Organize Multiple Models

Jake Thompson

wjakethompson.com

/ @wjakethompson

ATLAS

Accessible Teaching, Learning, and Assessment Systems

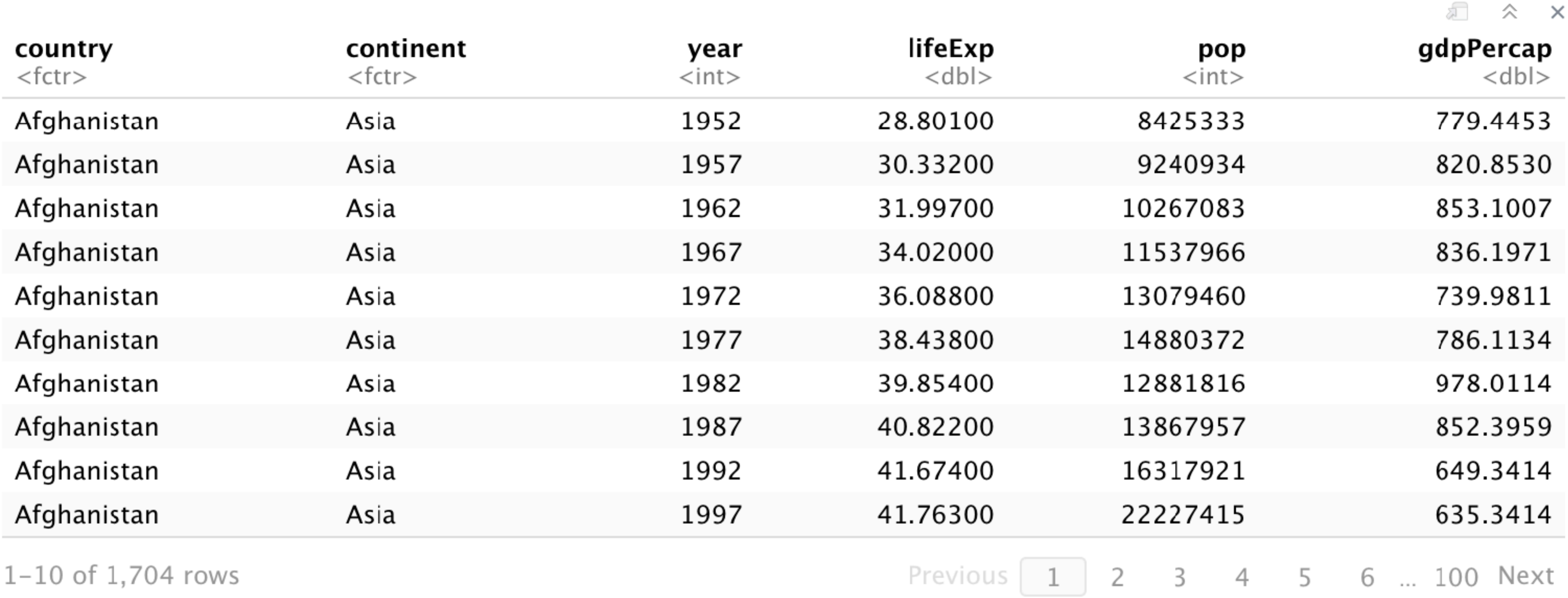
Your Turn 0

- Open **09-Organize.Rmd**
- Run the setup chunk

Adapted from Data Science in the tidyverse, CC BY Amelia McNamara

gap minder

```
gapminder
```



country	continent	year	lifeExp	pop	gdpPercap
<fctr>	<fctr>	<int>	<dbl>	<int>	<dbl>
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Afghanistan	Asia	1962	31.99700	10267083	853.1007
Afghanistan	Asia	1967	34.02000	11537966	836.1971
Afghanistan	Asia	1972	36.08800	13079460	739.9811
Afghanistan	Asia	1977	38.43800	14880372	786.1134
Afghanistan	Asia	1982	39.85400	12881816	978.0114
Afghanistan	Asia	1987	40.82200	13867957	852.3959
Afghanistan	Asia	1992	41.67400	16317921	649.3414
Afghanistan	Asia	1997	41.76300	22227415	635.3414

1–10 of 1,704 rows

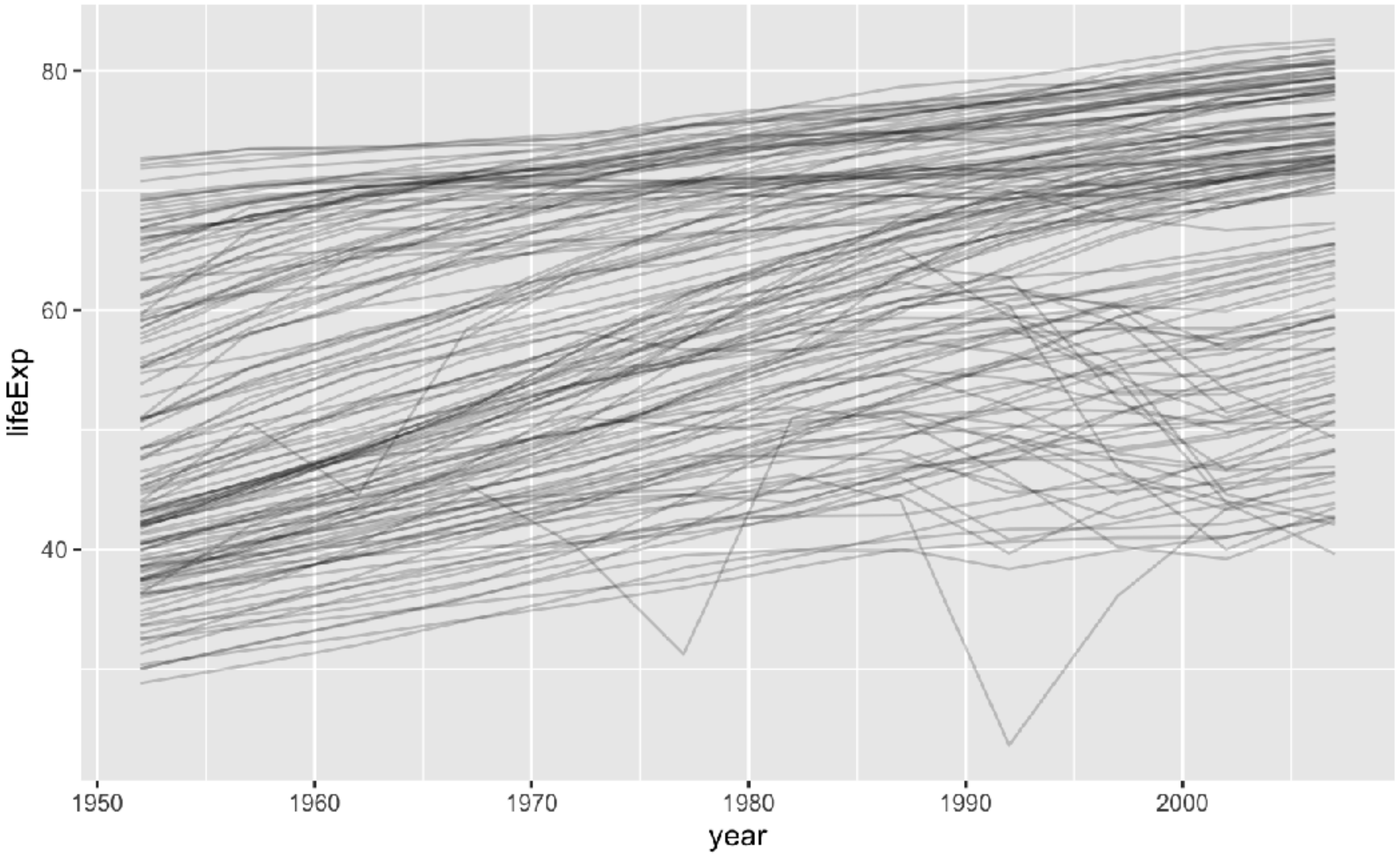
Your Turn 1

How has life expectancy changed over time?

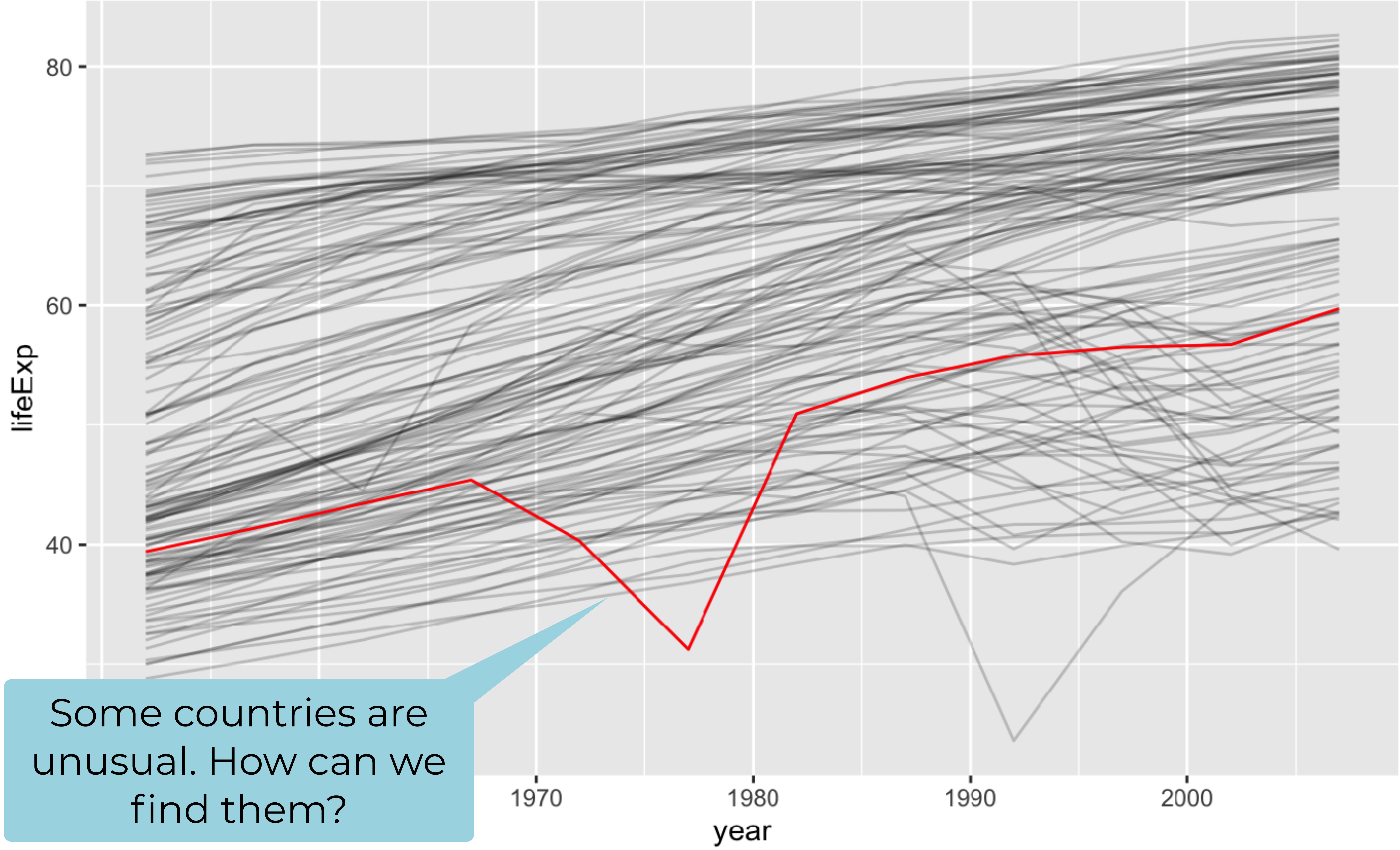
Make a line plot of **lifeExp** vs. **year** grouped by **country**. Set alpha to 0.2 to see the results better.

02 : 00

```
gapminder %>%
  ggplot(mapping = aes(x = year, y = lifeExp, group = country) +
    geom_line(alpha = 0.2)
```



Adapted from Master the Tidyverse, CC BY RStudio

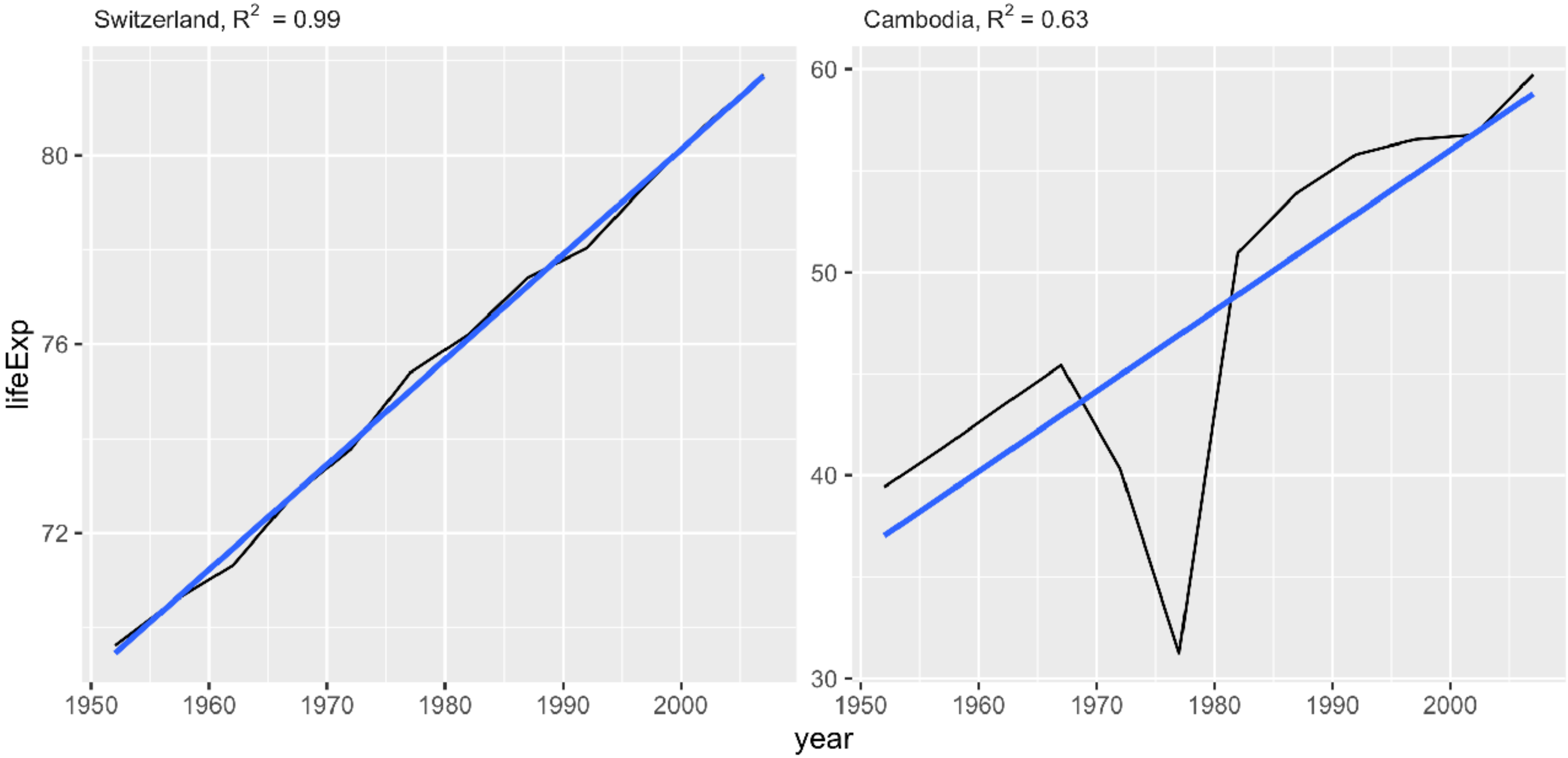


Some countries are unusual. How can we find them?

Adapted from Master the Tidyverse, CC BY RStudio

Idea 1

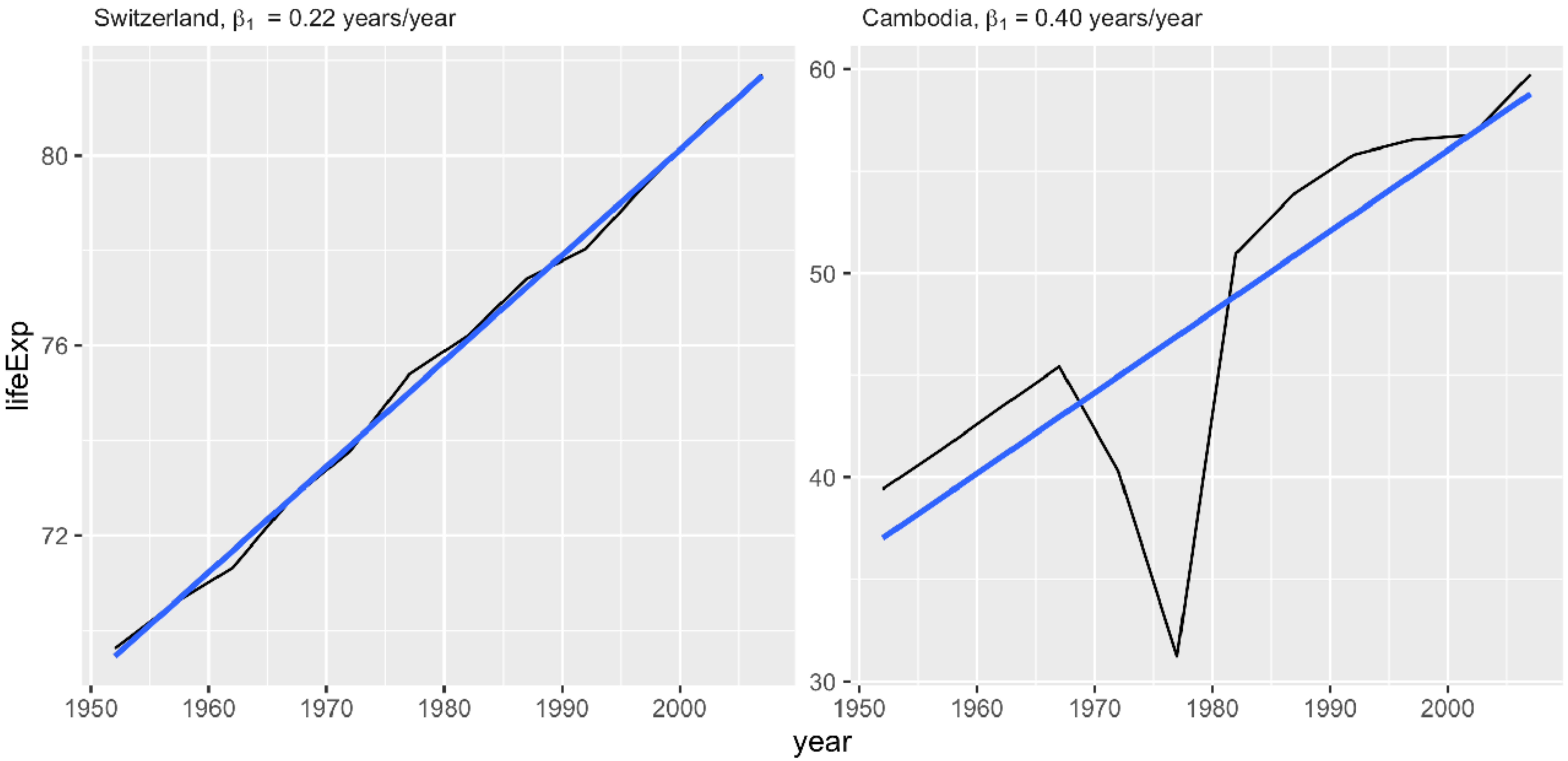
To quantify "linearity," fit a linear model, compare **r-squared**.



Adapted from Master the Tidyverse, CC BY RStudio

Idea 2

To quantify rate of change, fit a linear model, extract **coefficient on year**.



ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

Goal

Fit model, compute R^2, collect coefficient ***for every country***.

1. **dplyr** + **tidyr** grouping toolkit
2. **purrr** toolkit and list columns

Adapted from Master the Tidyverse, CC BY RStudio

List Columns

In R4DS

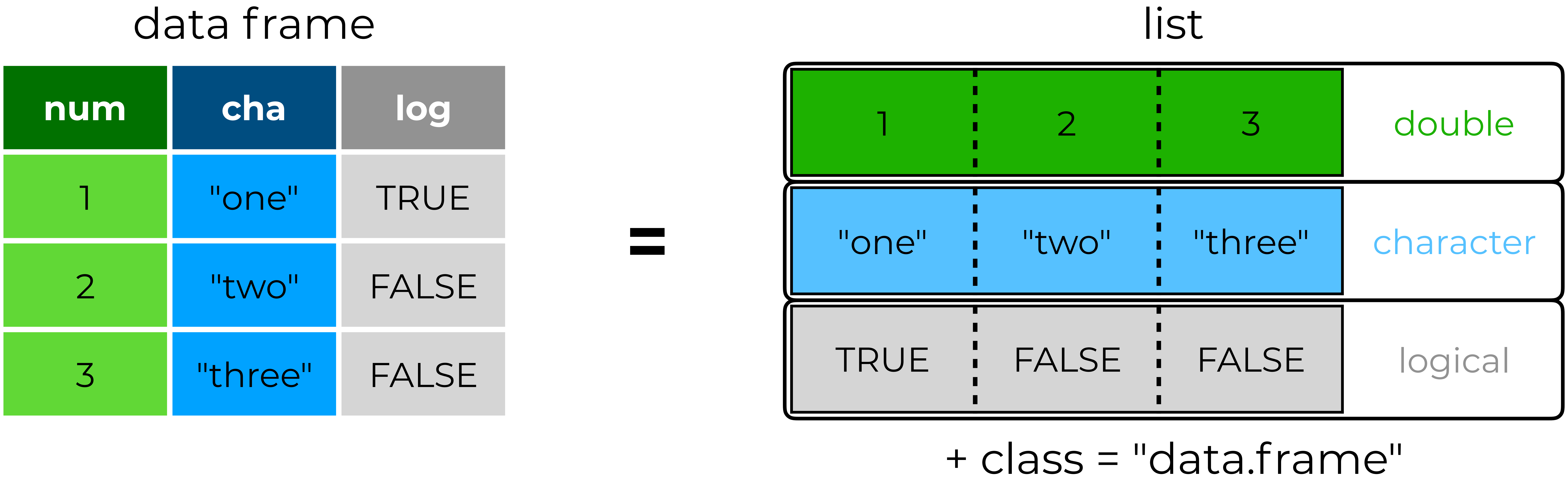
Many Models

Quiz

- How is a data frame/tibble similar to or different from a list?

Adapted from Data Science in the tidyverse, CC BY Amelia McNamara

A data frame / tibble is a list



data frame

num	cha	log
1	"one"	TRUE
2	"two"	FALSE
3	"three"	FALSE

=

list

1	2	3	double
"one"	"two"	"three"	character
TRUE	FALSE	FALSE	logical

+ class = "data.frame"

Adapted from Master the Tidyverse, CC BY RStudio

A data frame / tibble is a list

data frame

num	cha	log
1	"one"	TRUE
2	"two"	FALSE
3	"three"	FALSE

df["num"]

num
1
2
3

df[["num"]]
df$num

c(1, 2, 3)

Adapted from Master the Tidyverse, CC BY RStudio

A data frame / tibble is a list

data frame

num	cha	log
1	"one"	TRUE
2	"two"	FALSE
3	"three"	FALSE

```
df %>% select(num)
```

num
1
2
3

```
df[["num"]]
df$num
```

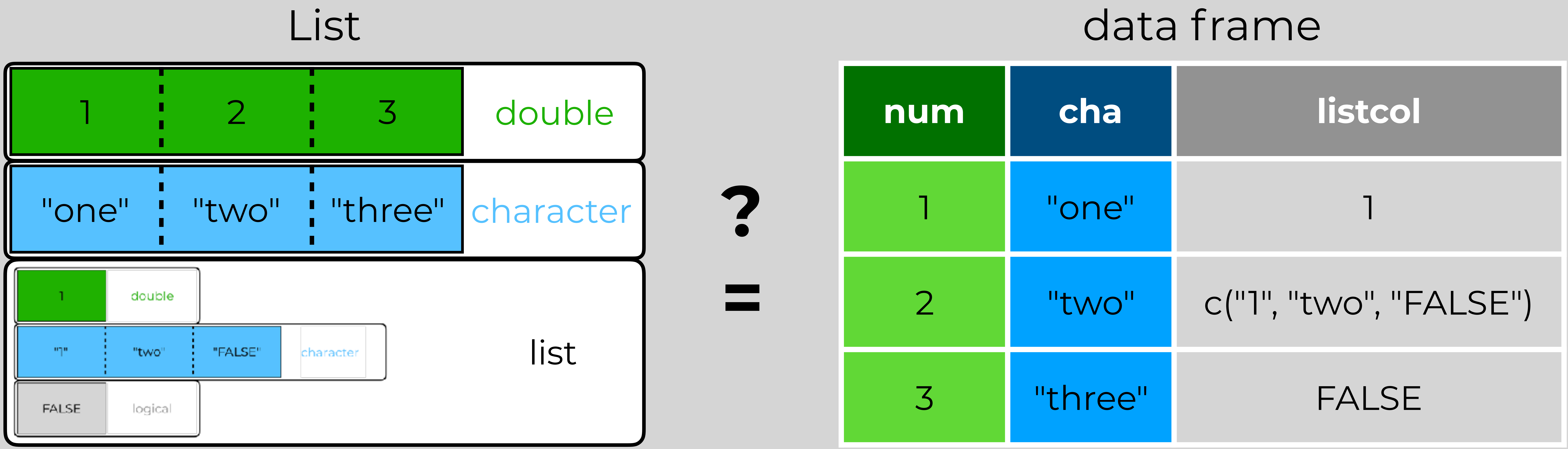
```
c(1, 2, 3)
```

ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

Quiz

If one of the elements of a list can be another list,can one of the columns of a data frame be another list?



num	cha	listcol
1	"one"	1
2	"two"	c("1", "two", "FALSE")
3	"three"	FALSE

ATLAS

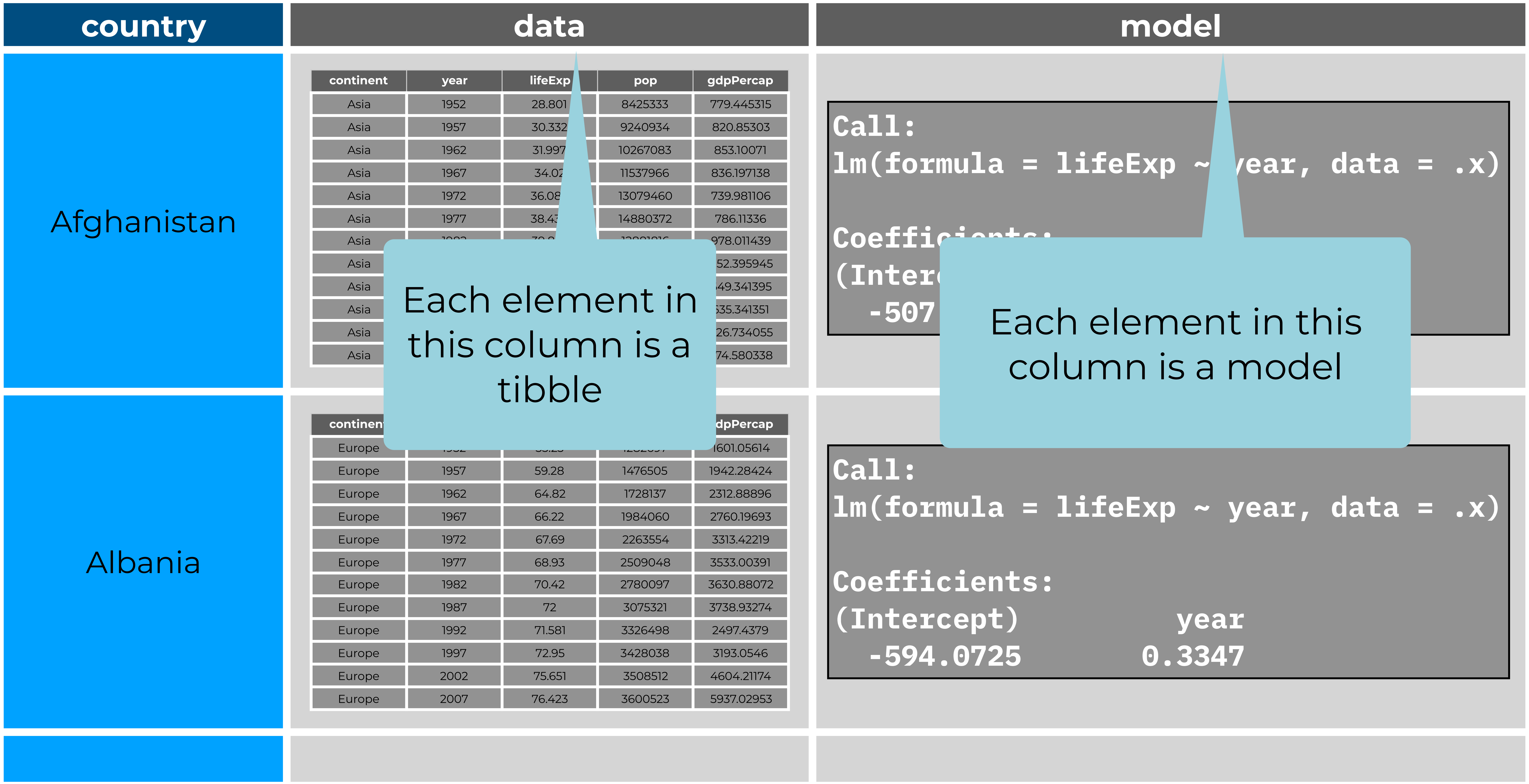
Adapted from Data Science in the tidyverse, CC BY Amelia McNamara

Yes

```
tibble(
  num = c(1, 2, 3),
  cha = c("one", "two", "three"),
  listcol = list(1, c("1", "two", "FALSE"), FALSE)
)
# A tibble: 3 x 3
    num cha   listcol
  <dbl> <chr> <list>
1     1 one   <dbl [1]>
2     2 two   <chr [3]>
3     3 three <lgl [1]>
```

Adapted from Master the Tidyverse, CC BY RStudio

Goal



country	data	model
Afghanistan	tibble	model
Albania	tibble	model

continent	year	lifeExp	pop	gdpPercap
Asia	1952	28.801	8425333	779.445315
Asia	1957	30.332	9240934	820.85303
Asia	1962	31.997	10267083	853.10071
Asia	1967	34.02	11537966	836.197138
Asia	1972	36.08	13079460	739.981106
Asia	1977	38.43	14880372	786.11336
Asia	[illegible]	[illegible]	[illegible]	978.011439
Asia	[illegible]	[illegible]	[illegible]	52.395945
Asia	[illegible]	[illegible]	[illegible]	49.341395
Asia	[illegible]	[illegible]	[illegible]	35.341351
Asia	[illegible]	[illegible]	[illegible]	26.734055
Asia	[illegible]	[illegible]	[illegible]	74.580338

Each element in this column is a tibble

continen				dpPercap
Europe	[illegible]	[illegible]	[illegible]	1601.05614
Europe	1957	59.28	1476505	1942.28424
Europe	1962	64.82	1728137	2312.88896
Europe	1967	66.22	1984060	2760.19693
Europe	1972	67.69	2263554	3313.42219
Europe	1977	68.93	2509048	3533.00391
Europe	1982	70.42	2780097	3630.88072
Europe	1987	72	3075321	3738.93274
Europe	1992	71.581	3326498	2497.4379
Europe	1997	72.95	3428038	3193.0546
Europe	2002	75.651	3508512	4604.21174
Europe	2007	76.423	3600523	5937.02953

```
Call:
lm(formula = lifeExp ~ year, data = .x)

Coefficients:
(Interc
  -507
```

Each element in this column is a model

```
Call:
lm(formula = lifeExp ~ year, data = .x)

Coefficients:
(Intercept)         year
  -594.0725       0.3347
```

Adapted from Master the Tidyverse, CC BY RStudio

Why?

country	data	model	r.squared
Afghanistan	see table below	see output below	0.034
Albania	see table below	see output below	0.493

Afghanistan data:

continent	year	lifeExp	pop	gdpPercap
Asia	1952	28.801	8425333	779.445315
Asia	1957	30.332	9240934	820.85303
Asia	1962	31.997	10267083	853.10071
Asia	1967	34.02	11537966	836.197138
Asia	1972	36.088	13079460	739.981106
Asia	1977	38.438	14880372	786.11336
Asia	1982	39.854	12881816	978.011439
Asia	1987	40.822	13867957	852.395945
Asia	1992	41.674	16317921	649.341395
Asia	1997	41.763	22227415	635.341351
Asia	2002	42.129	25268405	726.734055
Asia	2007	43.828	31889923	974.580338

Afghanistan model:

```
Call:
lm(formula = lifeExp ~ year, data = .x)

Coefficients:
(Intercept)         year
  -507.5343       0.2753
```

Albania data:

contin				
Europ				
Europ				
Europ				
Europe	1967	66.22	1984060	2760.19693
Europe	1972	67.69	2263554	3313.42219
Europe	1977	68.93	2509048	3533.00391
Europe	1982	70.42	2780097	3630.88072
Europe	1987	72	3075321	3738.93274
Europe	1992	71.581	3326498	2497.4379
Europe	1997	72.95	3428038	3193.0546
Europe	2002	75.651	3508512	4604.21174
Europe	2007	76.423	3600523	5937.02953

Albania model:

```
lm(formula = lifeExp ~ year, data = .x)

Coefficients:
(Intercept)         year
  -594.0725       0.3347
```

Organization.

We keep think that are related together.

Adapted from Master the Tidyverse, CC BY RStudio

Nesting

nest()

Nest rows into a list column by group.

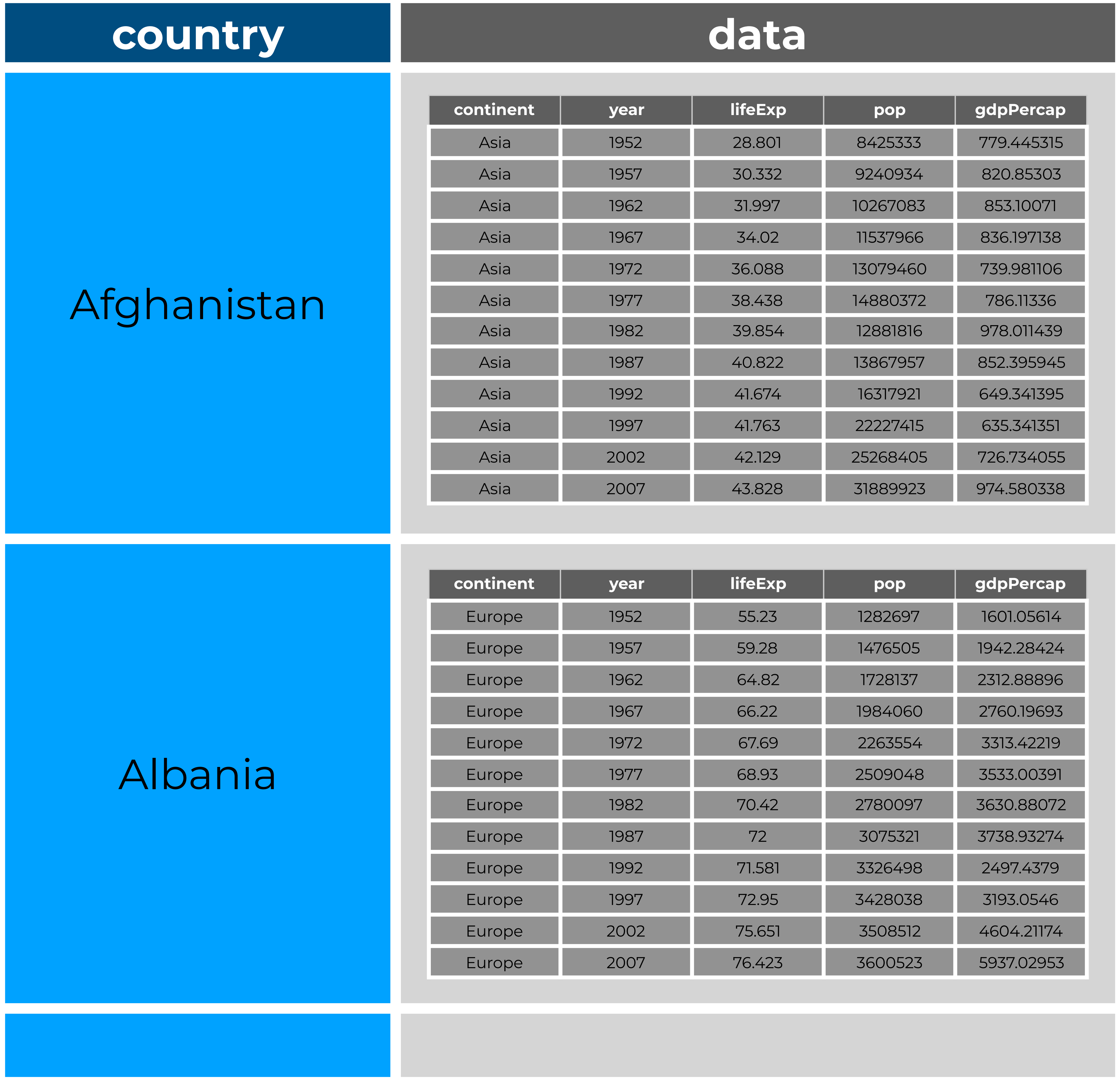
```
nest(.data, ...)
```

A grouped data frame

Adapted from Master the Tidyverse, CC BY RStudio

Places grouped cases into a list column.

```
gapminder %>%
  group_by(country) %>%
  nest()
```



country	data
Afghanistan	see table below
Albania	see table below

Afghanistan

continent	year	lifeExp	pop	gdpPercap
Asia	1952	28.801	8425333	779.445315
Asia	1957	30.332	9240934	820.85303
Asia	1962	31.997	10267083	853.10071
Asia	1967	34.02	11537966	836.197138
Asia	1972	36.088	13079460	739.981106
Asia	1977	38.438	14880372	786.11336
Asia	1982	39.854	12881816	978.011439
Asia	1987	40.822	13867957	852.395945
Asia	1992	41.674	16317921	649.341395
Asia	1997	41.763	22227415	635.341351
Asia	2002	42.129	25268405	726.734055
Asia	2007	43.828	31889923	974.580338

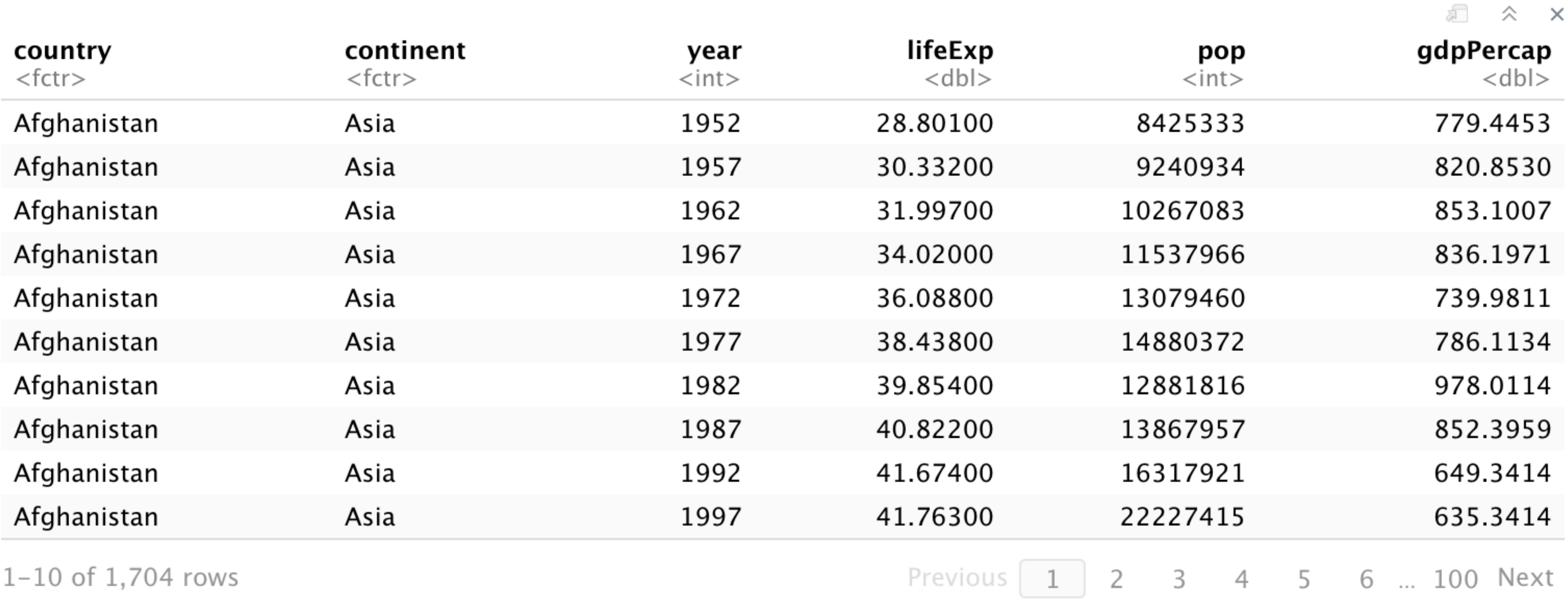
Albania

continent	year	lifeExp	pop	gdpPercap
Europe	1952	55.23	1282697	1601.05614
Europe	1957	59.28	1476505	1942.28424
Europe	1962	64.82	1728137	2312.88896
Europe	1967	66.22	1984060	2760.19693
Europe	1972	67.69	2263554	3313.42219
Europe	1977	68.93	2509048	3533.00391
Europe	1982	70.42	2780097	3630.88072
Europe	1987	72	3075321	3738.93274
Europe	1992	71.581	3326498	2497.4379
Europe	1997	72.95	3428038	3193.0546
Europe	2002	75.651	3508512	4604.21174
Europe	2007	76.423	3600523	5937.02953

ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

```
gapminder
```



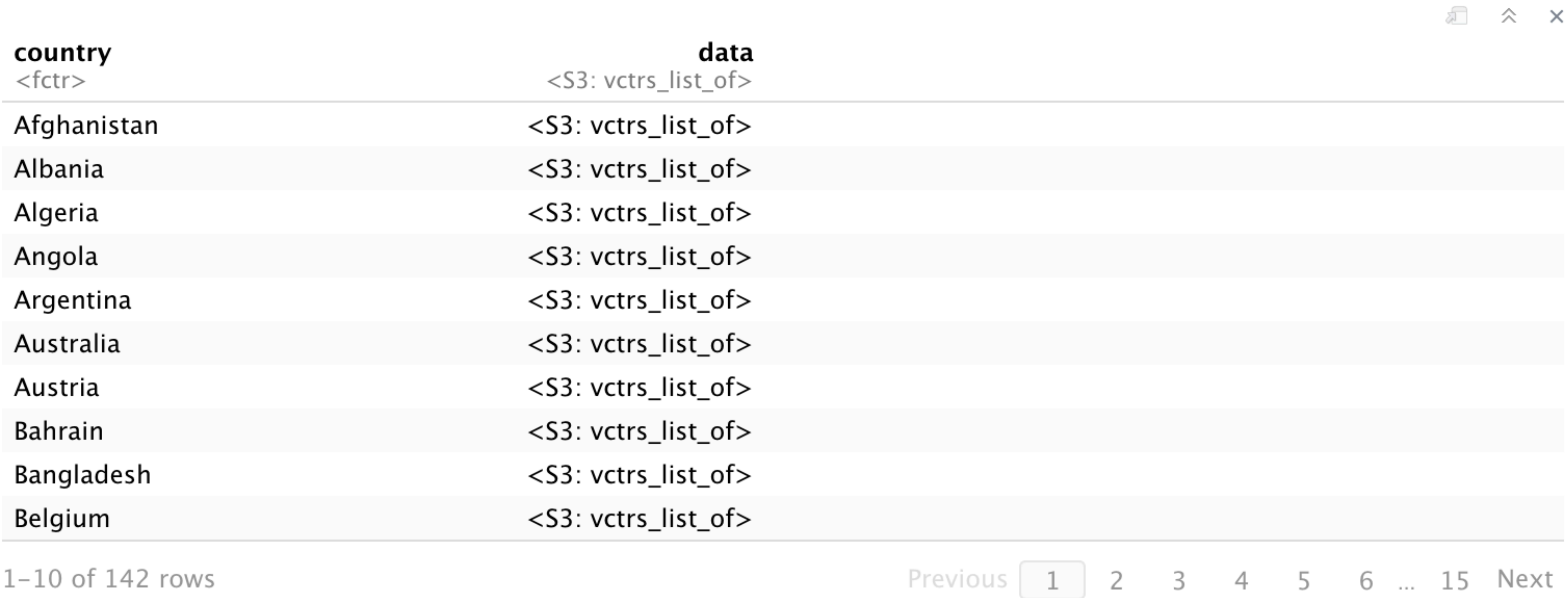
country	continent	year	lifeExp	pop	gdpPercap
<fctr>	<fctr>	<int>	<dbl>	<int>	<dbl>
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Afghanistan	Asia	1962	31.99700	10267083	853.1007
Afghanistan	Asia	1967	34.02000	11537966	836.1971
Afghanistan	Asia	1972	36.08800	13079460	739.9811
Afghanistan	Asia	1977	38.43800	14880372	786.1134
Afghanistan	Asia	1982	39.85400	12881816	978.0114
Afghanistan	Asia	1987	40.82200	13867957	852.3959
Afghanistan	Asia	1992	41.67400	16317921	649.3414
Afghanistan	Asia	1997	41.76300	22227415	635.3414

1–10 of 1,704 rows

ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

```
gapminder %>%
  group_by(country) %>%
  nest()
```



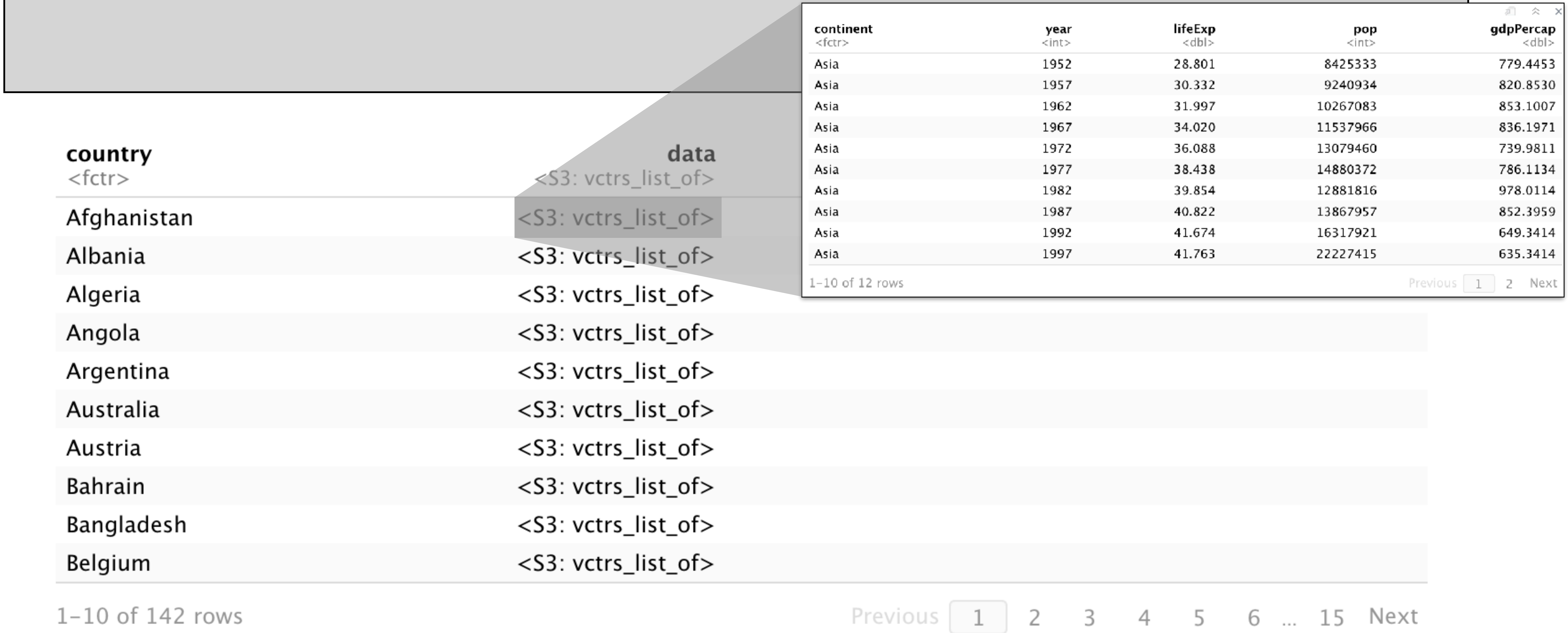
country <fctr>	data <S3: vctrs_list_of>
Afghanistan	<S3: vctrs_list_of>
Albania	<S3: vctrs_list_of>
Algeria	<S3: vctrs_list_of>
Angola	<S3: vctrs_list_of>
Argentina	<S3: vctrs_list_of>
Australia	<S3: vctrs_list_of>
Austria	<S3: vctrs_list_of>
Bahrain	<S3: vctrs_list_of>
Bangladesh	<S3: vctrs_list_of>
Belgium	<S3: vctrs_list_of>

1–10 of 142 rows

ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

```r
gapminder_nested$data[[1]]
```



continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>
Asia	1952	28.801	8425333	779.4453
Asia	1957	30.332	9240934	820.8530
Asia	1962	31.997	10267083	853.1007
Asia	1967	34.020	11537966	836.1971
Asia	1972	36.088	13079460	739.9811
Asia	1977	38.438	14880372	786.1134
Asia	1982	39.854	12881816	978.0114
Asia	1987	40.822	13867957	852.3959
Asia	1992	41.674	16317921	649.3414
Asia	1997	41.763	22227415	635.3414

1-10 of 12 rows

Previous 1 2 Next

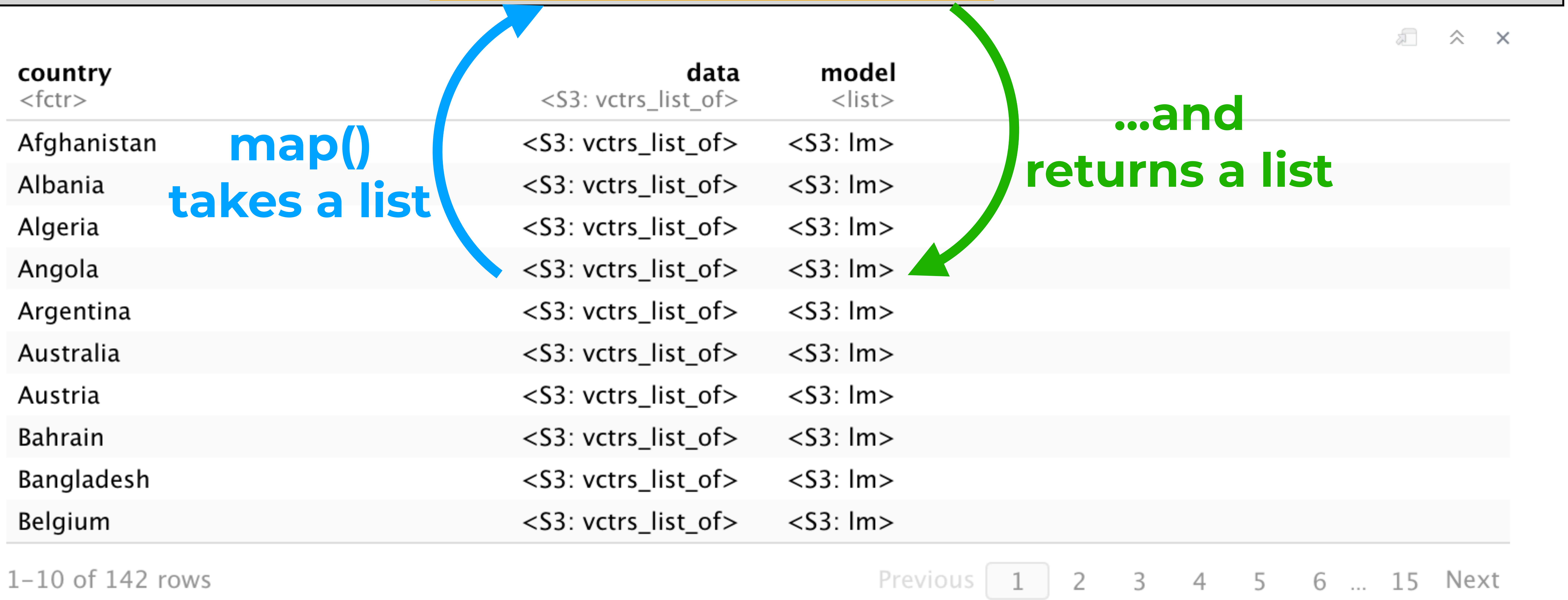
country <fctr>	data <S3: vctrs_list_of>
Afghanistan	<S3: vctrs_list_of>
Albania	<S3: vctrs_list_of>
Algeria	<S3: vctrs_list_of>
Angola	<S3: vctrs_list_of>
Argentina	<S3: vctrs_list_of>
Australia	<S3: vctrs_list_of>
Austria	<S3: vctrs_list_of>
Bahrain	<S3: vctrs_list_of>
Bangladesh	<S3: vctrs_list_of>
Belgium	<S3: vctrs_list_of>

1–10 of 142 rows

Previous 1 2 3 4 5 6 ... 15 Next

Adapted from Master the Tidyverse, CC BY RStudio

```r
fit_model <- function(df) lm(lifeExp ~ year, data = df)

gapminder_nested <- gapminder_nested %>%
  mutate(model = map(data, fit_model))
```



country <fctr>	data <S3: vctrs_list_of>	model <list>
Afghanistan	<S3: vctrs_list_of>	<S3: lm>
Albania	<S3: vctrs_list_of>	<S3: lm>
Algeria	<S3: vctrs_list_of>	<S3: lm>
Angola	<S3: vctrs_list_of>	<S3: lm>
Argentina	<S3: vctrs_list_of>	<S3: lm>
Australia	<S3: vctrs_list_of>	<S3: lm>
Austria	<S3: vctrs_list_of>	<S3: lm>
Bahrain	<S3: vctrs_list_of>	<S3: lm>
Bangladesh	<S3: vctrs_list_of>	<S3: lm>
Belgium	<S3: vctrs_list_of>	<S3: lm>

1-10 of 142 rows

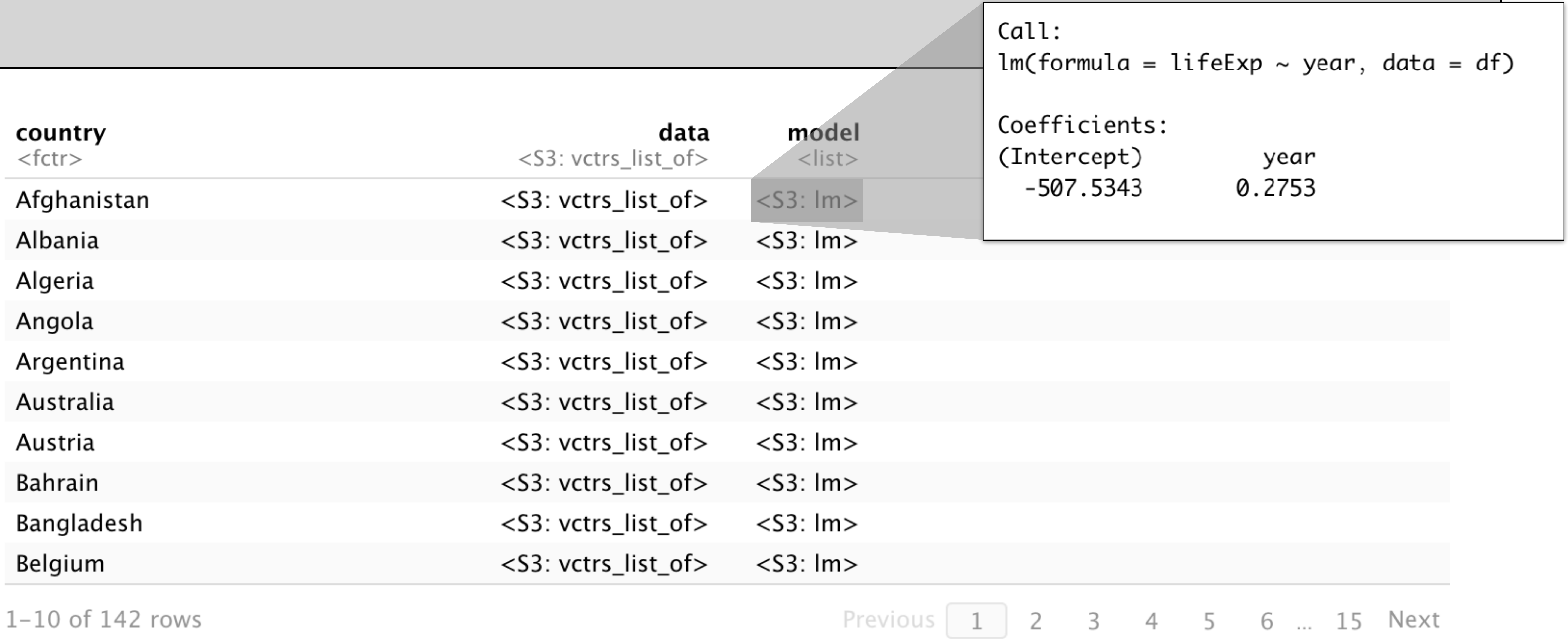
Previous 1 2 3 4 5 6 ... 15 Next

map() takes a list

...and returns a list

Adapted from Master the Tidyverse, CC BY RStudio

```
gapminder_nested$model[[1]]
```



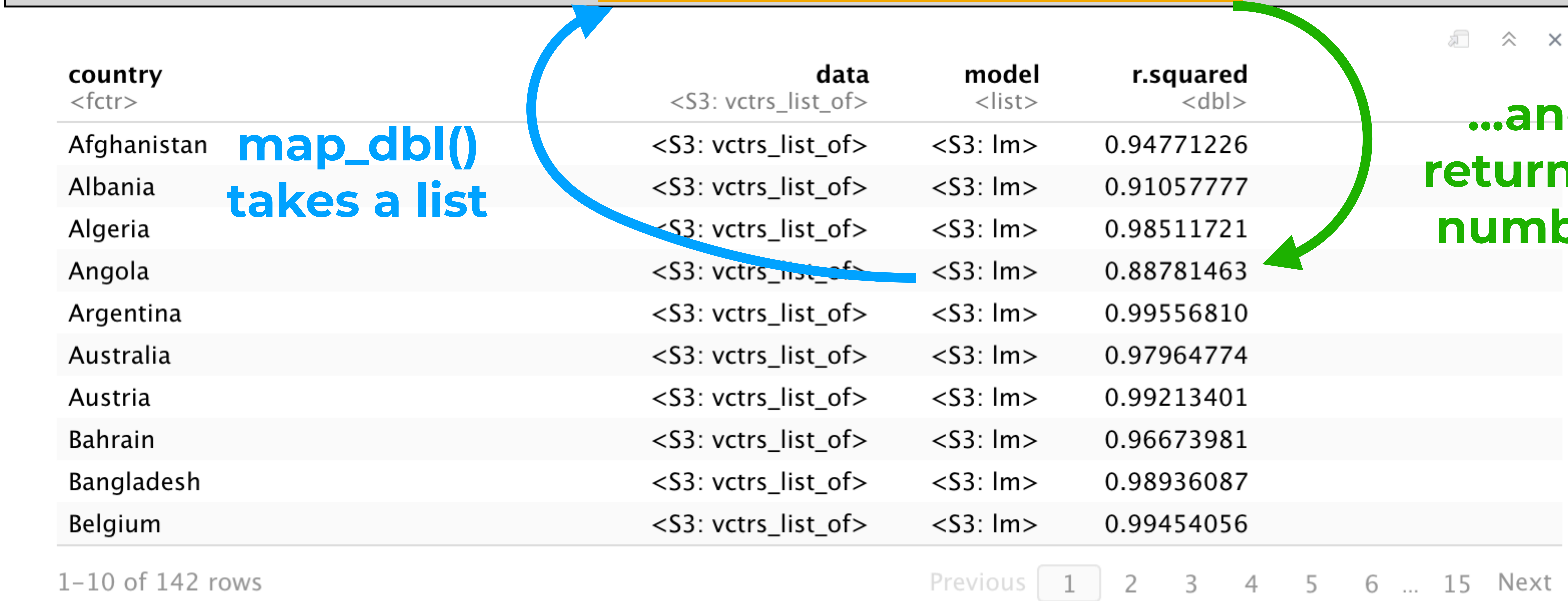
country <fctr>	data <S3: vctrs_list_of>	model <list>
Afghanistan	<S3: vctrs_list_of>	<S3: lm>
Albania	<S3: vctrs_list_of>	<S3: lm>
Algeria	<S3: vctrs_list_of>	<S3: lm>
Angola	<S3: vctrs_list_of>	<S3: lm>
Argentina	<S3: vctrs_list_of>	<S3: lm>
Australia	<S3: vctrs_list_of>	<S3: lm>
Austria	<S3: vctrs_list_of>	<S3: lm>
Bahrain	<S3: vctrs_list_of>	<S3: lm>
Bangladesh	<S3: vctrs_list_of>	<S3: lm>
Belgium	<S3: vctrs_list_of>	<S3: lm>

1–10 of 142 rows

```
Call:
lm(formula = lifeExp ~ year, data = df)

Coefficients:
(Intercept)         year
  -507.5343       0.2753
```

Adapted from Master the Tidyverse, CC BY RStudio

```r
get_rsq <- function(mod) glance(mod)$r.squared

gapminder_nested <- gapminder_nested %>%
  mutate(r.squared = map_dbl(model, get_rsq))
```



country <fctr>	data <S3: vctrs_list_of>	model <list>	r.squared <dbl>
Afghanistan	<S3: vctrs_list_of>	<S3: lm>	0.94771226
Albania	<S3: vctrs_list_of>	<S3: lm>	0.91057777
Algeria	<S3: vctrs_list_of>	<S3: lm>	0.98511721
Angola	<S3: vctrs_list_of>	<S3: lm>	0.88781463
Argentina	<S3: vctrs_list_of>	<S3: lm>	0.99556810
Australia	<S3: vctrs_list_of>	<S3: lm>	0.97964774
Austria	<S3: vctrs_list_of>	<S3: lm>	0.99213401
Bahrain	<S3: vctrs_list_of>	<S3: lm>	0.96673981
Bangladesh	<S3: vctrs_list_of>	<S3: lm>	0.98936087
Belgium	<S3: vctrs_list_of>	<S3: lm>	0.99454056

1–10 of 142 rows

map_dbl() takes a list

...and returns a number

ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

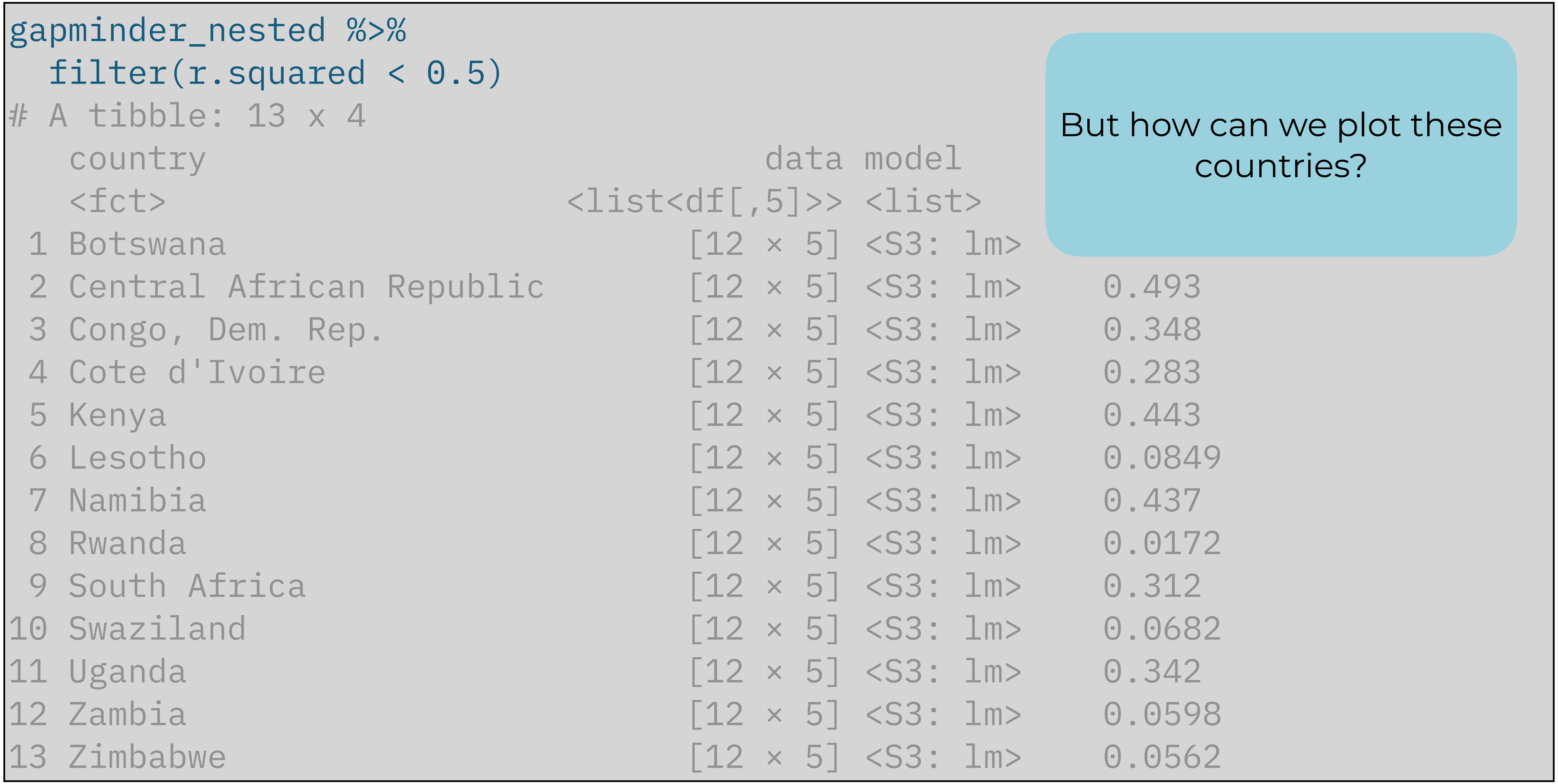
Your Turn 2

Run the chunk, then filter **gapminder_nested** to find the countries with **r.squared** less than 0.5.

02 : 00

ATLAS

Adapted from Data Science in the tidyverse, CC BY Amelia McNamara



```
gapminder_nested %>%
  filter(r.squared < 0.5)
# A tibble: 13 x 4
   country                                data model
   <fct>                       <list<df[,5]>> <list>
 1 Botswana                          [12 × 5] <S3: lm>
 2 Central African Republic          [12 × 5] <S3: lm>    0.493
 3 Congo, Dem. Rep.                  [12 × 5] <S3: lm>    0.348
 4 Cote d'Ivoire                     [12 × 5] <S3: lm>    0.283
 5 Kenya                             [12 × 5] <S3: lm>    0.443
 6 Lesotho                           [12 × 5] <S3: lm>    0.0849
 7 Namibia                           [12 × 5] <S3: lm>    0.437
 8 Rwanda                            [12 × 5] <S3: lm>    0.0172
 9 South Africa                      [12 × 5] <S3: lm>    0.312
10 Swaziland                         [12 × 5] <S3: lm>    0.0682
11 Uganda                            [12 × 5] <S3: lm>    0.342
12 Zambia                            [12 × 5] <S3: lm>    0.0598
13 Zimbabwe                          [12 × 5] <S3: lm>    0.0562
```

But how can we plot these countries?

Adapted from Master the Tidyverse, CC BY RStudio

unnest()

Flatten list columns into regular columns

```
unnest(data, cols, ...)
```

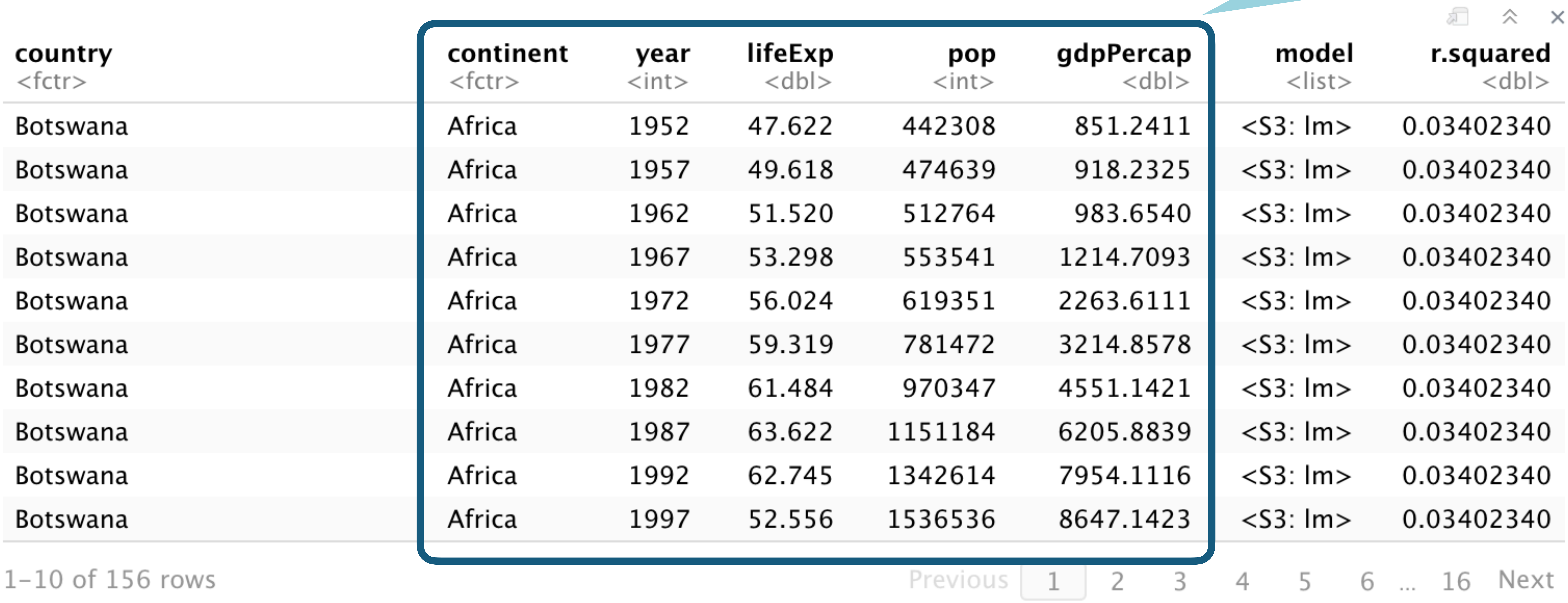
Nested data frame

Columns to unnest

Adapted from Master the Tidyverse, CC BY RStudio

```
poor_fit <- gapminder_nested %>%
  filter(r.squared < 0.5)

poor_fit %>% unnest(data)
```

Columns from inside **data**

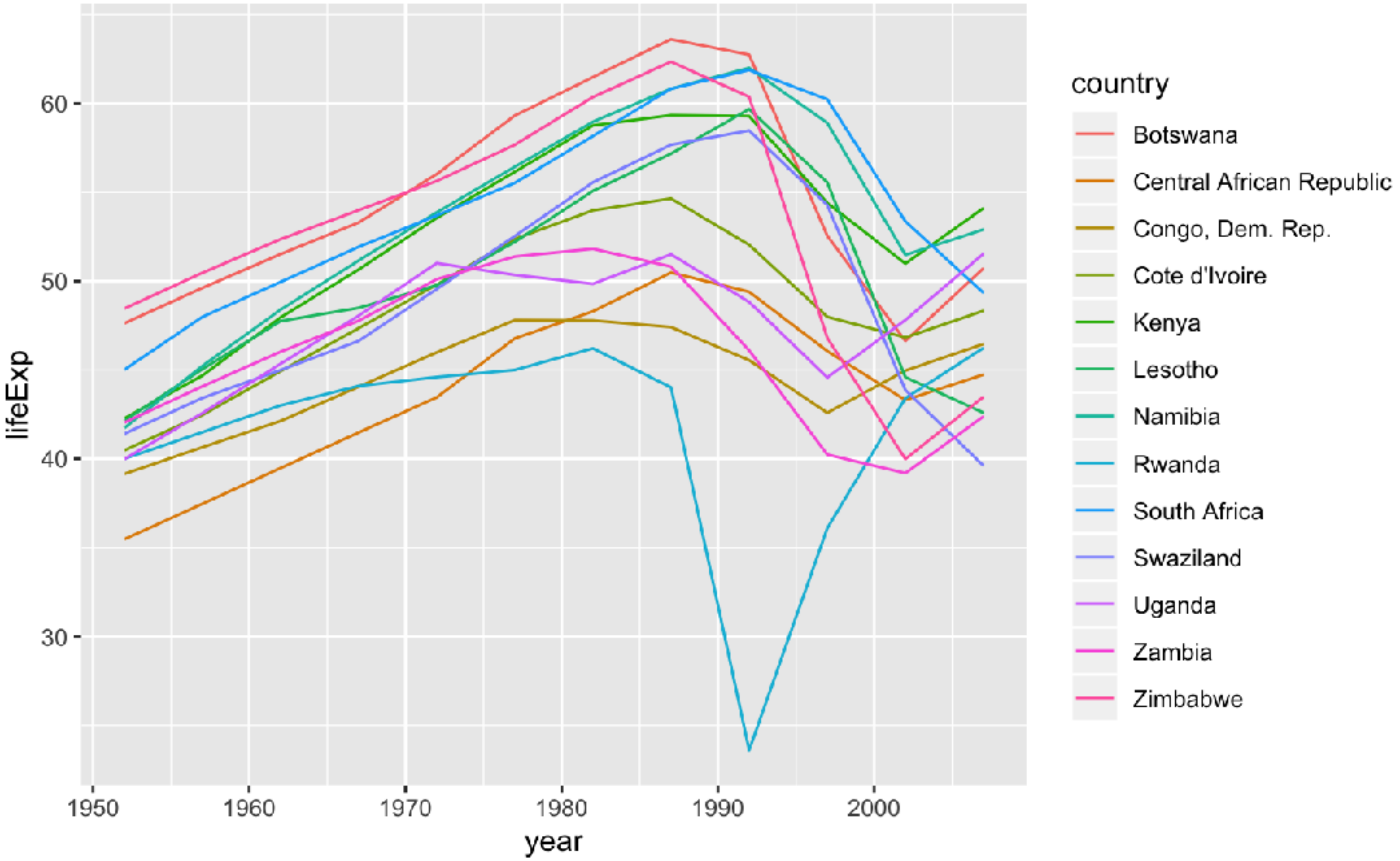


country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>	model <list>	r.squared <dbl>
Botswana	Africa	1952	47.622	442308	851.2411	<S3: lm>	0.03402340
Botswana	Africa	1957	49.618	474639	918.2325	<S3: lm>	0.03402340
Botswana	Africa	1962	51.520	512764	983.6540	<S3: lm>	0.03402340
Botswana	Africa	1967	53.298	553541	1214.7093	<S3: lm>	0.03402340
Botswana	Africa	1972	56.024	619351	2263.6111	<S3: lm>	0.03402340
Botswana	Africa	1977	59.319	781472	3214.8578	<S3: lm>	0.03402340
Botswana	Africa	1982	61.484	970347	4551.1421	<S3: lm>	0.03402340
Botswana	Africa	1987	63.622	1151184	6205.8839	<S3: lm>	0.03402340
Botswana	Africa	1992	62.745	1342614	7954.1116	<S3: lm>	0.03402340
Botswana	Africa	1997	52.556	1536536	8647.1423	<S3: lm>	0.03402340

1-10 of 156 rows

Adapted from Master the Tidyverse, CC BY RStudio

```
unnest(poor_fit, data) %>%
  ggplot(aes(x = year, y = lifeExp)) +
    geom_line(aes(color = country))
```



Adapted from Master the Tidyverse, CC BY RStudio

Your Turn 3

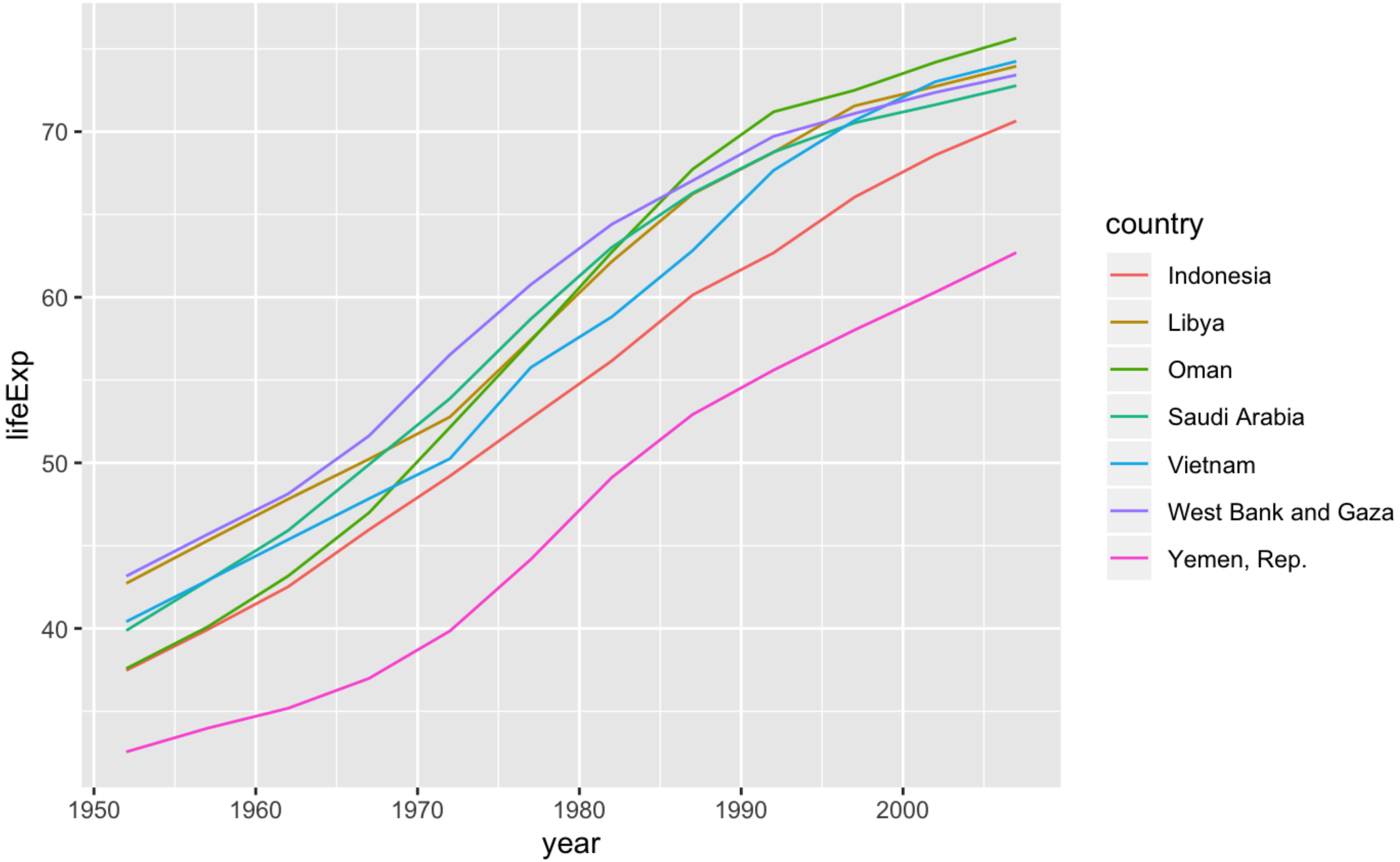
Edit the code in the chunk provided to instead find and plot countries with a slope above 0.6 years/year.

I've provided a **get_slope()** function:

```
get_slope <- function(mod) {
  tidy(mod) %>% filter(term == "year") %>% pull(estimate)
}
```

06:00

```
gapminder_nested <- gapminder_nested %>%
  mutate(slope = map_dbl(model, get_slope))

big_slope <- gapminder_nested %>%
  filter(slope > 0.6)

unnest(big_slope, data) %>%
  ggplot(aes(x = year, y = lifeExp)) +
    geom_line(aes(color = country))
```



ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

Recap

A table is...an organizational structure...that you can manipulate.

<table>
<tr><th>country</th><th>data</th><th>model</th><th>r.squared</th></tr>
<tr><td>Afghanistan</td><td>
<table>
<tr><th>continent</th><th>year</th><th>lifeExp</th><th>pop</th><th>gdpPercap</th></tr>
<tr><td>Asia</td><td>1952</td><td>28.801</td><td>8425333</td><td>779.445315</td></tr>
<tr><td>Asia</td><td>1957</td><td>30.332</td><td>9240934</td><td>820.85303</td></tr>
<tr><td>Asia</td><td>1962</td><td>31.997</td><td>10267083</td><td>853.10071</td></tr>
<tr><td>Asia</td><td>1967</td><td>34.02</td><td>11537966</td><td>836.197138</td></tr>
<tr><td>Asia</td><td>1972</td><td>36.088</td><td>13079460</td><td>739.981106</td></tr>
<tr><td>Asia</td><td>1977</td><td>38.438</td><td>14880372</td><td>786.11336</td></tr>
<tr><td>Asia</td><td>1982</td><td>39.854</td><td>12881816</td><td>978.011439</td></tr>
<tr><td>Asia</td><td>1987</td><td>40.822</td><td>13867957</td><td>852.395945</td></tr>
<tr><td>Asia</td><td>1992</td><td>41.674</td><td>16317921</td><td>649.341395</td></tr>
<tr><td>Asia</td><td>1997</td><td>41.763</td><td>22227415</td><td>635.341351</td></tr>
<tr><td>Asia</td><td>2002</td><td>42.129</td><td>25268405</td><td>726.734055</td></tr>
<tr><td>Asia</td><td>2007</td><td>43.828</td><td>31889923</td><td>974.580338</td></tr>
</table>
</td><td>Call:
lm(formula = lifeExp ~ year, data = .x)

Coefficients:
(Intercept)   year
  -507.5343   0.2753</td><td>0.034</td></tr>
<tr><td>Albania</td><td>
<table>
<tr><th>continent</th><th>year</th><th>lifeExp</th><th>pop</th><th>gdpPercap</th></tr>
<tr><td>Europe</td><td>1952</td><td>55.23</td><td>1282697</td><td>1601.05614</td></tr>
<tr><td>Europe</td><td>1957</td><td>59.28</td><td>1476505</td><td>1942.28424</td></tr>
<tr><td>Europe</td><td>1962</td><td>64.82</td><td>1728137</td><td>2312.88896</td></tr>
<tr><td>Europe</td><td>1967</td><td>66.22</td><td>1984060</td><td>2760.19693</td></tr>
<tr><td>Europe</td><td>1972</td><td>67.69</td><td>2263554</td><td>3313.42219</td></tr>
<tr><td>Europe</td><td>1977</td><td>68.93</td><td>2509048</td><td>3533.00391</td></tr>
<tr><td>Europe</td><td>1982</td><td>70.42</td><td>2780097</td><td>3630.88072</td></tr>
<tr><td>Europe</td><td>1987</td><td>72</td><td>3075321</td><td>3738.93274</td></tr>
<tr><td>Europe</td><td>1992</td><td>71.581</td><td>3326498</td><td>2497.4379</td></tr>
<tr><td>Europe</td><td>1997</td><td>72.95</td><td>3428038</td><td>3193.0546</td></tr>
<tr><td>Europe</td><td>2002</td><td>75.651</td><td>3508512</td><td>4604.21174</td></tr>
<tr><td>Europe</td><td>2007</td><td>76.423</td><td>3600523</td><td>5937.02953</td></tr>
</table>
</td><td>Call:
lm(formula = lifeExp ~ year, data = .x)

Coefficients:
(Intercept)   year
  -594.0725   0.3347</td><td>0.493</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

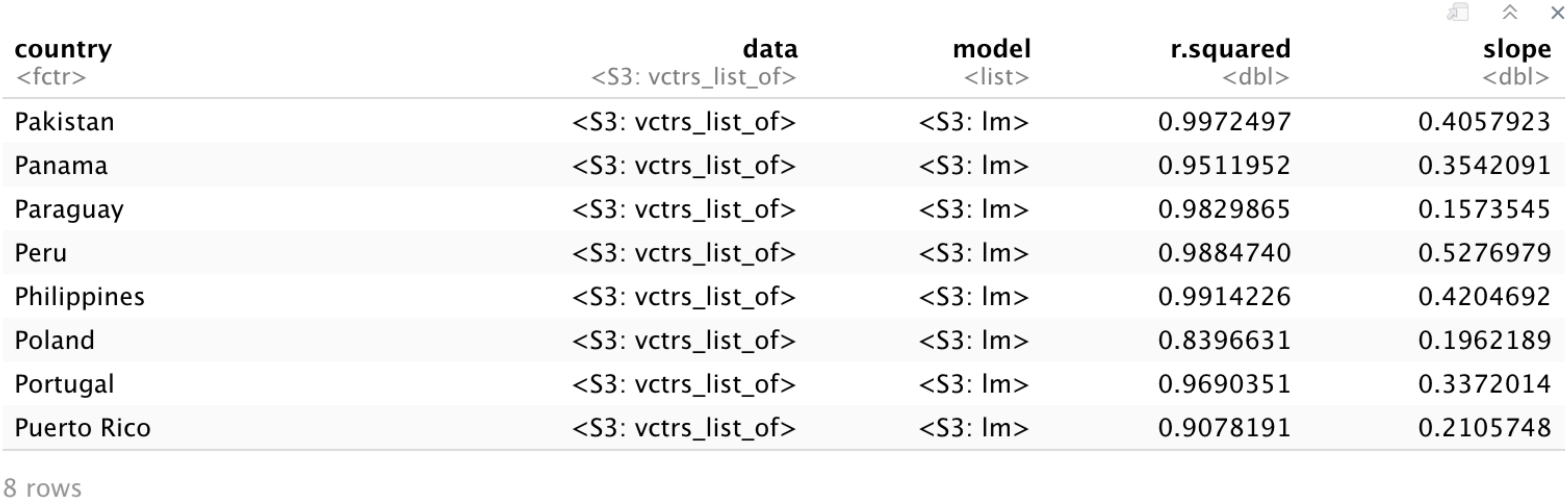
ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

Benefits

Data and models stay in correspondence across manipulations

```
gapminder_nested %>% filter(str_sub(country, 1, 1) == "P")
```



country <fctr>	data <S3: vctrs_list_of>	model <list>	r.squared <dbl>	slope <dbl>
Pakistan	<S3: vctrs_list_of>	<S3: lm>	0.9972497	0.4057923
Panama	<S3: vctrs_list_of>	<S3: lm>	0.9511952	0.3542091
Paraguay	<S3: vctrs_list_of>	<S3: lm>	0.9829865	0.1573545
Peru	<S3: vctrs_list_of>	<S3: lm>	0.9884740	0.5276979
Philippines	<S3: vctrs_list_of>	<S3: lm>	0.9914226	0.4204692
Poland	<S3: vctrs_list_of>	<S3: lm>	0.8396631	0.1962189
Portugal	<S3: vctrs_list_of>	<S3: lm>	0.9690351	0.3372014
Puerto Rico	<S3: vctrs_list_of>	<S3: lm>	0.9078191	0.2105748

8 rows

Adapted from Master the Tidyverse, CC BY RStudio

Your Turn 4

Challenge:

1. Create your own copy of **gapminder_nested** and then add one more list column: **output** which contains the output of **augment()** for each model.

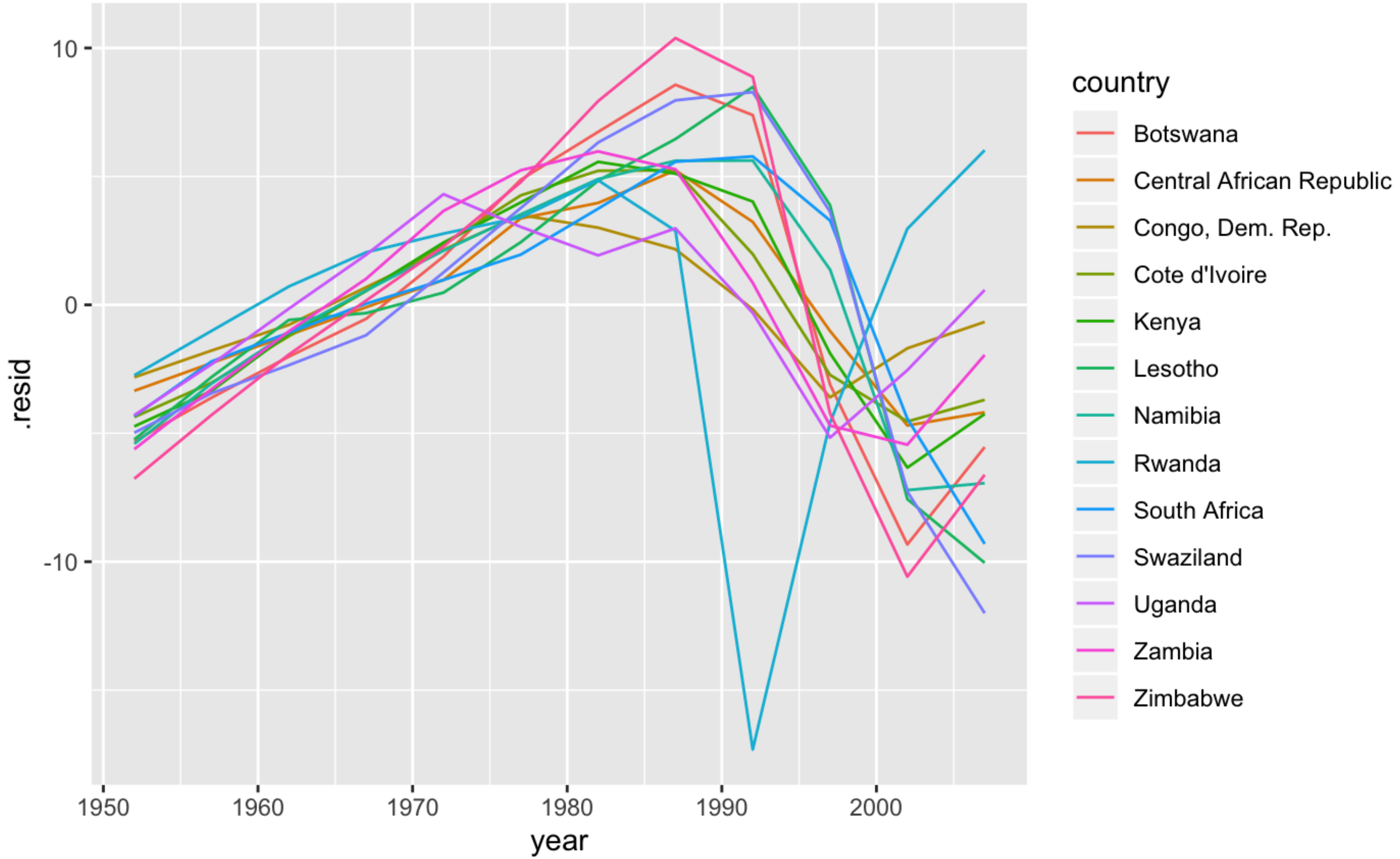
2. Plot the residuals against time for the countries with small r-squared.

06 : 00

ATLAS

Adapted from Master the Tidyverse, CC BY RStudio

```
jake_gapminder <- gapminder_nested

jake_gapminder %>%
  mutate(output = map(model, augment)) %>%
  filter(r.squared < 0.5) %>%
  unnest(output) %>%
  ggplot(aes(x = year, y = .resid)) +
    geom_line(aes(color = country))
```



Adapted from Master the Tidyverse, CC BY RStudio

Resampling

rsample
www.rstudio.com

Bootstrapping

id	x	y	z
1	0.73	-0.76	0.86
2	-0.24	0.59	0.93
3	-0.24	-1.81	0.46
4	-1.12	-0.17	1.71
5	0.21	-0.73	-1.25
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72
8	0.81	-0.76	0.93
9	-0.18	2.75	-0.14

id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
1	0.73	-0.76	0.86
3	-0.24	-1.81	0.46
5	0.21	-0.73	-1.25
7	-0.62	-0.57	-1.72
6	0.13	-1.41	1.73
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73

id	x	y	z
4	-1.12	-0.17	1.71
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
4	-1.12	-0.17	1.71
1	0.73	-0.76	0.86
5	0.21	-0.73	-1.25
7	-0.62	-0.57	-1.72
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72

...

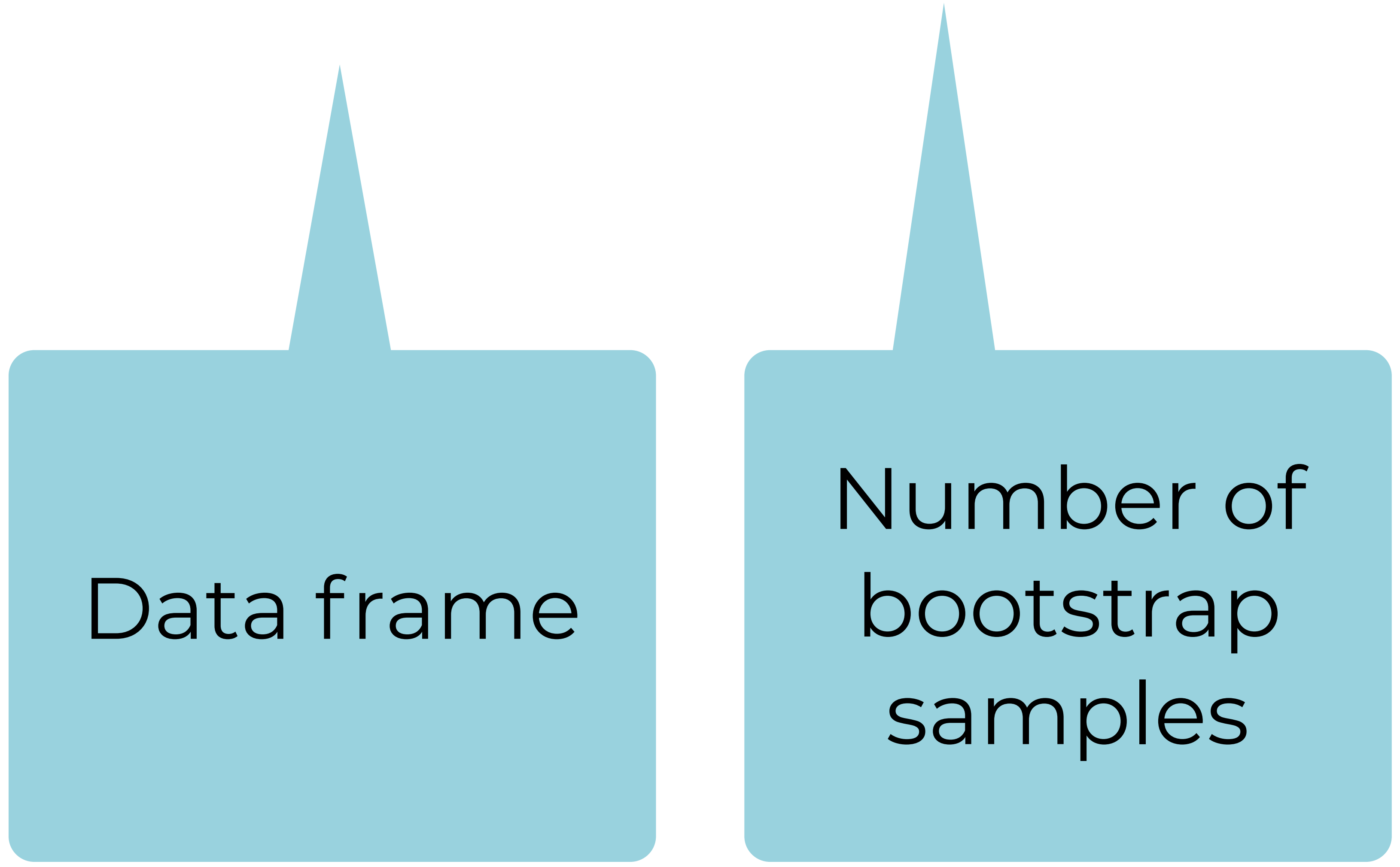
ATLAS

rsample

bootstraps()

Randomly same the data **with replacement**.

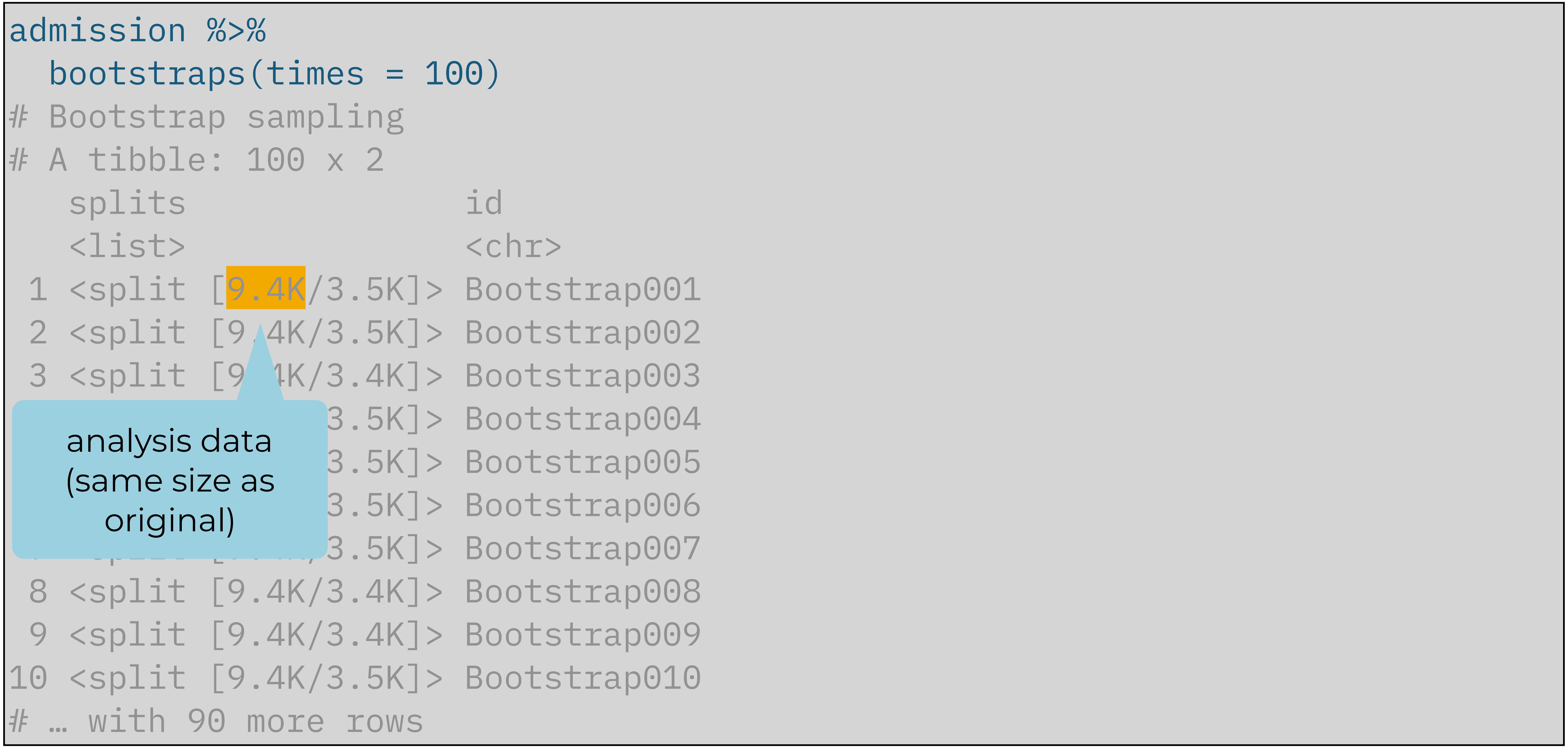
```
bootstraps(data, times = 25, ...)
```



Data frame

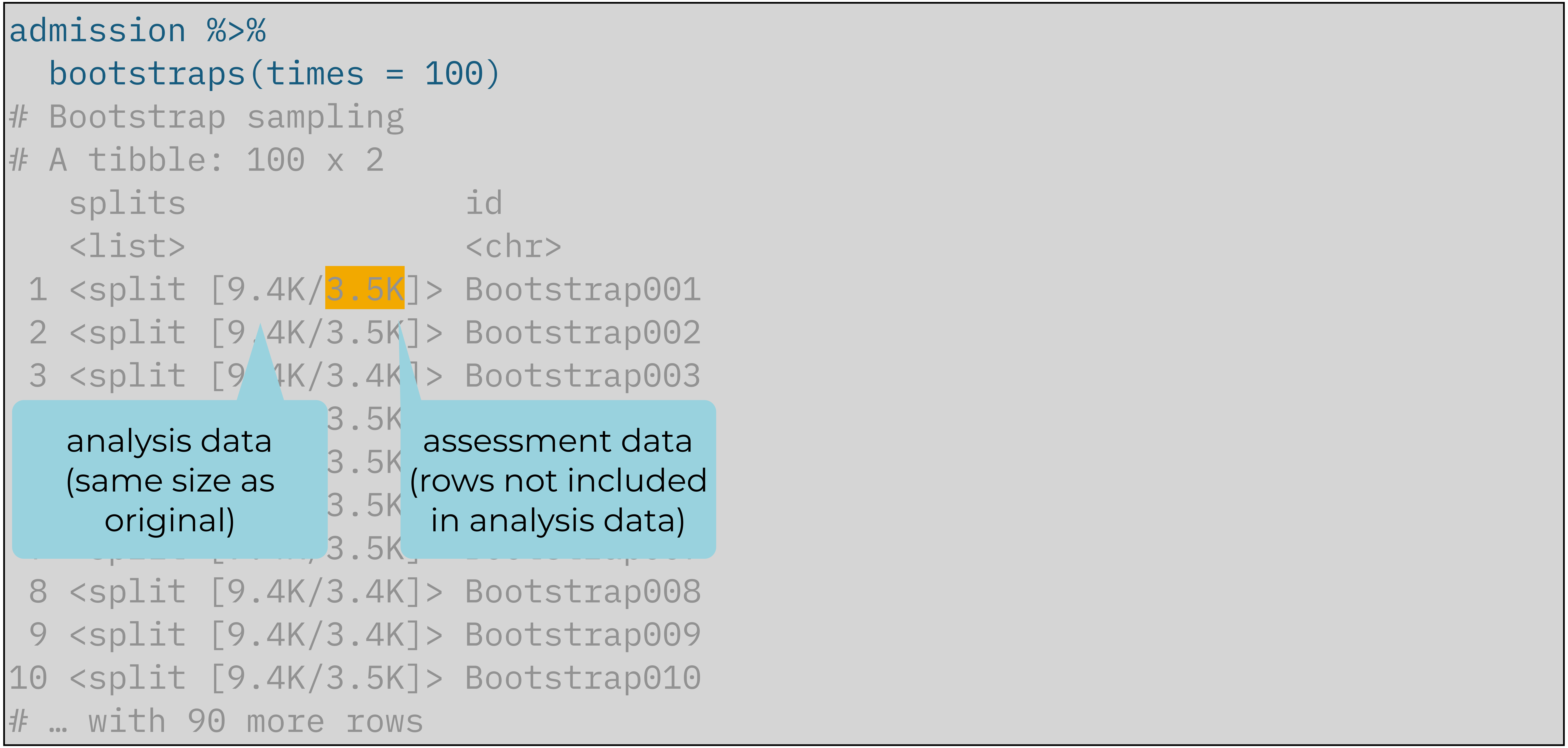
Number of bootstrap samples

```
admission %>%
  bootstraps(times = 100)
# Bootstrap sampling
# A tibble: 100 x 2
   splits               id
   <list>               <chr>
 1 <split [9.4K/3.5K]> Bootstrap001
 2 <split [9.4K/3.5K]> Bootstrap002
 3 <split [9.4K/3.4K]> Bootstrap003
 4 <split [9.4K/3.5K]> Bootstrap004
 5 <split [9.4K/3.5K]> Bootstrap005
 6 <split [9.4K/3.5K]> Bootstrap006
 7 <split [9.4K/3.5K]> Bootstrap007
 8 <split [9.4K/3.4K]> Bootstrap008
 9 <split [9.4K/3.4K]> Bootstrap009
10 <split [9.4K/3.5K]> Bootstrap010
# … with 90 more rows
```



```
admission %>%
  bootstraps(times = 100)
# Bootstrap sampling
# A tibble: 100 x 2
   splits             id
   <list>             <chr>
 1 <split [9.4K/3.5K]> Bootstrap001
 2 <split [9.4K/3.5K]> Bootstrap002
 3 <split [9.4K/3.4K]> Bootstrap003
                 3.5K]> Bootstrap004
                 3.5K]> Bootstrap005
                 3.5K]> Bootstrap006
                 3.5K]> Bootstrap007
 8 <split [9.4K/3.4K]> Bootstrap008
 9 <split [9.4K/3.4K]> Bootstrap009
10 <split [9.4K/3.5K]> Bootstrap010
# … with 90 more rows
```

analysis data (same size as original)



```
admission %>%
  bootstraps(times = 100)
# Bootstrap sampling
# A tibble: 100 x 2
   splits                id
   <list>                <chr>
 1 <split [9.4K/3.5K]> Bootstrap001
 2 <split [9.4K/3.5K]> Bootstrap002
 3 <split [9.4K/3.4K]> Bootstrap003
 8 <split [9.4K/3.4K]> Bootstrap008
 9 <split [9.4K/3.4K]> Bootstrap009
10 <split [9.4K/3.5K]> Bootstrap010
# … with 90 more rows
```

analysis data (same size as original)

assessment data (rows not included in analysis data)

```
models <- admission %>%
  bootstraps(times = 100)

models$splits[[1]]
# <9416/3418/9416>
```

Size of analysis data

Size of assessment data

Size of total data

```
admission %>%
  bootstraps(times = 100)
```

splits	id
<split [9.4K/3.5K]>	Bootstrap001
<split [9.4K/3.5K]>	Bootstrap002
<split [9.4K/3.4K]>	Bootstrap003

```
admission %>%
  bootstraps(times = 100) %>%
  mutate(model = map(splits, function(x) glm(admit ~ gender, data = analysis(x), family = binomial)))
```

splits	id	model
<split [9.4K/3.5K]>	Bootstrap001	Call: glm(formula = admit ~ gender, family = binomial, data = analysis(x)) Coefficients: (Intercept) genderFemale -0.7762 -0.4731 Degrees of Freedom: 9415 Total (i.e. Null); 9414 Residual Null Deviance: 11150 Residual Deviance: 11050 AIC: 11050
<split [9.4K/3.5K]>	Bootstrap002	Call: glm(formula = admit ~ gender, family = binomial, data = analysis(x)) Coefficients: (Intercept) genderFemale -0.7944 -0.4518 Degrees of Freedom: 9415 Total (i.e. Null); 9414 Residual Null Deviance: 11100 Residual Deviance: 11020 AIC: 11020
<split [9.4K/3.4K]>	Bootstrap003	Call: glm(formula = admit ~ gender, family = binomial, data = analysis(x)) Coefficients: (Intercept) genderFemale -0.7314 -0.4857 Degrees of Freedom: 9415 Total (i.e. Null); 9414 Residual Null Deviance: 11280 Residual Deviance: 11180 AIC: 11180

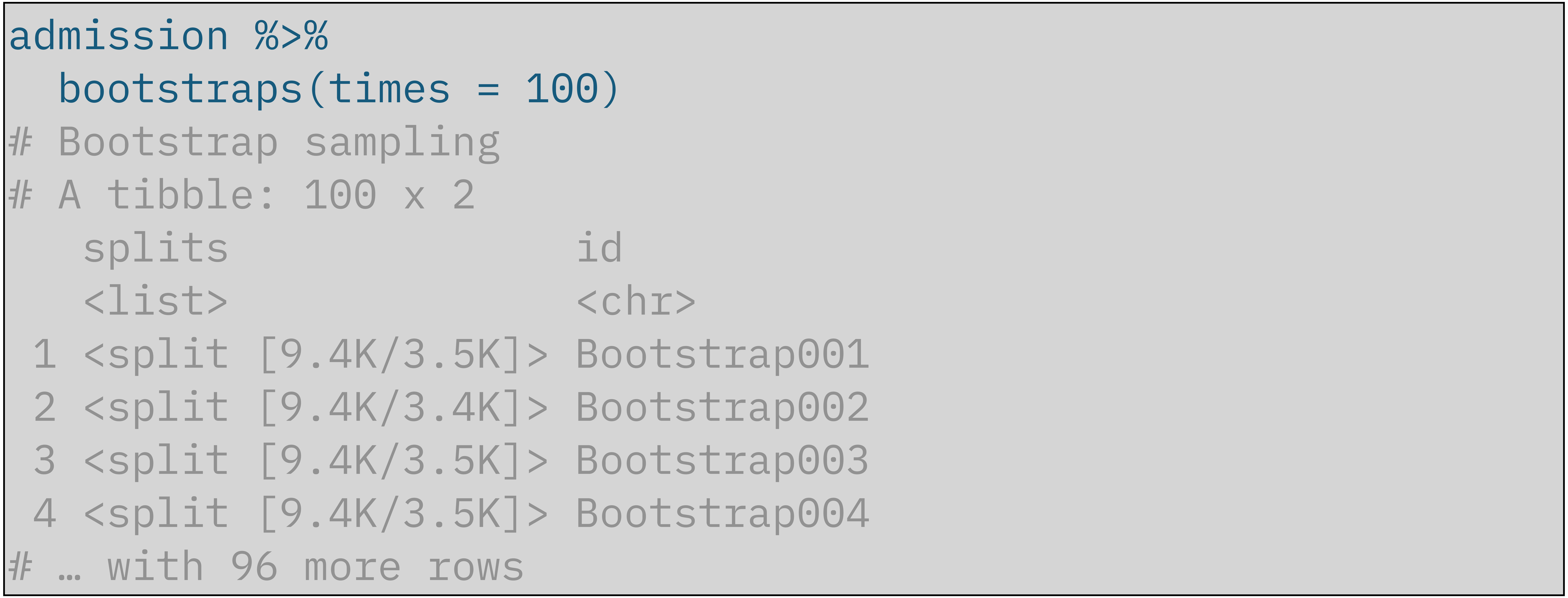
Bootstrapped Comparisons

1. Write a function to calculate the comparison of interest on your observed data
2. Create bootstrapped samples
3. Apply the function to each sample
4. Create a distribution of comparison value from each bootstrapped sample

1. Write a function to calculate the comparison of interest on your observed data

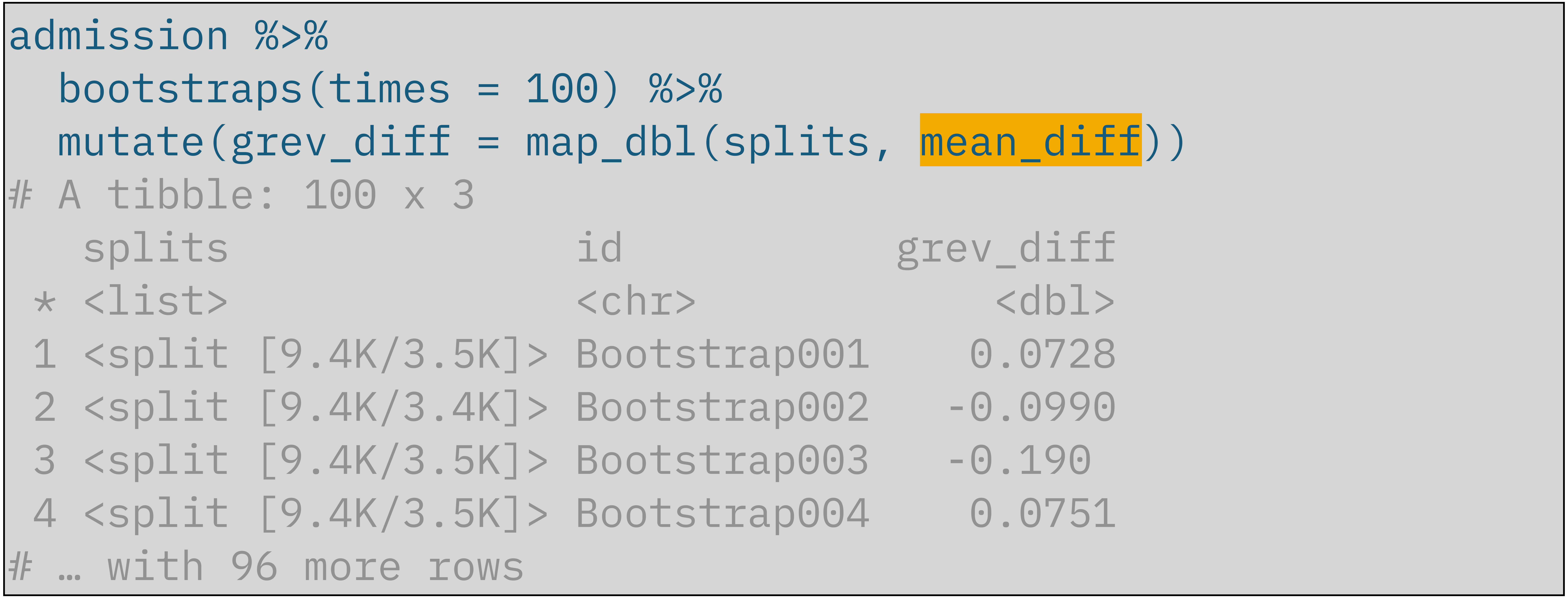
```
mean(admission$gre_v[admission$gender == "Male"]) -
  mean(admission$gre_v[admission$gender == "Female"])
[1] 0.02895073

mean_diff <- function(splits) {
  x <- analysis(splits)
  mean(x$gre_v[x$gender == "Male"]) -
    mean(x$gre_v[x$gender == "Female"])
}
```

2. Create bootstrapped samples



```
admission %>%
  bootstraps(times = 100)
# Bootstrap sampling
# A tibble: 100 x 2
   splits              id
   <list>              <chr>
 1 <split [9.4K/3.5K]> Bootstrap001
 2 <split [9.4K/3.4K]> Bootstrap002
 3 <split [9.4K/3.5K]> Bootstrap003
 4 <split [9.4K/3.5K]> Bootstrap004
# … with 96 more rows
```

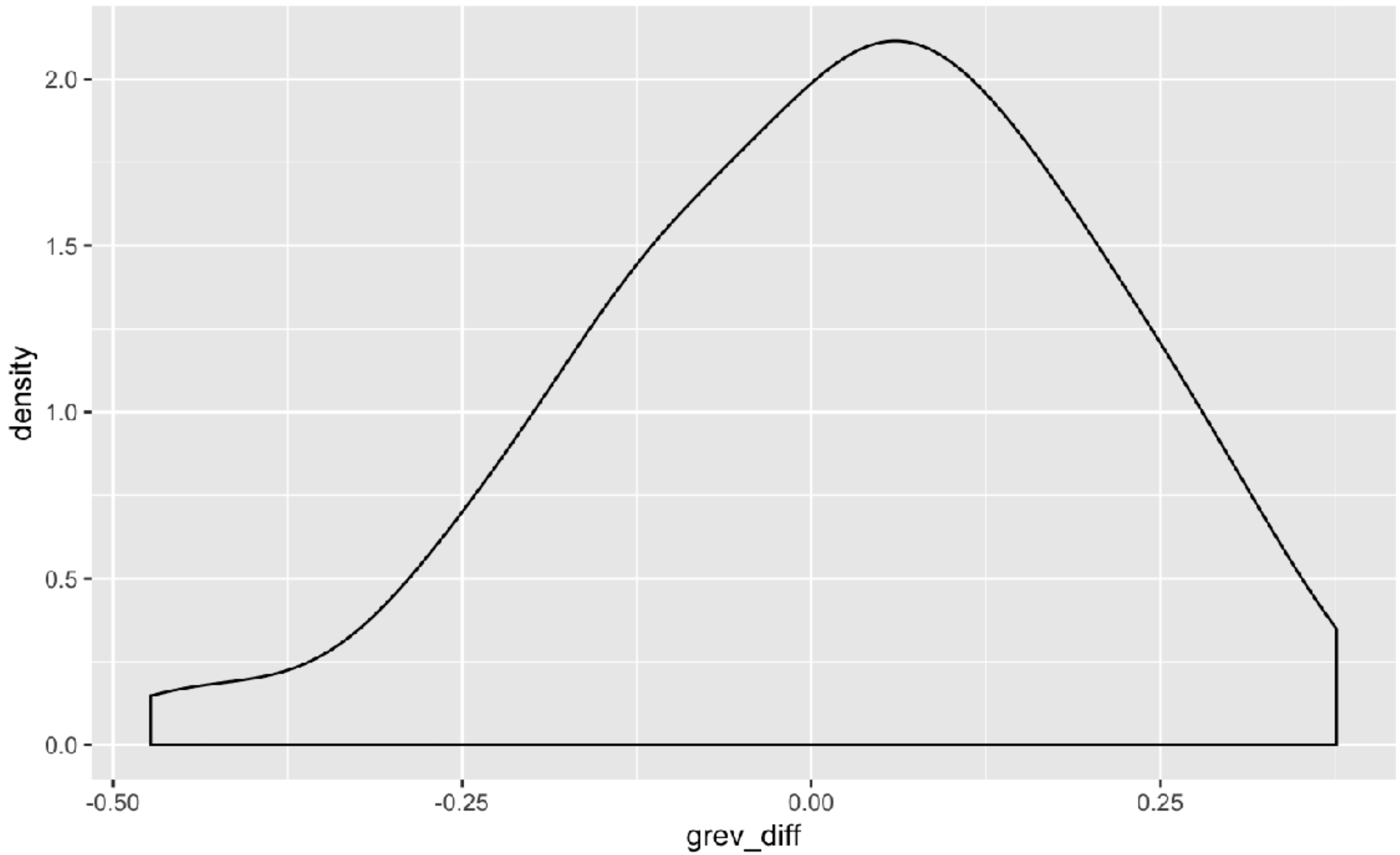
3. Apply the function to each sample



```
admission %>%
  bootstraps(times = 100) %>%
  mutate(grev_diff = map_dbl(splits, mean_diff))
# A tibble: 100 x 3
   splits              id           grev_diff
 * <list>              <chr>            <dbl>
 1 <split [9.4K/3.5K]> Bootstrap001    0.0728
 2 <split [9.4K/3.4K]> Bootstrap002   -0.0990
 3 <split [9.4K/3.5K]> Bootstrap003   -0.190
 4 <split [9.4K/3.5K]> Bootstrap004    0.0751
# … with 96 more rows
```

4. Create a distribution of comparison value from each bootstrapped sample

```
set.seed(32011)
grev_gender <- admission %>%
  bootstraps(times = 100) %>%
  mutate(grev_diff = map_dbl(splits, mean_diff))

ggplot(grev_gender, mapping = aes(x = grev_diff)) +
  geom_density()
```



```
quantile(grev_gender$grev_diff, probs = c(0.025, 0.500, 0.975))
       2.5%         50%       97.5% 
-0.34975576  0.04544041  0.32019353 
```

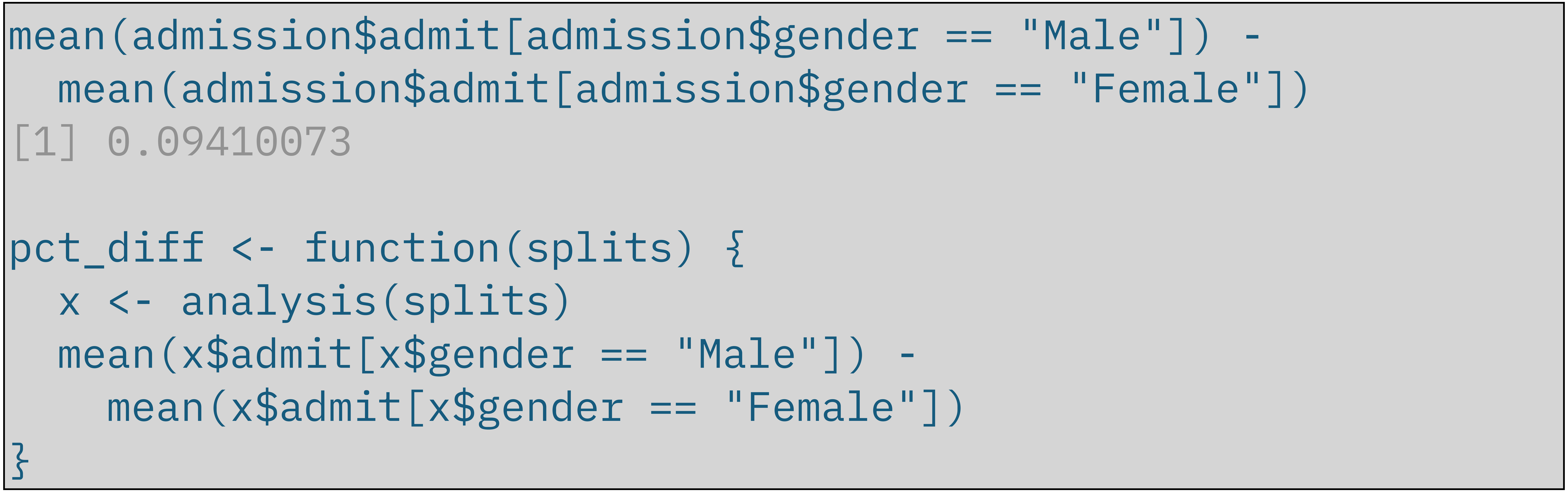
Your Turn 5

Is there a difference between the percentage of male and female applicants admitted?

- Modify the function to calculate the difference between the percentage of males admitted and the percentage of females admitted
- Apply the function to 100 bootstrap samples
- Create a density plot of the results

05:00

1. Write a function to calculate the comparison of interest on your observed data



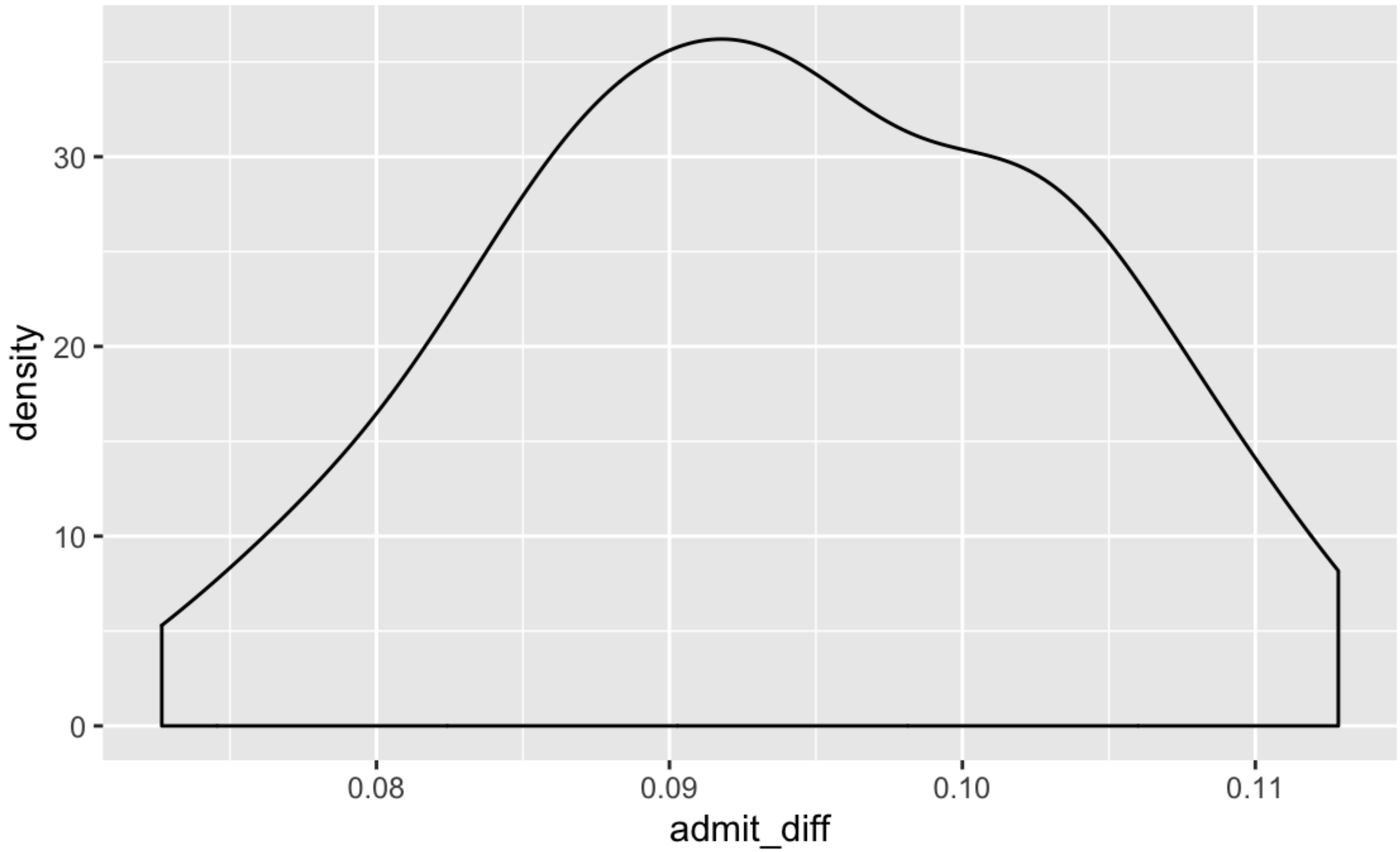
```
mean(admission$admit[admission$gender == "Male"]) -
  mean(admission$admit[admission$gender == "Female"])
[1] 0.09410073

pct_diff <- function(splits) {
  x <- analysis(splits)
  mean(x$admit[x$gender == "Male"]) -
    mean(x$admit[x$gender == "Female"])
}
```

2. Apply the function to each sample

```
admission %>%
  bootstraps(times = 100) %>%
  mutate(admit_diff = map_dbl(splits, pct_diff))
# A tibble: 100 x 3
   splits                   id           admit_diff
 * <list>                   <chr>             <dbl>
 1 <split [9.4K/3.5K]> Bootstrap001      0.0923
 2 <split [9.4K/3.5K]> Bootstrap002      0.0879
 3 <split [9.4K/3.4K]> Bootstrap003      0.0964
 4 <split [9.4K/3.5K]> Bootstrap004      0.0991
# … with 96 more rows
```

3. Create a distribution of comparison value from each bootstrapped sample

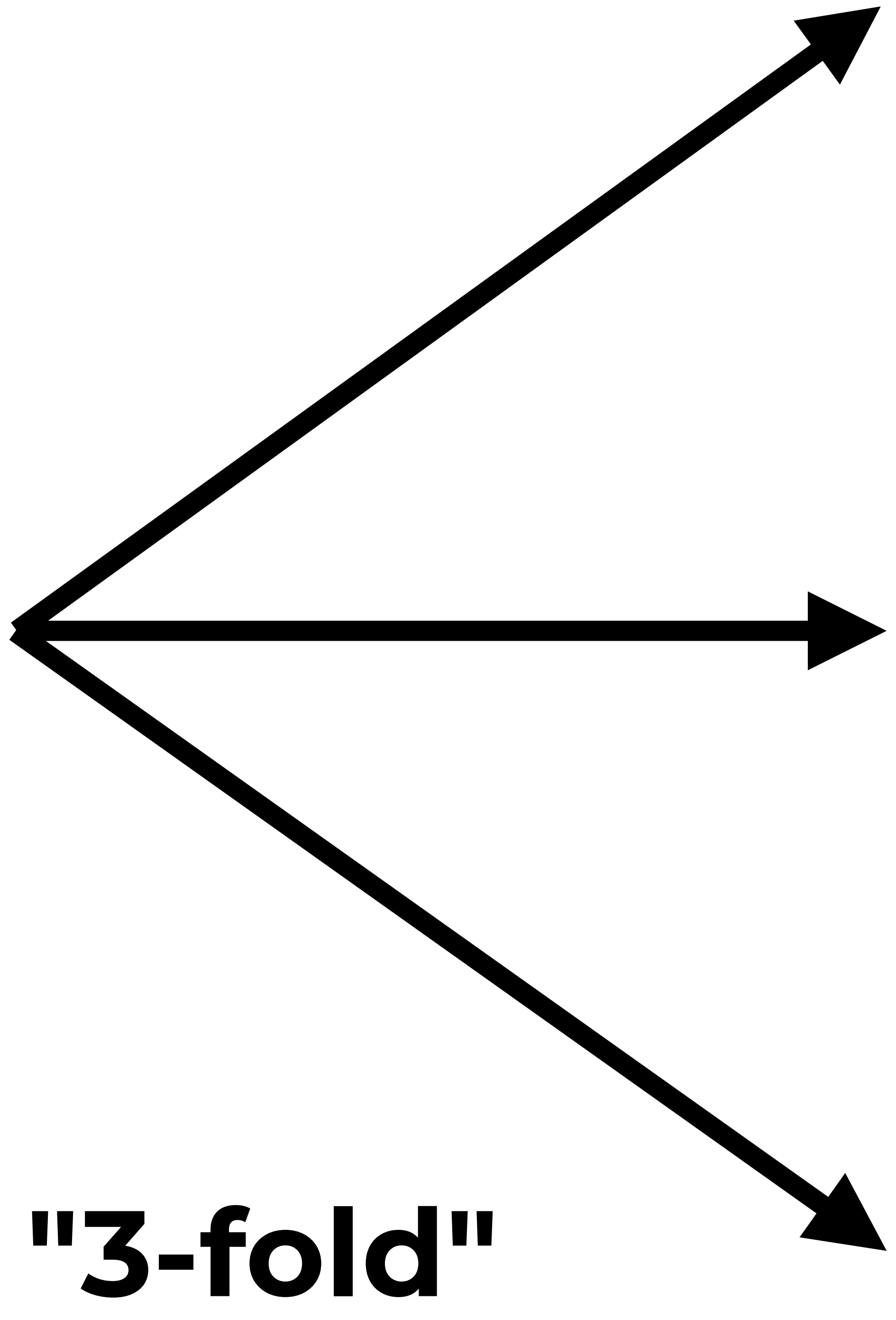
```
set.seed(32011)
admit_gender <- admission %>%
  bootstraps(times = 100) %>%
  mutate(admit_diff = map_dbl(splits, pct_diff))

ggplot(admit_gender, mapping = aes(x = admit_diff)) +
  geom_density()
```



```
quantile(admit_gender$admit_diff, probs = c(0.025, 0.500, 0.975))
      2.5%        50%      97.5%
0.07596896 0.09386976 0.11140315
```

Cross Validation

id	x	y	z
1	0.73	-0.76	0.86
2	-0.24	0.59	0.93
3	-0.24	-1.81	0.46
4	-1.12	-0.17	1.71
5	0.21	-0.73	-1.25
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72
8	0.81	-0.76	0.93
9	-0.18	2.75	-0.14

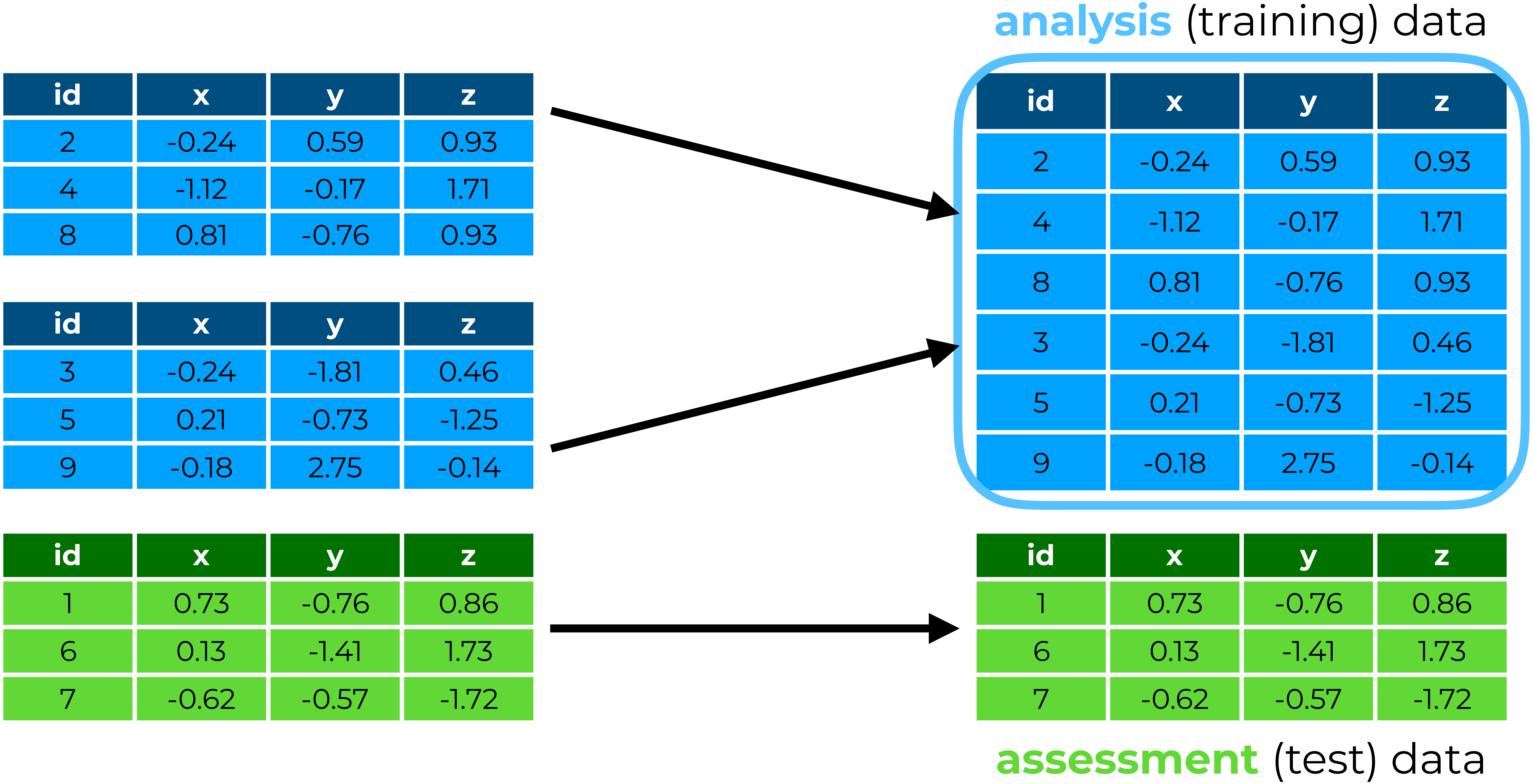


"3-fold"

id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93

id	x	y	z
3	-0.24	-1.81	0.46
5	0.21	-0.73	-1.25
9	-0.18	2.75	-0.14

id	x	y	z
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72



id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93

id	x	y	z
3	-0.24	-1.81	0.46
5	0.21	-0.73	-1.25
9	-0.18	2.75	-0.14

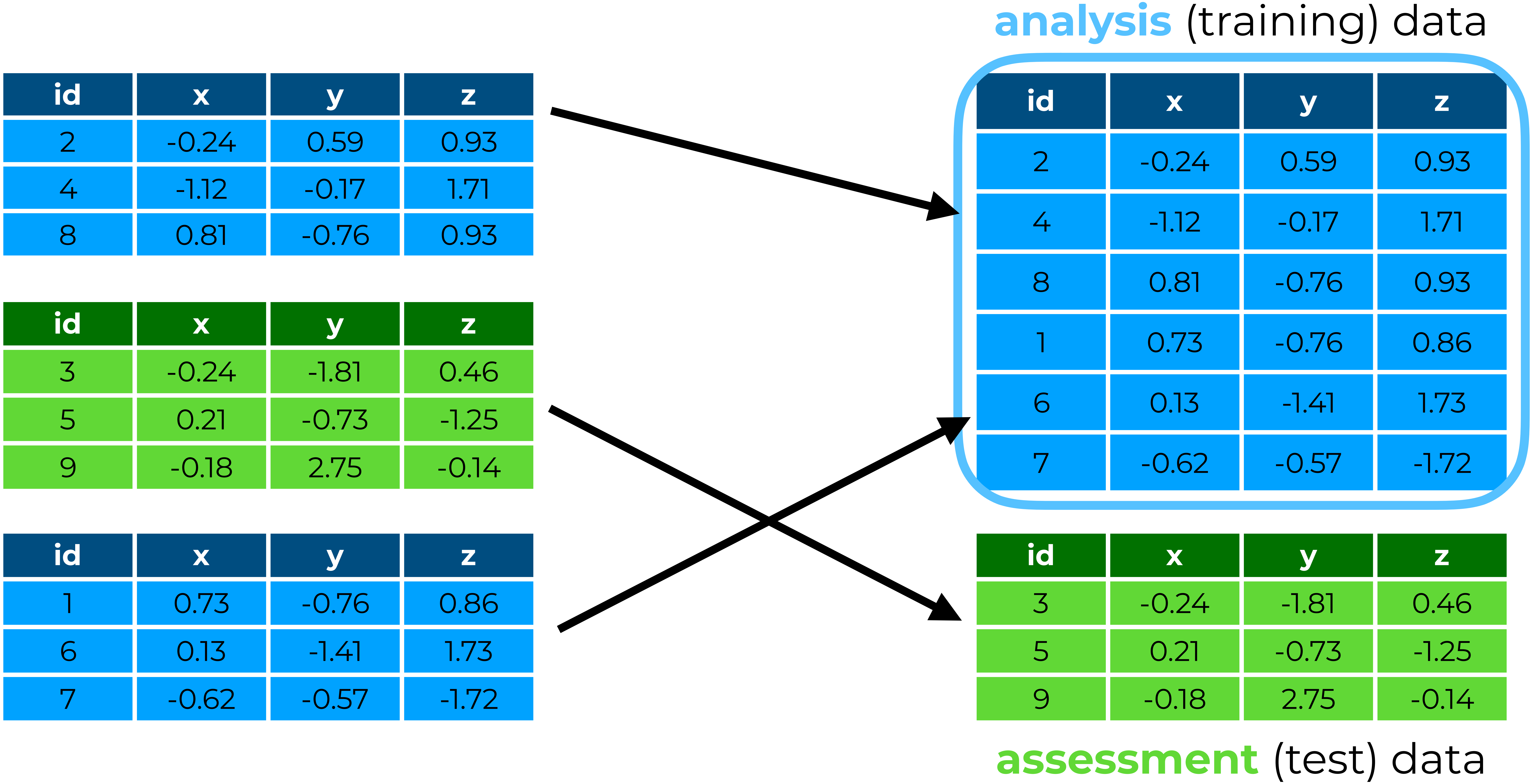
id	x	y	z
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72

analysis (training) data

id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93
3	-0.24	-1.81	0.46
5	0.21	-0.73	-1.25
9	-0.18	2.75	-0.14

id	x	y	z
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72

assessment (test) data



id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93

id	x	y	z
3	-0.24	-1.81	0.46
5	0.21	-0.73	-1.25
9	-0.18	2.75	-0.14

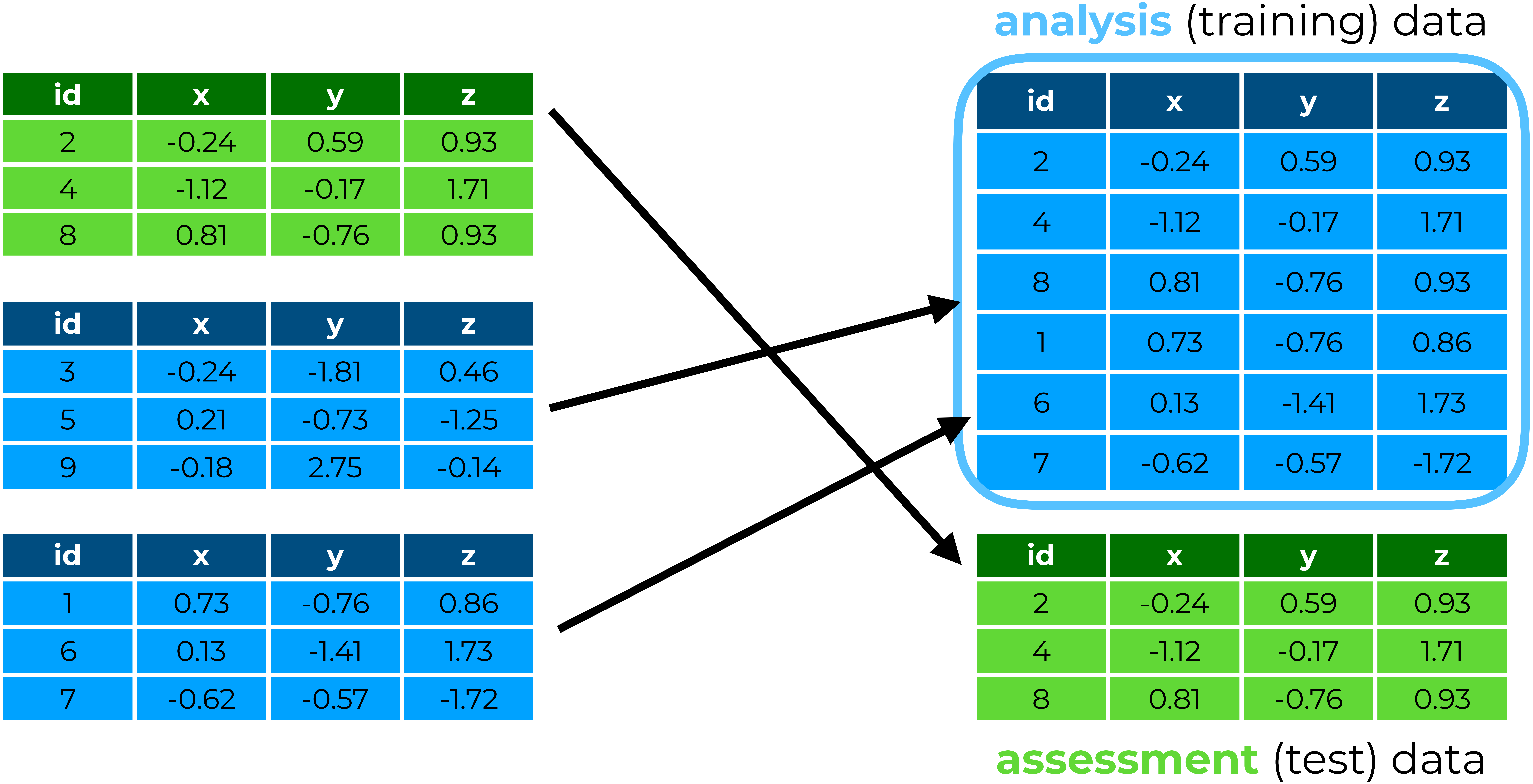
id	x	y	z
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72

analysis (training) data

id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72

id	x	y	z
3	-0.24	-1.81	0.46
5	0.21	-0.73	-1.25
9	-0.18	2.75	-0.14

assessment (test) data



id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93

id	x	y	z
3	-0.24	-1.81	0.46
5	0.21	-0.73	-1.25
9	-0.18	2.75	-0.14

id	x	y	z
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72

analysis (training) data

id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93
1	0.73	-0.76	0.86
6	0.13	-1.41	1.73
7	-0.62	-0.57	-1.72

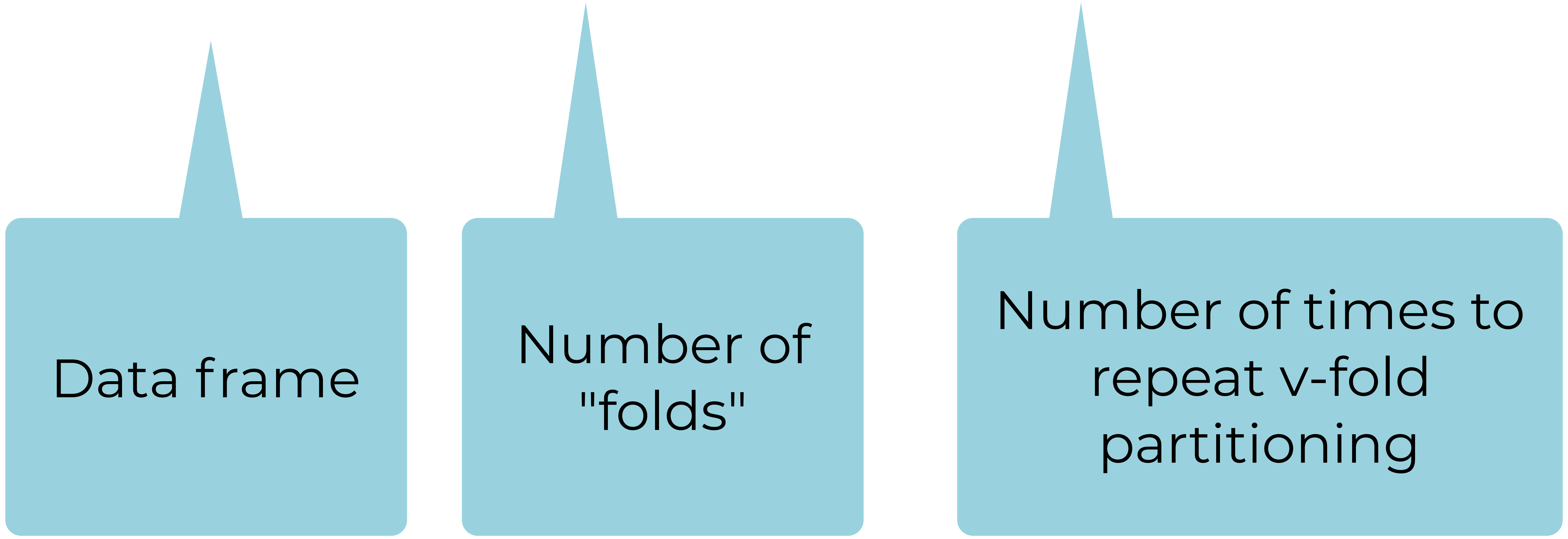
id	x	y	z
2	-0.24	0.59	0.93
4	-1.12	-0.17	1.71
8	0.81	-0.76	0.93

assessment (test) data

vfold_cv()

Randomly split the data into folds for cross validation.

```
vfold_cv(data, v = 10, repeats = 1, ...)
```



Data frame

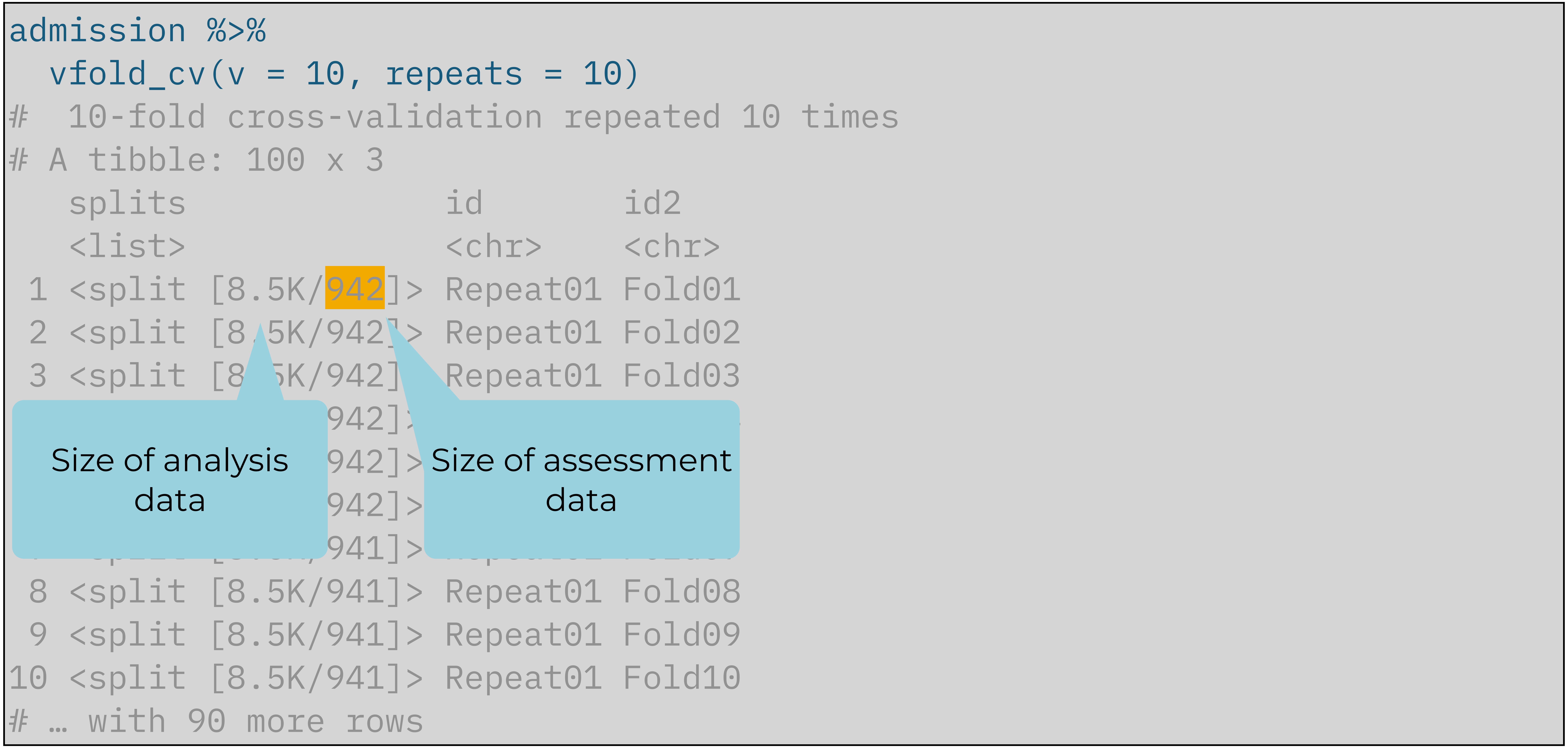
Number of "folds"

Number of times to repeat v-fold partitioning

```
admission %>%
  vfold_cv(v = 10, repeats = 10)
#  10-fold cross-validation repeated 10 times
# A tibble: 100 x 3
   splits             id       id2  
   <list>             <chr>    <chr>
 1 <split [8.5K/942]> Repeat01 Fold01
 2 <split [8.5K/942]> Repeat01 Fold02
 3 <split [8.5K/942]> Repeat01 Fold03
 4 <split [8.5K/942]> Repeat01 Fold04
 5 <split [8.5K/942]> Repeat01 Fold05
 6 <split [8.5K/942]> Repeat01 Fold06
 7 <split [8.5K/941]> Repeat01 Fold07
 8 <split [8.5K/941]> Repeat01 Fold08
 9 <split [8.5K/941]> Repeat01 Fold09
10 <split [8.5K/941]> Repeat01 Fold10
# … with 90 more rows
```

```
admission %>%
  vfold_cv(v = 10, repeats = 10)
#  10-fold cross-validation repeated 10 times
# A tibble: 100 x 3
   splits               id       id2
   <list>               <chr>    <chr>
 1 <split [8.5K/942]> Repeat01 Fold01
 2 <split [8.5K/942]> Repeat01 Fold02
 3 <split [8.5K/942]> Repeat01 Fold03
               942]> Repeat01 Fold04
               942]> Repeat01 Fold05
               942]> Repeat01 Fold06
               941]> Repeat01 Fold07
 8 <split [8.5K/941]> Repeat01 Fold08
 9 <split [8.5K/941]> Repeat01 Fold09
10 <split [8.5K/941]> Repeat01 Fold10
# … with 90 more rows
```

Size of analysis data

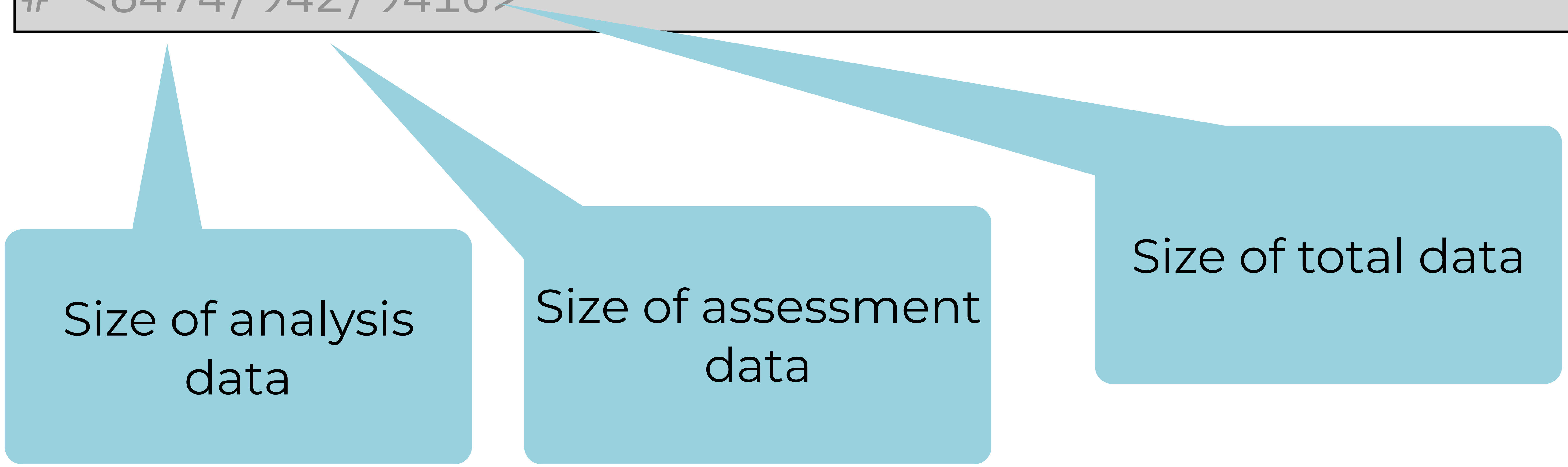


```
admission %>%
  vfold_cv(v = 10, repeats = 10)
#  10-fold cross-validation repeated 10 times
# A tibble: 100 x 3
   splits               id       id2
   <list>               <chr>    <chr>
 1 <split [8.5K/942]> Repeat01 Fold01
 2 <split [8.5K/942]> Repeat01 Fold02
 3 <split [8.5K/942]> Repeat01 Fold03
                 942]>
                 942]>
                 942]>
                 941]>
 8 <split [8.5K/941]> Repeat01 Fold08
 9 <split [8.5K/941]> Repeat01 Fold09
10 <split [8.5K/941]> Repeat01 Fold10
# … with 90 more rows
```

Size of analysis data

Size of assessment data

```
models <- admission %>%
  vfold_cv(v = 10, repeats = 10)

models$splits[[1]]
# <8474/942/9416>
```



Size of analysis data

Size of assessment data

Size of total data

```
admission %>%
  vfold_cv(v = 10, repeats = 10)
```

splits	id	id2
<split [8.5K/942]>	Repeat01	Fold01
<split [8.5K/942]>	Repeat01	Fold02
<split [8.5K/942]>	Repeat01	Fold03

```
admission %>%
  vfold_cv(v = 10, repeats = 10) %>%
  mutate(model = map(splits, function(x) glm(admit ~ gender, data = analysis(x), family = binomial)))
```

splits	id	id2	model
<split [8.5K/942]>	Repeat01	Fold01	Call: glm(formula = admit ~ gender, family = binomial, data = analysis(x)) Coefficients: (Intercept) genderMale -0.21453 -0.02388 Degrees of Freedom: 8473 Total (i.e. Null); 8472 Residual Null Deviance: 11640 Residual Deviance: 11640 AIC: 11640
<split [8.5K/942]>	Repeat01	Fold02	Call: glm(formula = admit ~ gender, family = binomial, data = analysis(x)) Coefficients: (Intercept) genderMale -0.21665 -0.01874 Degrees of Freedom: 8473 Total (i.e. Null); 8472 Residual Null Deviance: 11640 Residual Deviance: 11640 AIC: 11640
<split [8.5K/942]>	Repeat01	Fold03	Call: glm(formula = admit ~ gender, family = binomial, data = analysis(x)) Coefficients: (Intercept) genderMale -0.232365 -0.008608 Degrees of Freedom: 8473 Total (i.e. Null); 8472 Residual Null Deviance: 11630 Residual Deviance: 11630 AIC: 11630

Consider

How would you assess model performance?

More to come in Case Study 2….

Recap

- Store objects and other lists in list-columns of data frames
- Use **bootsraps()** to recreate resampled data objects
- Use **vfold_cv()** to create *analysis* and *assessment* sub-samples of your data to assessment model performance
- Use **purrr** to iterate over bootstrapped samples and cross validation folds

Organize Multiple Models

rsample

www.rstudio.com

ATLAS

Accessible Teaching, Learning, and Assessment Systems