Data Wrangling

@JennyBryan
@jennybc

Data Wrangling

Rect

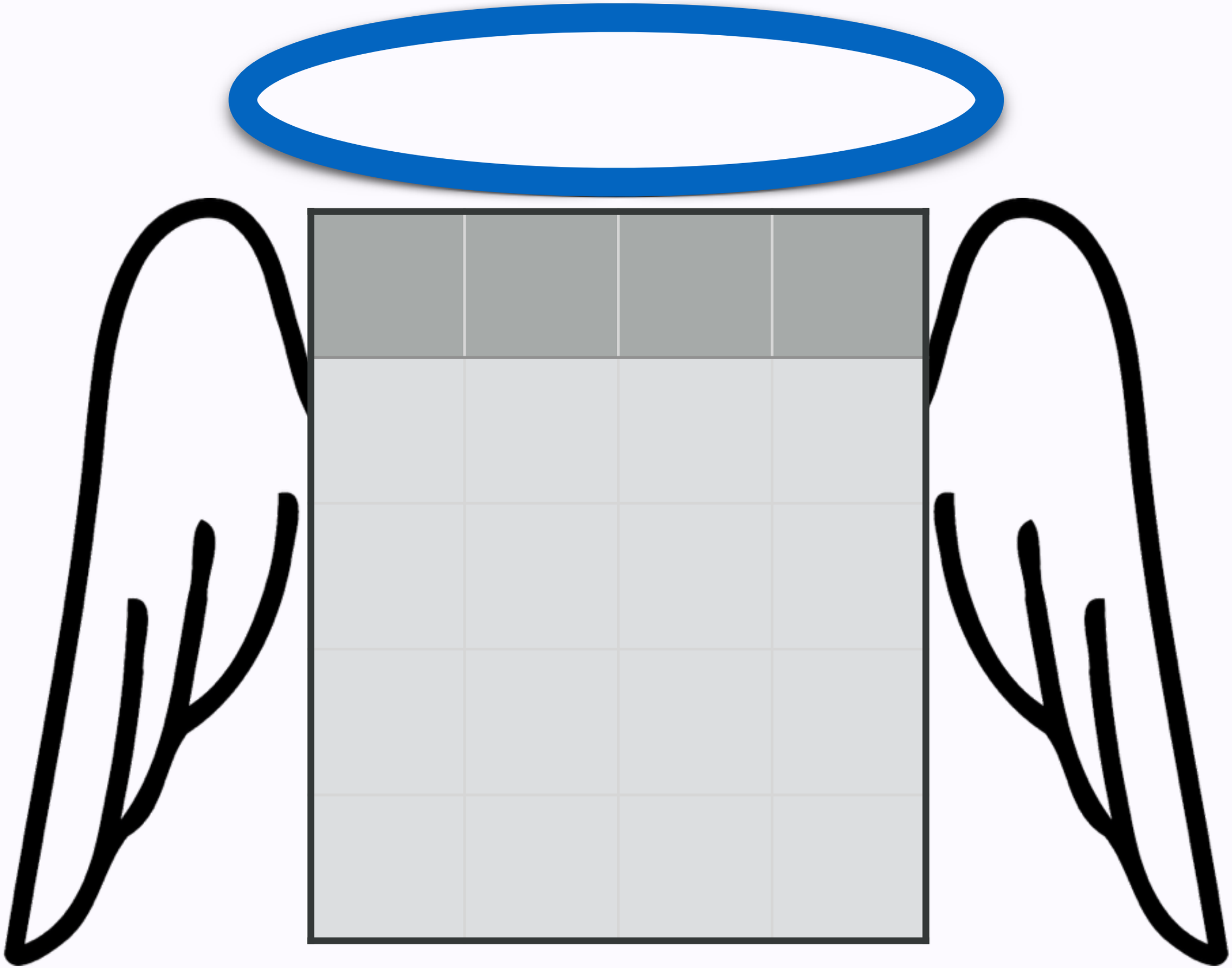
@JennyBryan
@jennybc

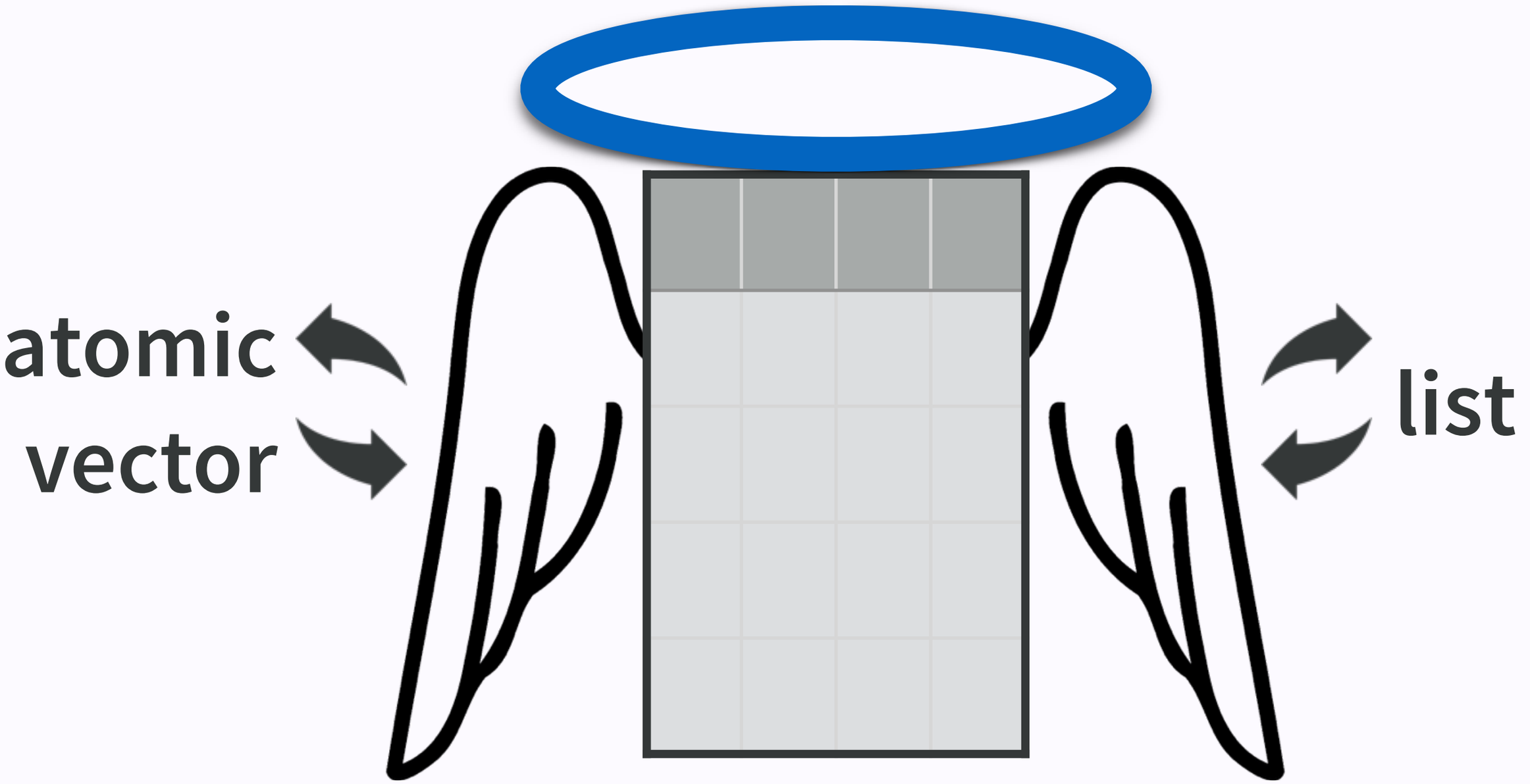
Big Data Borat:

80% time spent prepare data

20% time spent complain

about need for prepare data.




atomic
vector
list

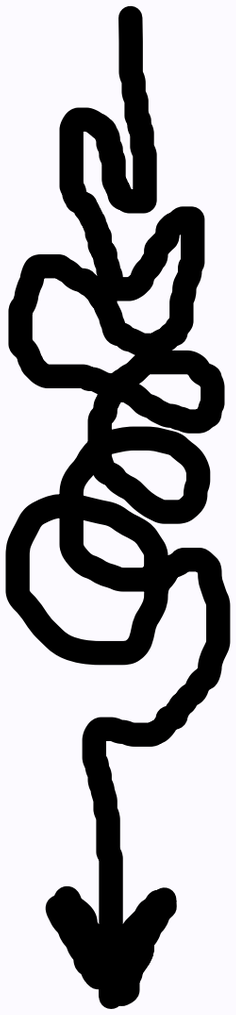
data cleaning

data wrangling

descriptive stats

inferential stats

reporting



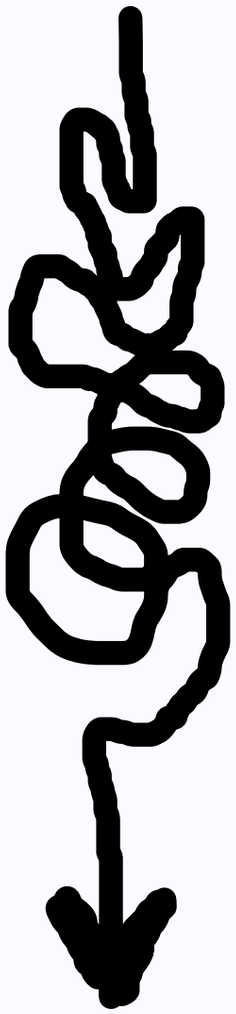
data cleaning

data wrangling

descriptive stats

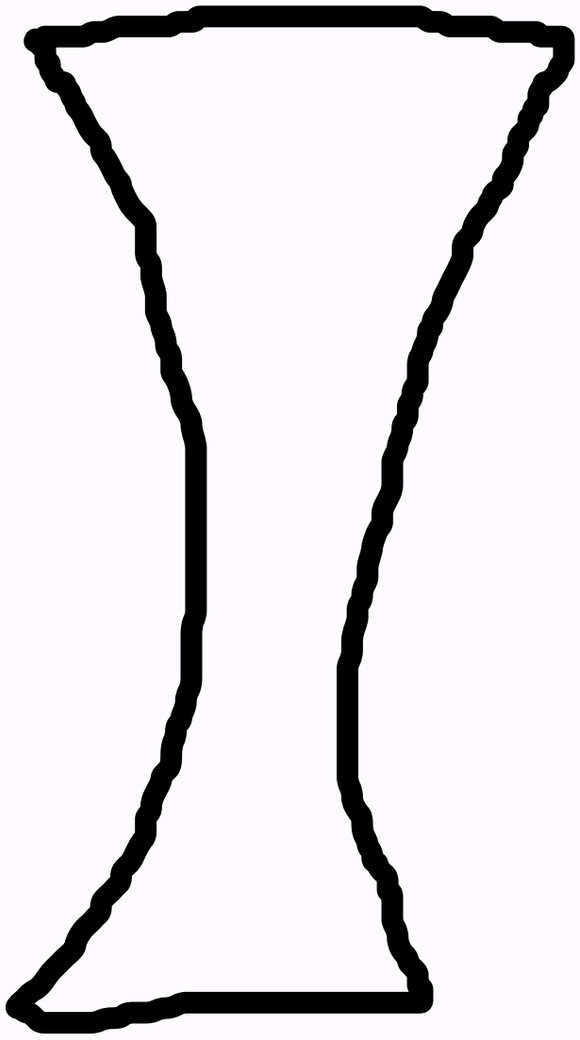
inferential stats

reporting



data cleaning
data wrangling
descriptive stats
inferential stats
reporting

programming difficulty



better exp. design → simpler stats

better data model → simpler analysis

Hadley Wickham
Lionel Henry

+ dplyr
+ tidyr
+ tibble
+ broom

https://cran.r-project.org/package=purrr
https://github.com/hadley/purrr

Lessons from my fall 2016 teaching:

https://jennybc.github.io/purrr-tutorial/

repurrrsive package (non-boring examples):

https://github.com/jennybc/repurrrsive

I am the Annie Leibovitz of lego mini-figures:

https://github.com/jennybc/lego-rstats

`x`

`x[i]`

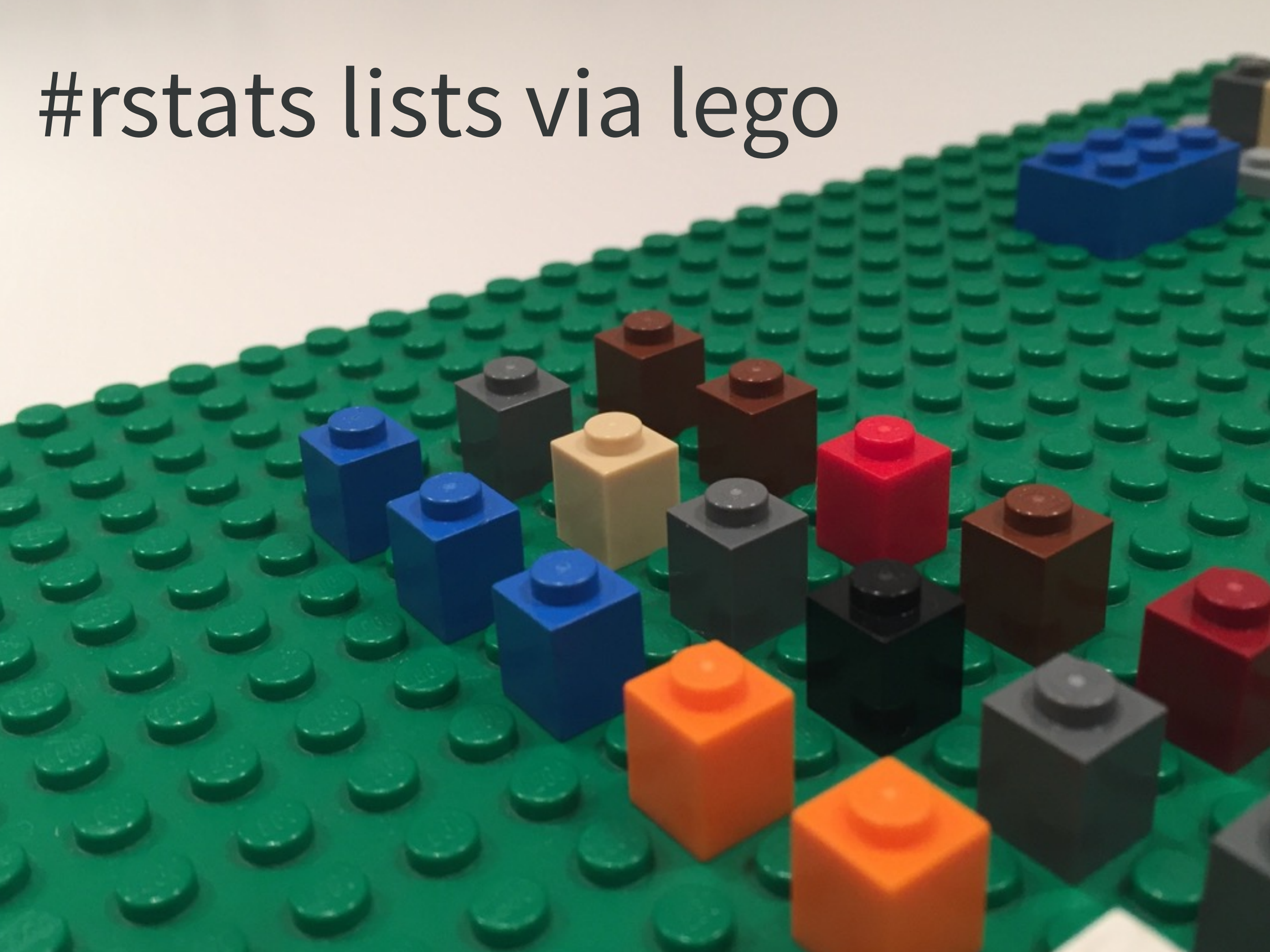
`x[[i]]`

from

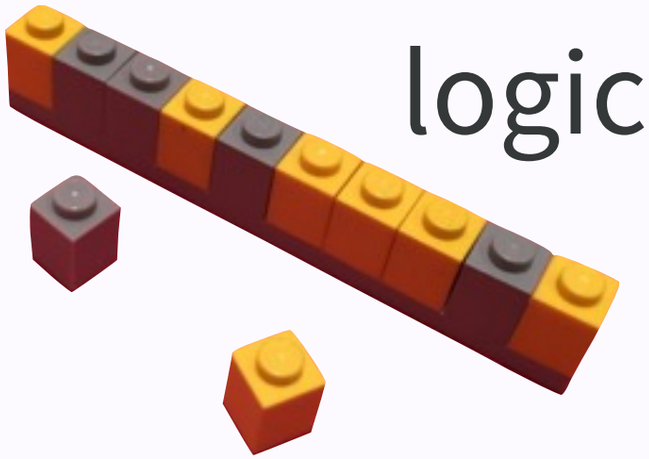
http://r4ds.had.co.nz/vectors.html#lists-of-condiments

http://legogradstudent.tumblr.com

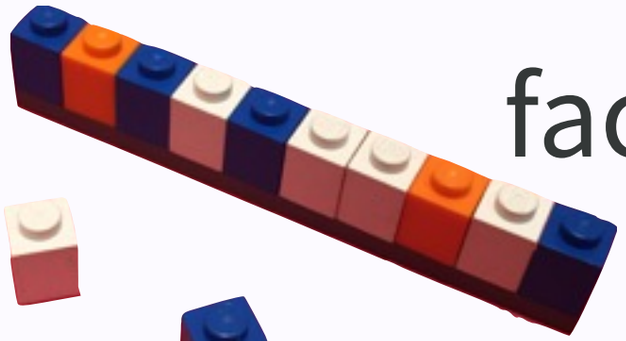
#rstats lists via lego



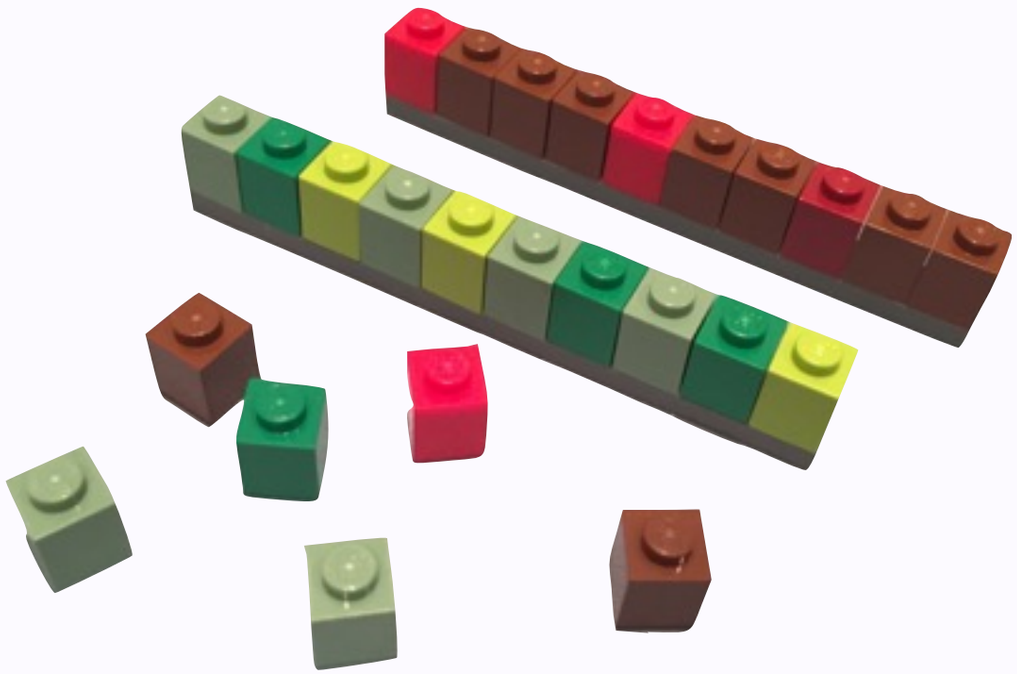
atomic vectors



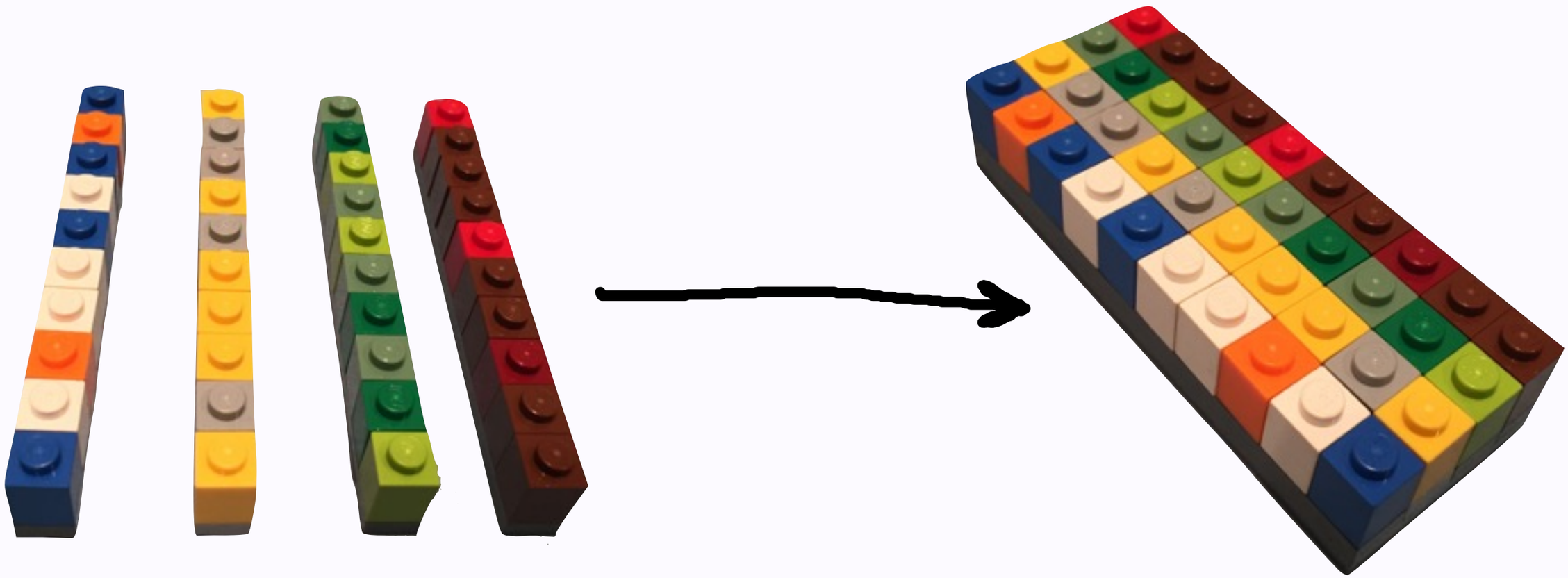
logical



factor



integer, double

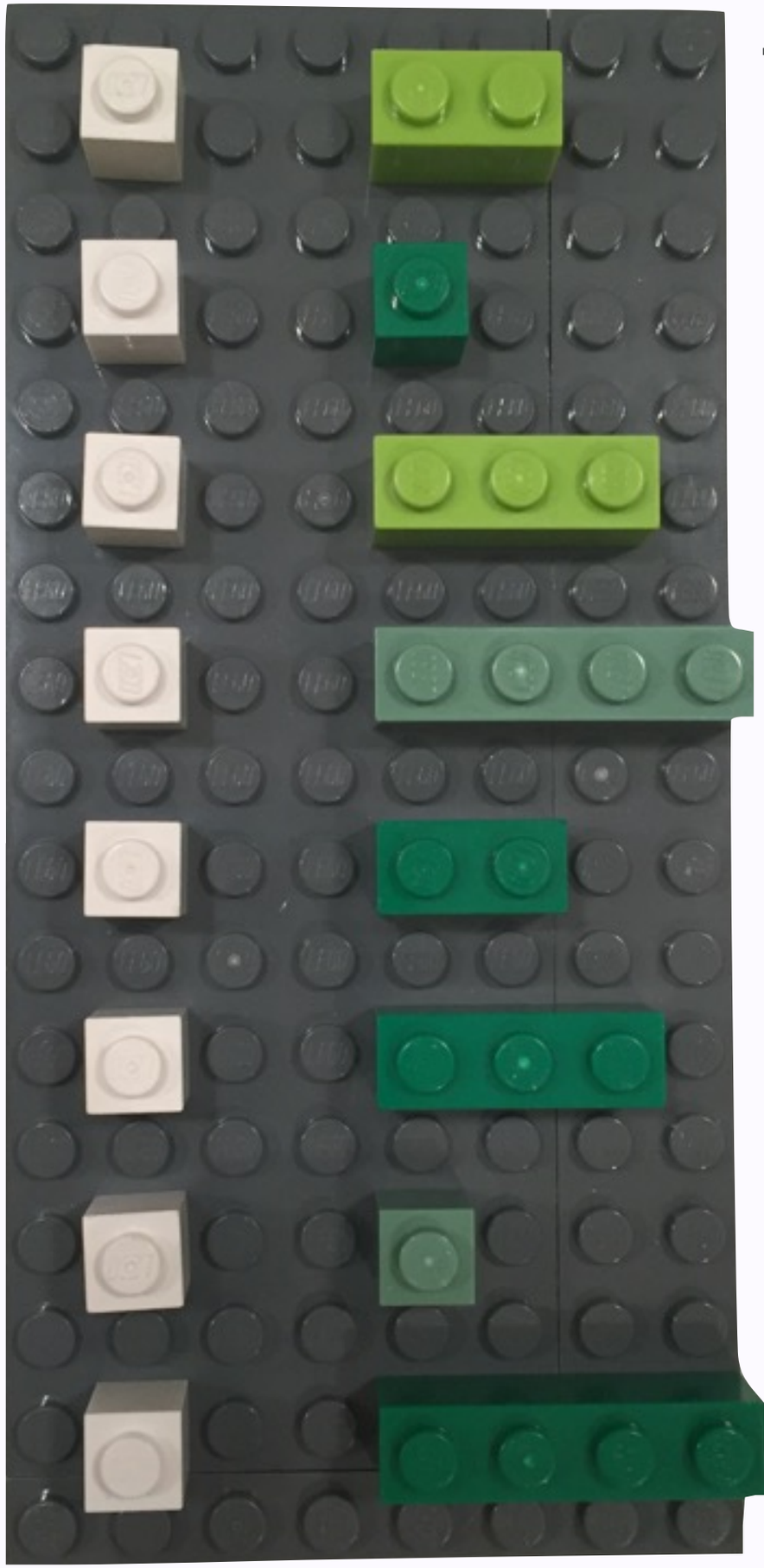


vectors of same length? **DATA FRAME!**

vectors don’t have to be atomic

works for lists too! LOVE THE **LIST COLUMN!**

atomic
vector

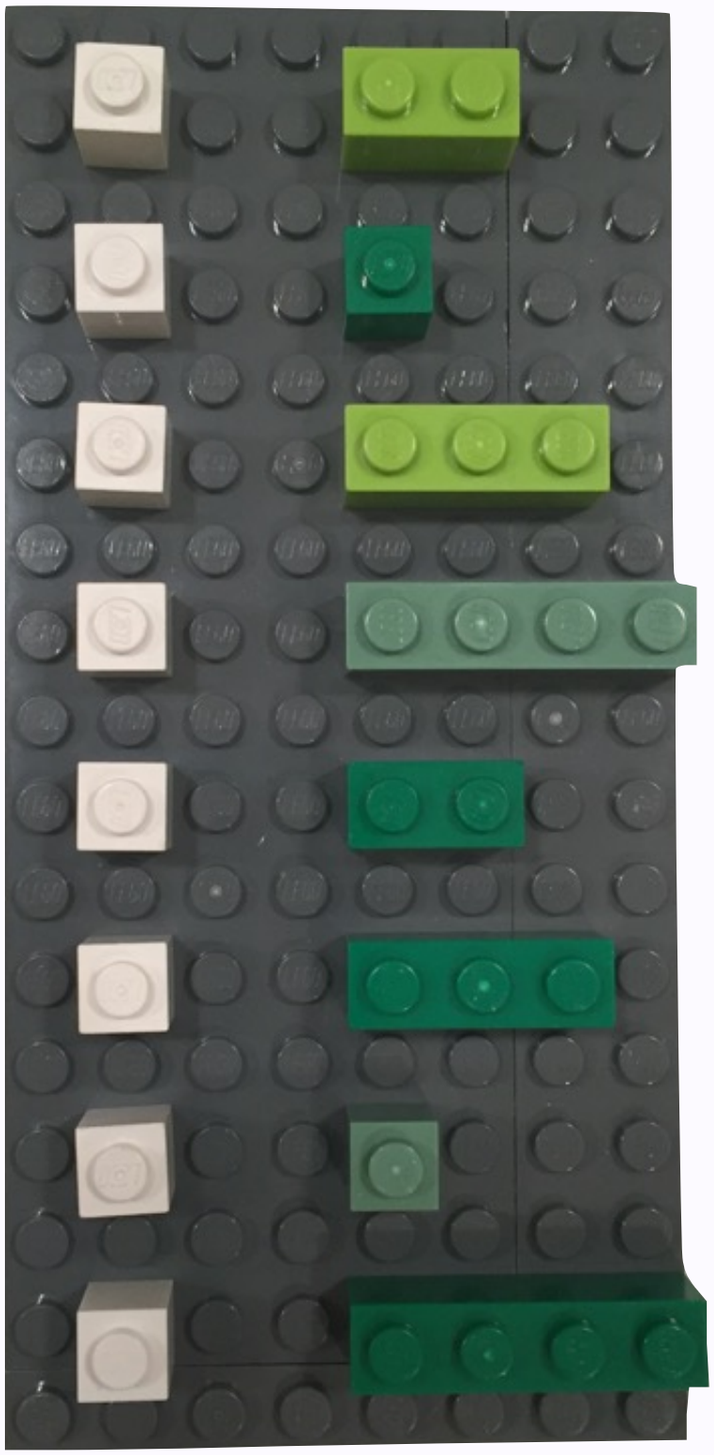


this is a data frame!

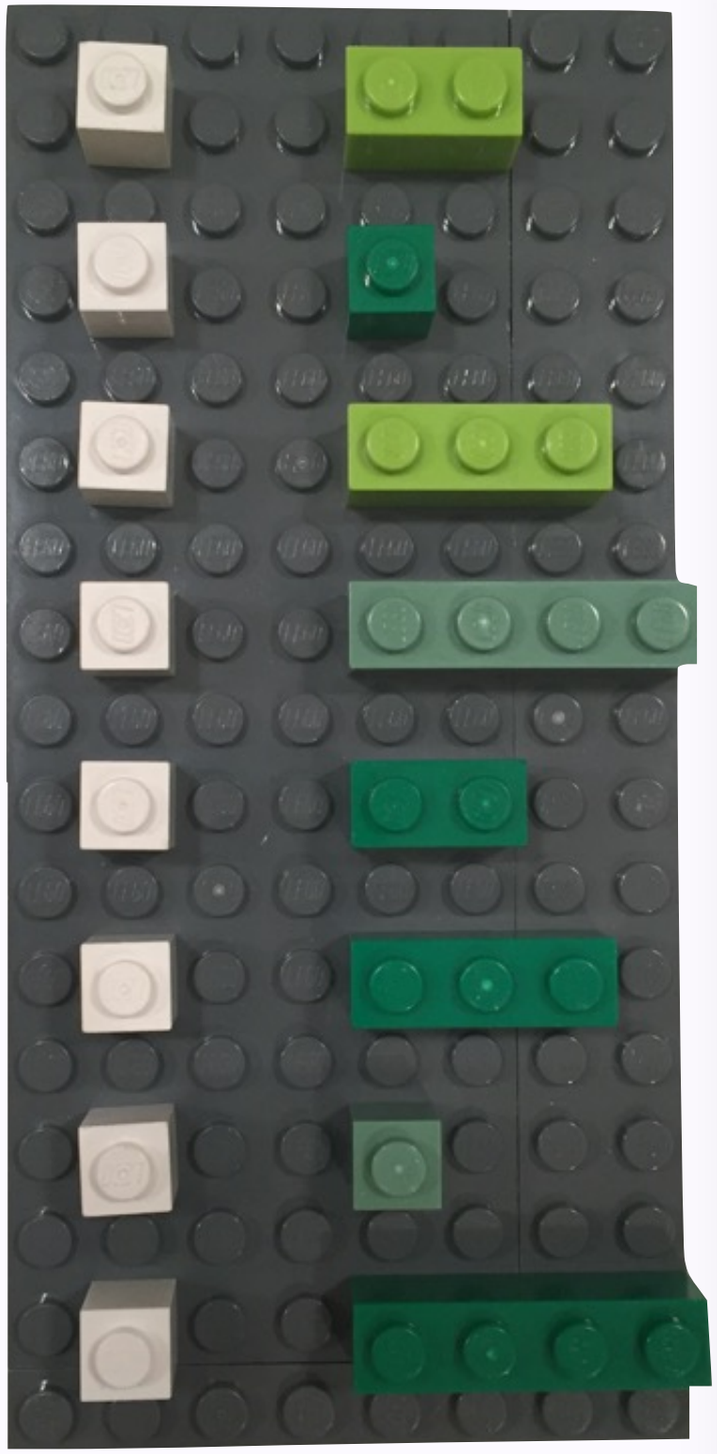
list
column

An API Of Ice And Fire | https://anapioficeandfire.com

```
{
 "url": "http://www.anapioficeandfire.com/api/characters/1303",
 "id": 1303,
 "name": "Daenerys Targaryen",
 "gender": "Female",
 "culture": "Valyrian",
 "born": "In 284 AC, at Dragonstone",
 "died": "",
 "alive": true,
 "titles": [
    "Queen of the Andals and the Rhoynar and the First Men,
     Lord of the Seven Kingdoms",
    "Khaleesi of the Great Grass Sea",
    "Breaker of Shackles/Chains",
    "Queen of Meereen",
    "Princess of Dragonstone"
],
"aliases": [
    "Dany",
    "Daenerys Stormborn",
```



```
titles
#> # A tibble: 29 × 2
#>                        name      titles
#>                       <chr>      <list>
#> 1          Theon Greyjoy  <chr [3]>
#> 2    Tyrion Lannister  <chr [2]>
#> 3   Victarion Greyjoy  <chr [2]>
#> 4                  Will <list [0]>
#> 5           Areo Hotah  <chr [1]>
#> 6                Chett <list [0]>
#> 7              Cressen  <chr [1]>
#> 8    Arianne Martell  <chr [1]>
#> 9  Daenerys Targaryen  <chr [5]>
#> 10     Davos Seaworth  <chr [4]>
#> # ... with 19 more rows
```



~/teaching/purrr-tutorial - gh-pages - RStudio Source Editor

titles

Filter

	name	titles
9	Daenerys Targaryen	c("Queen of the Andals and the Rhoynar and the First ...
10	Davos Seaworth	c("Ser", "Lord of the Rainwood", "Admiral of the Narrow...
11	Arya Stark	Princess
12	Arys Oakheart	Ser
13	Asha Greyjoy	c("Princess", "Captain of the Black Wind", "Conqueror o...
14	Barristan Selmy	c("Ser", "Hand of the Queen")
15	Varamyr	list()
16	Brandon Stark	Prince of Winterfell
17	Brienne of Tarth	list()
18	Catelyn Stark	Lady of Winterfell
19	Cersei Lannister	c("Light of the West", "Queen Dowager", "Protector of t...
20	Eddard Stark	c("Lord of Winterfell", "Warden of the North", "Hand of ...
21	Jaime Lannister	c("Ser", "Lord Commander of the Kingsguard", "Warden...
22	Jon Connington	c("Lord of Griffin's Roost", "Hand of the King", "Hand of...
23	Jon Snow	Lord Commander of the Night's Watch
24	Aeron Greyjoy	c("Priest of the Drowned God", "Captain of the Golden ...
25	Kevan Lannister	c("Ser", "Master of laws", "Lord Regent", "Protector of t...
26	Melisandre	list()
27	Merrett Frey	list()
28	Quentyn Martell	Prince

Showing 9 to 29 of 29 entries

Why would you do this to yourself?

The list is forced on you by the problem.

- String processing, e.g., regex
- JSON or XML
- Split-Apply-Combine

But why lists in a data frame?

All the usual reasons!

- Keep multiple vectors intact and “in sync”
- Use existing toolkit for filter, select, ….

What happens in the

DATA FRAME

Stays in the data frame

you have a list-column

congratulations!

1 inspect

2 query

3 modify

4 simplify

inspect

```
my_list[1:3]
my_list[[2]]
View()
str(my_list, max.level = 1)
str(my_list[[i]], list.len = 10)
listviewer::jsonedit()
```

1 inspect

2 query

3 modify

4 simplify

```
purrr::
map(.x, .f, ...)
```

`map(.x, .f, ...)`

for every element of **`.x`**

apply **`.f`**

return results **like so**

.x = minis

```
map(minis, antennate)
```

.x = minis

```
map(minis, "pants")
```





.y = hair

.x = minis

```
map2(minis, hair, enhair)
```

.y = weapons

.x = minis

```
map2(minis, weapons, arm)
```

```
minis %>%
   map2(hair, enhair) %>%
   map2(weapons, arm)
```

```
df <- tibble(pants, torso, head)
embody <- function(pants, torso, head)
  insert(insert(pants, torso), head)
```

pmap(df, embody)

```
map_df(minis, `[`,
       c("pants", "torso", "head"))
```

query

```
map(got_chars, "name")
#> [[1]]
#> [1] "Theon Greyjoy"
#>
#> [[2]]
#> [1] "Tyrion Lannister"
#>
#> [[3]]
#> [1] "Victarion Greyjoy"
```

simplify

```
map_chr(got_chars, "name")
```

```
#>  [1] "Theon Greyjoy"      "Tyrion Lannister"   "Victarion Greyjoy"
#>  [4] "Will"               "Areo Hotah"         "Chett"
#>  [7] "Cressen"            "Arianne Martell"    "Daenerys Targaryen"
#> [10] "Davos Seaworth"     "Arya Stark"         "Arys Oakheart"
#> [13] "Asha Greyjoy"       "Barristan Selmy"    "Varamyr"
#> [16] "Brandon Stark"      "Brienne of Tarth"   "Catelyn Stark"
#> [19] "Cersei Lannister"   "Eddard Stark"       "Jaime Lannister"
#> [22] "Jon Connington"     "Jon Snow"           "Aeron Greyjoy"
#> [25] "Kevan Lannister"    "Melisandre"         "Merrett Frey"
#> [28] "Quentyn Martell"    "Sansa Stark"
```

simplify

```
> map_df(got_chars, `[`,
         c("name", "culture", "gender", "born"))
```

```
#>  # A tibble: 29 × 4
#>                    name  culture gender                                  born
#>                   <chr>    <chr>  <chr>                                 <chr>
#>  1      Theon Greyjoy Ironborn   Male          In 278 AC or 279 AC, at Pyke
#>  2    Tyrion Lannister           Male           In 273 AC, at Casterly Rock
#>  3   Victarion Greyjoy Ironborn   Male          In 268 AC or before, at Pyke
#>  4               Will            Male
#>  5         Areo Hotah Norvoshi   Male        In 257 AC or before, at Norvos
#>  6              Chett            Male                        At Hag's Mire
#>  7            Cressen            Male                  In 219 AC or 220 AC
#>  8     Arianne Martell  Dornish Female                In 276 AC, at Sunspear
#>  9  Daenerys Targaryen Valyrian Female              In 284 AC, at Dragonstone
#> 10      Davos Seaworth Westeros   Male In 260 AC or before, at King's Landing
#> # ... with 19 more rows
```

simplify

```
got_chars %>% {
  tibble(name = map_chr(., "name"),
         houses = map(., "allegiances"))
} %>%
  filter(lengths(houses) > 1) %>%
  unnest()
#> # A tibble: 15 × 2
#>                 name                                houses
#>                <chr>                                 <chr>
#> 1    Davos Seaworth     House Baratheon of Dragonstone
#> 2    Davos Seaworth       House Seaworth of Cape Wrath
#> 3      Asha Greyjoy                House Greyjoy of Pyke
#> 4      Asha Greyjoy                    House Ironmaker
```

@JennyBryan

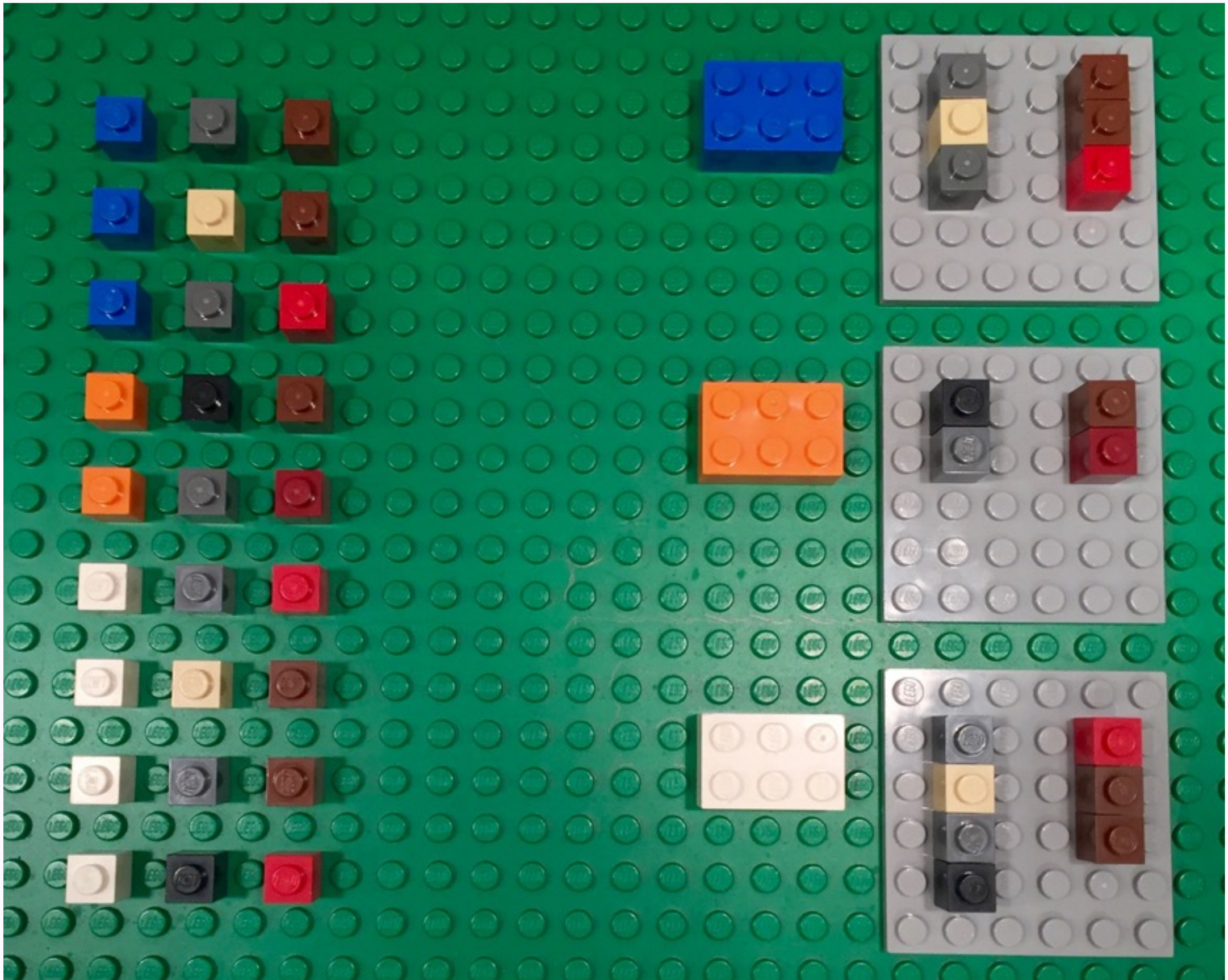
@jennybc

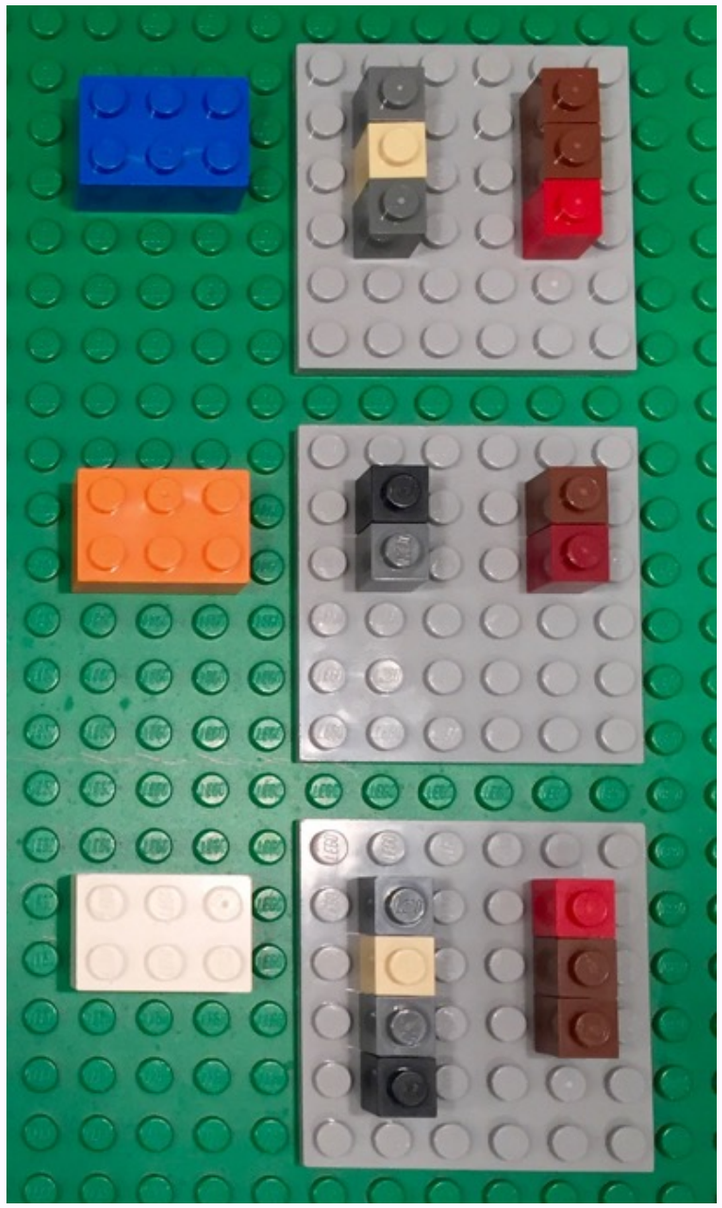
@STAT545

http://stat545.com

data frame

nested data frame





```
gap_nested <- gapminder %>%
  group_by(country, continent) %>%
  nest()
gap_nested
#>  # A tibble: 142 × 3
#>         country continent                 data
#>          <fctr>    <fctr>               <list>
#>  1  Afghanistan      Asia <tibble [12 × 4]>
#>  2      Albania    Europe <tibble [12 × 4]>
#>  3      Algeria    Africa <tibble [12 × 4]>
#>  4       Angola    Africa <tibble [12 × 4]>
#>  5    Argentina  Americas <tibble [12 × 4]>
#>  6    Australia   Oceania <tibble [12 × 4]>
#>  7      Austria    Europe <tibble [12 × 4]>
#>  8      Bahrain      Asia <tibble [12 × 4]>
#>  9   Bangladesh      Asia <tibble [12 × 4]>
#> 10      Belgium    Europe <tibble [12 × 4]>
#> # ... with 132 more rows
```

modify

```
gap_nested %>%
  mutate(fit = map(data, ~ lm(lifeExp ~ year, data = .x))) %>%
  filter(continent == "Oceania") %>%
  mutate(coefs = map(fit, coef))
```

```
#> # A tibble: 2 × 5
#>       country continent                data      fit     coefs
#>        <fctr>    <fctr>              <list>   <list>    <list>
#> 1   Australia   Oceania <tibble [12 × 4]> <S3: lm> <dbl [2]>
#> 2 New Zealand   Oceania <tibble [12 × 4]> <S3: lm> <dbl [2]>
```

simplify

```
gap_nested %>%
  …
  mutate(intercept = map_dbl(coefs, 1),
         slope = map_dbl(coefs, 2)) %>%
  select(country, continent,
         intercept, slope)
#> # A tibble: 2 × 4
#>       country continent intercept      slope
#>        <fctr>    <fctr>     <dbl>      <dbl>
#> 1   Australia   Oceania -376.1163 0.2277238
#> 2 New Zealand   Oceania -307.6996 0.1928210
```

GLASS
BOTTLES +
JARS
PUT LIDS IN
BLUE BOX

How are you doing such things today?

maybe you don’t, because you don’t know how 😔

`for` loops

`apply()`, `[slvmt]apply()`, `split()`, `by()`

with plyr: `[adl][adl_]ply()`

with dplyr: `df %>% group_by() %>% do()`

`map(.x, .f, ...)`

`.x` is a vector

lists are vectors

data frames are lists

```
map(.x, .f, ...)
```

.f is function to apply

name & position shortcuts

concise ~ formula syntax

“return results like so”

```
map(.x, .f, …)
```

can be thought of as

```
map_list(.x, .f, …)
```

```
map_lgl(.x, .f, ...)
map_chr(.x, .f, ...)
map_int(.x, .f, ...)
map_dbl(.x, .f, …)

map_df(.x, .f, …)
```

`walk(.x, .f, …)`

can be thought of as

`map_nothing(.x, .f, …)`

`map2(.x, .y, .f, …)`

`f(.x[[i]], .y[[i]], …)`

`pmap(.l, .f, …)`

`f` (tuple of `i`-th elements of the vectors in `.l, …`)

friends don’t let friends

use `do.call()`

workflow

1 do something easy with the iterative machine

2 do the real, hard thing with one representative unit

3 insert logic from 2 into template from 1