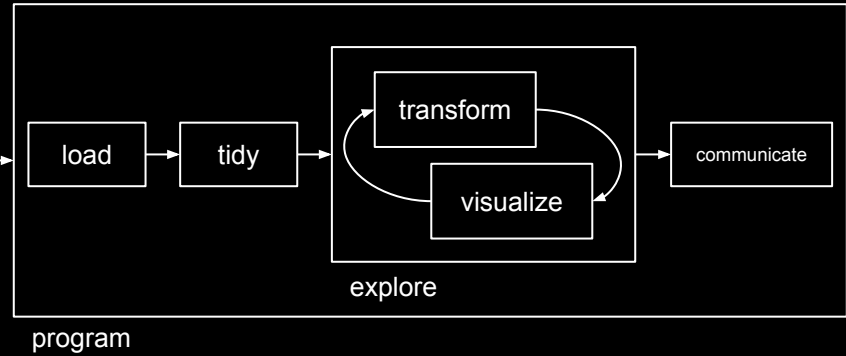


program

pre-process  
unstructured data

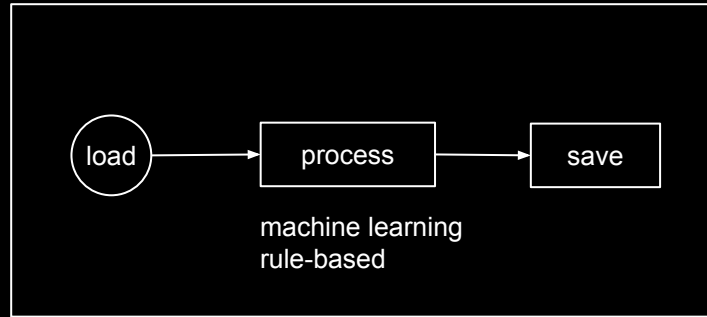


exploratory data  
analysis





pre-process  
unstructured data



program

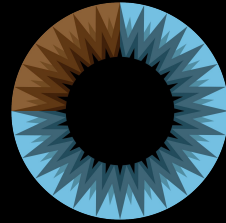


exploratory data  
analysis



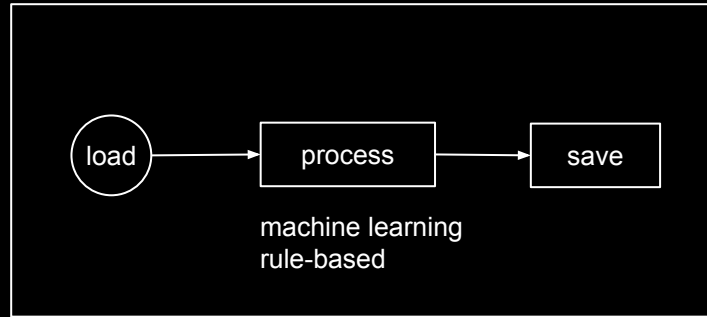
# MACHINE LEARNING

Highly recommended for  
background information



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

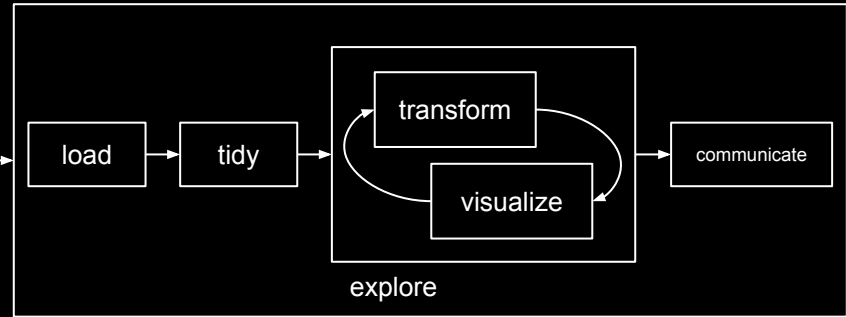
pre-process  
unstructured data



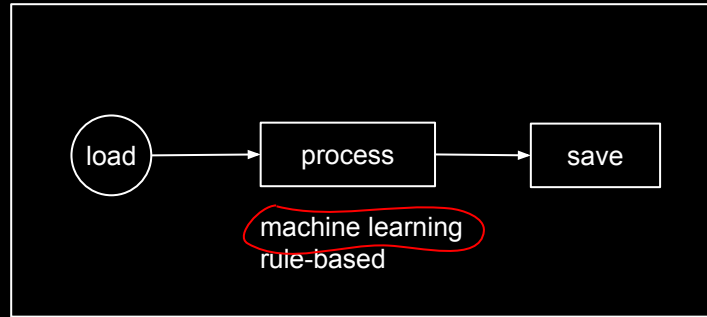
program



exploratory data  
analysis



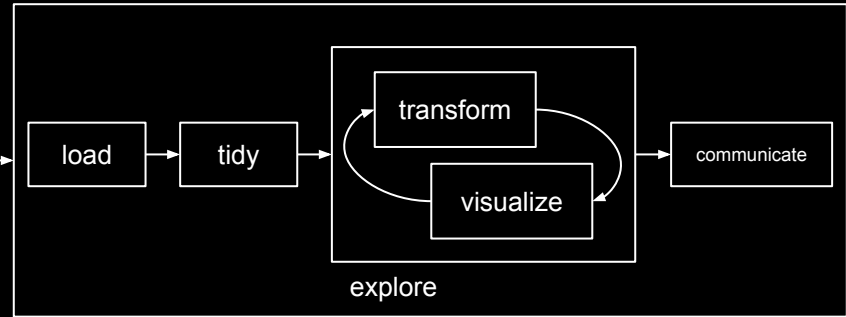
pre-process  
unstructured data



program



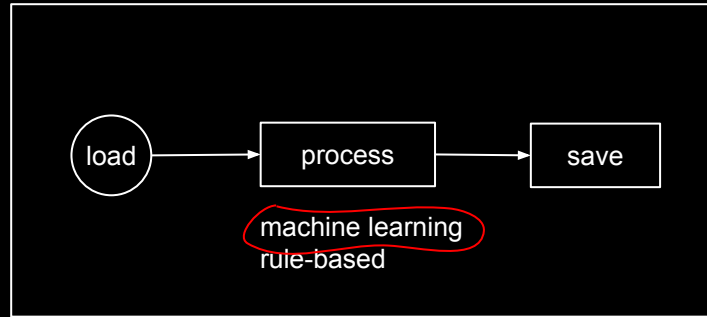
exploratory data  
analysis



program



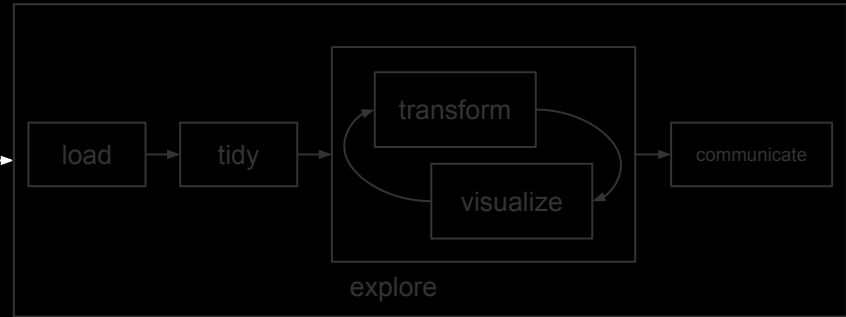
pre-process  
unstructured data



program



exploratory data  
analysis



program





machine learning

program



YouTube



process

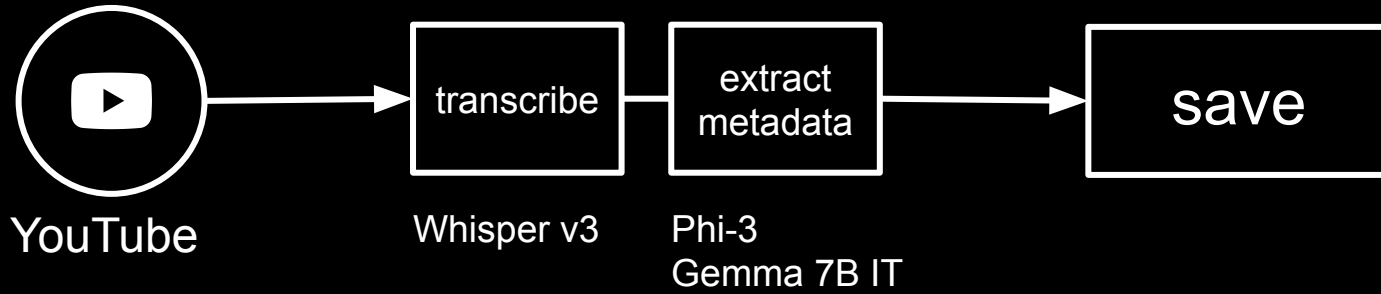
machine learning



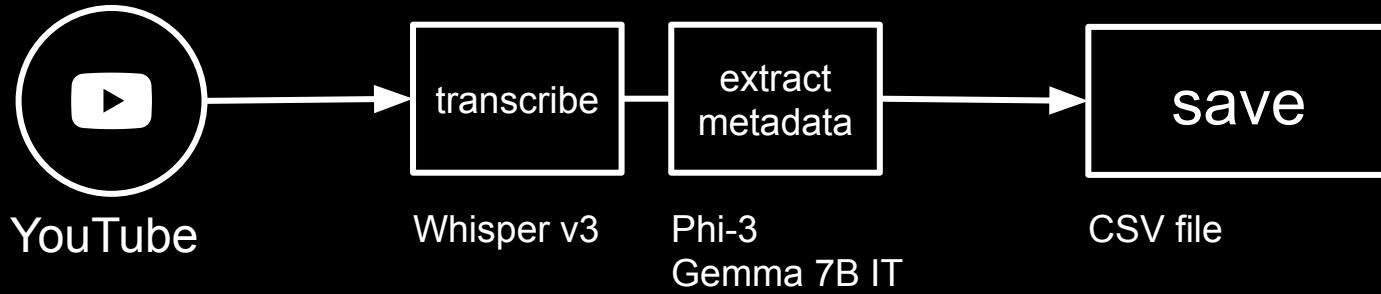
save

program





program



program

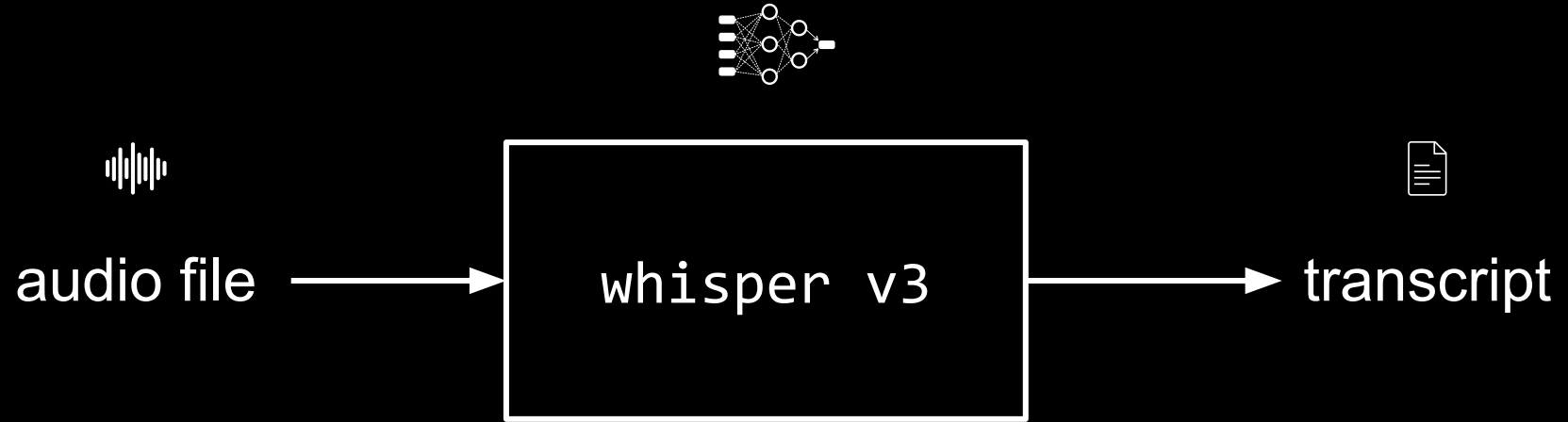
# YouTube API

# Whisper v3

<https://arxiv.org/abs/2212.04356>



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



# 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/>



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

Explain the binary number system.

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

<instruction>

<context>

<input data>

<output indicator>

## example prompt

Classify the text into neutral,  
negative or positive.

Text: "What a great dinner!"

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>

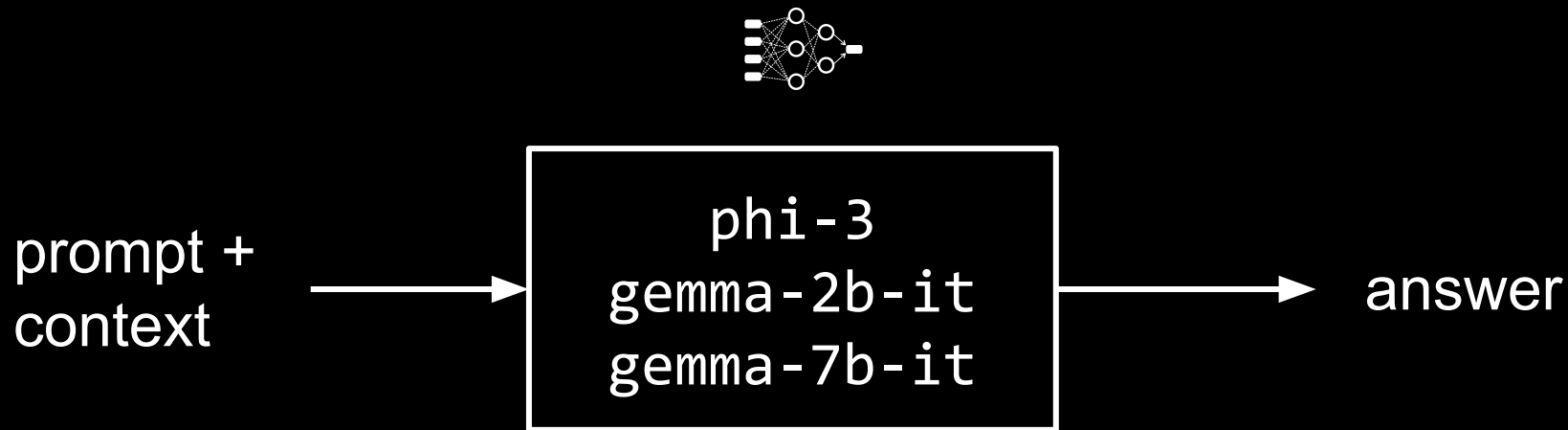
# Gemma 2B / 7B Instruct

<https://arxiv.org/abs/2403.08295>



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

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



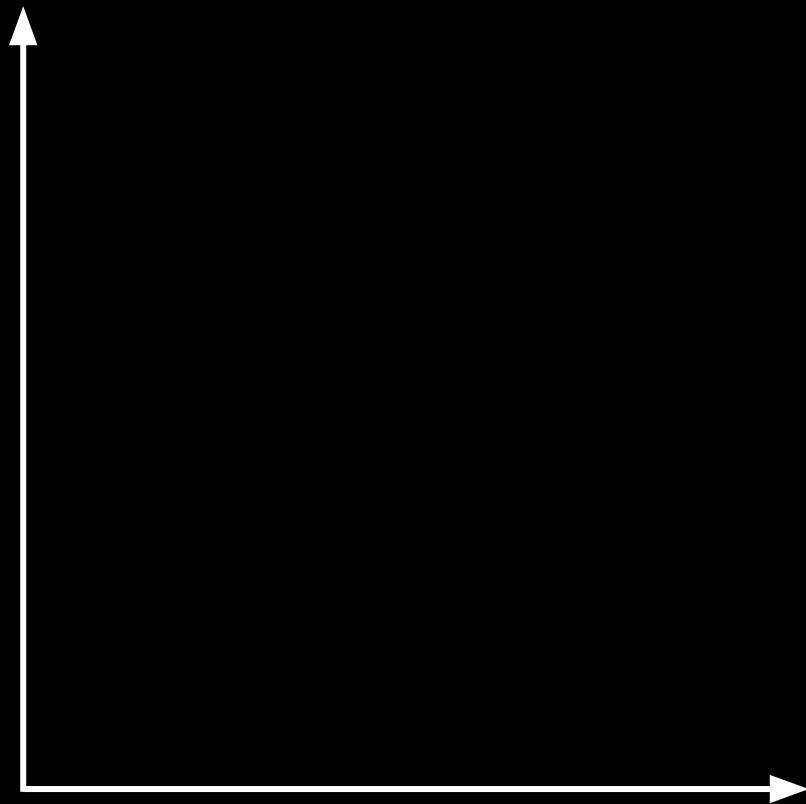


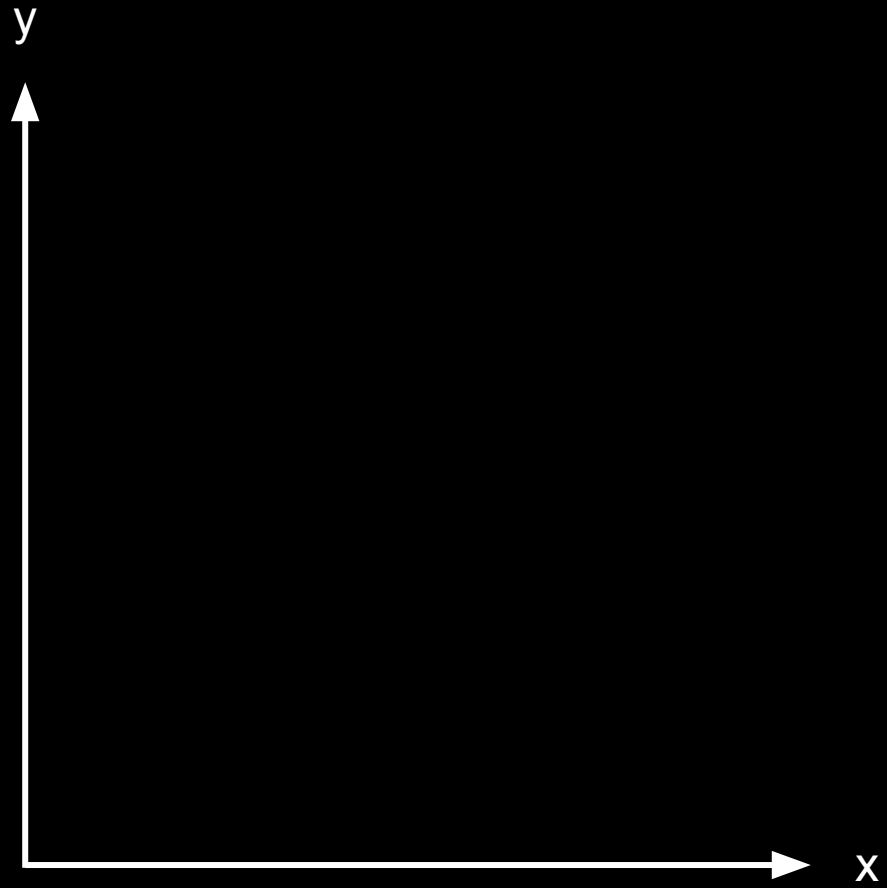
OpenAI GPT-4o

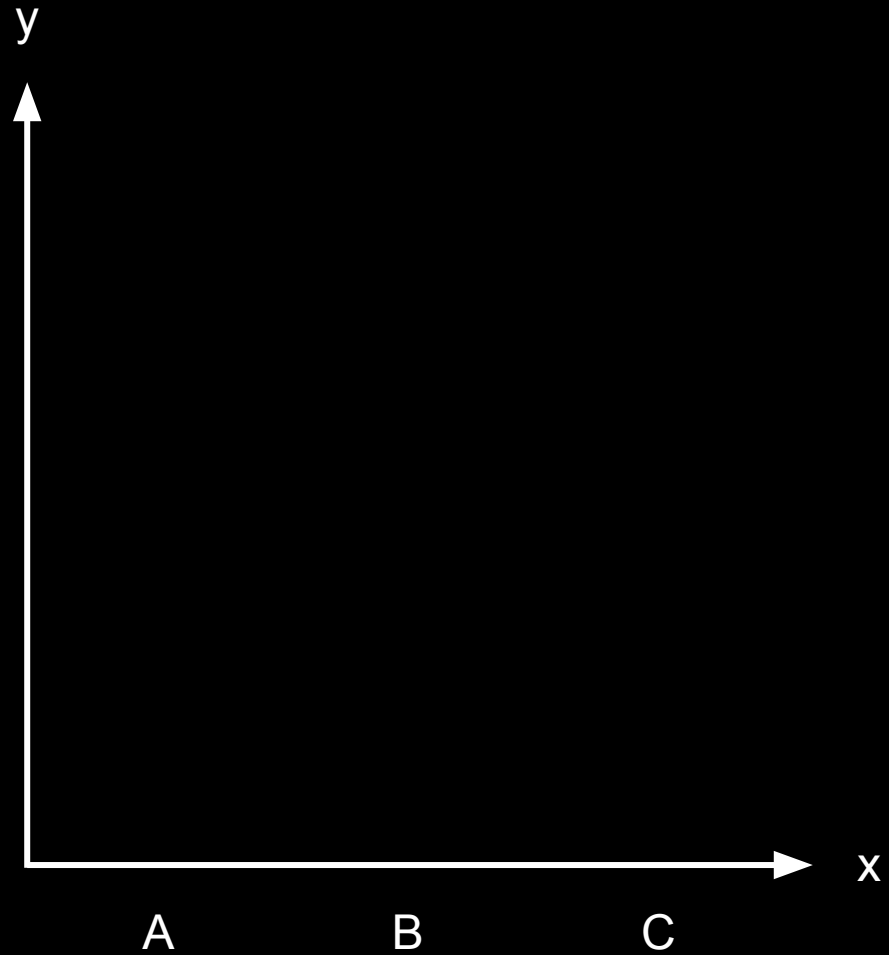
# VISUALIZE DATA

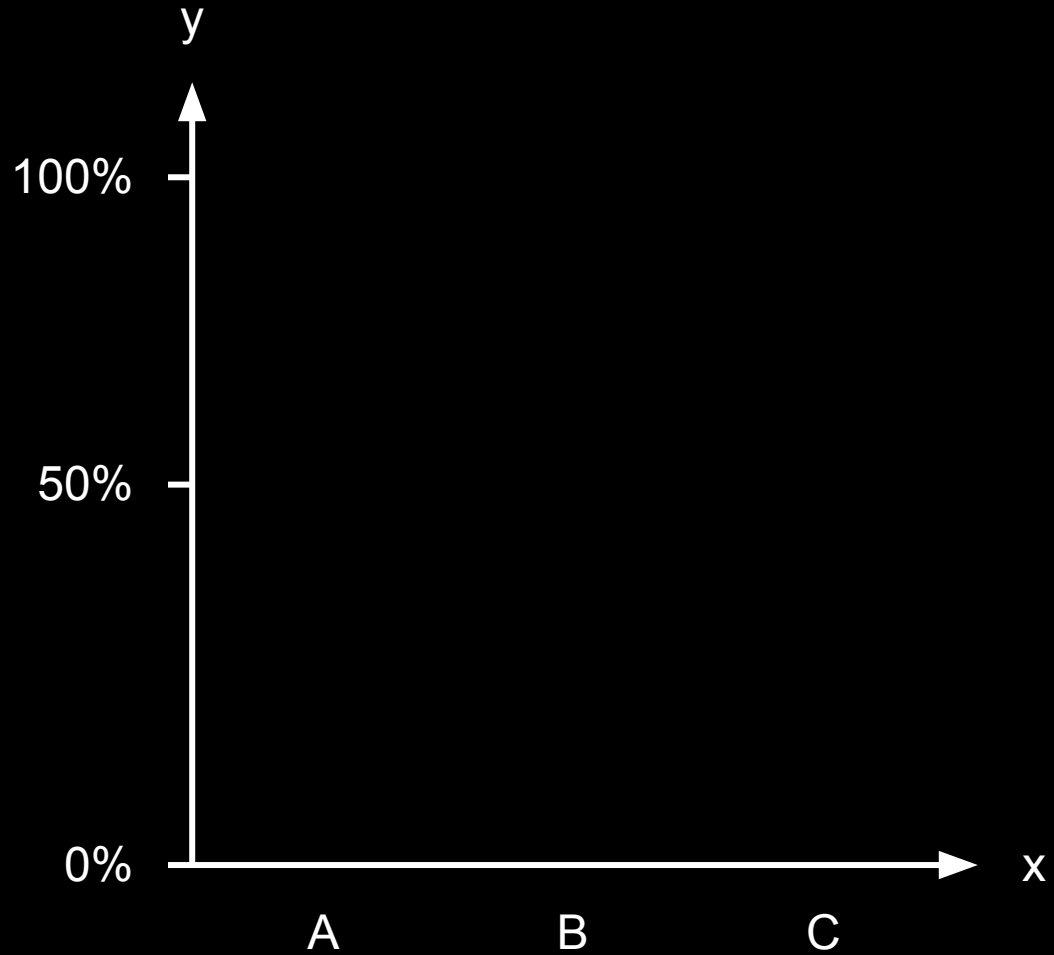
# data

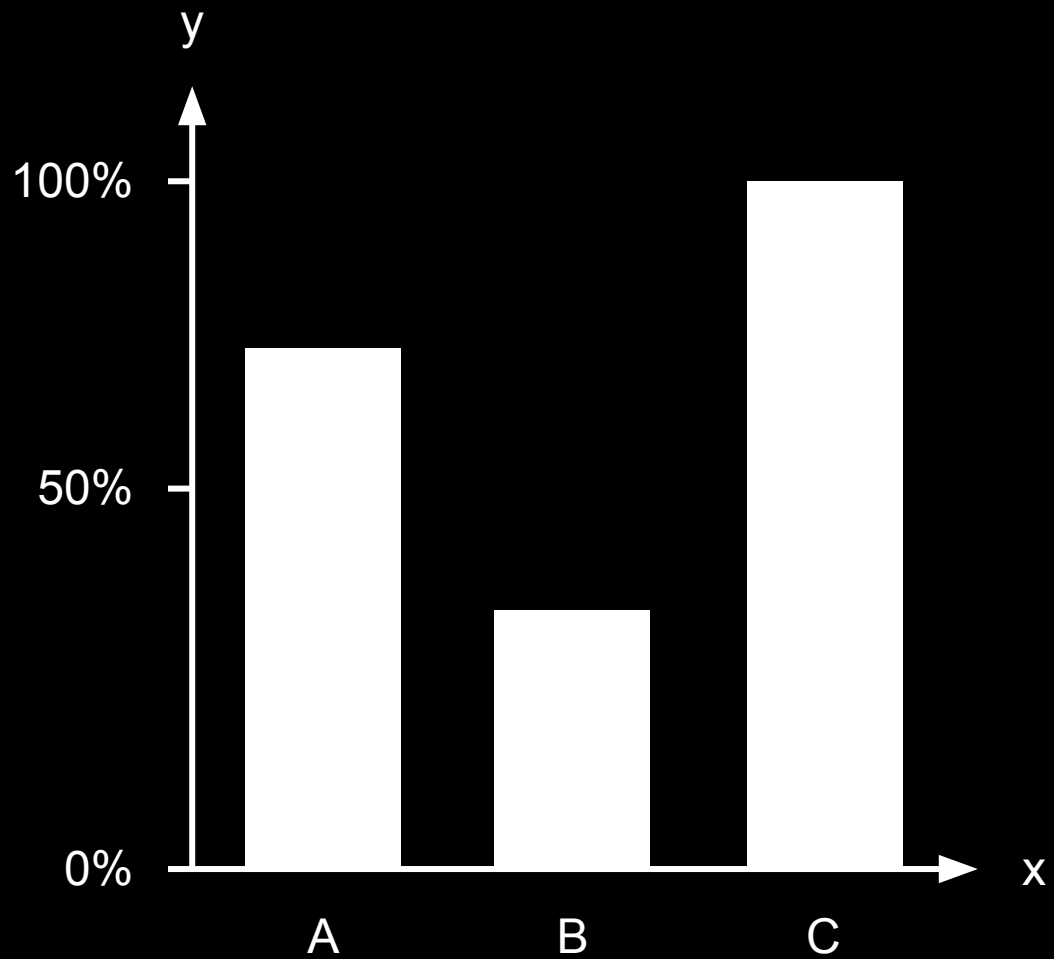
category	pct
A	75
B	33
C	100





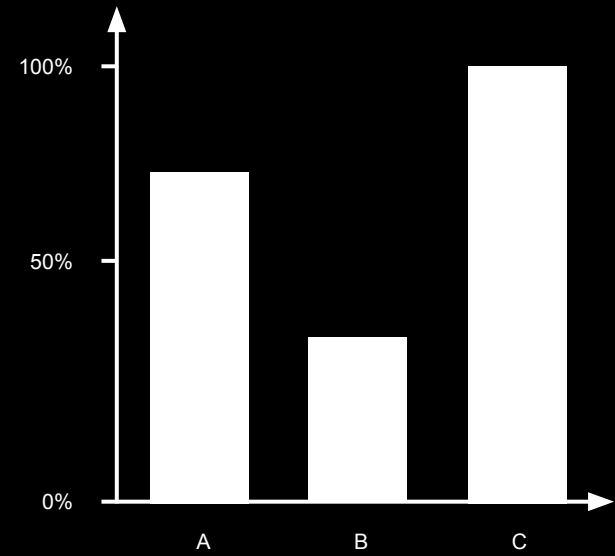








category	pct
A	75
B	33
C	100



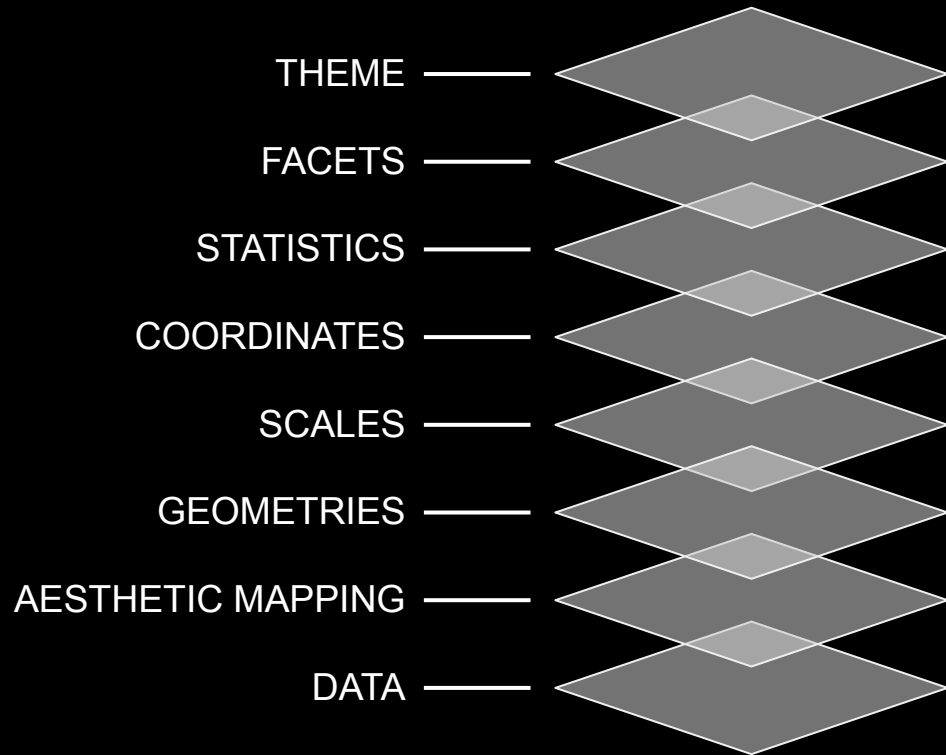
{{ ggplot2 }}

why visualize?

{{ ggplot2 }}

grammar of graphics

any  
data  
visualization



has useful defaults

mandatory



THEME

FACETS

STATISTICS

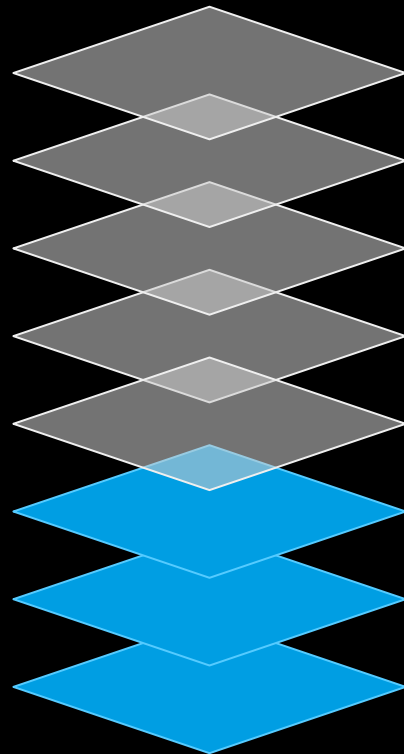
COORDINATES

SCALES

GEOMETRIES

AESTHETIC MAPPING

DATA



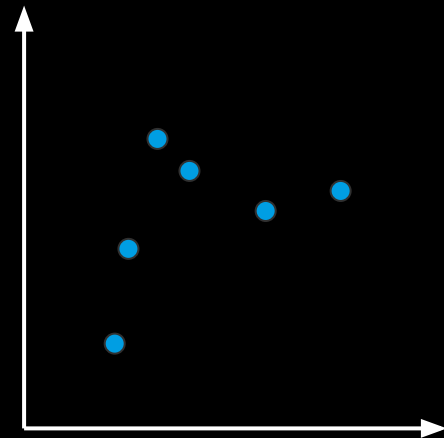
`ggplot()`

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

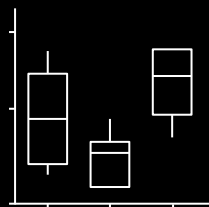
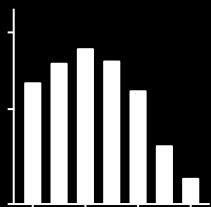
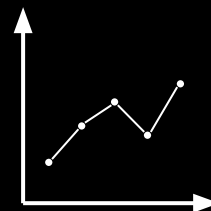
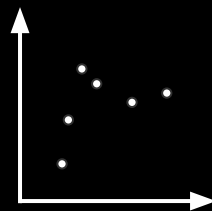
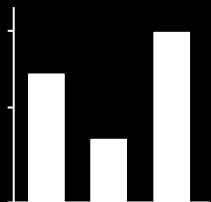


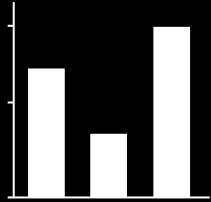
```
ggplot() +  
  aes() +  
  geom_point()
```

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

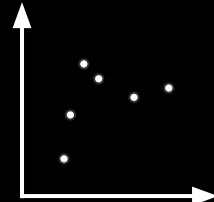


basic plots

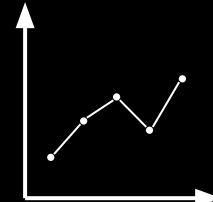




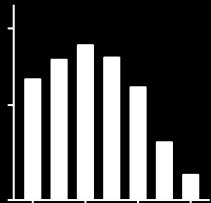
bar chart



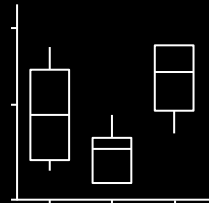
scatter plot



line chart

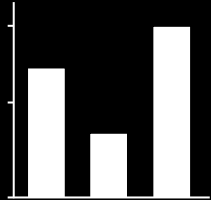


histogram



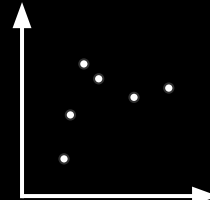
box plot

amounts  
proportions  
distributions (discrete)



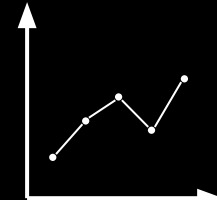
bar chart

associations  
patterns



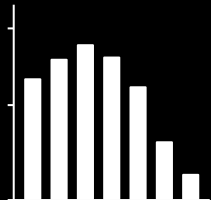
scatter plot

trends  
developments



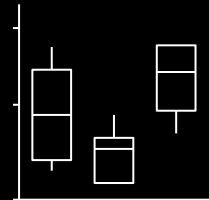
line chart

distributions (continuous)



histogram

compare distributions (continuous)



box plot

# COMMUNICATE FINDINGS

# Quarto



# PYTHON

{{ reticulate }}