

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?



NOT a data analysis

did you summarize the data?

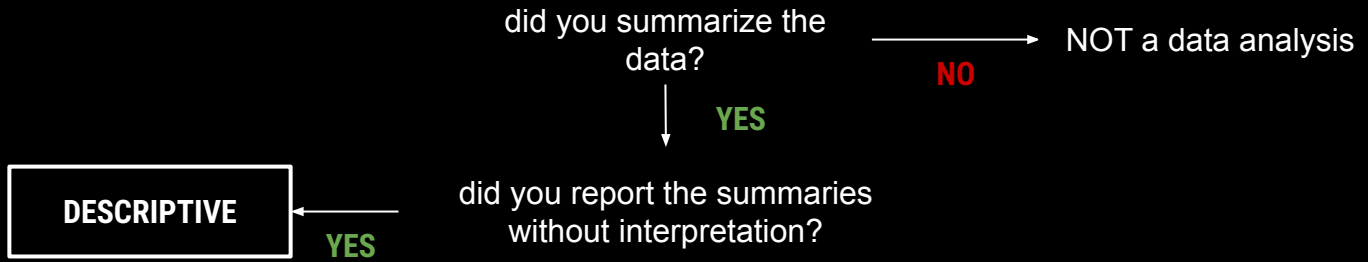


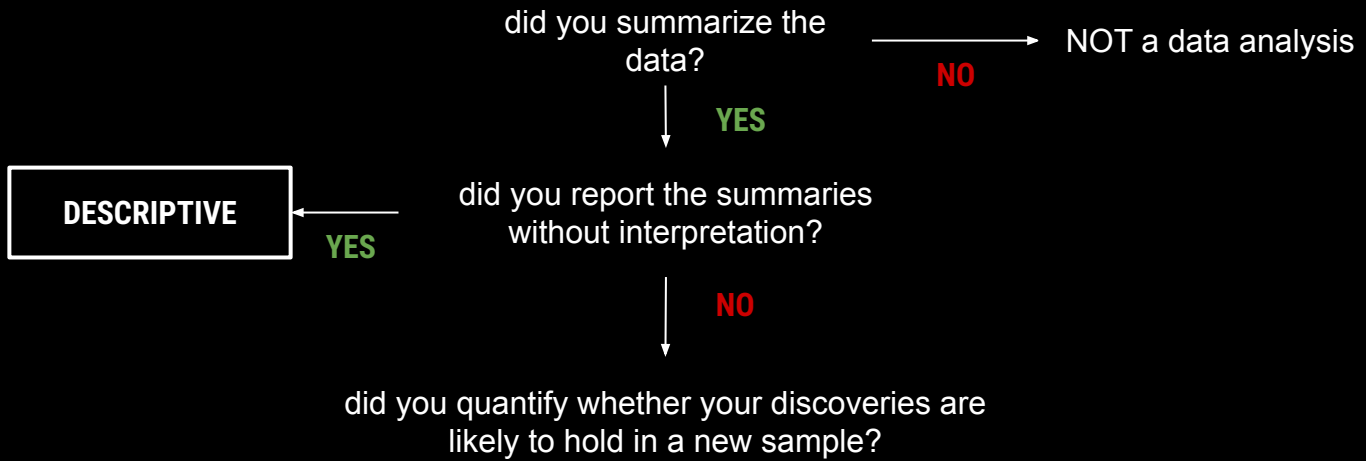
NOT a data analysis

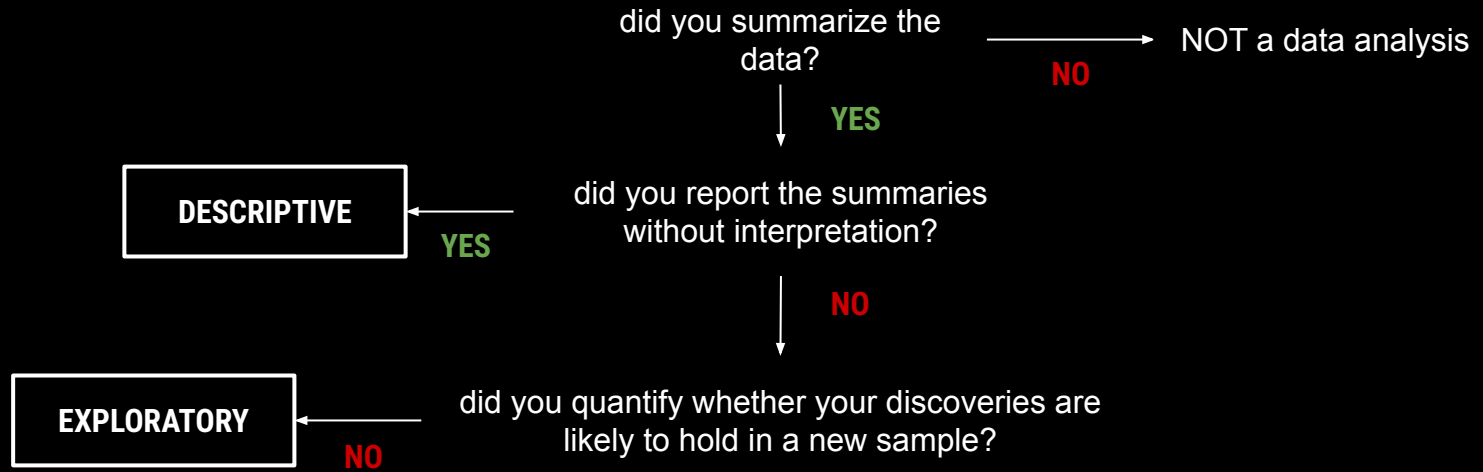


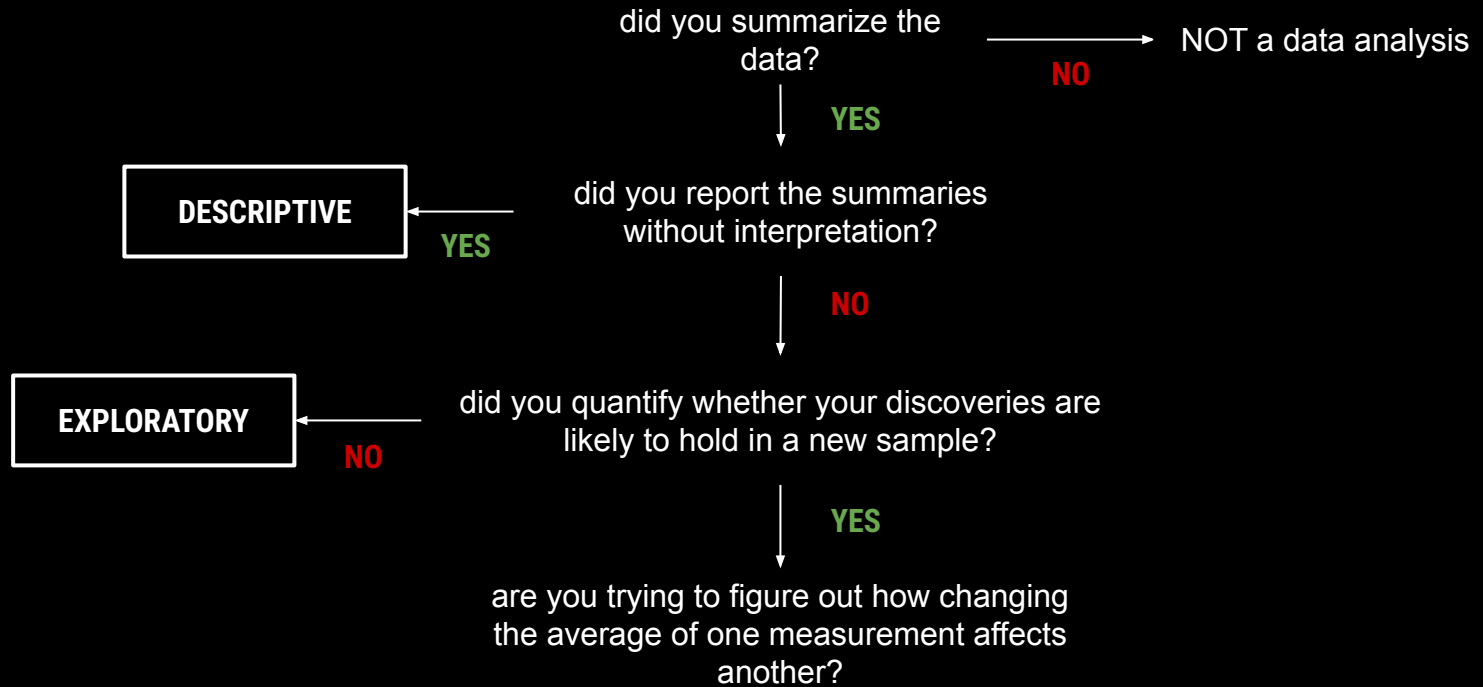
YES

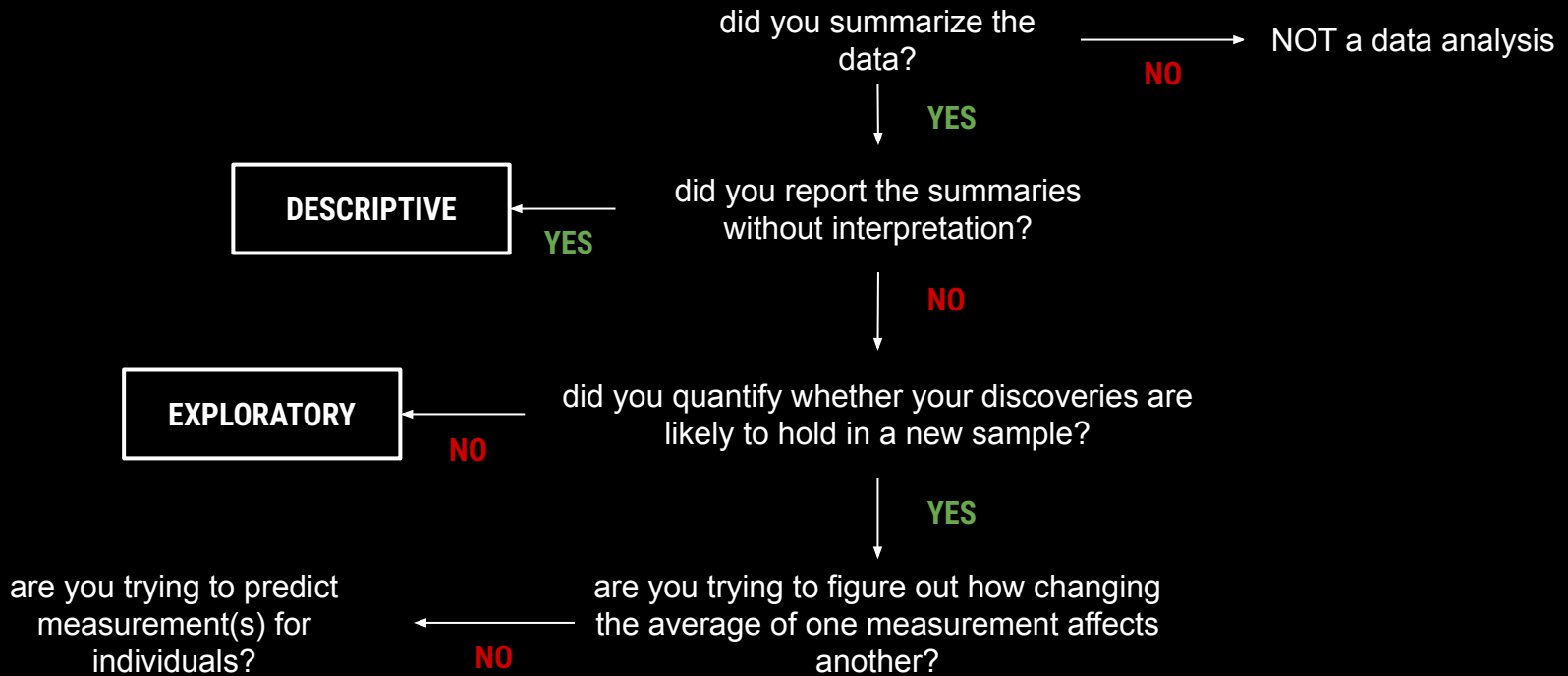
did you report the summaries without interpretation?

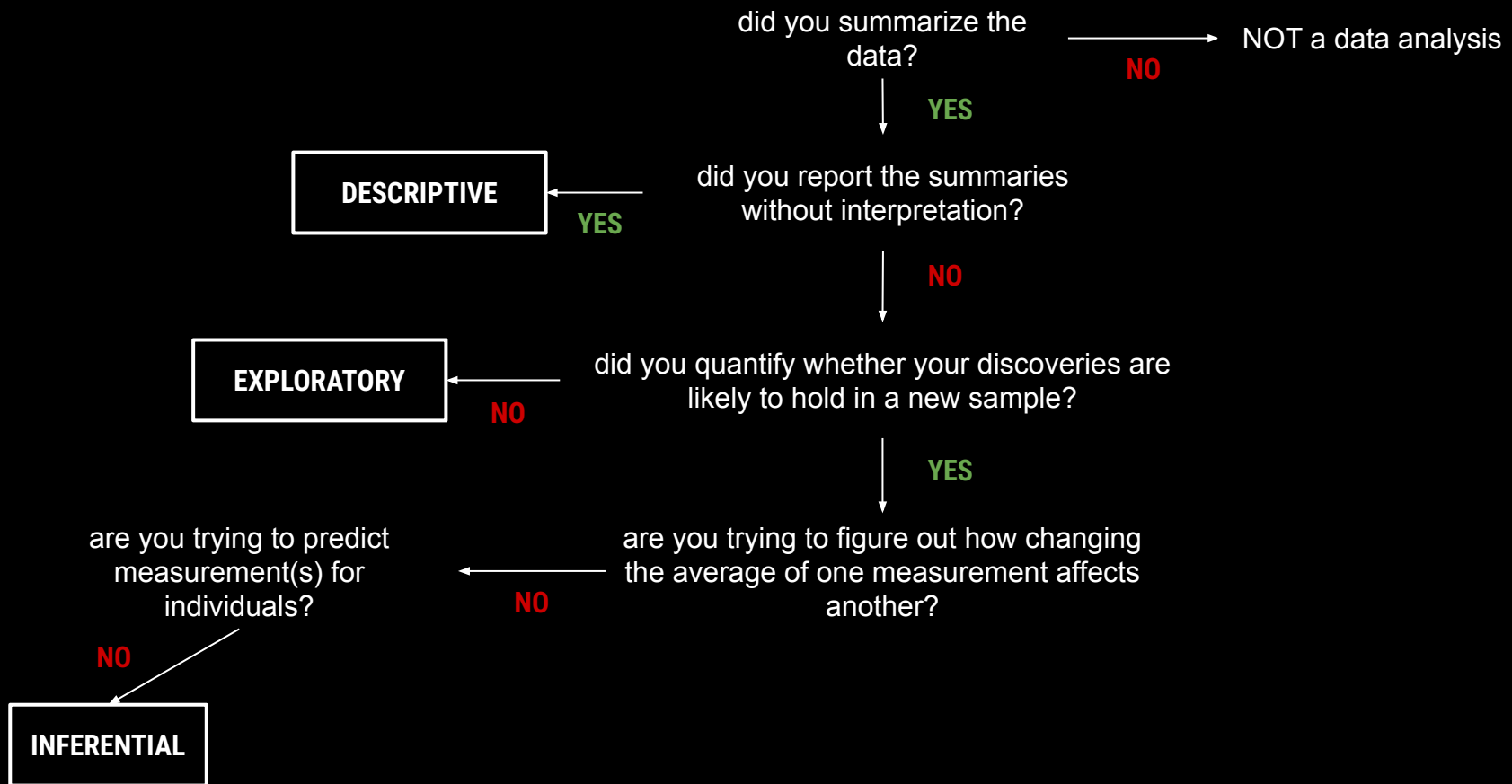


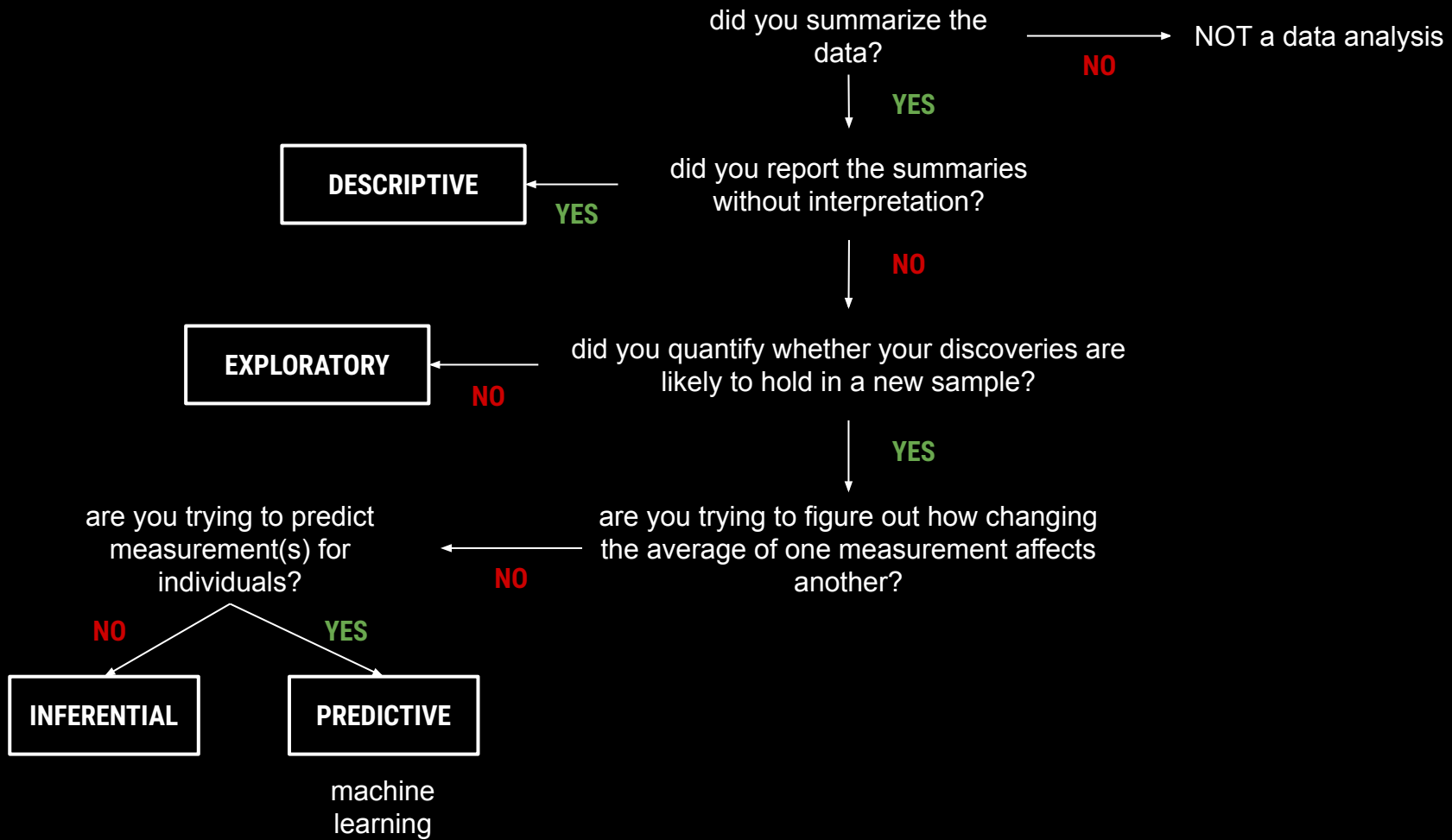


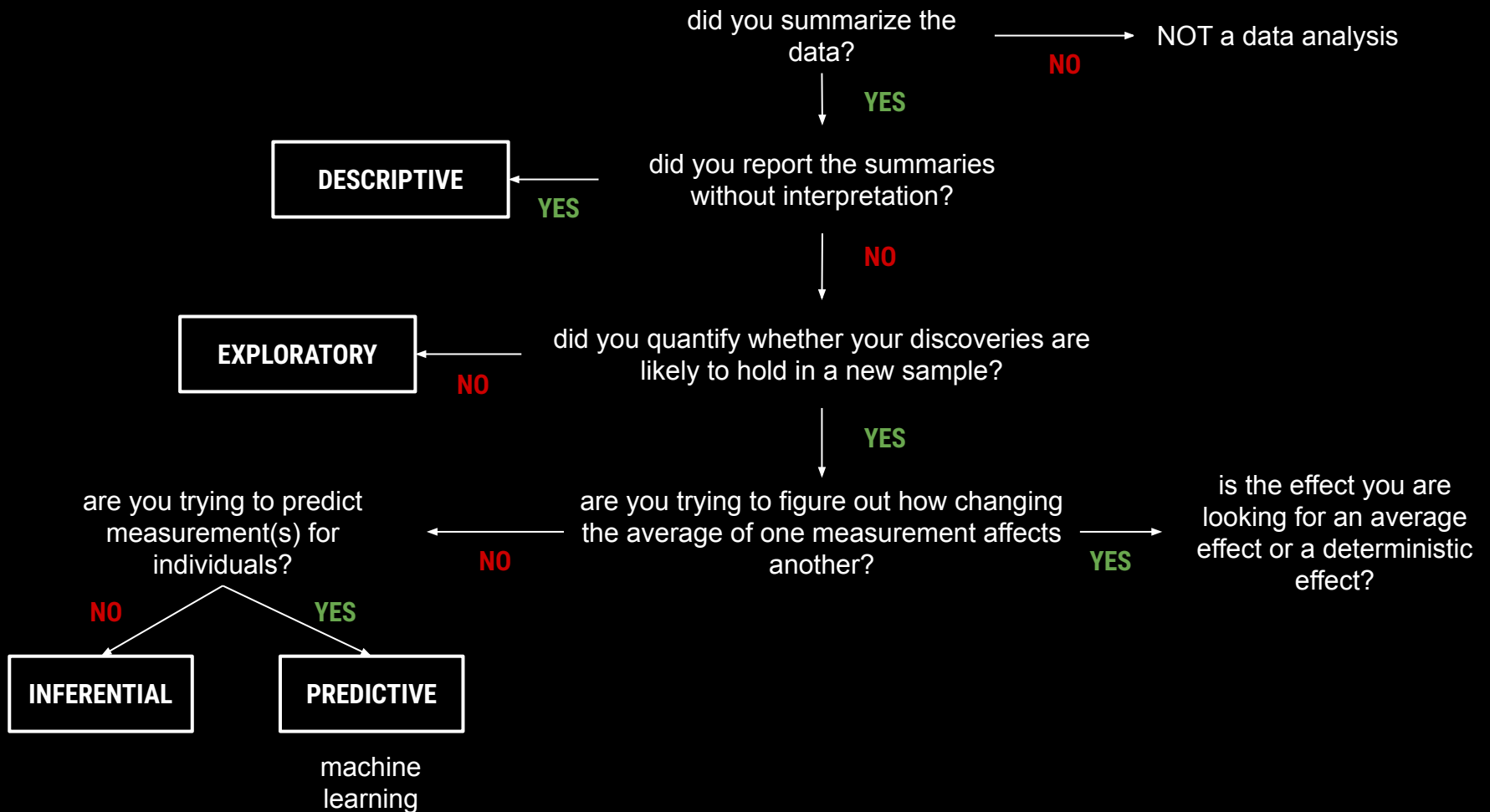


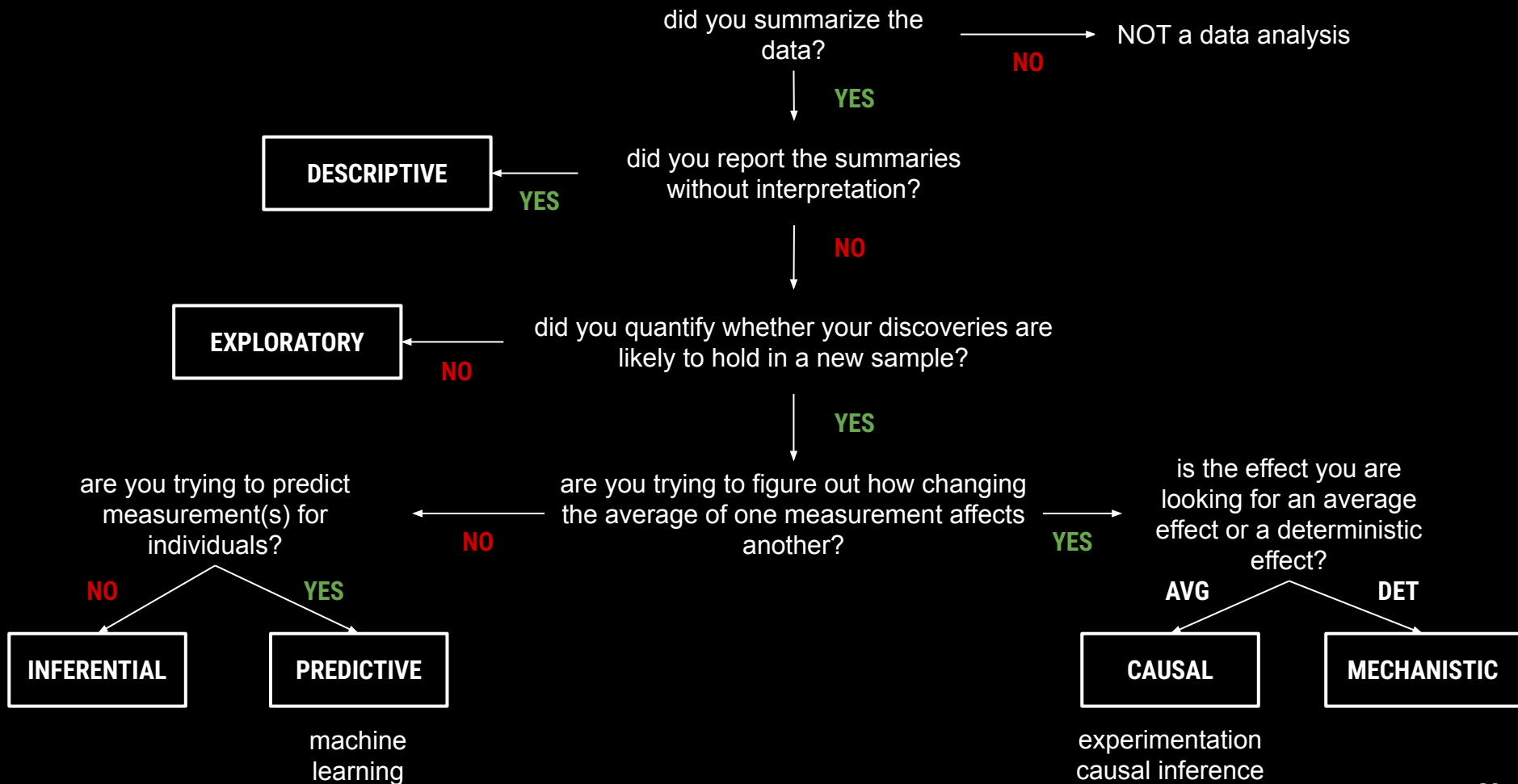


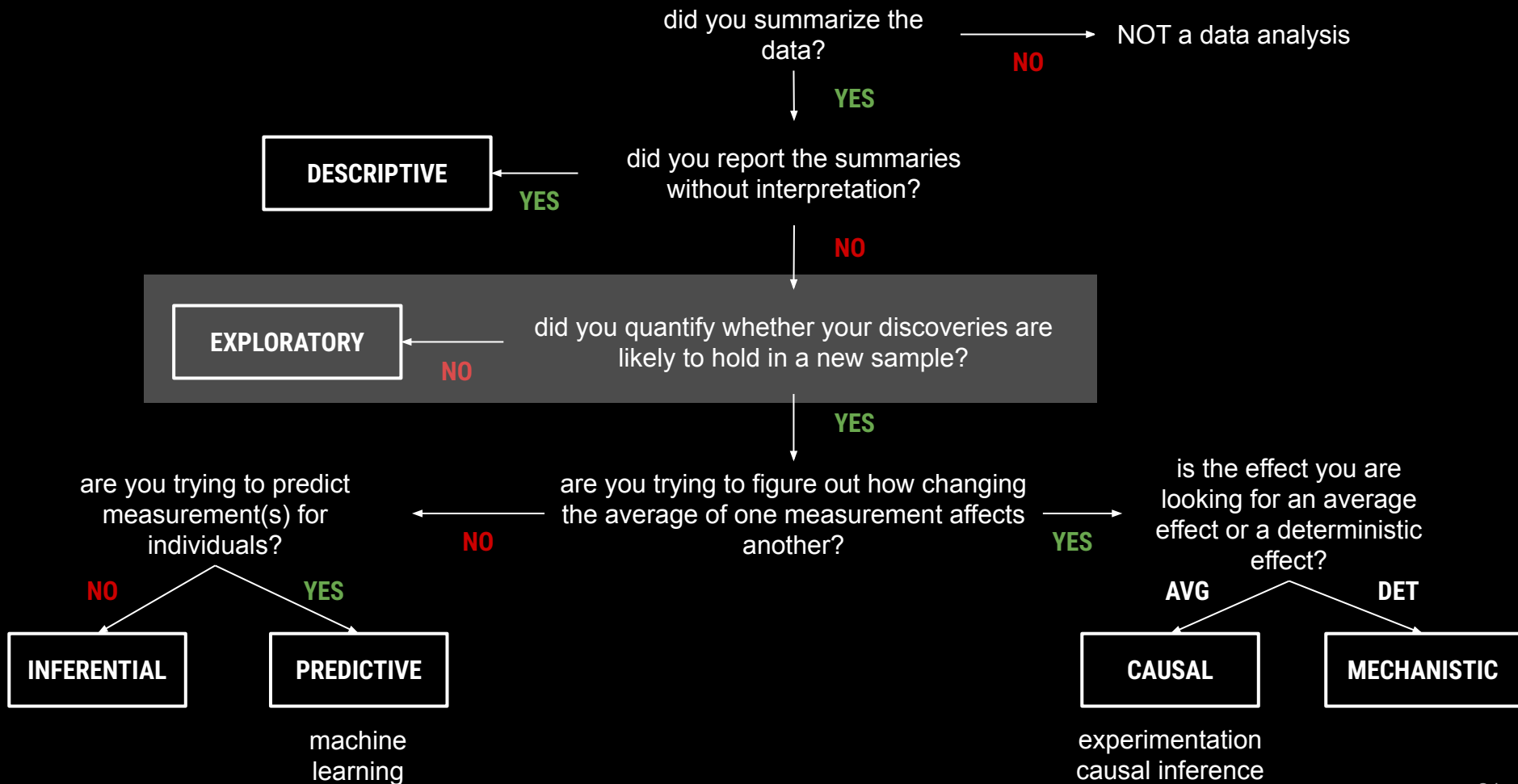






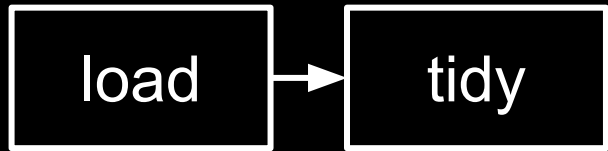


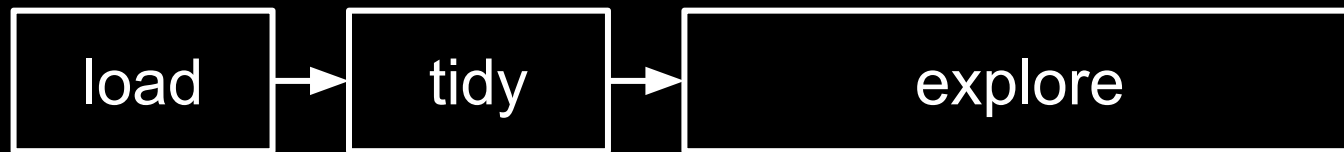




EXPLORATORY DATA ANALYSIS

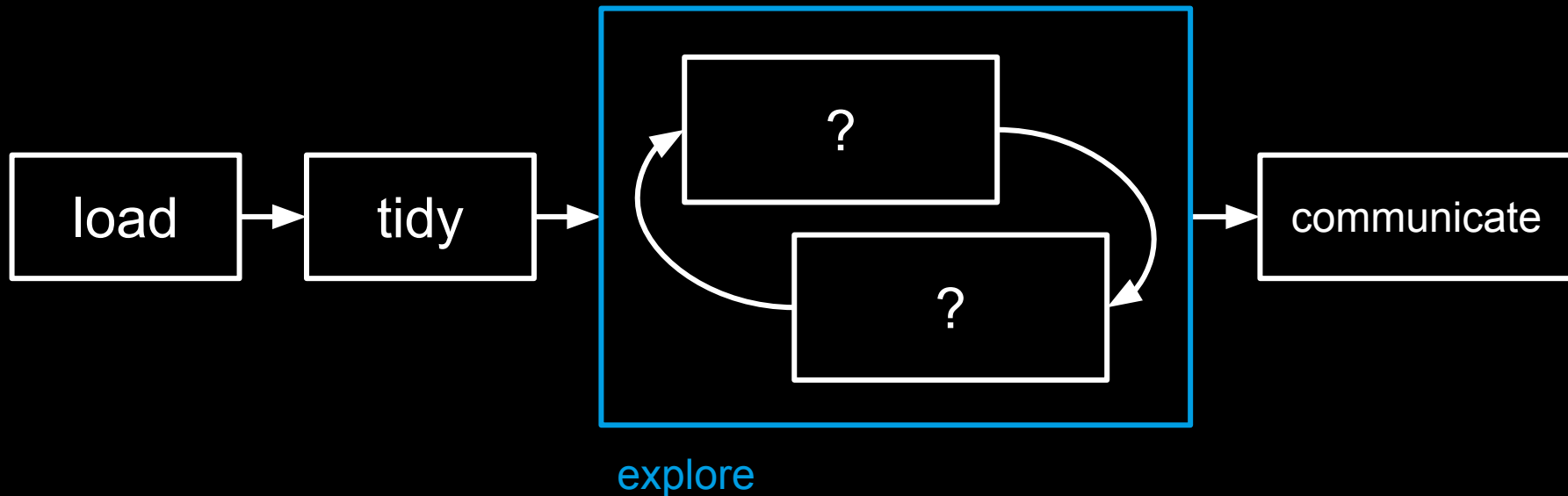
load

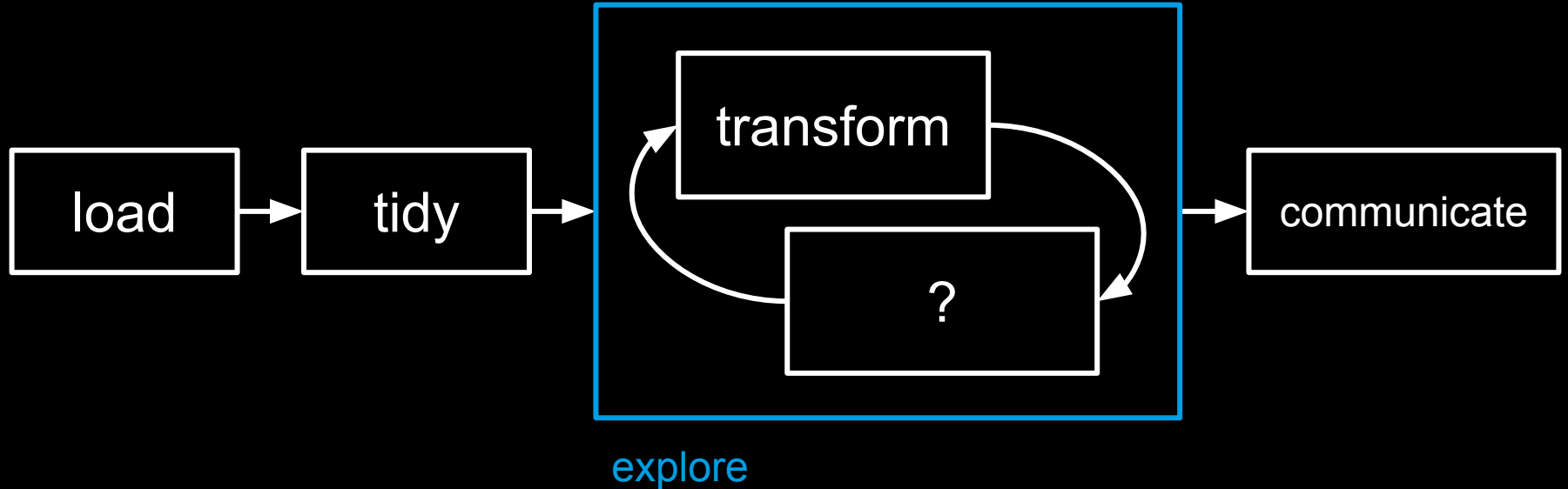


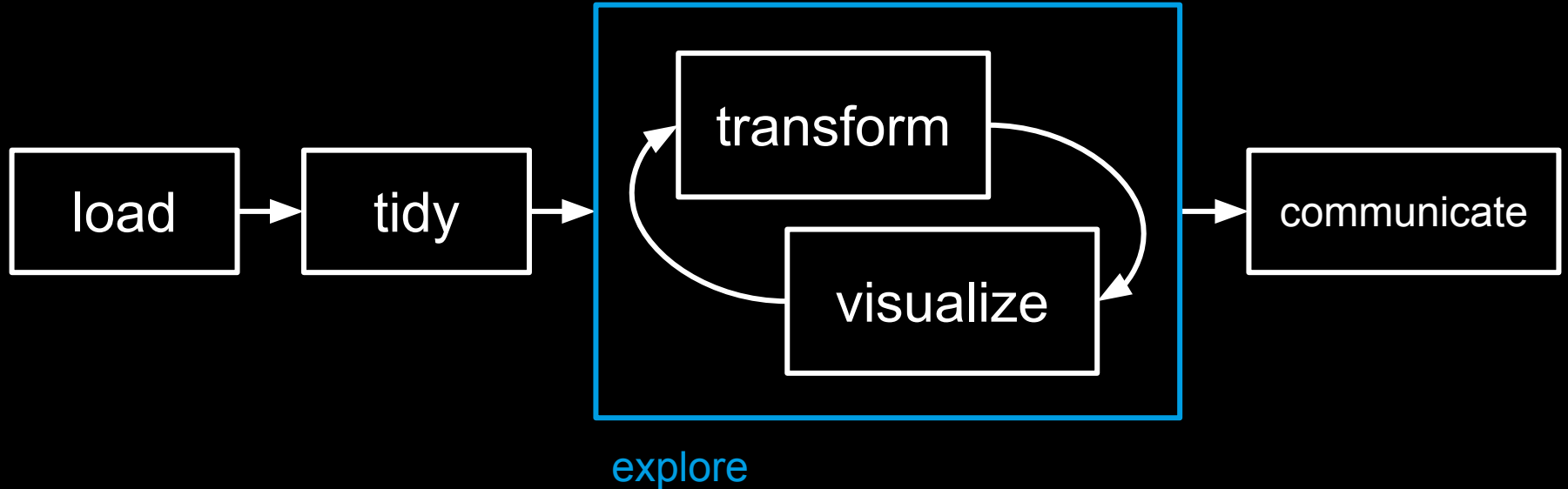


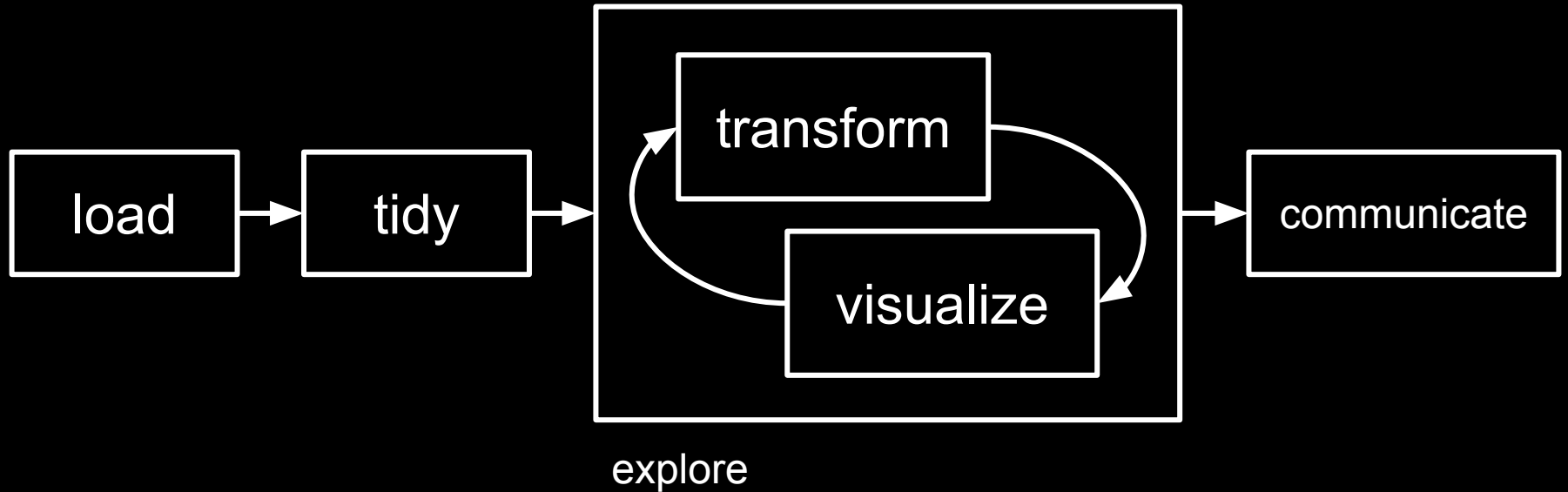


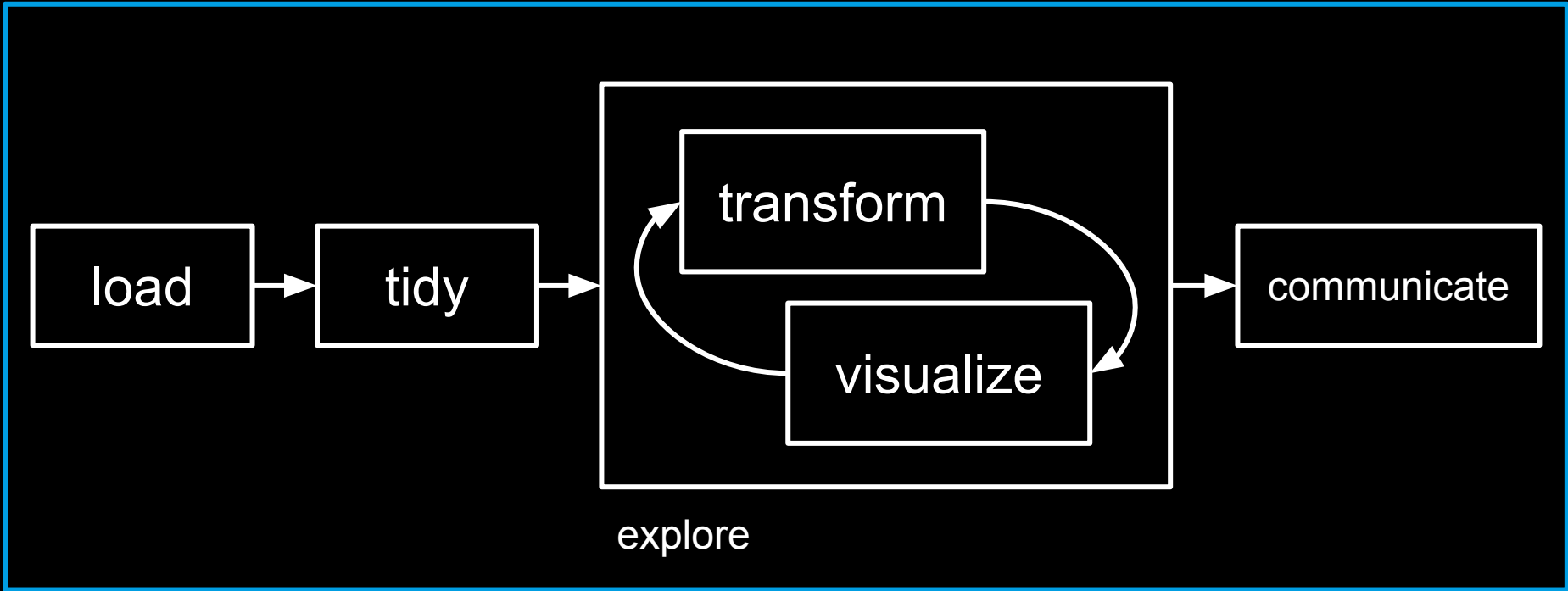




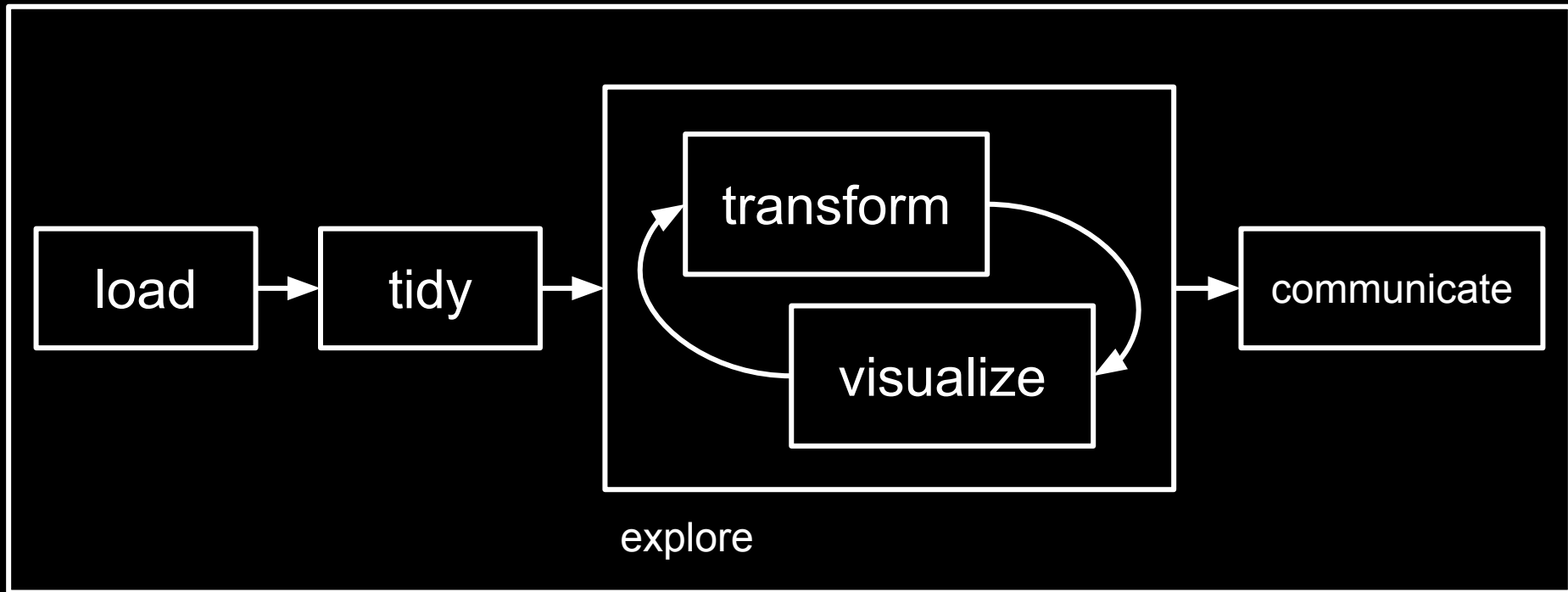








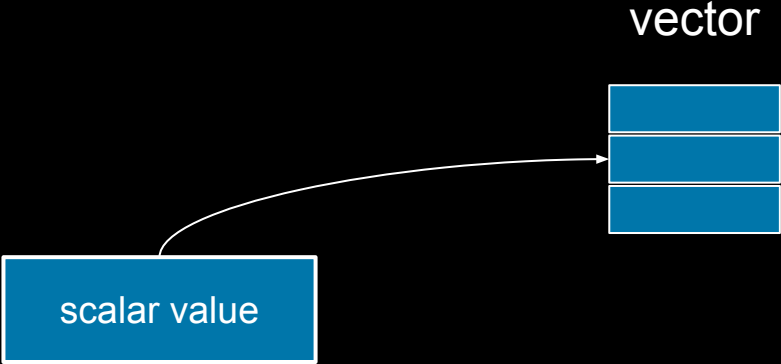
program

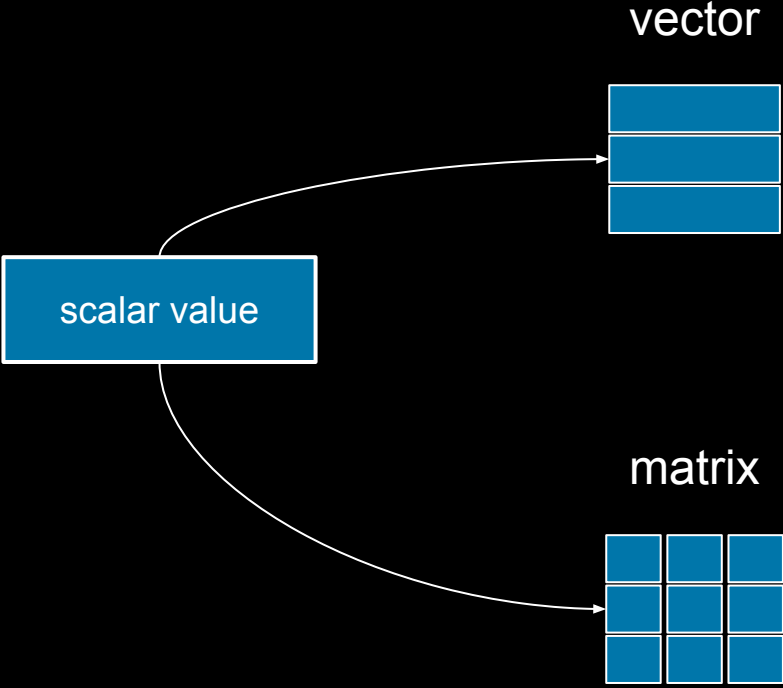


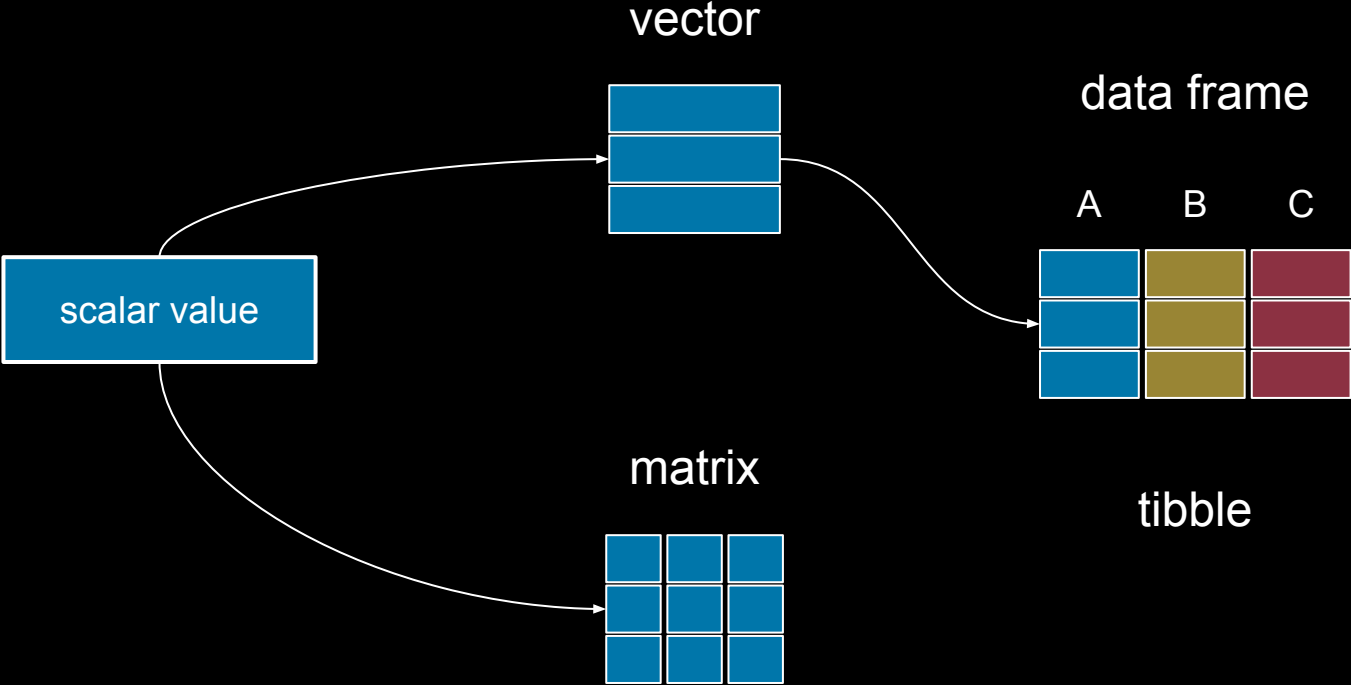
program

DATA REPRESENTATION

scalar value







VECTORS

apple

pear

orange

list of values with
the same storage mode

list of values with
the same storage mode

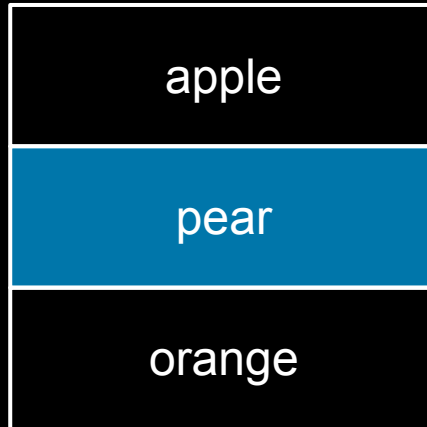
character
double
integer
logical

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

v[1]

apple
pear
orange

v[2]



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

91
75.5
61
88.5
120

```
weight_after_diet <-  
  c(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

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)
```

{{ forcats }}

`as_factor()`

fct_reorder
fct_relevel
fct_infreq
fct_rev
fct_lump

DATA FRAMES

"apple"

"pear"

"orange"

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

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

fruit

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

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"

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

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"

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

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

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	205995360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898

variables

country	year	cases	population
Afghanistan	1999	745	1997071
Afghanistan	2000	2000	2095360
Brazil	1999	37737	17296362
Brazil	2000	60480	17494898

observations

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	7737	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



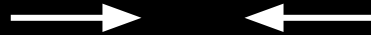
country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898

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


```
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”

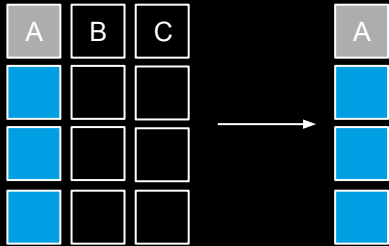
```
str_detect(txt, "Miller")
```

TRANSFORM DATA

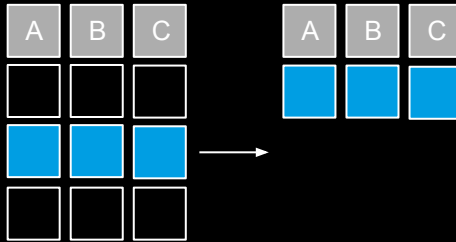
{{ dplyr }}

types of transformations

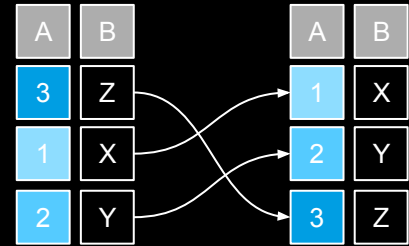
`select()`



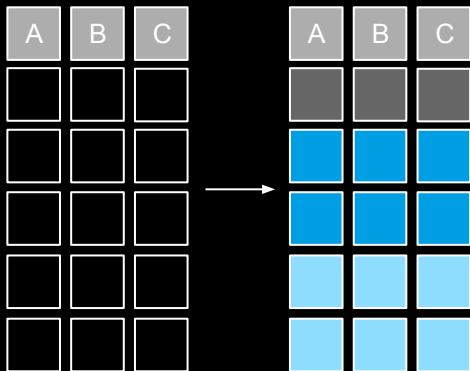
`filter()`



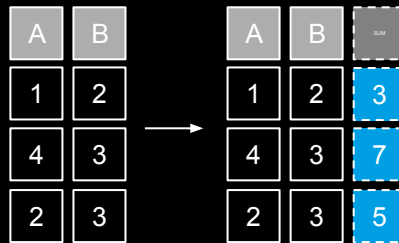
`arrange()`



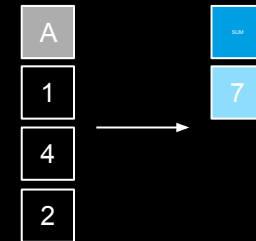
`group_by()`



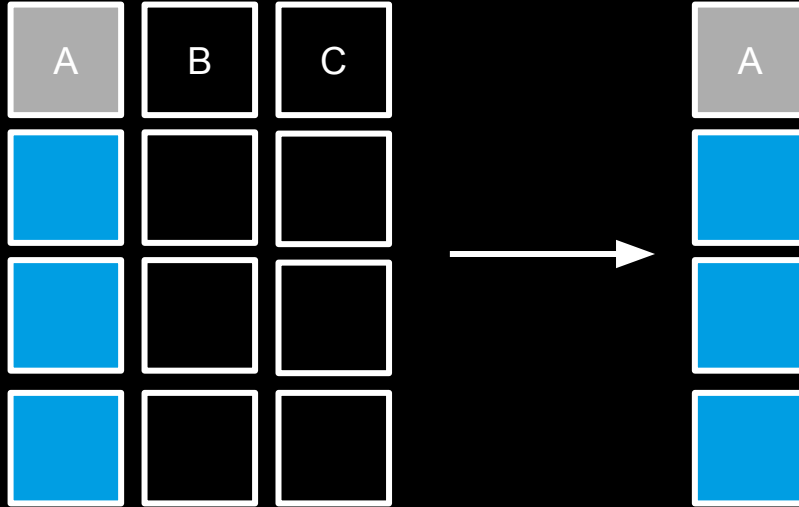
`mutate()`



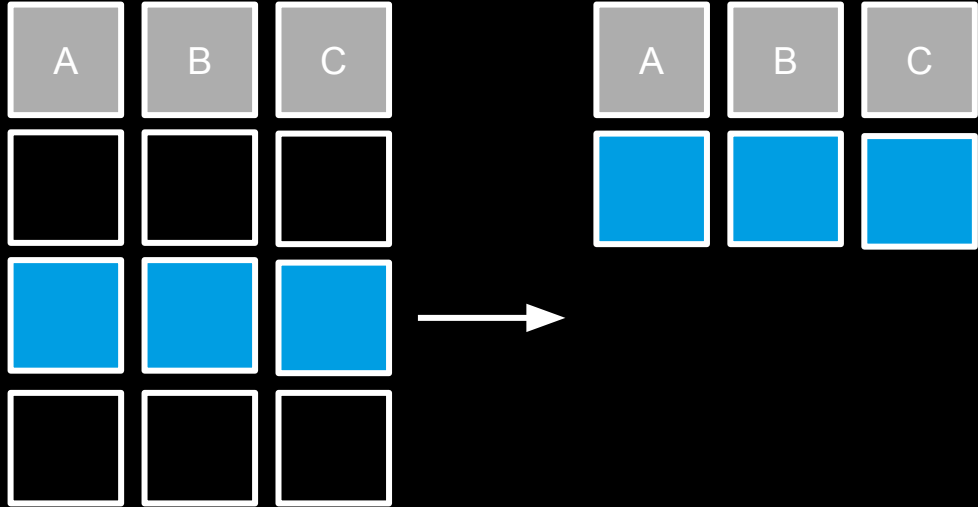
`summarize()`



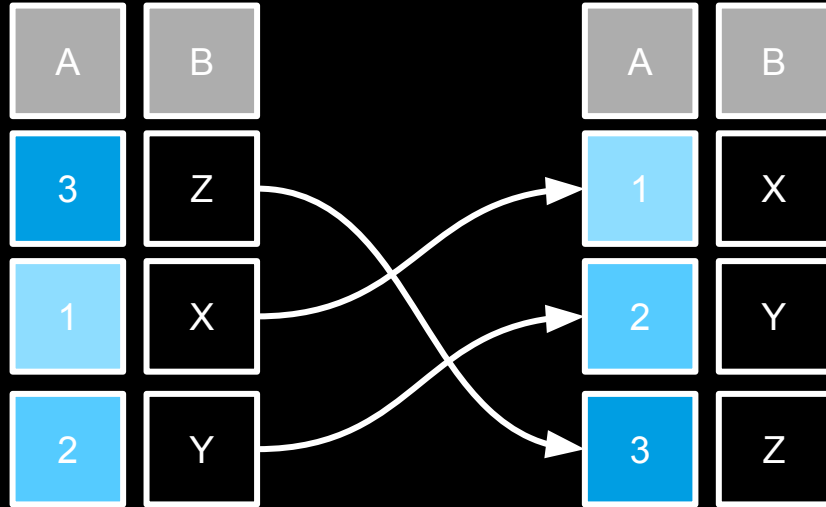
`select()`



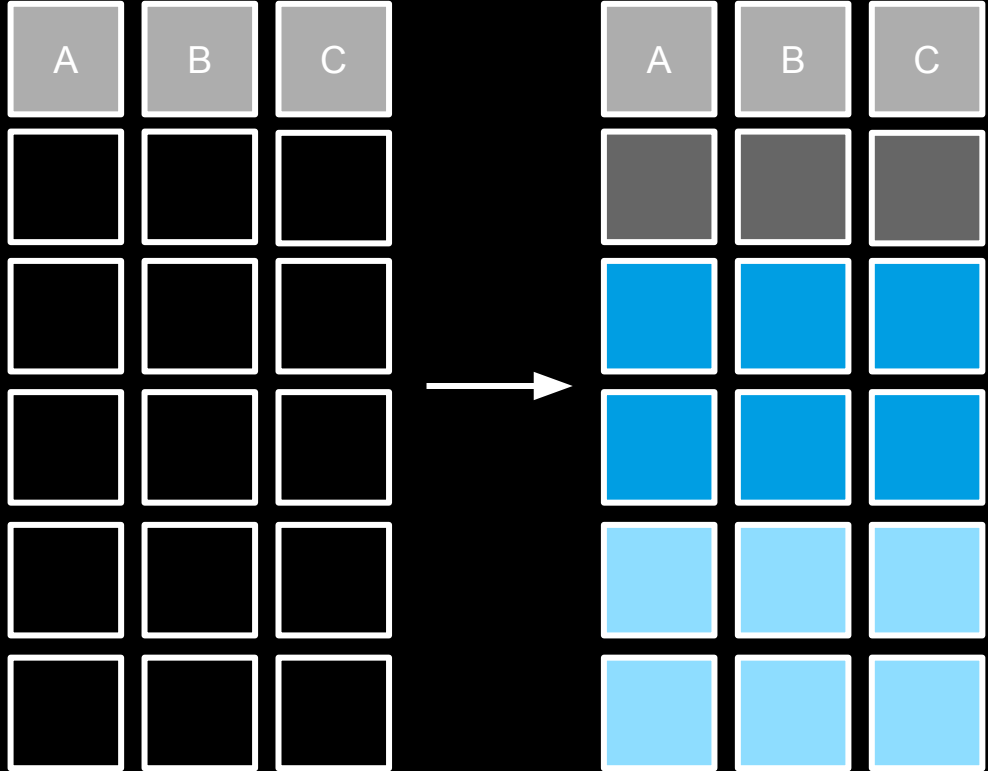
`filter()`



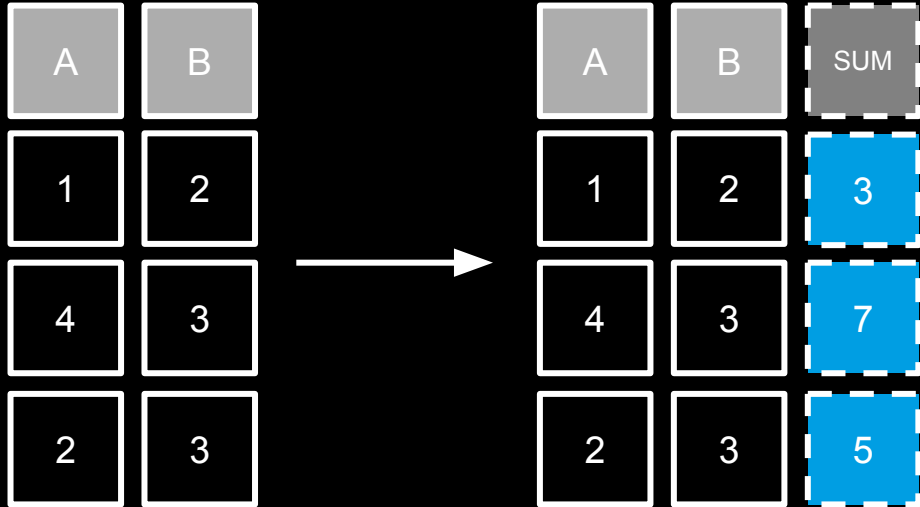
arrange()



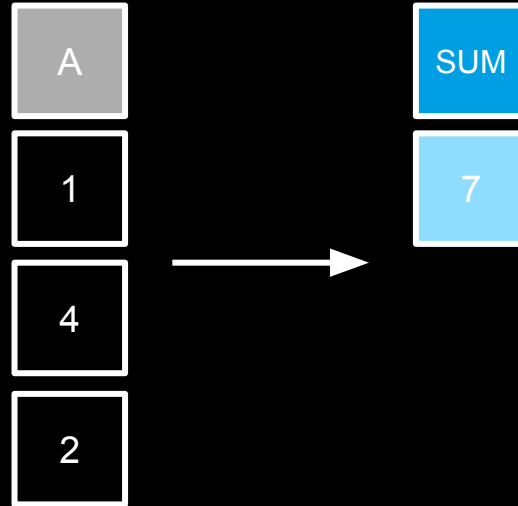
`group_by()`



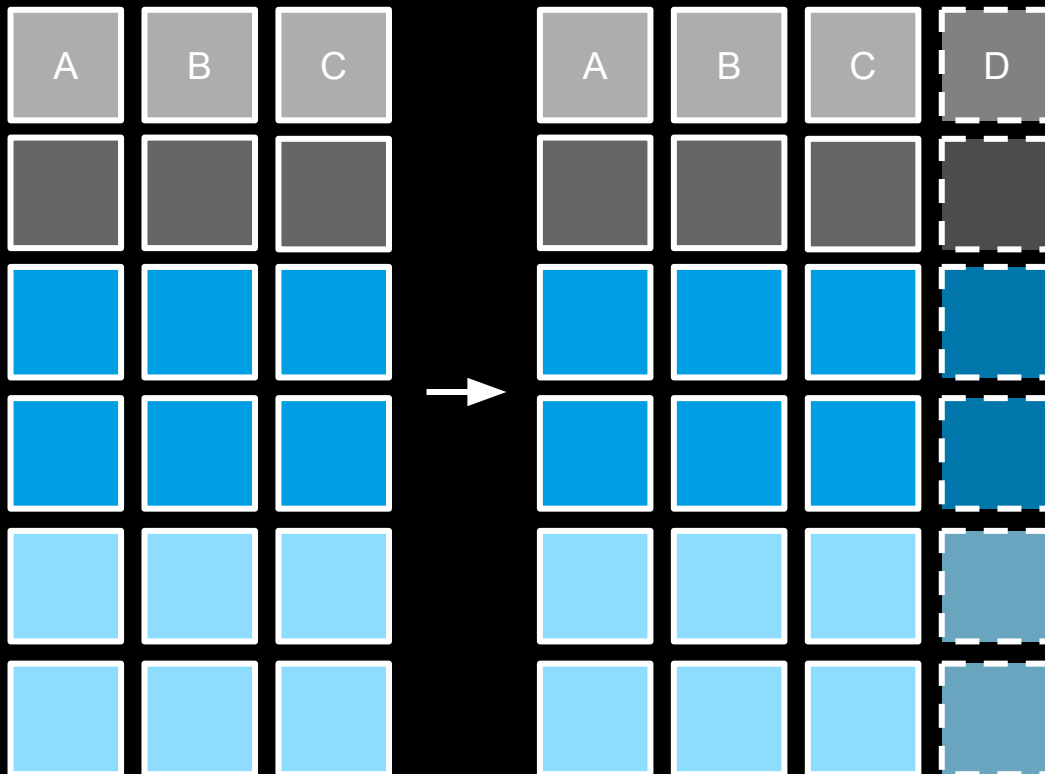
mutate()



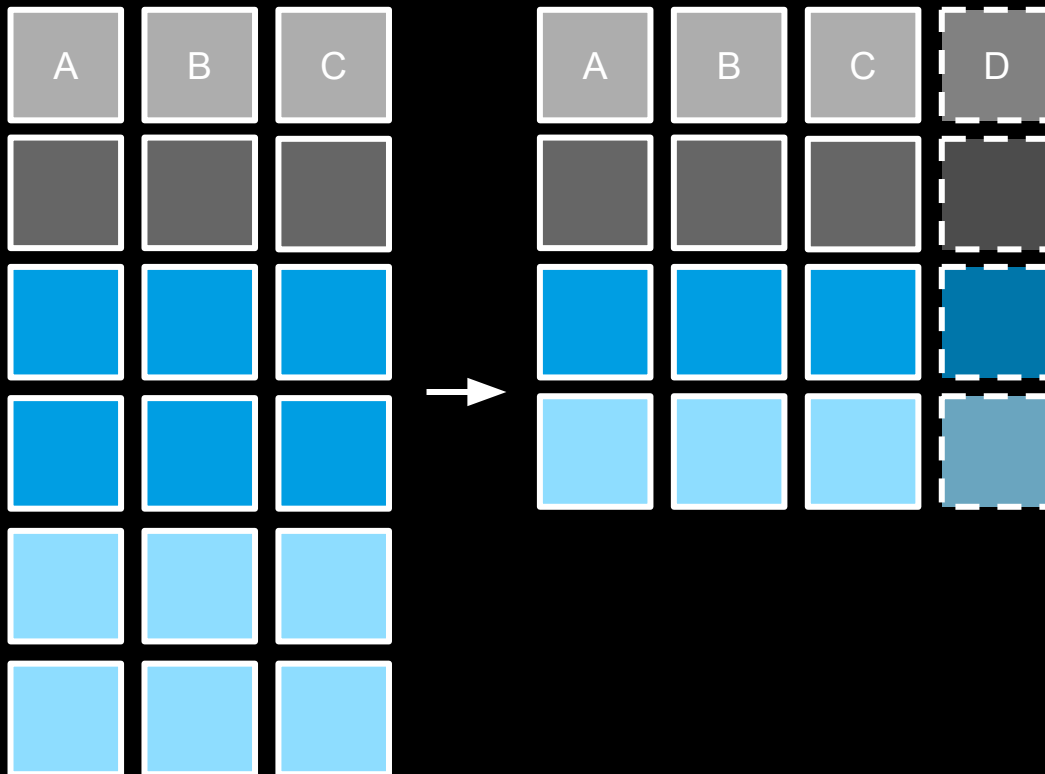
`summarize()`



`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

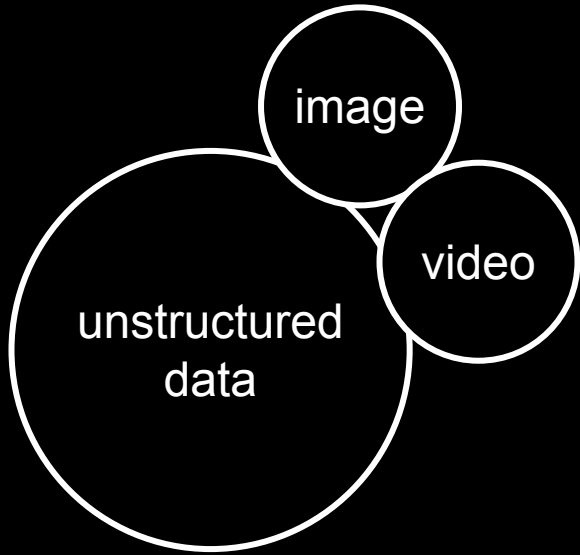


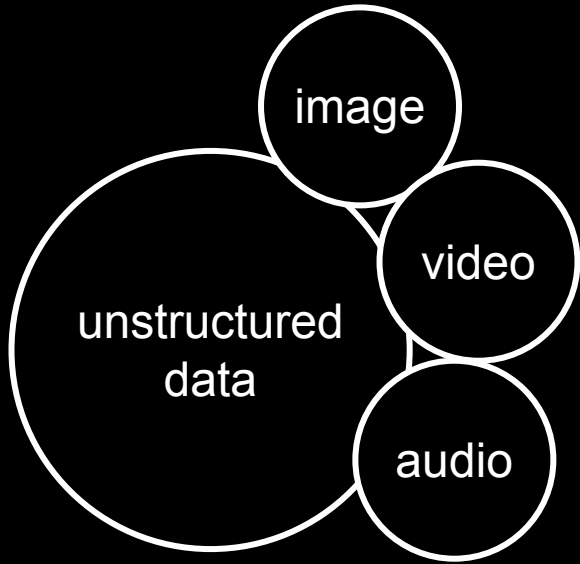
unstructured
data

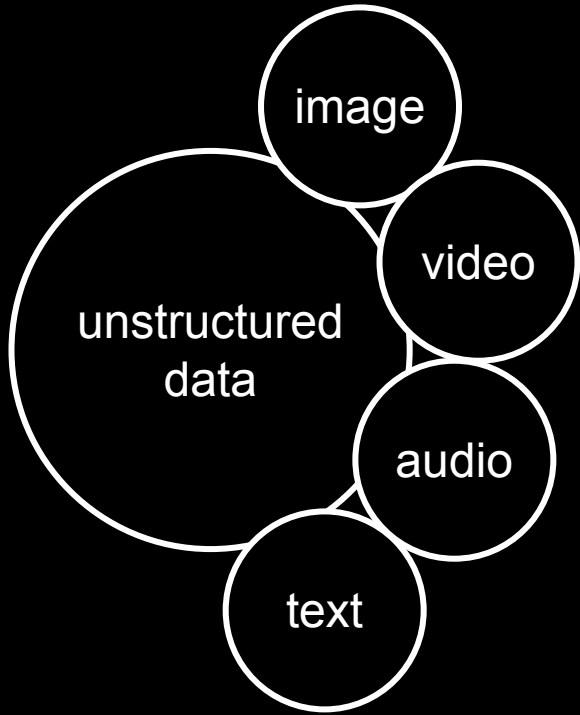
A diagram consisting of two white circles on a black background. The larger circle on the left contains the text 'unstructured data'. A smaller circle is positioned to the upper right of the larger circle, overlapping its top edge, and contains the text 'image'.

image

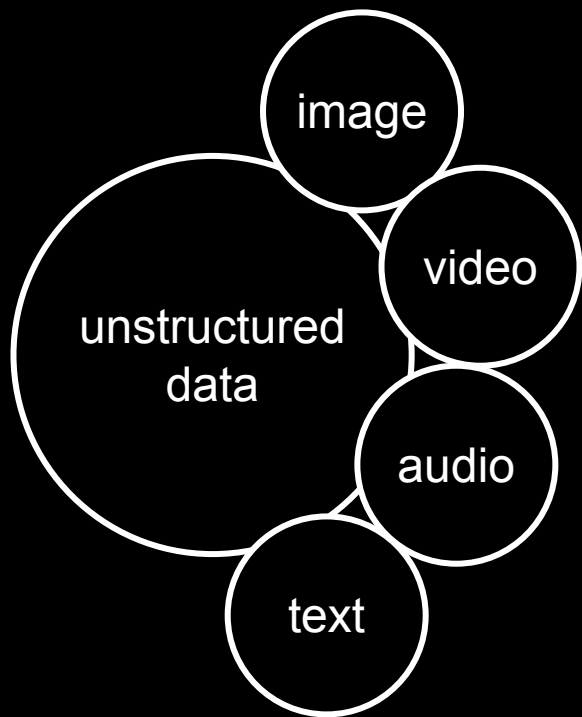
unstructured
data



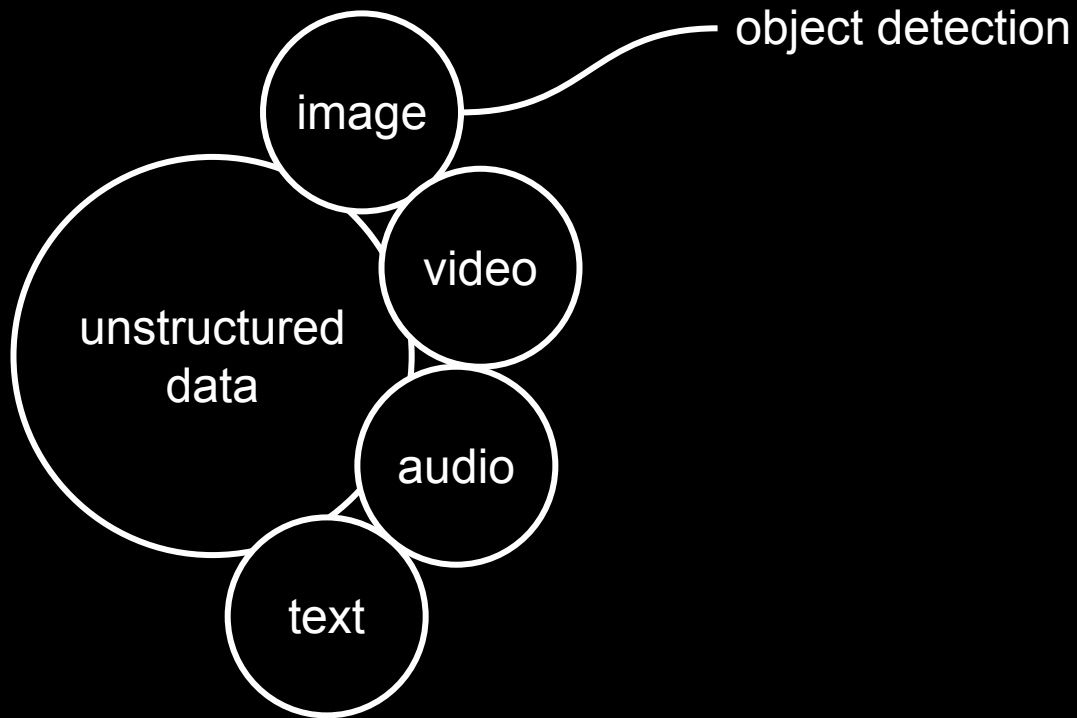




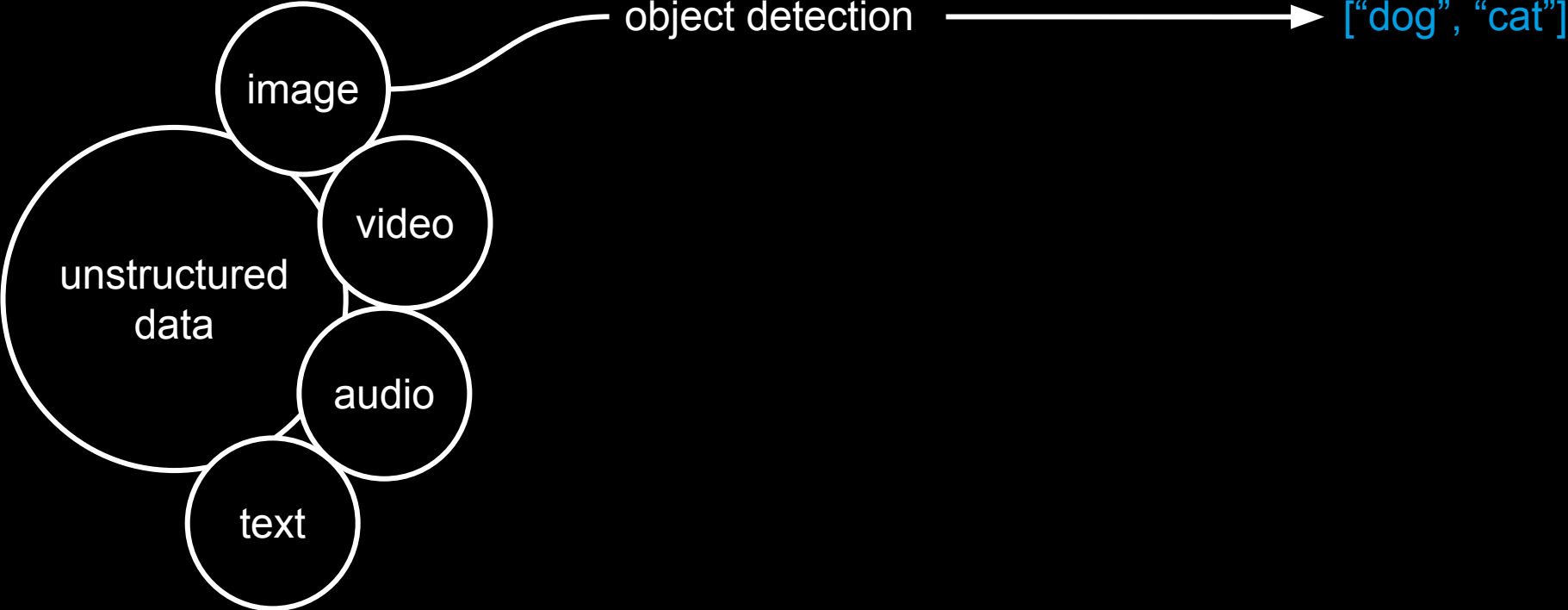
no handles to grab



no handles to grab



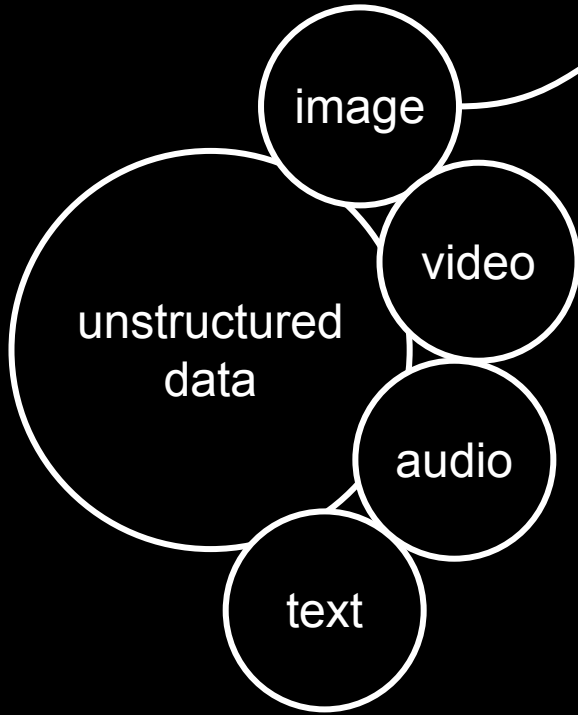
no handles to grab



no handles to grab

algorithm

extracted, structured information



object detection

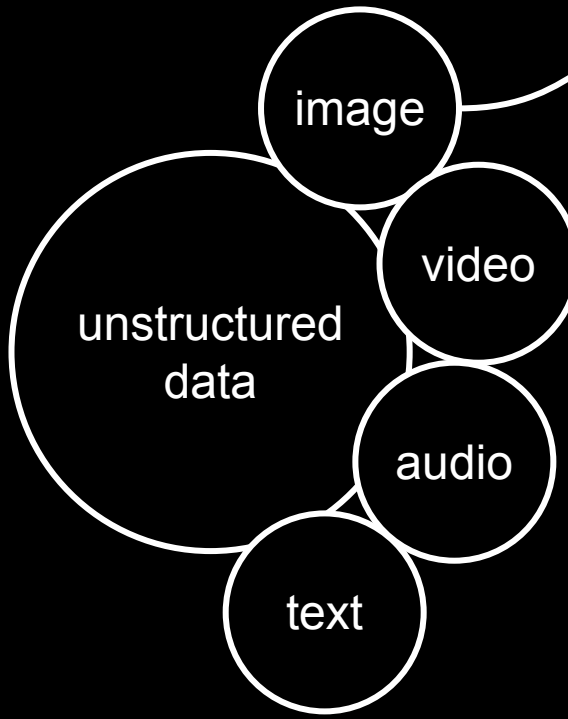


["dog", "cat"]

no handles to grab

algorithm

extracted, structured information



object detection



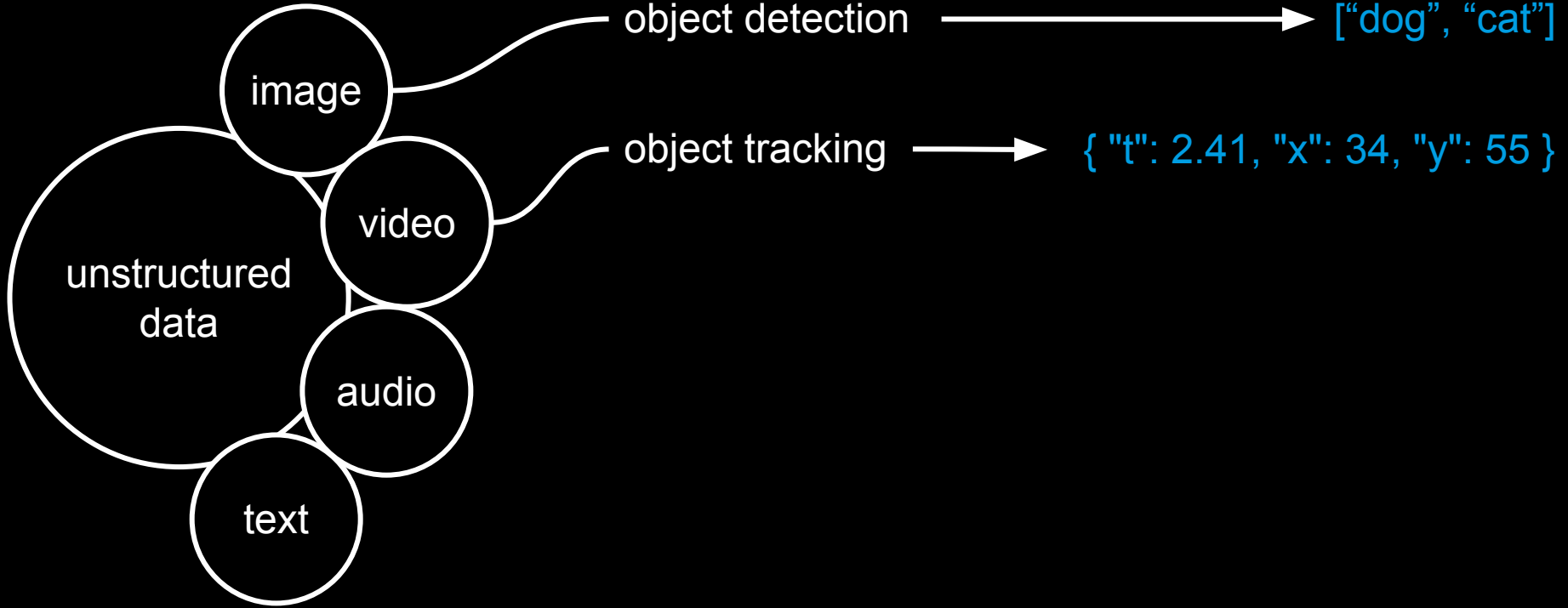
["dog", "cat"]

object tracking

no handles to grab

algorithm

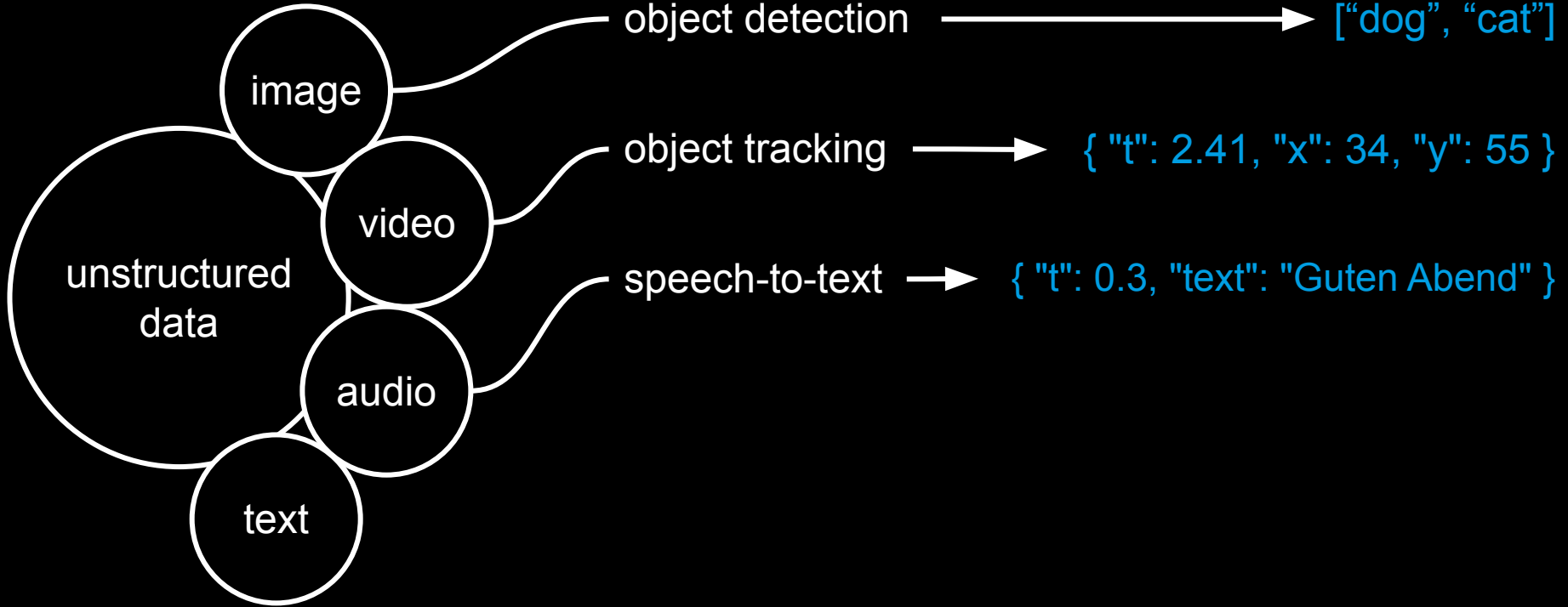
extracted, structured information



no handles to grab

algorithm

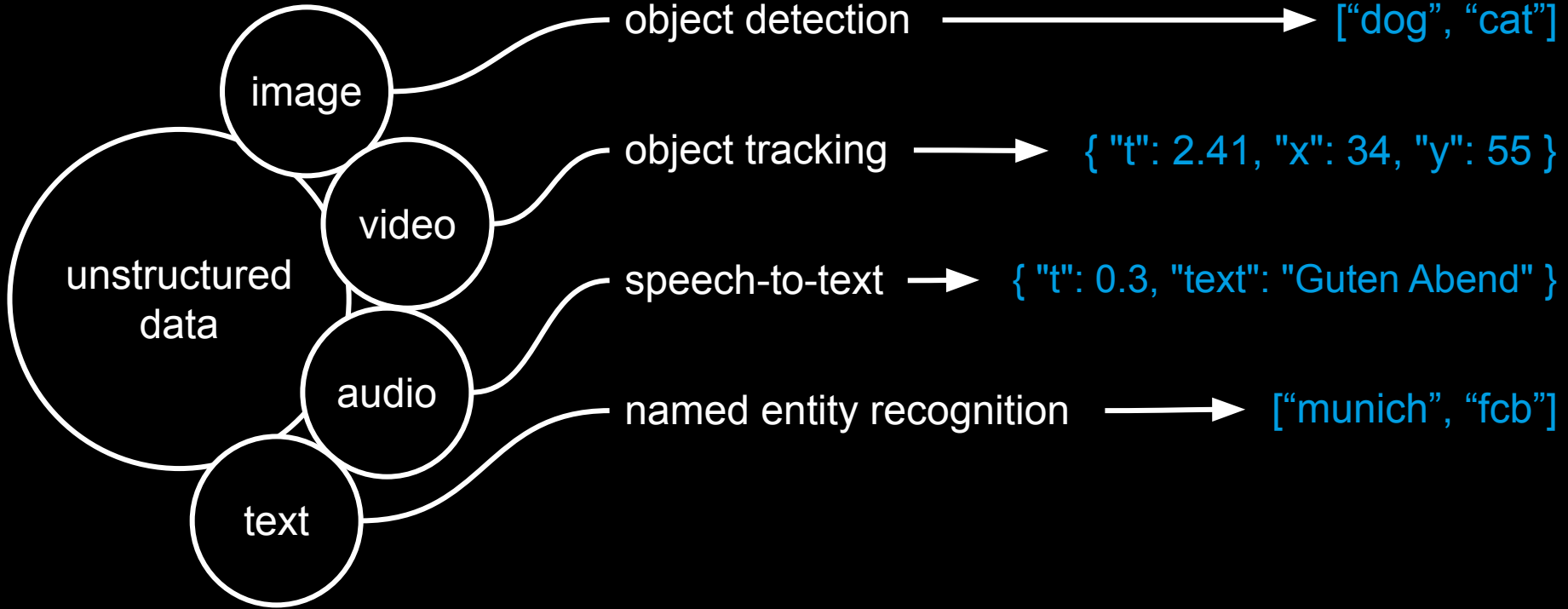
extracted, structured information



no handles to grab

algorithm

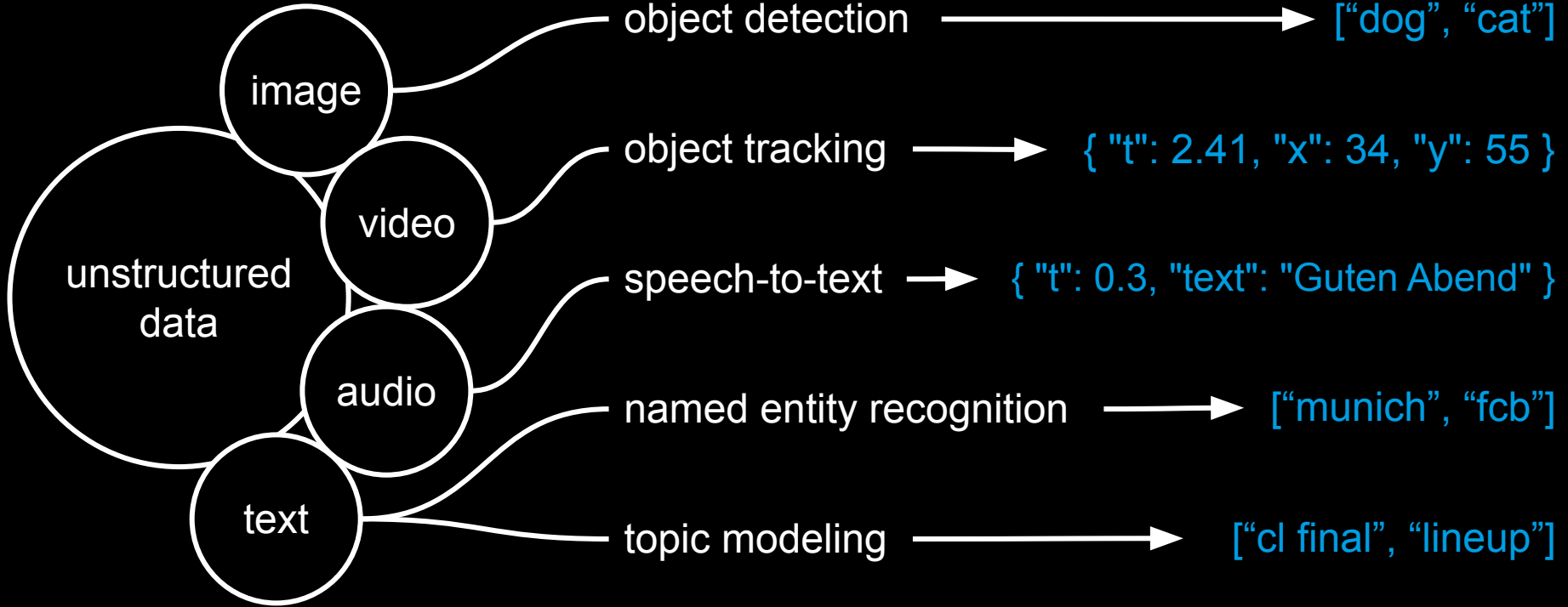
extracted, structured information



no handles to grab

algorithm

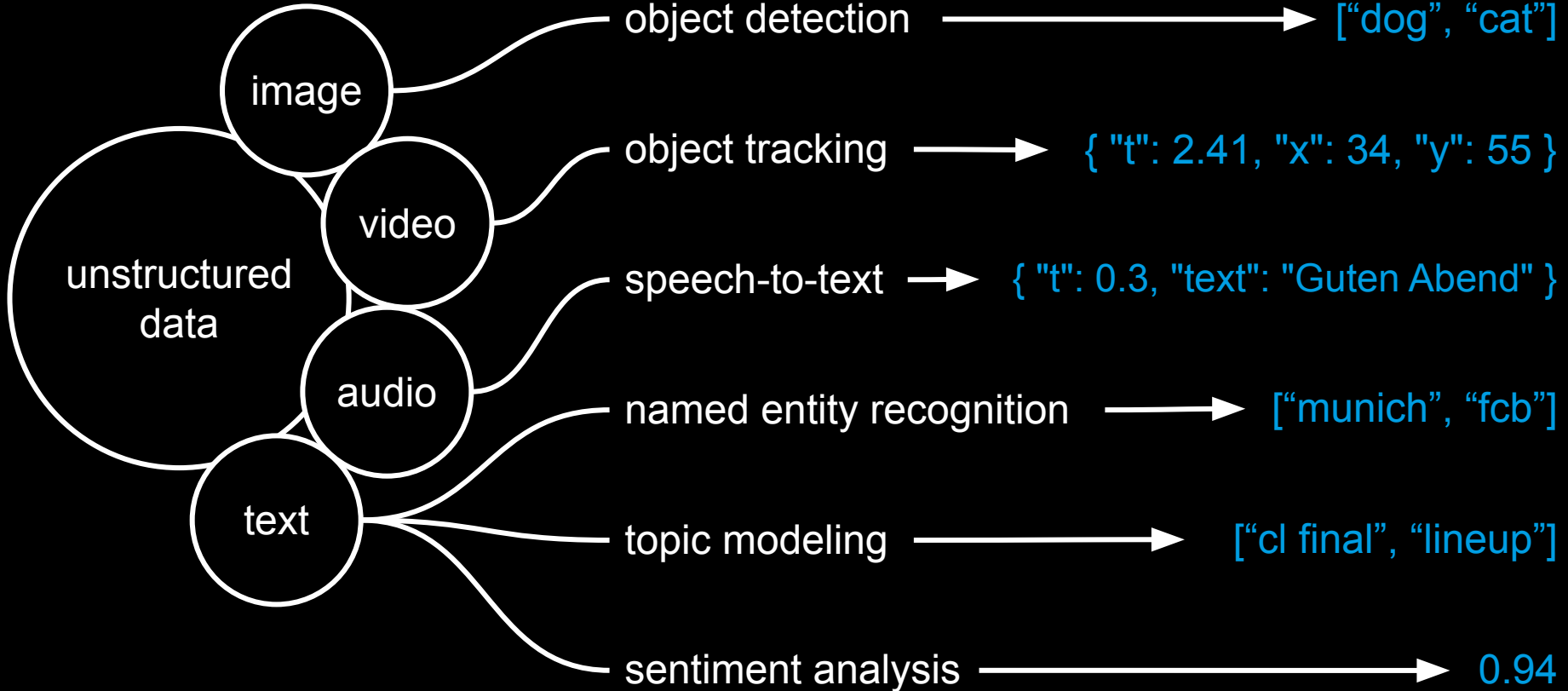
extracted, structured information



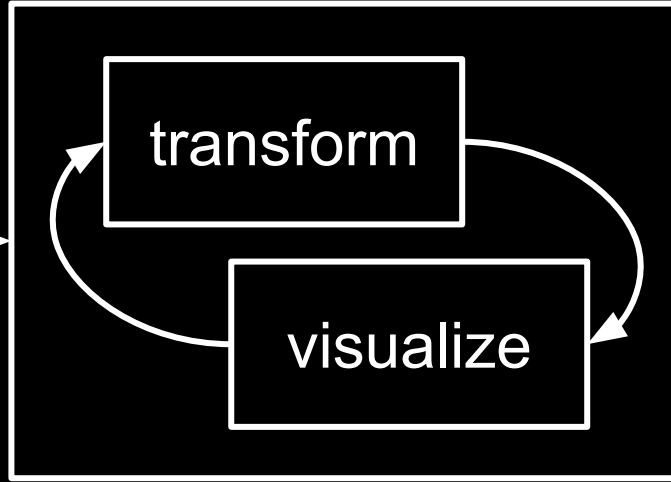
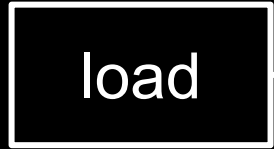
no handles to grab

algorithm

extracted, structured information



this assumes
structured data

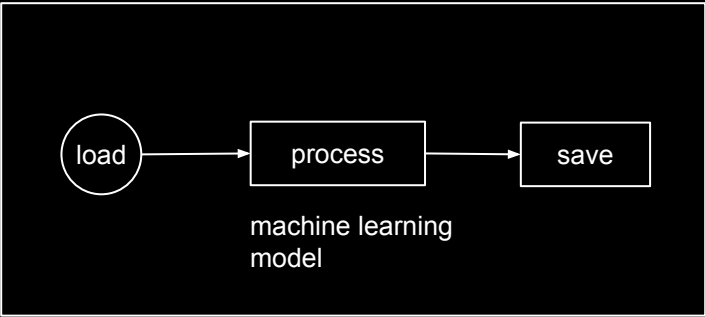


explore



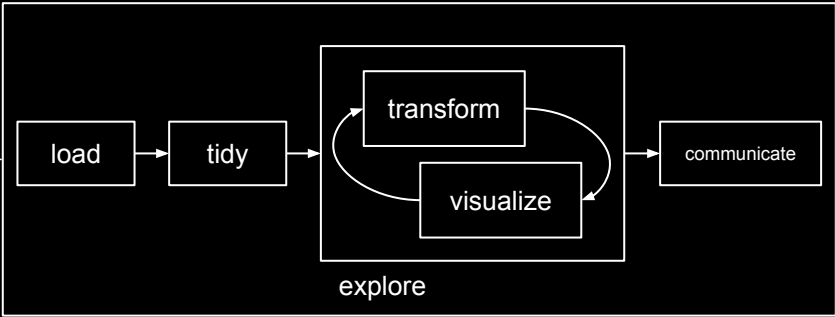
program

pre-process
unstructured data



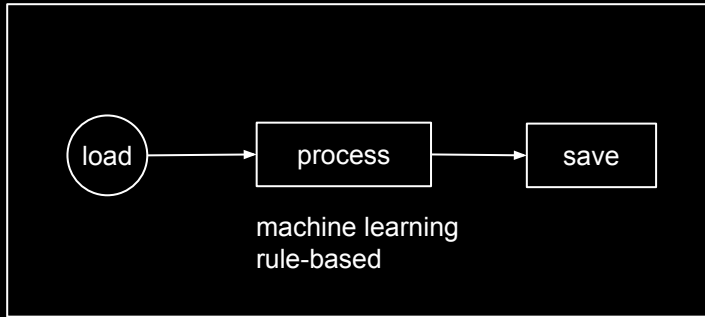
program

exploratory data
analysis



program

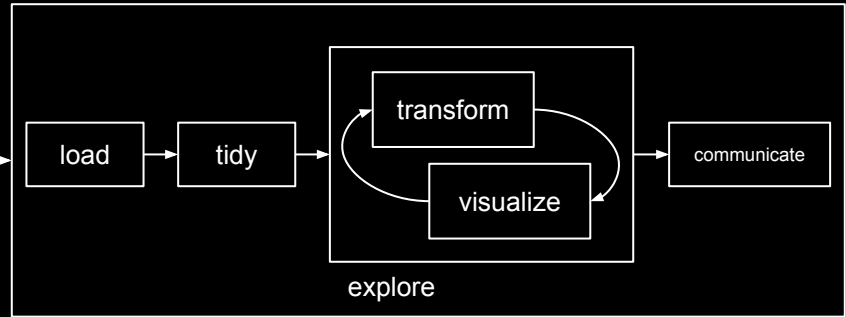
pre-process
unstructured data



program



exploratory data
analysis

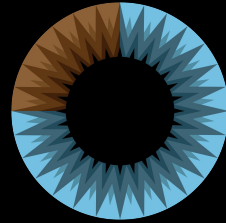


program



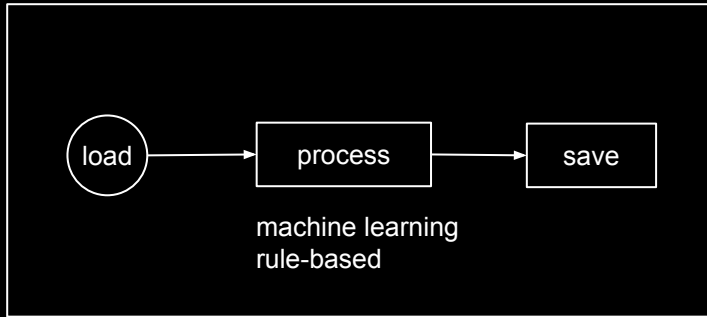
MACHINE LEARNING

Highly recommended for
background information



[3Blue1Brown's YouTube Course on Neural
Networks and Deep Learning](#)

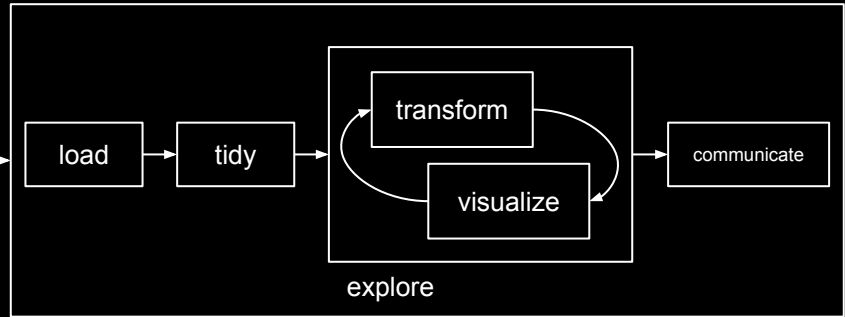
pre-process unstructured data



program



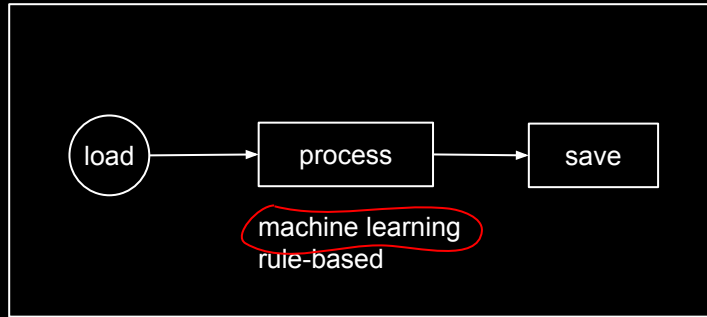
exploratory data analysis



program



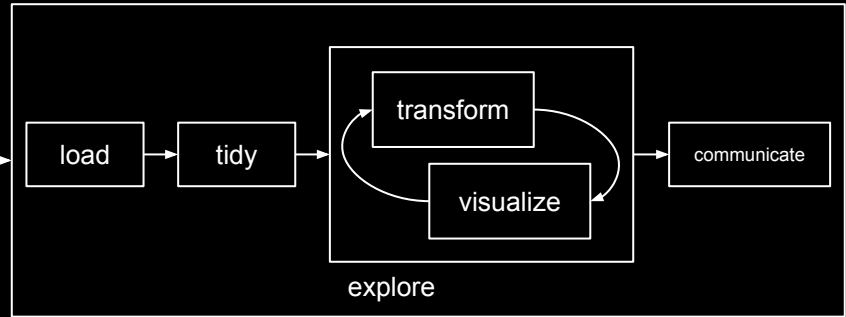
pre-process
unstructured data



program



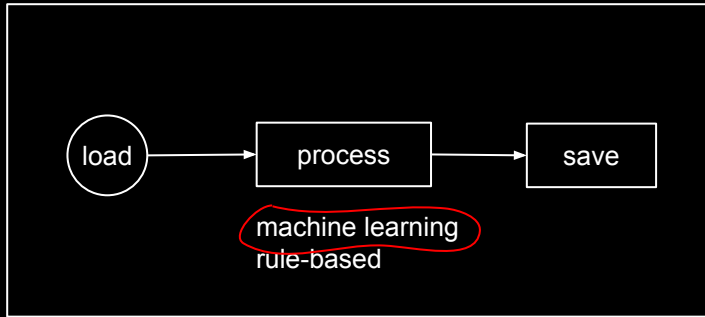
exploratory data
analysis



program



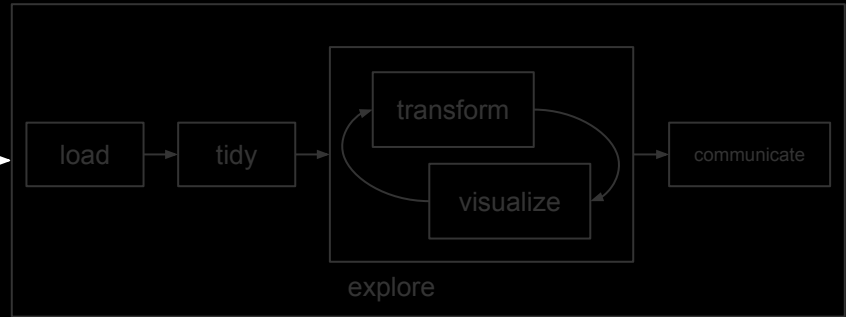
pre-process
unstructured data



program

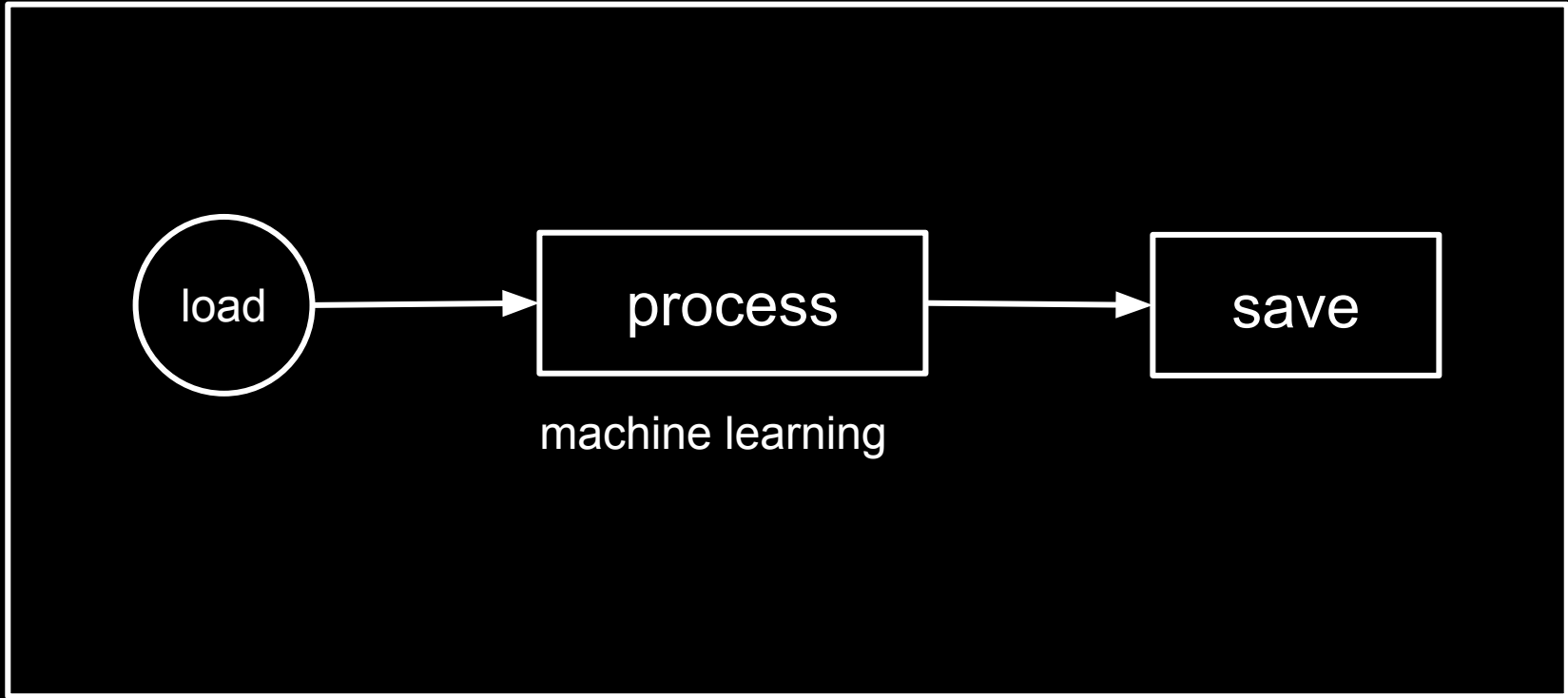


exploratory data
analysis



program





program



YouTube



machine learning



save

program



YouTube



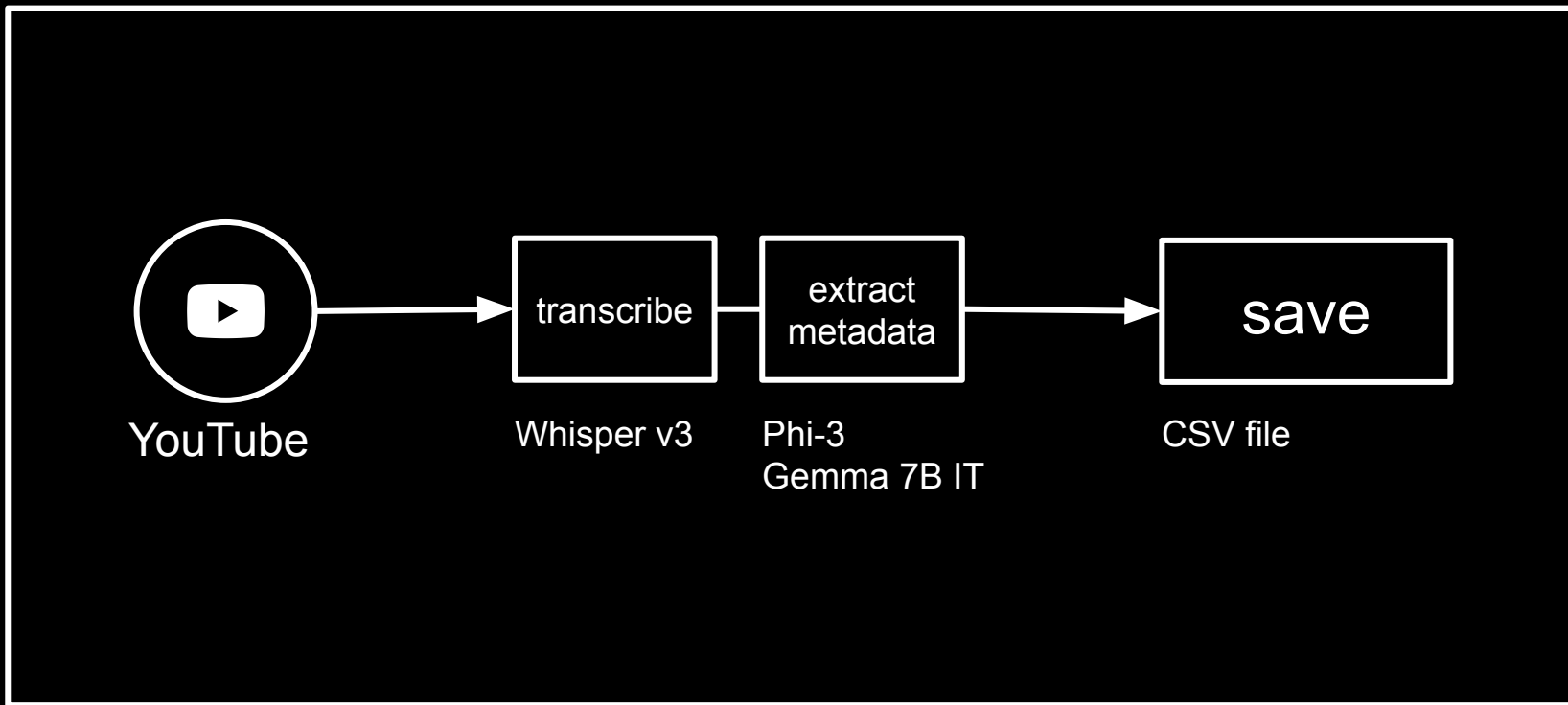
Whisper v3



Phi-3
Gemma 7B IT



program



program

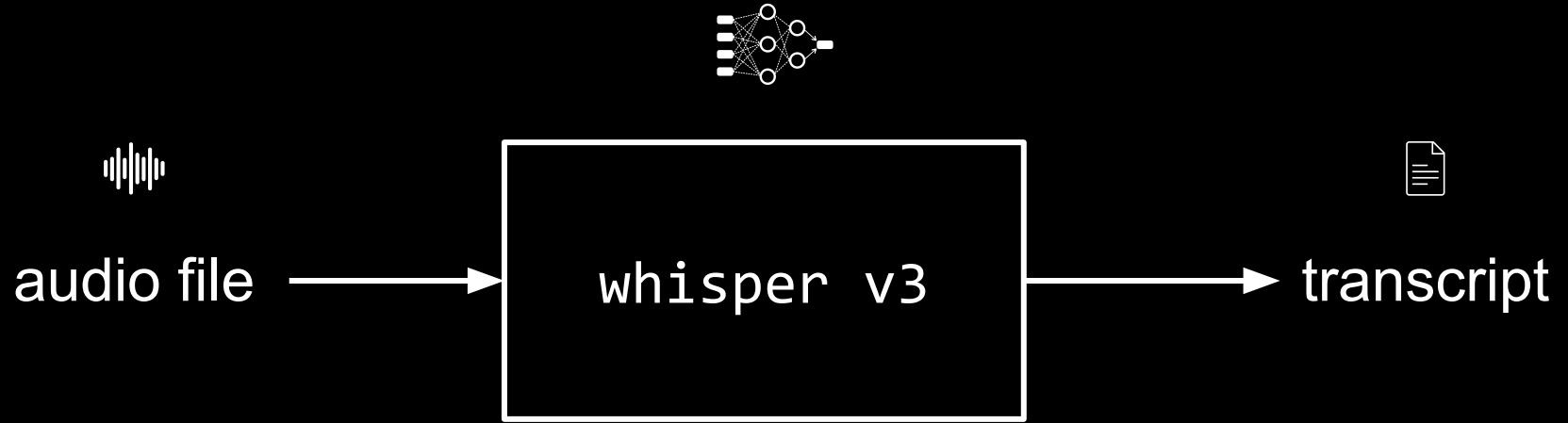
YouTube API

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>

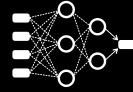
Gemma 2B / 7B Instruct

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

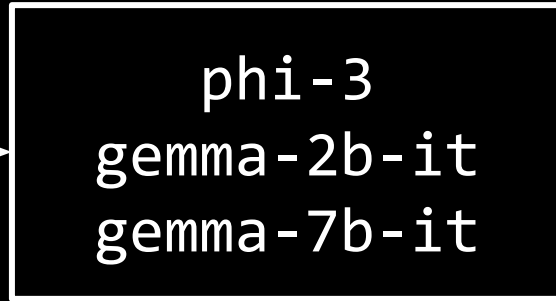


<https://huggingface.co/google/gemma-2b-it>

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



prompt +
context



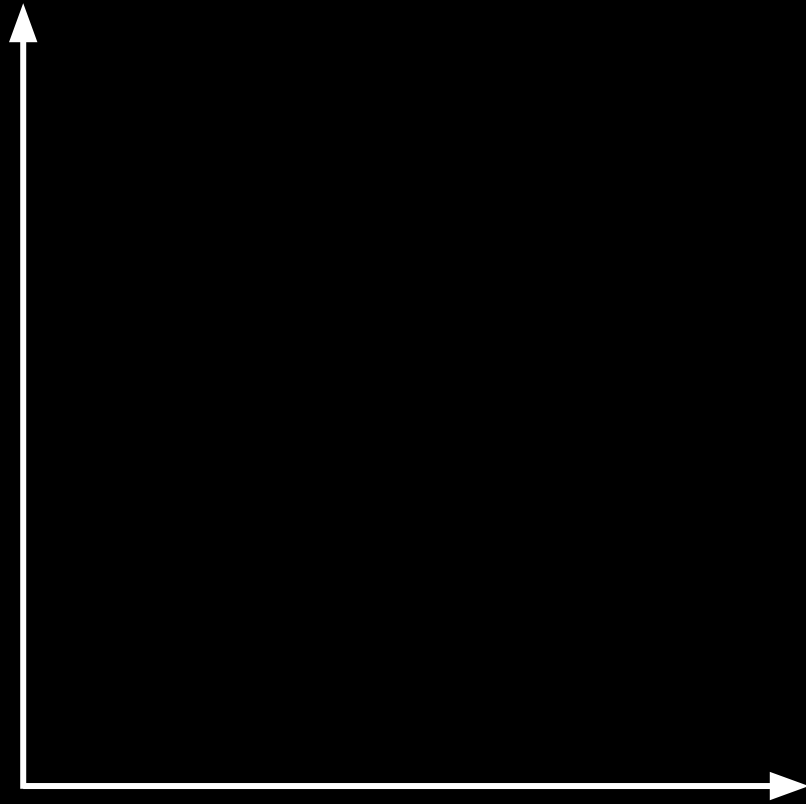
answer

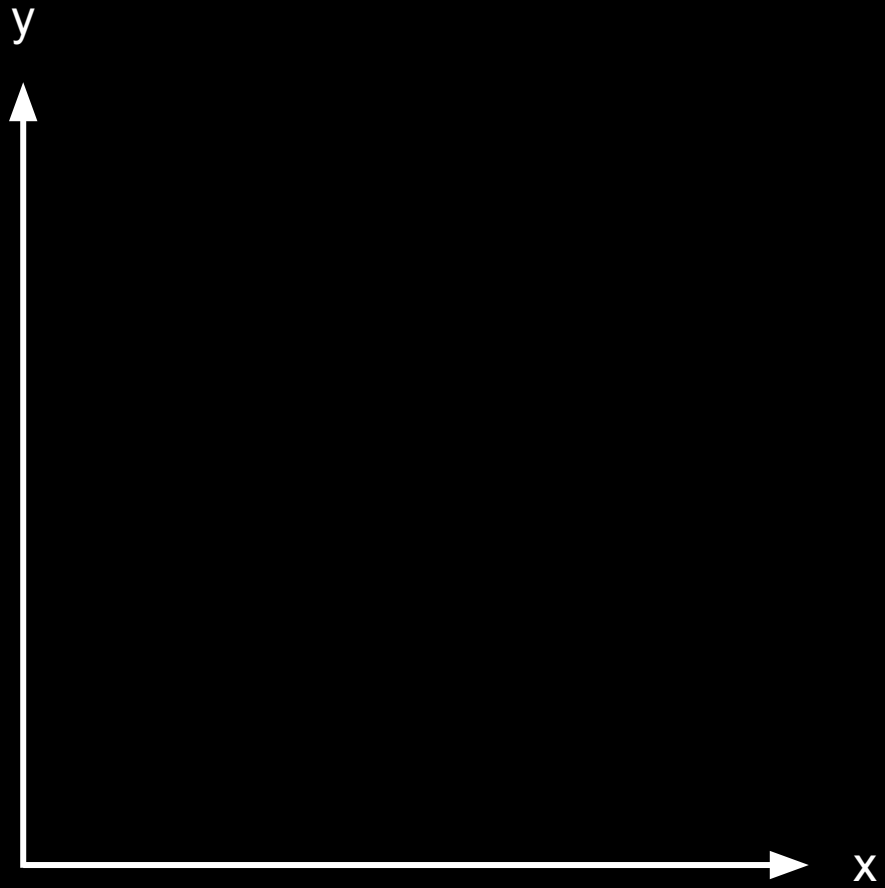
OpenAI GPT-4o

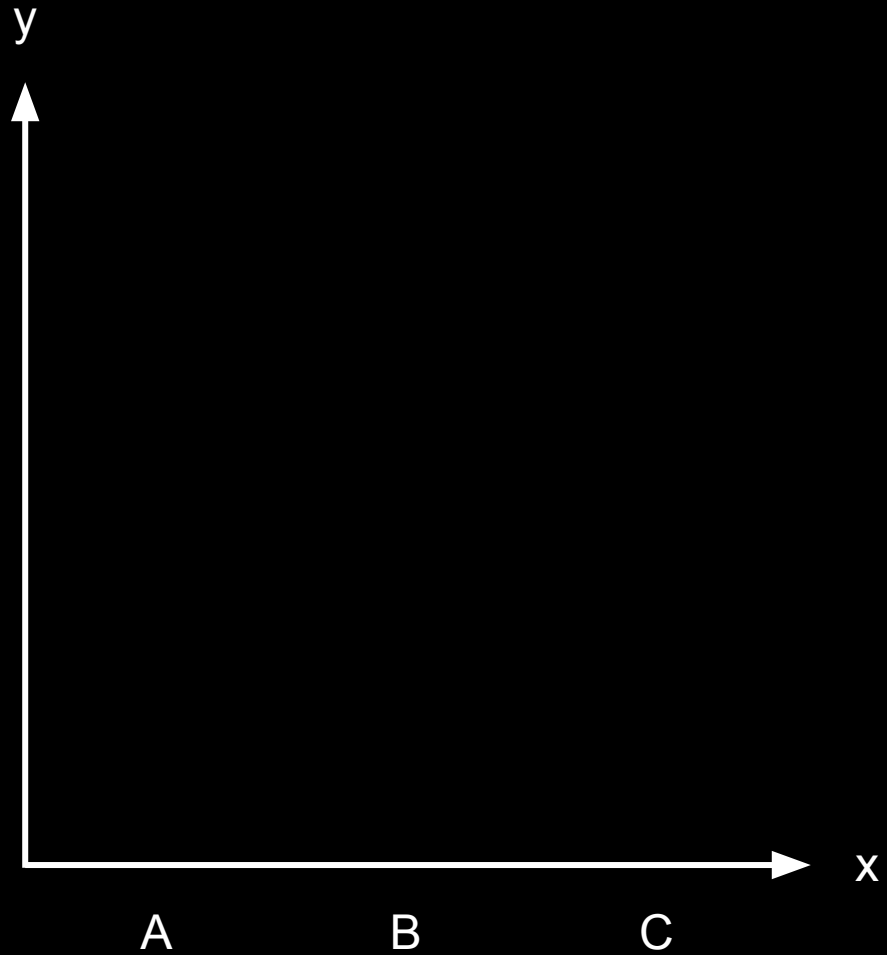
VISUALIZE DATA

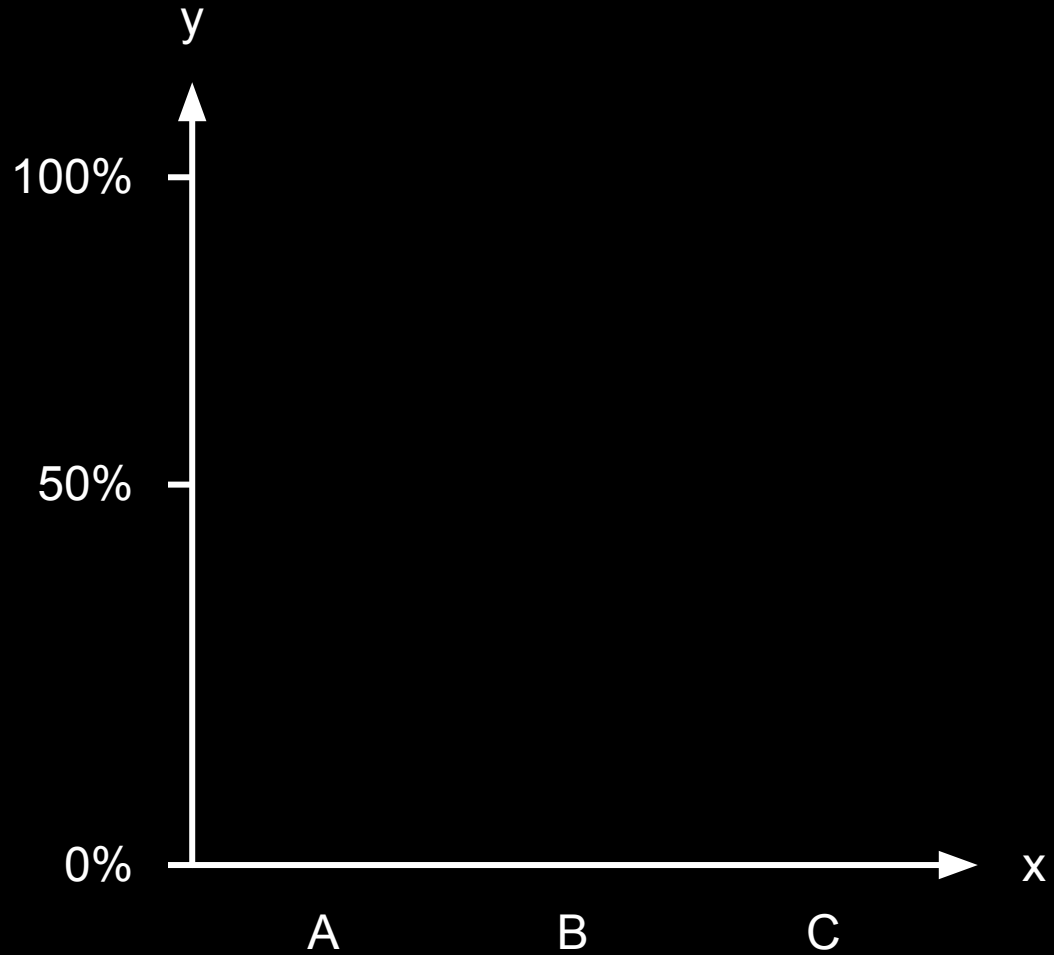
data

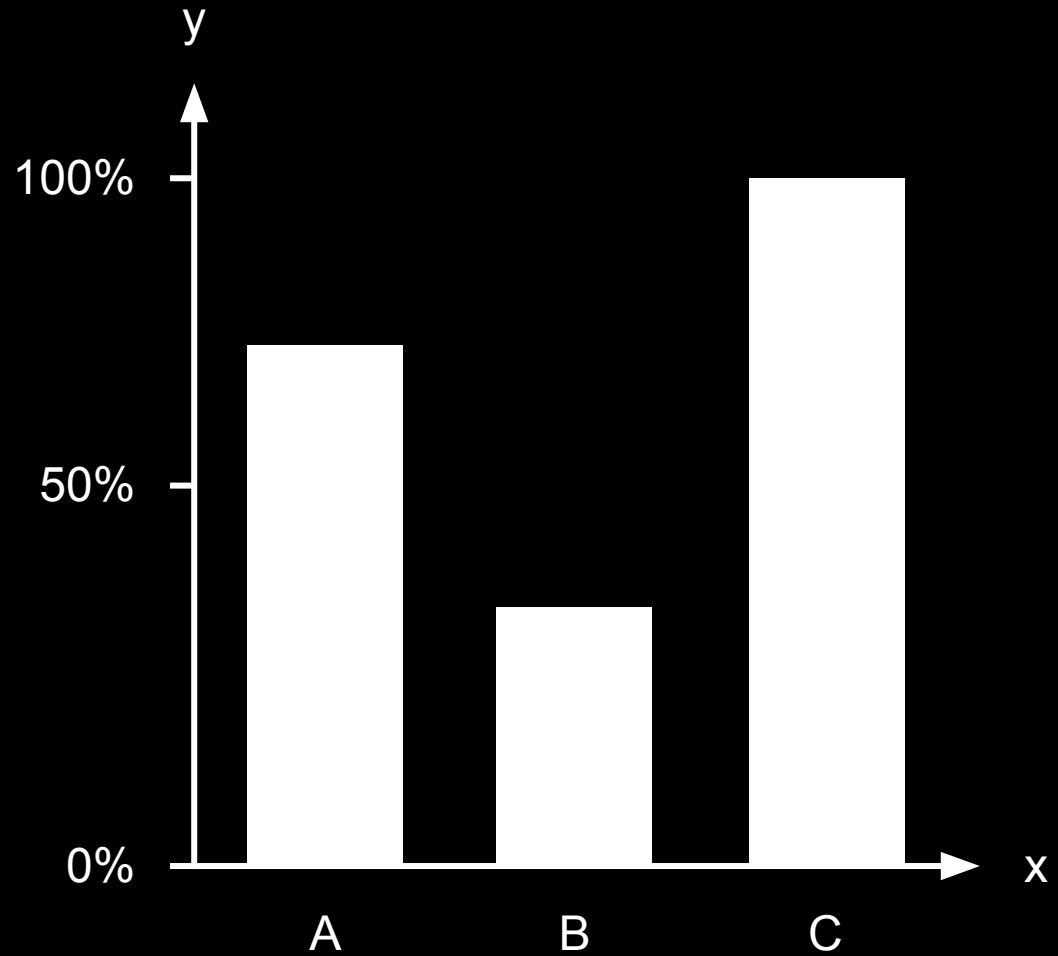
category	pct
A	75
B	33
C	100



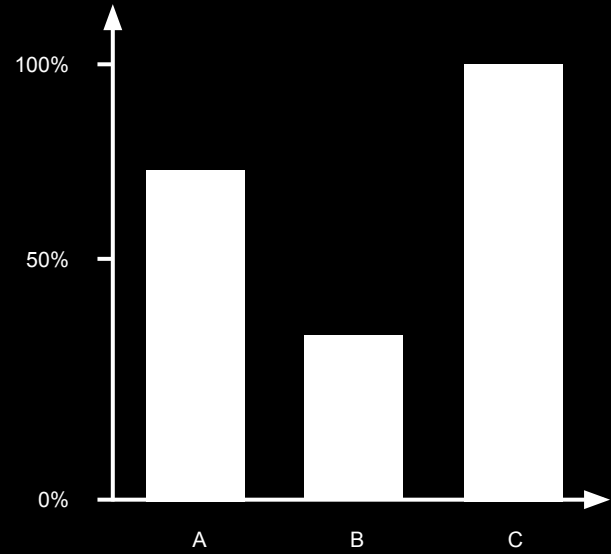








category	pct
A	75
B	33
C	100



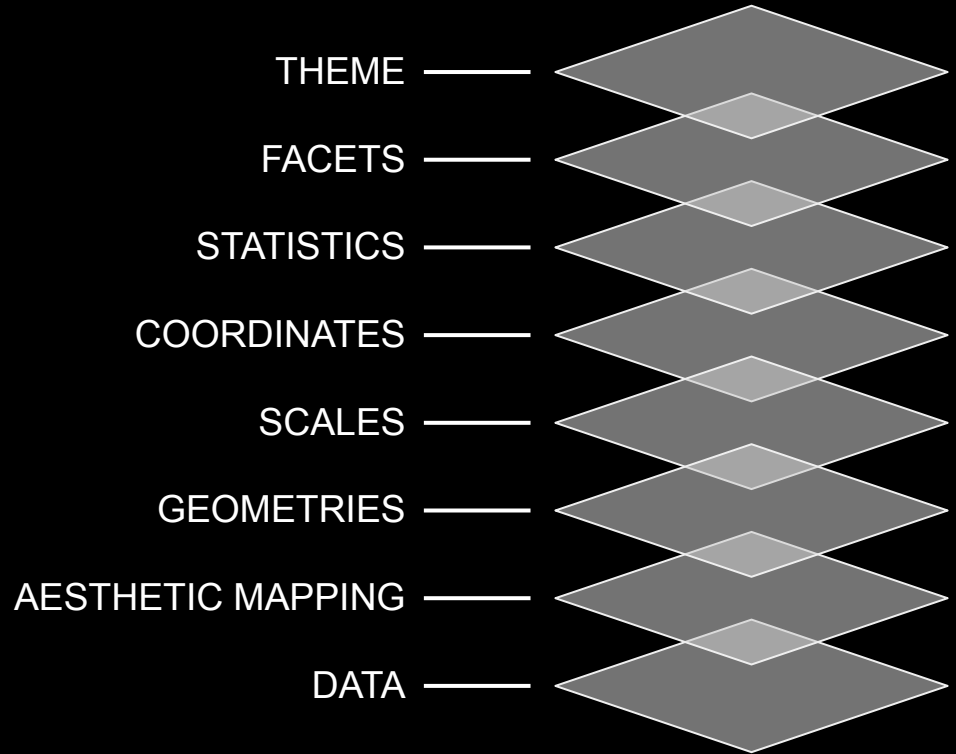
`{{ ggplot2 }}`

why visualize?

{{ ggplot2 }}

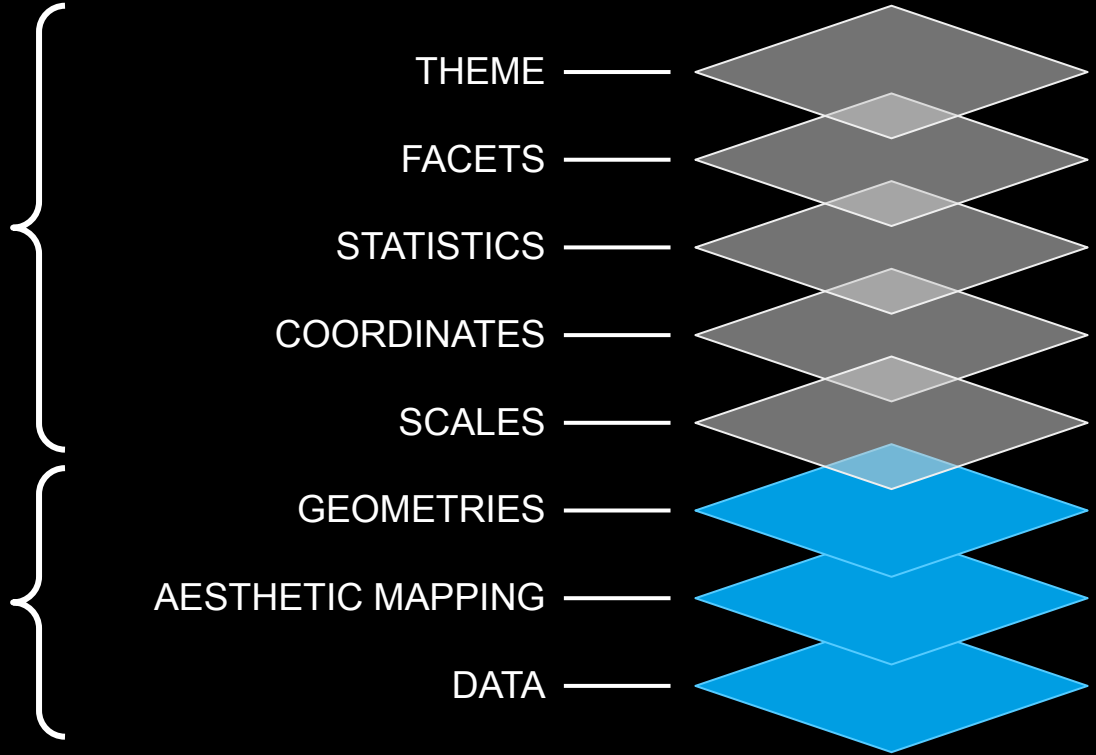
grammar of graphics

any
data
visualization



has useful defaults

mandatory



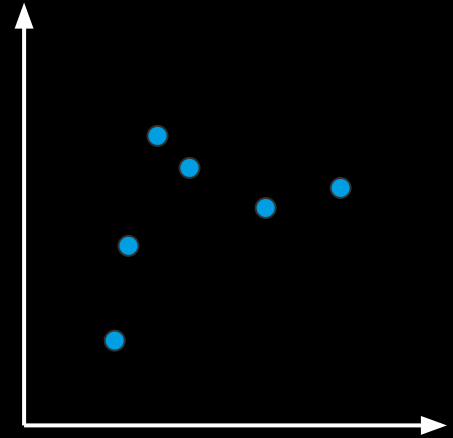
`ggplot()`

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

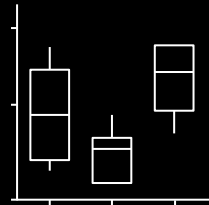
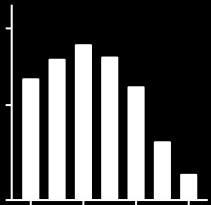
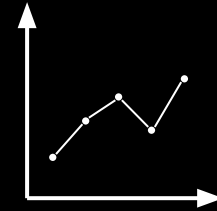
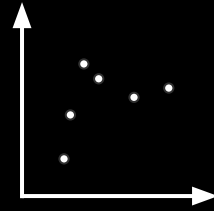
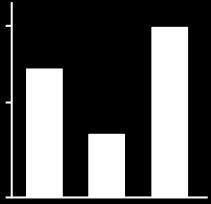


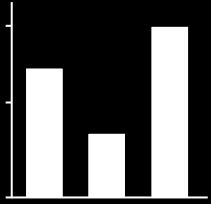
```
ggplot() +  
  aes() +  
  geom_point()
```

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

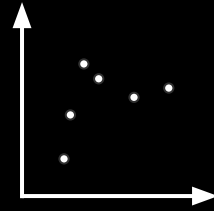


basic plots

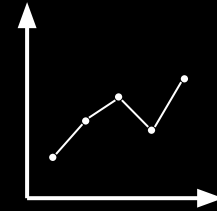




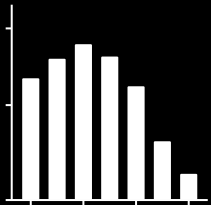
bar chart



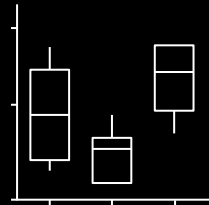
scatter plot



line chart

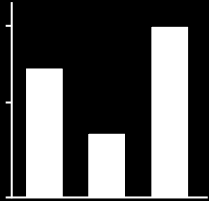


histogram



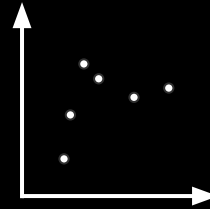
box plot

amounts
proportions
distributions (discrete)



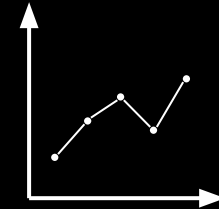
bar chart

associations
patterns



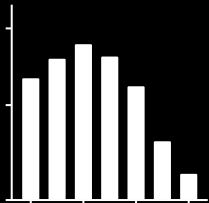
scatter plot

trends
developments



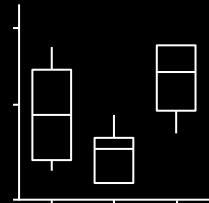
line chart

distributions (continuous)



histogram

compare distributions (continuous)



box plot

COMMUNICATE FINDINGS

quarto

{{ quarto }}

markdown

+

R

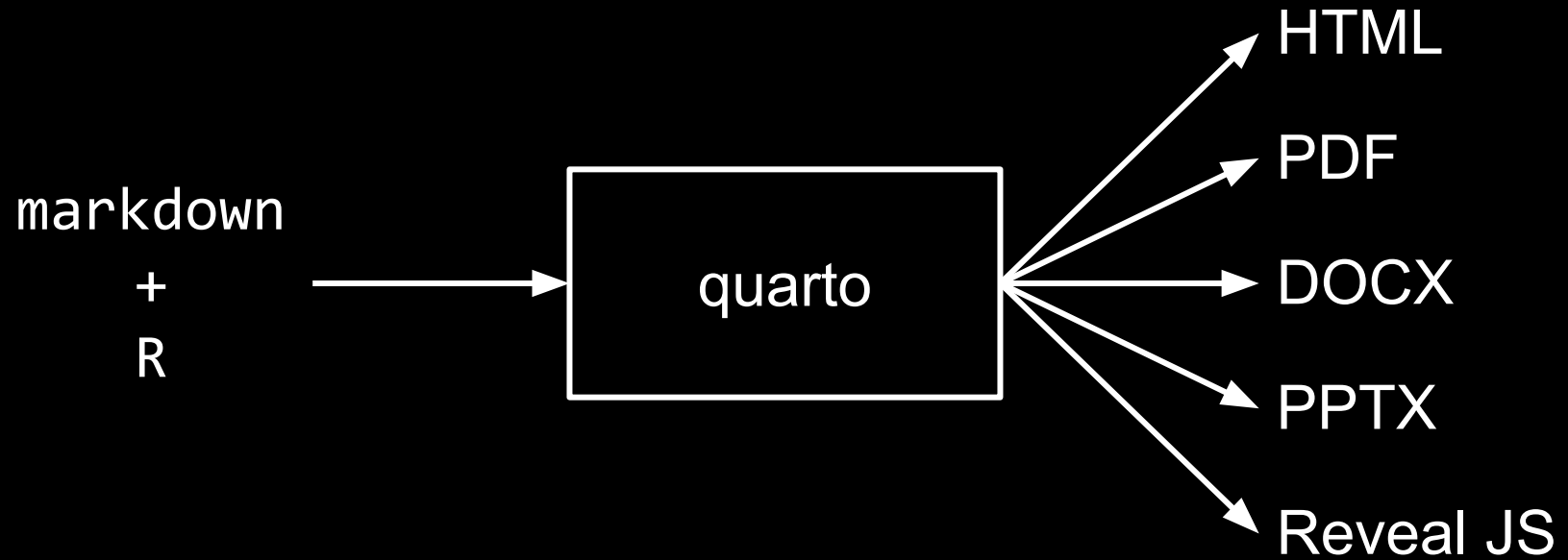
markdown

+

R



quarto



markdown

Heading 1

Heading 2

Heading 3

Heading 4

This is **italic**,

and this ****bold****

This is rendered
as `code`.

- List item A
- List item B
- List item C

1. First

2. Second

3. Third

```
! [Image title](path/to/image.png)
```

```
![[Image title](path/to/image.png){width=200}
```

```
![Image title](path/to/image.png){#fig-myimage}
```

For more details see [@fig-myimage](#).

[Linked text](https://quarto.org)

... {r}

1 + 1

...

code options


```
```{r}
```

```
#| echo: false
```

```
1 + 1
```

```
```
```

```
```{r}
```

```
#| eval: false
```

```
x <- 1 + 1
```

```
```
```

```
```{r}
```

```
#| include: false
```

```
library(tidyverse)
```

```
```
```

```
```{r}
```

```
#| message: false
```

```
data <- read_csv("data.csv")
```

```
```
```

```
```{r}
```

```
#| warning: false
```

```
data <- read_csv("data.csv")
```

```
```
```

figures

```
```{r}
#| label: fig-tweets-per-user
#| fig-cap: "Tweets per User"
tweets |>
 ggplot() +
 aes(x = screen_name) +
 geom_bar()
```
```

```
```{r}
#| code-fold: true
#| code-summary: "Show code"
tweets |>
 ggplot() +
 aes(x = screen_name) +
 geom_bar()
```
```


cross references

```
```{r}
#| label: fig-tweets-per-user
#| fig-cap: "Tweets per User"
...
```
```

In `@fig-tweets-per-user` you can see an overview of the number of tweets per user in the data set.

```
# Introduction {#sec-introduction}
```

```
...
```

```
# Analysis
```

```
As stated in @sec-introduction, the goal of this paper is  
to analyze the user behavior with regard to the content  
they tweet.
```

citation & bibliography

<https://quarto.org/docs/authoring/citations.html>

output formats

PYTHON

{{ reticulate }}