list-columns

list-columns

embrace

the

awkward

@JennyBryan
@jennybc

RStudio

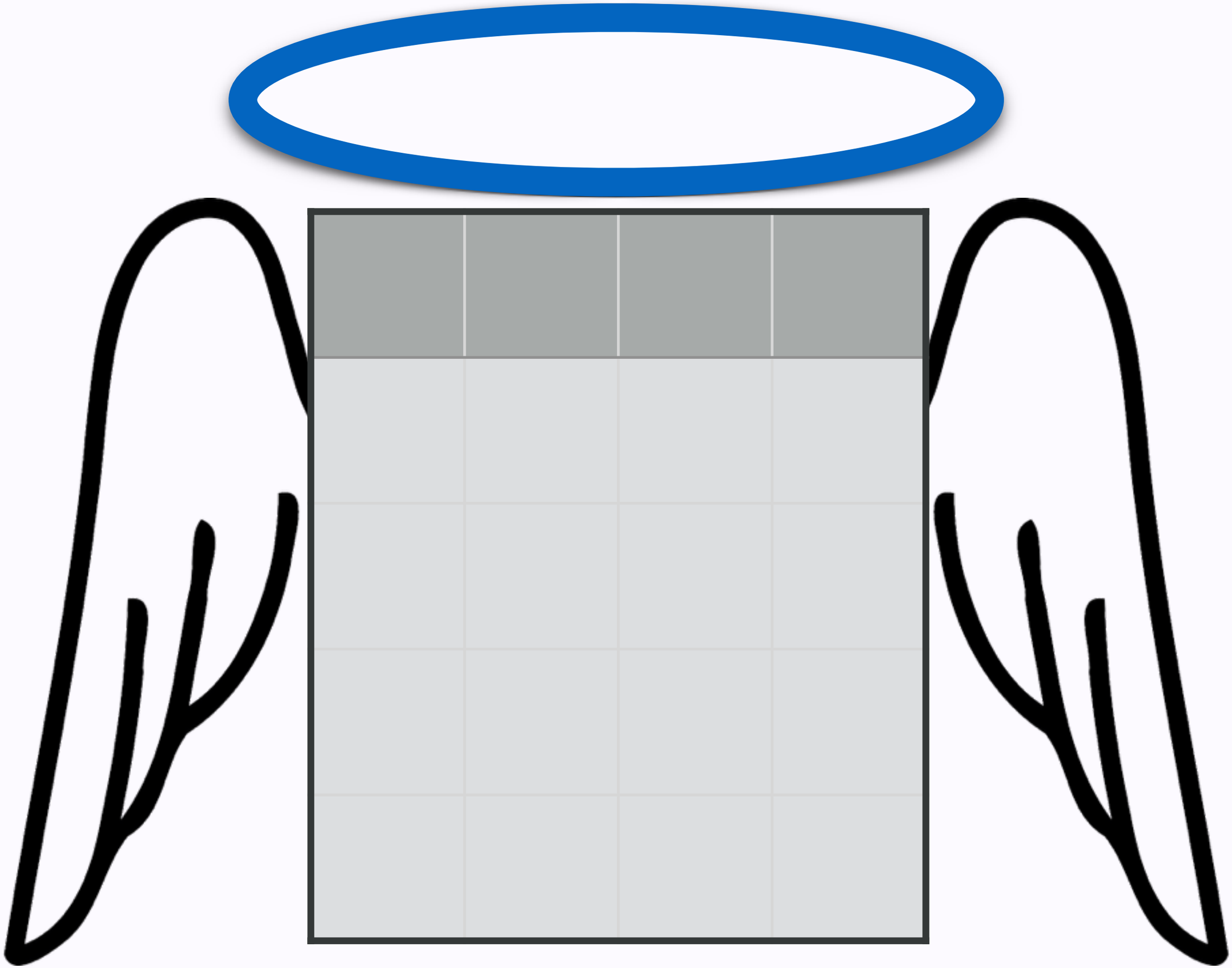
Lessons, links to resources, talks:

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

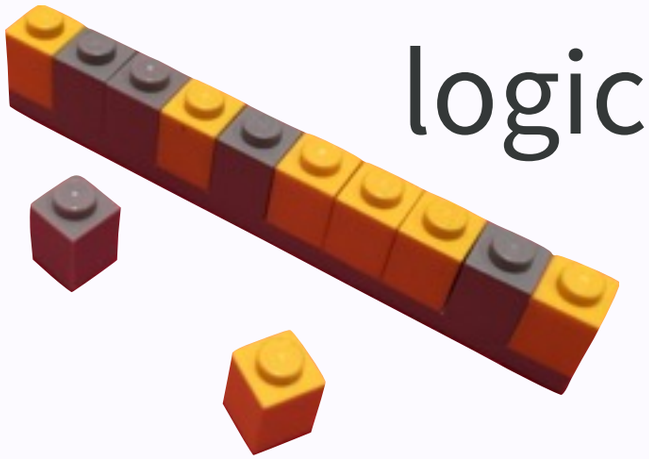
Hadley Wickham
Lionel Henry

+ dplyr
+ tidyr
+ tibble
+ broom

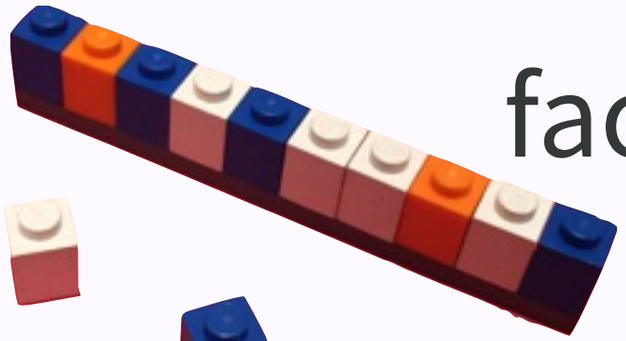
https://cran.r-project.org/package=purrr
https://github.com/hadley/purrr



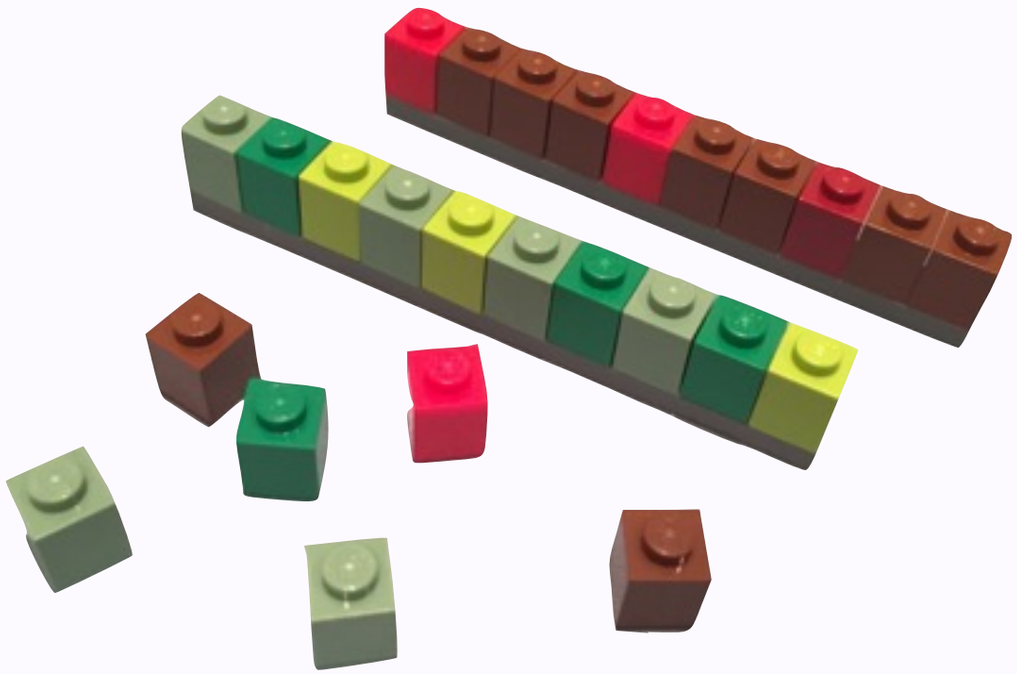
atomic vectors



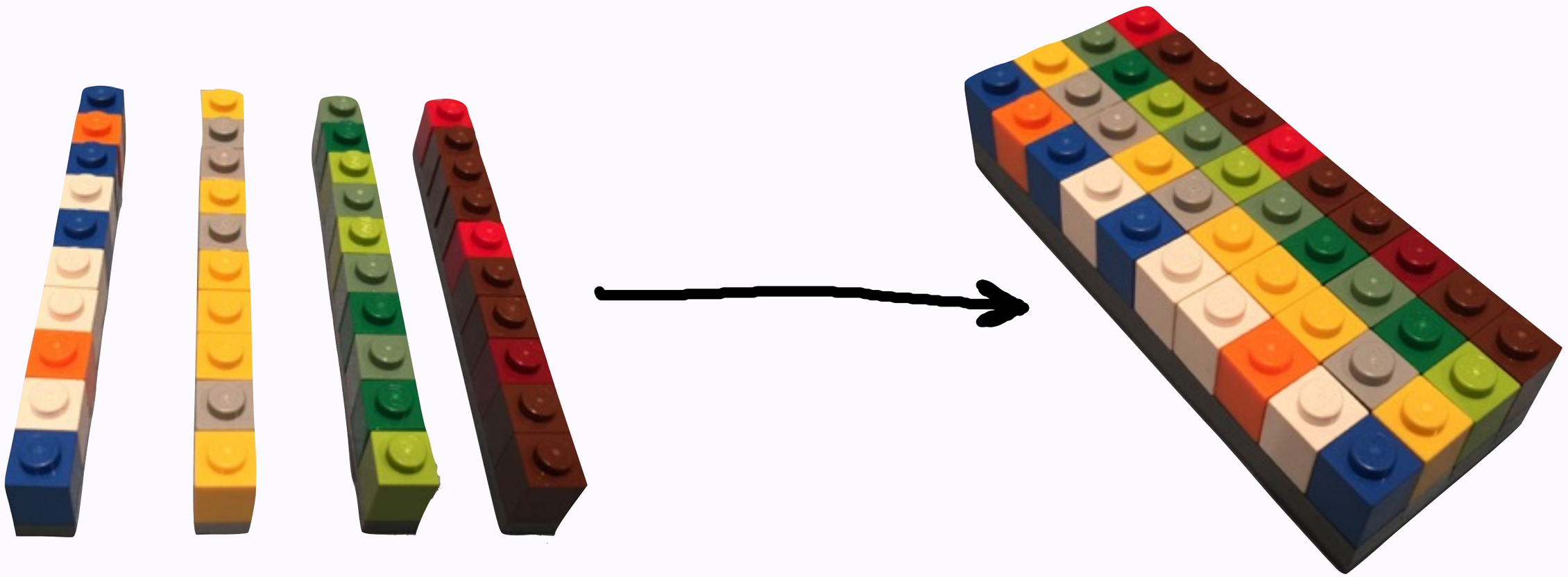
logical



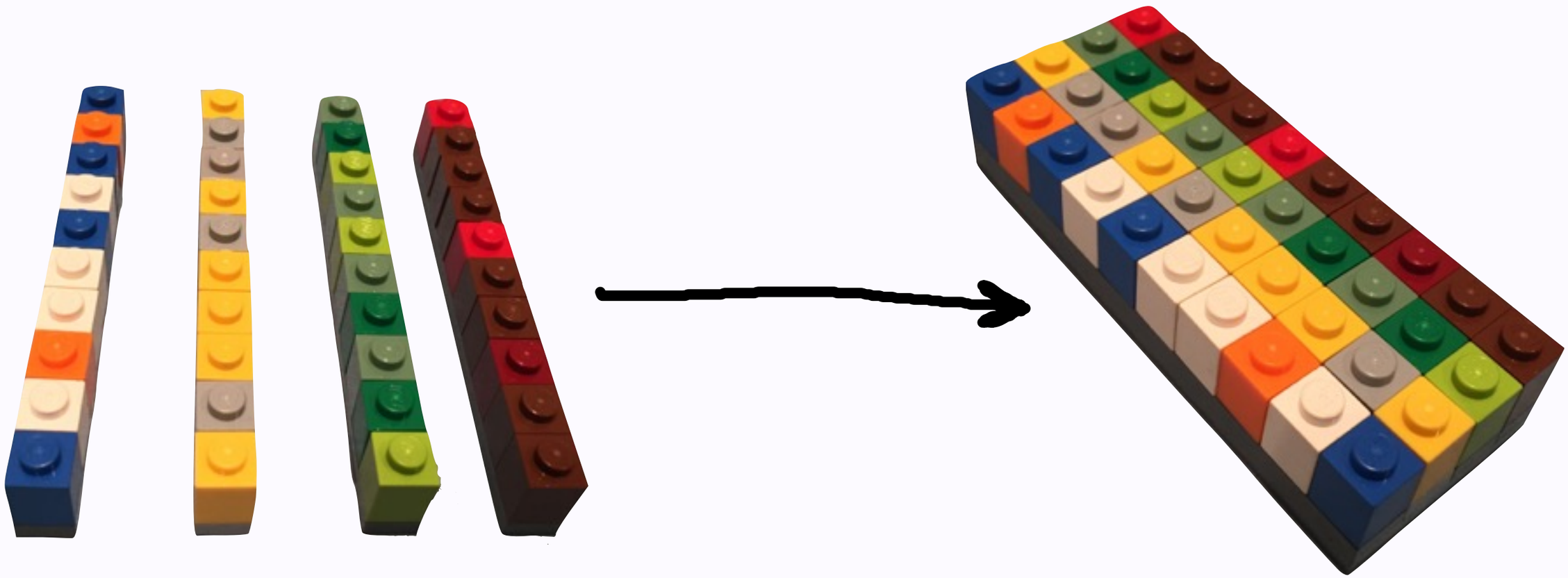
factor



integer, double



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



vectors don’t have to be atomic

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

Why would you do this to yourself?

The list is forced on you by the problem.

- String processing, e.g., regex
- JSON or XML, e.g. web APIs
- 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, mutate, ….

1 inspect

2 index

3 compute

4 simplify

Text analysis of Trump's tweets confirms he writes only the (angrier) Android half

David Robinson dgrtwo drob

http://varianceexplained.org/r/trump-tweets/

https://jennybc.github.io/purrr-tutorial/ls08_trump-tweets.html

https://jennybc.github.io/purrr-tutorial/ls13_list-columns.html

Text analysis of Trump's tweets confirms he writes only the (angrier) Android half

David Robinson dgrtwo drob

http://varianceexplained.org/r/trump-tweets/

https://jennybc.github.io/purrr-tutorial/ls08_trump-tweets.html

https://jennybc.github.io/purrr-tutorial/ls13_list-columns.html

No, I'm sorry.

This example has been cancelled. SAD!

Check out the links.

An API Of Ice And Fire

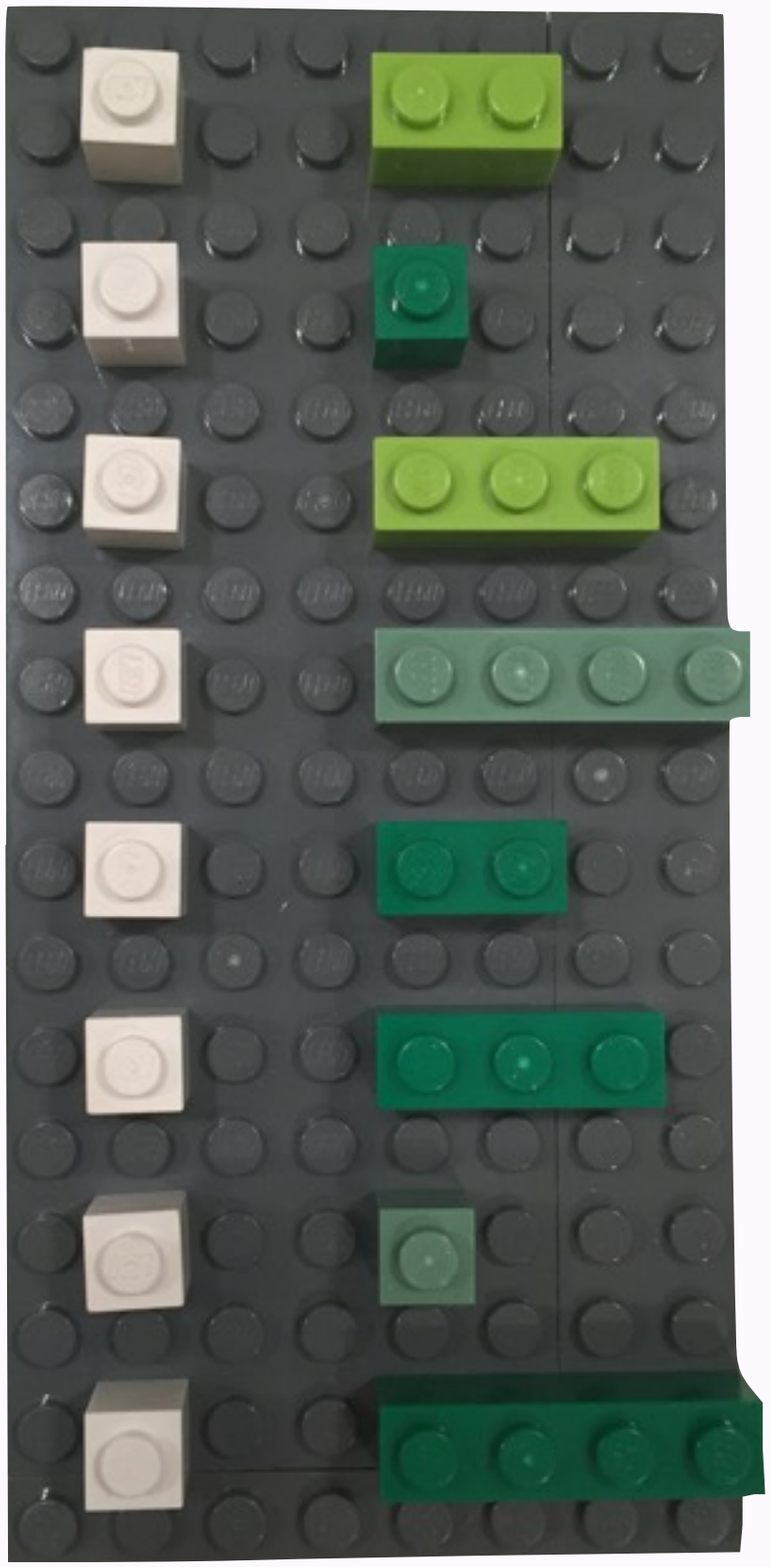
https://anapioficeandfire.com

https://github.com/jennybc/repurrrsive

```
{
 "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",
```

```
ice
#> # A tibble: 29 × 2
#>                      name        stuff
#>                     <chr>       <list>
#> 1          Theon Greyjoy <list [18]>
#> 2       Tyrion Lannister <list [18]>
#> 3     Victarion Greyjoy <list [18]>
#> 4                  Will <list [18]>
#> 5            Areo Hotah <list [18]>
#> 6                 Chett <list [18]>
#> 7               Cressen <list [18]>
#> 8       Arianne Martell <list [18]>
#> 9  Daenerys Targaryen <list [18]>
#> 10      Davos Seaworth <list [18]>
#> # ... with 19 more rows
```

```
name    stuff
<chr>   <list>
```



this **is** a data frame!

a tibble, specifically

```
str(ice$stuff[[9]], max.level = 1)    ## list.len also good stuff!
#> List of 18
#>  $ url         : chr “http://www.anapioficeandfire.com/.../1303”
#>  $ id          : int 1303
#>  $ name        : chr "Daenerys Targaryen"
#>  $ gender      : chr "Female"
#>  $ culture     : chr "Valyrian"
#>  $ born        : chr "In 284 AC, at Dragonstone"
#>  $ died        : chr ""
#>  $ alive       : logi TRUE
#>  $ titles      : chr [1:5] "Queen of the Andals and the Rhoynar ...
#>  $ aliases     : chr [1:11] "Dany" "Daenerys Stormborn” ...
#>  $ father      : chr ""
#>  $ mother      : chr ""
#>  $ spouse      : chr “http://www.anapioficeandfire.com/.../1346”
#>  $ allegiances: chr "House Targaryen of King's Landing"
#>  $ books       : chr "A Feast for Crows"
#>  $ povBooks    : chr [1:4] "A Game of Thrones" "A Clash of Kings” ...
#>  $ tvSeries    : chr [1:6] "Season 1" "Season 2" "Season 3" ...
#>  $ playedBy    : chr "Emilia Clarke"
```

```
str(ice$stuff[[9]], max.level = 1)    ## list.len also good stuff!
#> List of 18
#>  $ url          : chr “http://www.anapioficeandfire.com/.../1303”
#>  $ id           : int 1303
#>  $ name         : chr "Daenerys Targaryen"
#>  $ gender       : chr "Female"
#>  $ culture      : c
#>  $ born         :
#>  $ died         :
#>  $ alive        :
#>  $ titles       :                                    d the Rhoynar ...
#>  $ aliases      :                                    born” ...
#>  $ father       :
#>  $ mother       :
#>  $ spouse       : chr “http://www.anapioficeandfire.com/.../1346”
#>  $ allegiances: chr "House Targaryen of King's Landing"
#>  $ books        : chr "A Feast for Crows"
#>  $ povBooks     : chr [1:4] "A Game of Thrones" "A Clash of Kings” ...
#>  $ tvSeries     : chr [1:6] "Season 1" "Season 2" "Season 3" ...
#>  $ playedBy     : chr "Emilia Clarke"
```

str() is your friend

-max.level = ?

-list.len = ?

`x`

`x[i]`

`x[[i]]`

from

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

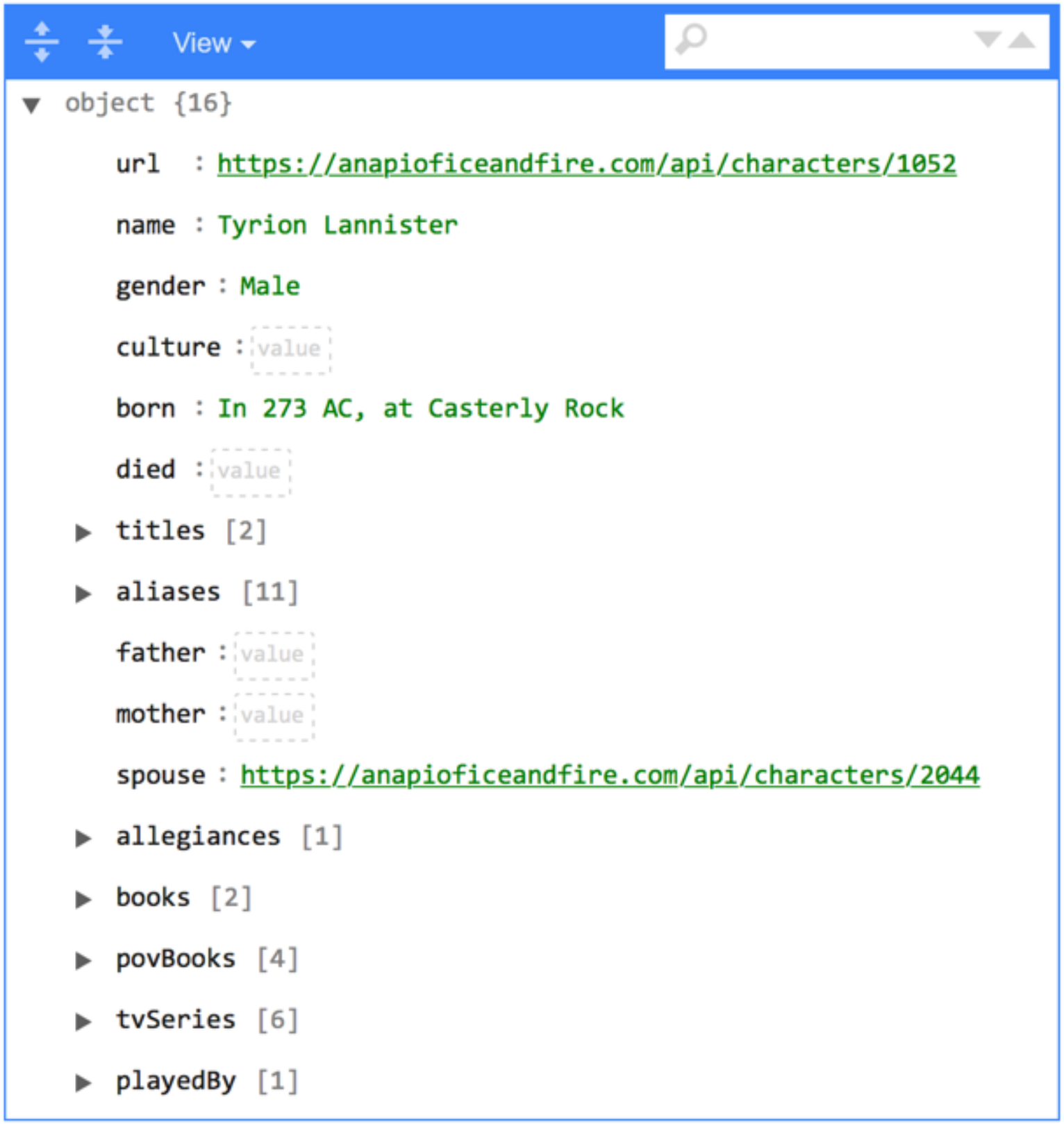
`x` `x[i]`

`[` and `[[`
are your friends too!

`x[[i]]`

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

```
listviewer::jsonedit(ice$stuff[[2]])
```



```
template <- "${name} was born ${born}."
birth_announcements <- ice %>%
  mutate(birth = map_chr(stuff, str_interp, string = template))


birth_announcements$birth
#>  [1] "Theon Greyjoy was born In 278 AC or 279 AC, at Pyke."
#>  [2] "Tyrion Lannister was born In 273 AC, at Casterly Rock."
#>  [3] "Victarion Greyjoy was born In 268 AC or before, at Pyke."
#>  [4] "Will was born ."
#>  [5] "Areo Hotah was born In 257 AC or before, at Norvos."
#>  [6] "Chett was born At Hag's Mire."
#>  [7] "Cressen was born In 219 AC or 220 AC."
#>  [8] "Arianne Martell was born In 276 AC, at Sunspear."
#>  [9] "Daenerys Targaryen was born In 284 AC, at Dragonstone."
#> [10] "Davos Seaworth was born In 260 AC or before, at King's Landing."
#> [11] "Arya Stark was born In 289 AC, at Winterfell."
#> and so on and so forth . . .
```

```
template <- "${name} was born ${born}."
birth_announcements <- ice %>%
  mutate(birth = map_chr(stuff, str_interp, string = template))

birth_announcements$bi
#>  [1] "Theon Greyjoy                                  e."
#>  [2] "Tyrion Lannis                                  ock."
#>  [3] "Victarion Gre                                   Pyke."
#>  [4] "Will was born
#>  [5] "Areo Hotah wa                                  ."
#>  [6] "Chett was bor
#>  [7] "Cressen was b
#>  [8] "Arianne Martell
#>  [9] "Daenerys Targaryen was born In 284 AC, at Dragonstone."
#> [10] "Davos Seaworth was born In 260 AC or before, at King's Landing."
#> [11] "Arya Stark was born In 289 AC, at Winterfell."
#> and so on and so forth . . .
```

reach into list-col and create simple strings from template

```
allegiances <- ice %>%
  transmute(name,
            houses = map(stuff, "allegiances")) %>%
  filter(lengths(houses) > 1) %>%
  unnest()
allegiances
#> # 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
#> 5   Barristan Selmy          House Selmy of Harvest Hall
#> 6   Barristan Selmy   House Targaryen of King's Landing
#> 7  Brienne of Tarth      House Baratheon of Storm's End
#> 8  Brienne of Tarth           House Stark of Winterfell
#> 9  Brienne of Tarth         House Tarth of Evenfall Hall
#> 10    Catelyn Stark           House Stark of Winterfell
#> 11    Catelyn Stark             House Tully of Riverrun
#> 12   Jon Connington House Connington of Griffin's Roost
#> 13   Jon Connington   House Targaryen of King's Landing
```

```
allegiances <- ice %>%
  transmute(name,
            houses = map(stuff, "allegiances")) %>%
  filter(lengths(houses) > 1) %>%
  unnest()
allegiances
#> # A tibble: 15 × 2
#>                  name                                houses
#>                 <chr>                                 <chr>
#> 1    Davos Seaworth                 atheon of Dragonstone
#> 2    Davos Seaworth                eaworth of Cape Wrath
#> 3      Asha Greyjoy                House Greyjoy of Pyke
#> 4      Asha Greyjoy                      House Ironmaker
#> 5   Barristan Selmy                Selmy of Harvest Hall
#> 6   Barristan Selmy          ryen of King's Landing
#> 7  Brienne of Tarth     House Baratheon of Storm's End
#> 8  Brienne of Tarth          House Stark of Winterfell
#> 9  Brienne of Tarth      House Tarth of Evenfall Hall
#> 10   Catelyn Stark          House Stark of Winterfell
#> 11   Catelyn Stark            House Tully of Riverrun
#> 12   Jon Connington House Connington of Griffin's Roost
#> 13   Jon Connington   House Targaryen of King's Landing
```

extract, filter, unnest

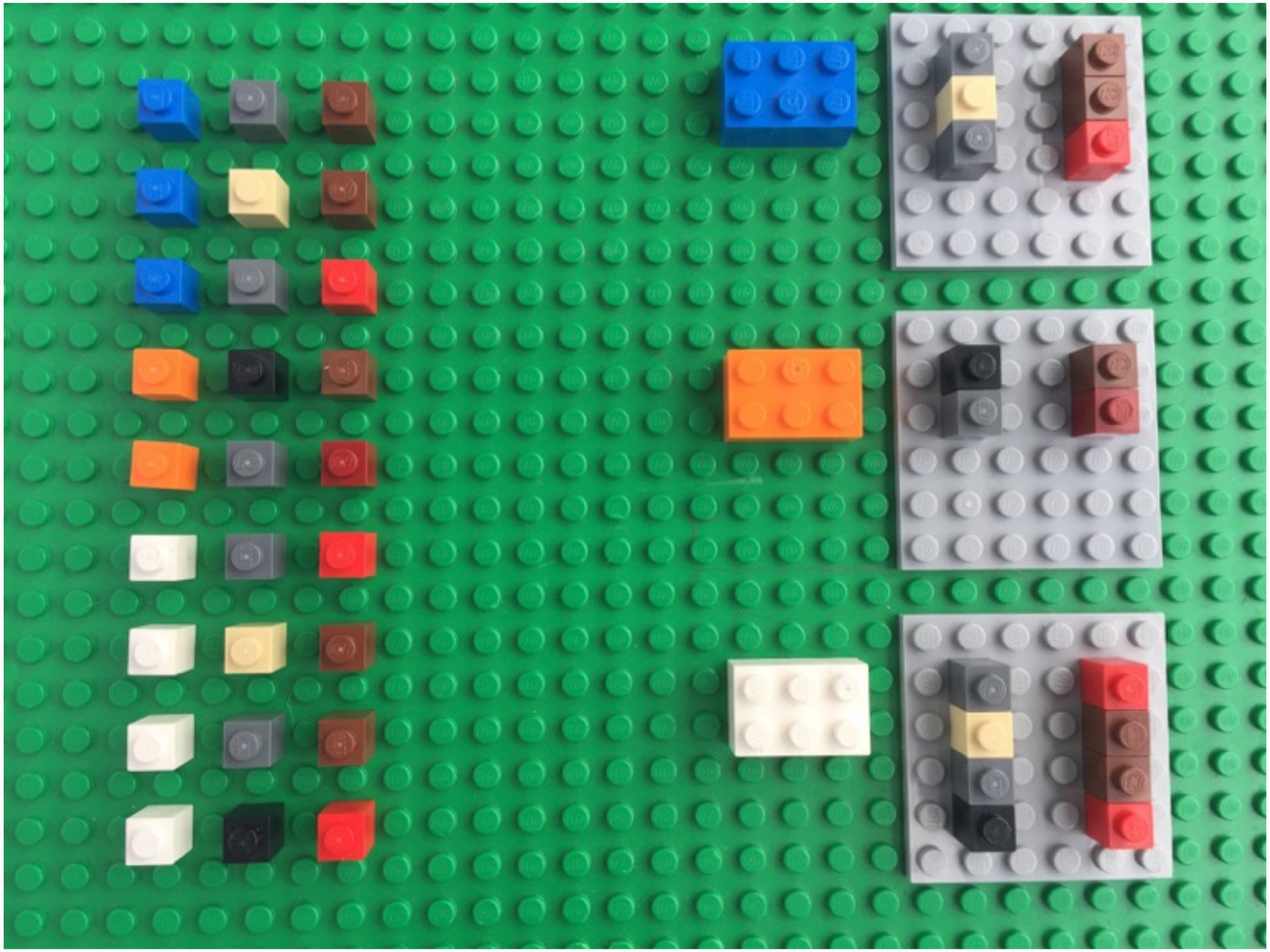
What happens in the

DATA FRAME

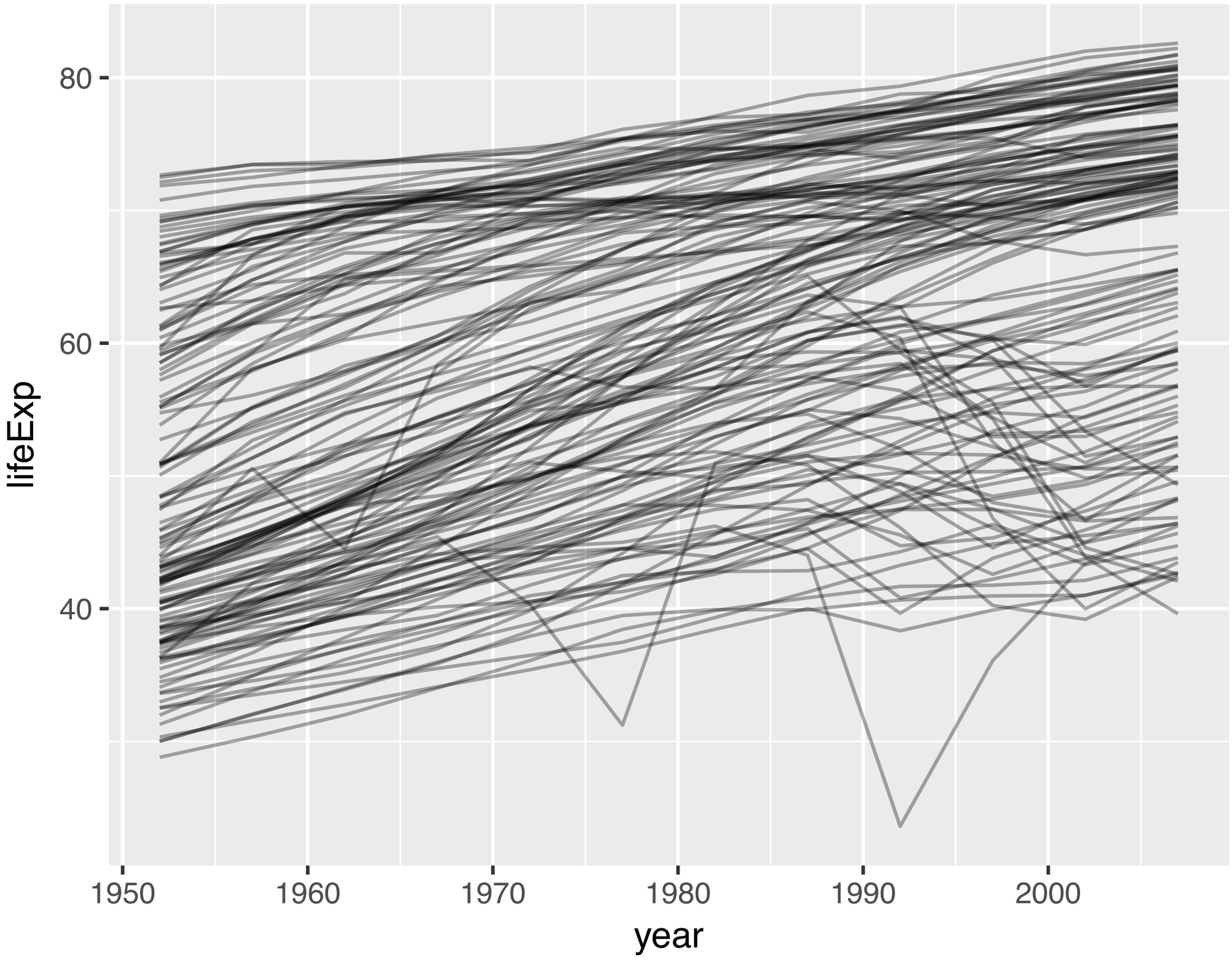
Stays in the data frame

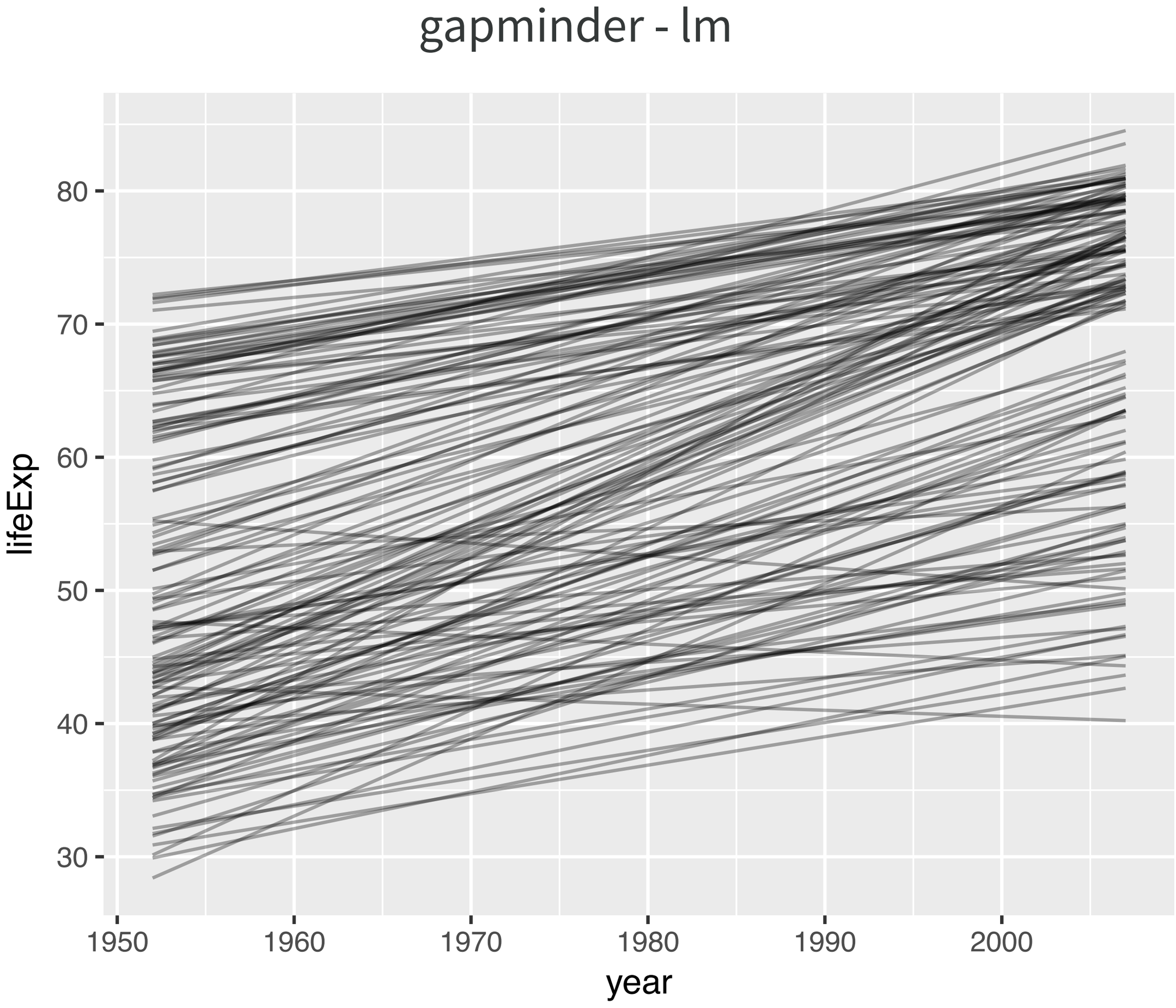
data frame

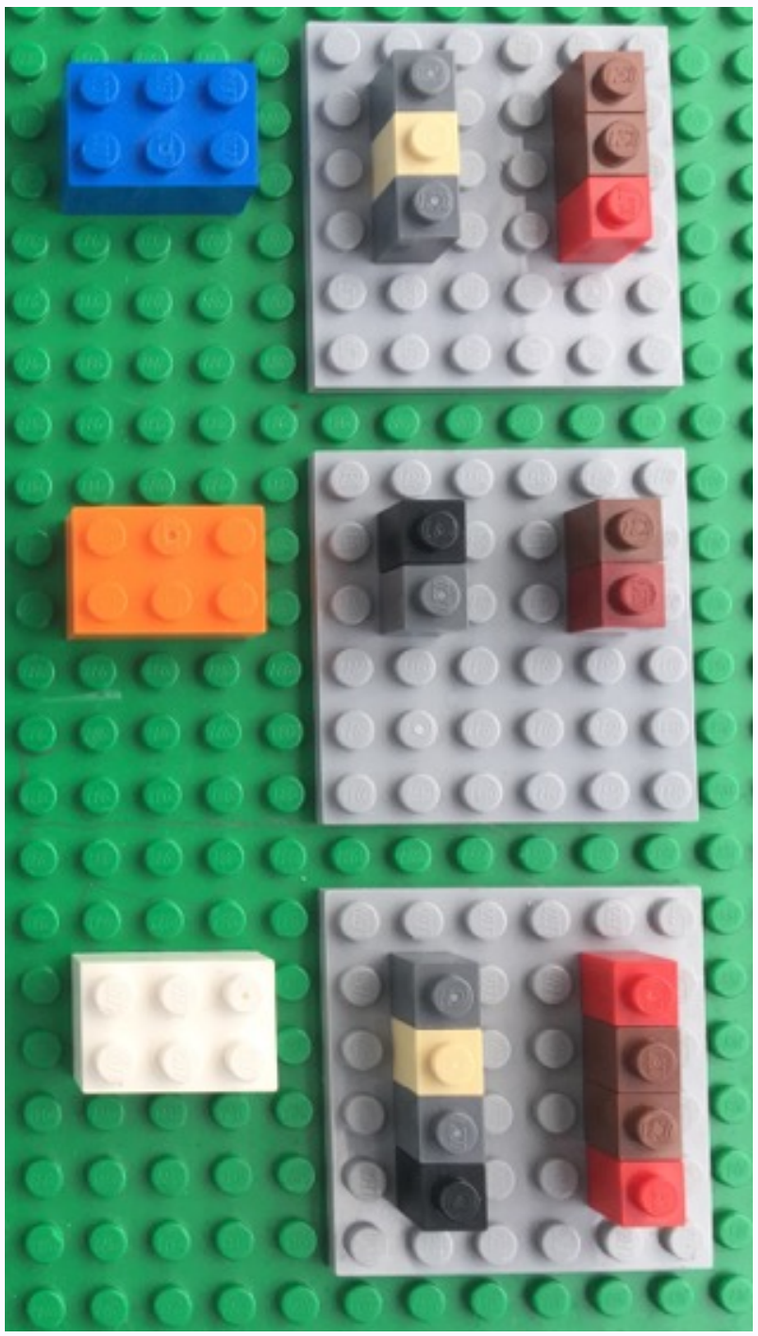
nested data frame



gapminder - raw




gapminder - lm
lifeExp
80
70
60
50
40
30
1950
1960
1970
1980
1990
2000
year



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

```
gap_fits <- gap_nested %>%
  mutate(fit = map(data, ~ lm(lifeExp ~ year, data = .x)))

gap_fits %>% tail(3)
#> # A tibble: 3 × 3
#>       country               data      fit
#>        <fctr>             <list>   <list>
#> 1 Yemen, Rep. <tibble [12 × 5]> <S3: lm>
#> 2      Zambia <tibble [12 × 5]> <S3: lm>
#> 3    Zimbabwe <tibble [12 × 5]> <S3: lm>
```

```
canada <- which(gap_fits$country == "Canada")
summary(gap_fits$fit[[canada]])
#> . . .
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -3.583e+02  8.252e+00  -43.42 1.01e-12 ***
#> year         2.189e-01  4.169e-03   52.50 1.52e-13 ***
#> . . .
#> Residual standard error: 0.2492 on 10 degrees of freedom
#> Multiple R-squared:  0.9964, Adjusted R-squared:  0.996
#> F-statistic:  2757 on 1 and 10 DF,  p-value: 1.521e-1
```

```
gap_fits %>%
  mutate(rsq = map_dbl(fit, ~ summary(.x)[["r.squared"]])) %>%
  arrange(rsq)
#> # A tibble: 142 × 4
#>              country                data      fit         rsq
#>               <fctr>              <list>   <list>       <dbl>
#> 1             Rwanda <tibble [12 × 5]> <S3: lm> 0.01715964
#> 2           Botswana <tibble [12 × 5]> <S3: lm> 0.03402340
#> 3           Zimbabwe <tibble [12 × 5]> <S3: lm> 0.05623196
#> 4             Zambia <tibble [12 × 5]> <S3: lm> 0.05983644
#> 5          Swaziland <tibble [12 × 5]> <S3: lm> 0.06821087
#> 6            Lesotho <tibble [12 × 5]> <S3: lm> 0.08485635
#> 7      Cote d'Ivoire <tibble [12 × 5]> <S3: lm> 0.28337240
#> 8       South Africa <tibble [12 × 5]> <S3: lm> 0.31246865
#> 9             Uganda <tibble [12 × 5]> <S3: lm> 0.34215382
#> 10 Congo, Dem. Rep. <tibble [12 × 5]> <S3: lm> 0.34820278
#> # ... with 132 more rows
```

```
gap_fits %>%
  mutate(coef = map(fit, broom::tidy)) %>%
  unnest(coef)
#> # A tibble: 284 × 6
#>       country         term      estimate    std.error  statistic
#>        <fctr>        <chr>         <dbl>        <dbl>      <dbl>
#> 1  Afghanistan (Intercept)  -507.5342716 40.484161954 -12.536613
#> 2  Afghanistan         year     0.2753287  0.020450934  13.462890
#> 3      Albania (Intercept)  -594.0725110 65.655359062  -9.048348
#> 4      Albania         year     0.3346832  0.033166387  10.091036
#> 5      Algeria (Intercept) -1067.8590396 43.802200843 -24.379118
#> 6      Algeria         year     0.5692797  0.022127070  25.727749
#> 7       Angola (Intercept)  -376.5047531 46.583370599  -8.082385
#> 8       Angola         year     0.2093399  0.023532003   8.895964
#> 9    Argentina (Intercept)  -389.6063445  9.677729641 -40.258031
#> 10   Argentina         year     0.2317084  0.004888791  47.395847
#> # ... with 274 more rows, and 1 more variables: p.value <dbl>
```

inspect

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

1 inspect

2 index

3 compute

4 simplify

purrr::map_*(list-column,…)

dplyr::mutate()

dplyr::filter()

etc …

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