Managing many models

February 2017

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio

You've never seen data presented like this. With the drama and urgency of a sportscaster, statistics guru Hans Rosling debunks myths about the so-called "developing world."

https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen

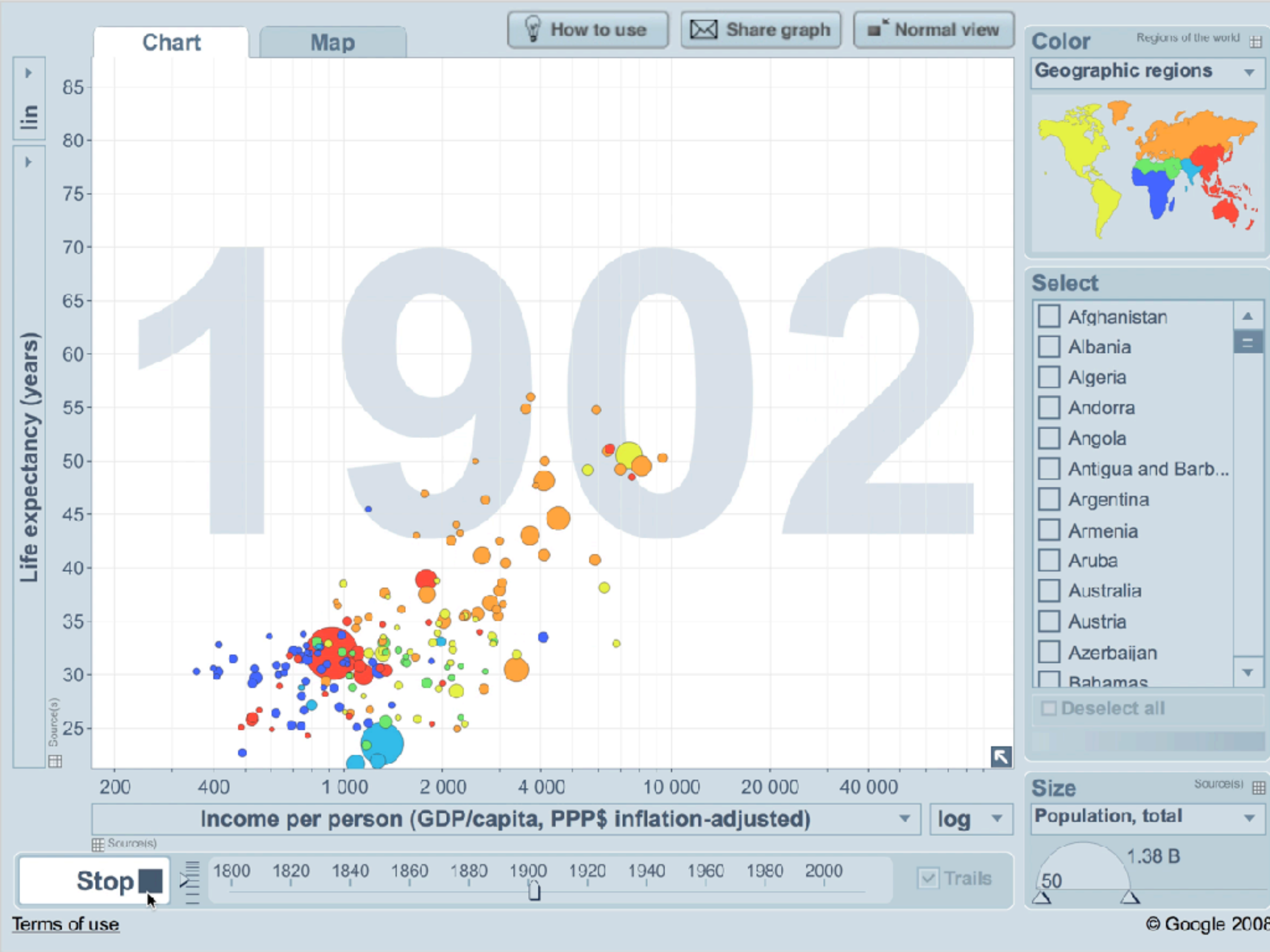
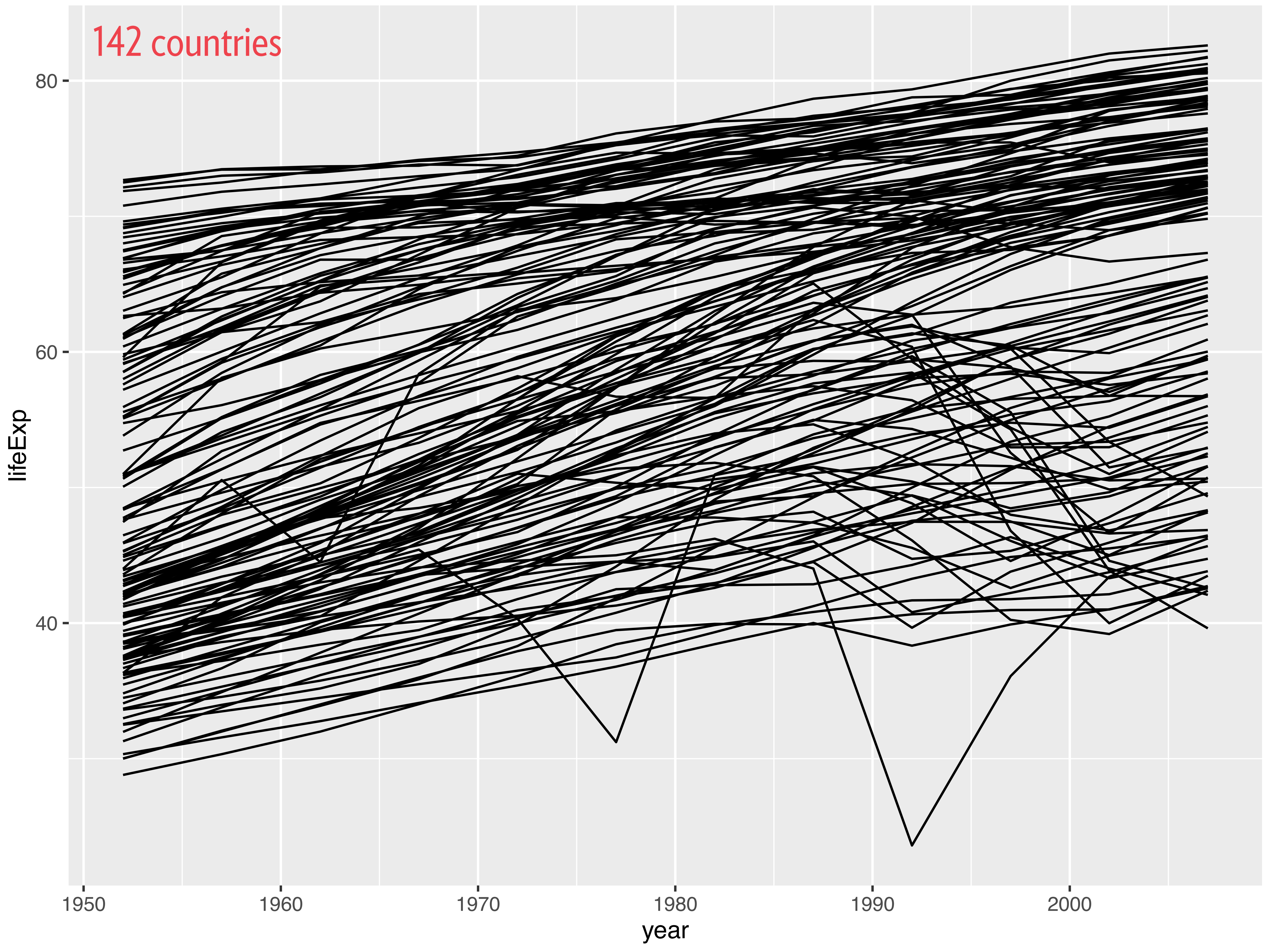
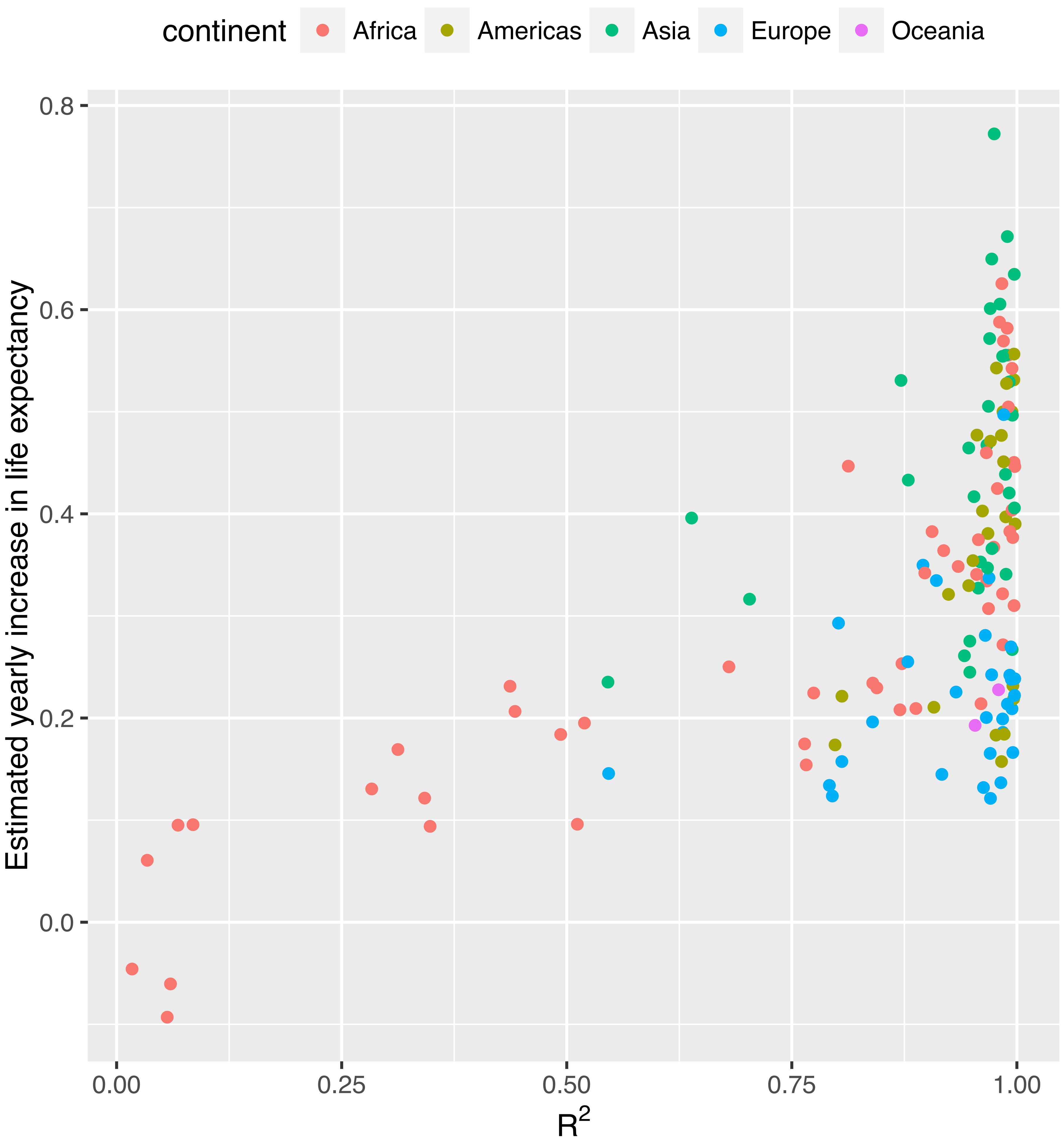


Chart
Map
How to use
Share graph
Normal view
Color
Regions of the world
Geographic regions
Select
Afghanistan
Albania
Algeria
Andorra
Angola
Antigua and Barb...
Argentina
Armenia
Aruba
Australia
Austria
Azerbaijan
Bahamas
Deselect all
Size
Sources
Population, total
1.38 B
50
lin
Life expectancy (years)
85
80
75
70
65
60
55
50
45
40
35
30
25
1902
200
400
1 000
2 000
4 000
10 000
20 000
40 000
Income per person (GDP/capita, PPP$ inflation-adjusted)
log
Source(s)
Stop
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
2000
Trails
Terms of use
© Google 2008


142 countries
80
60
40
lifeExp
1950
1960
1970
1980
1990
2000
year


continent
Africa
Americas
Asia
Europe
Oceania
Estimated yearly increase in life expectancy
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
R^2

But...

Arbitrarily complicated models

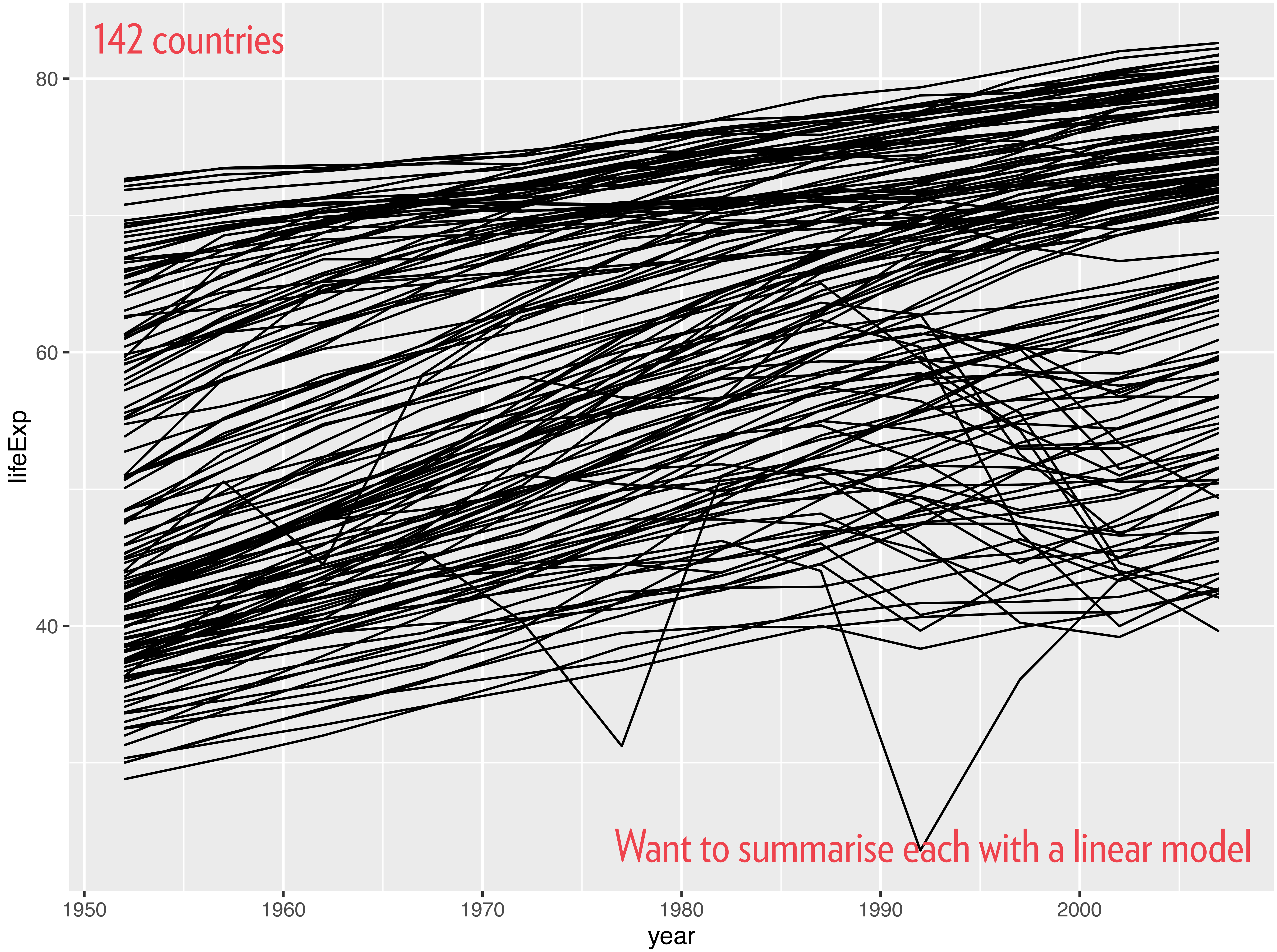
Scales to big data

Three simple underlying ideas

Each idea is partnered with a package

1. Nested data (tidyr)
2. Functional programming (purrr)
3. Models → tidy data (broom)

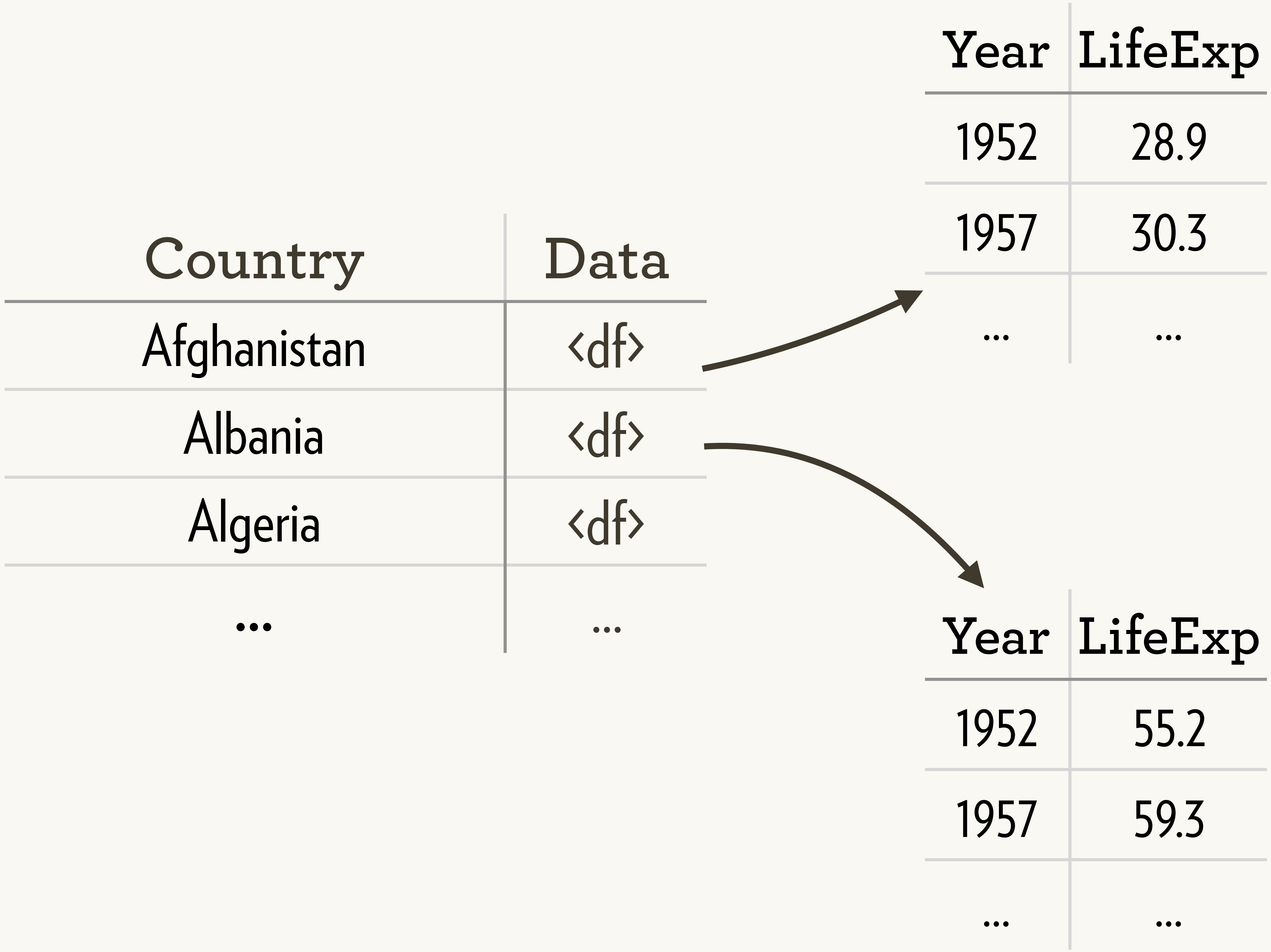
Nested data


142 countries
Want to summarise each with a linear model
80
60
40
lifeExp
1950
1960
1970
1980
1990
2000
year

Currently our data has one row per observation

Country	Year	LifeEx
Afghanistan	1952	28.9
Afghanistan	1957	30.3
Afghanistan	...	...
Albania	1952	55.2
Albania	1957	59.3
Albania	...	...
Algeria	...	...
...	...	...

More convenient to one row per group



Country	Data
Afghanistan	<df>
Albania	<df>
Algeria	<df>
...	...

Year	LifeExp
1952	28.9
1957	30.3
...	...

Year	LifeExp
1952	55.2
1957	59.3
...	...

I call this a **nested data frame**

In R:

```
library(dplyr)
library(tidyr)

by_country <- gapminder %>%
  group_by(continent, country) %>%
  nest()
```

Haven’t seen pipes?

```r
x %>% f(y)
# is the same as:
f(x, y)

gapminder %>%
  group_by(continent, country) %>%
  nest()
# same as:
nest(group_by(gapminder, continent, country))
```

Each country will have an associated model

Country	Data
Afghanistan	<df>
Albania	<df>
Algeria	<df>
...	...

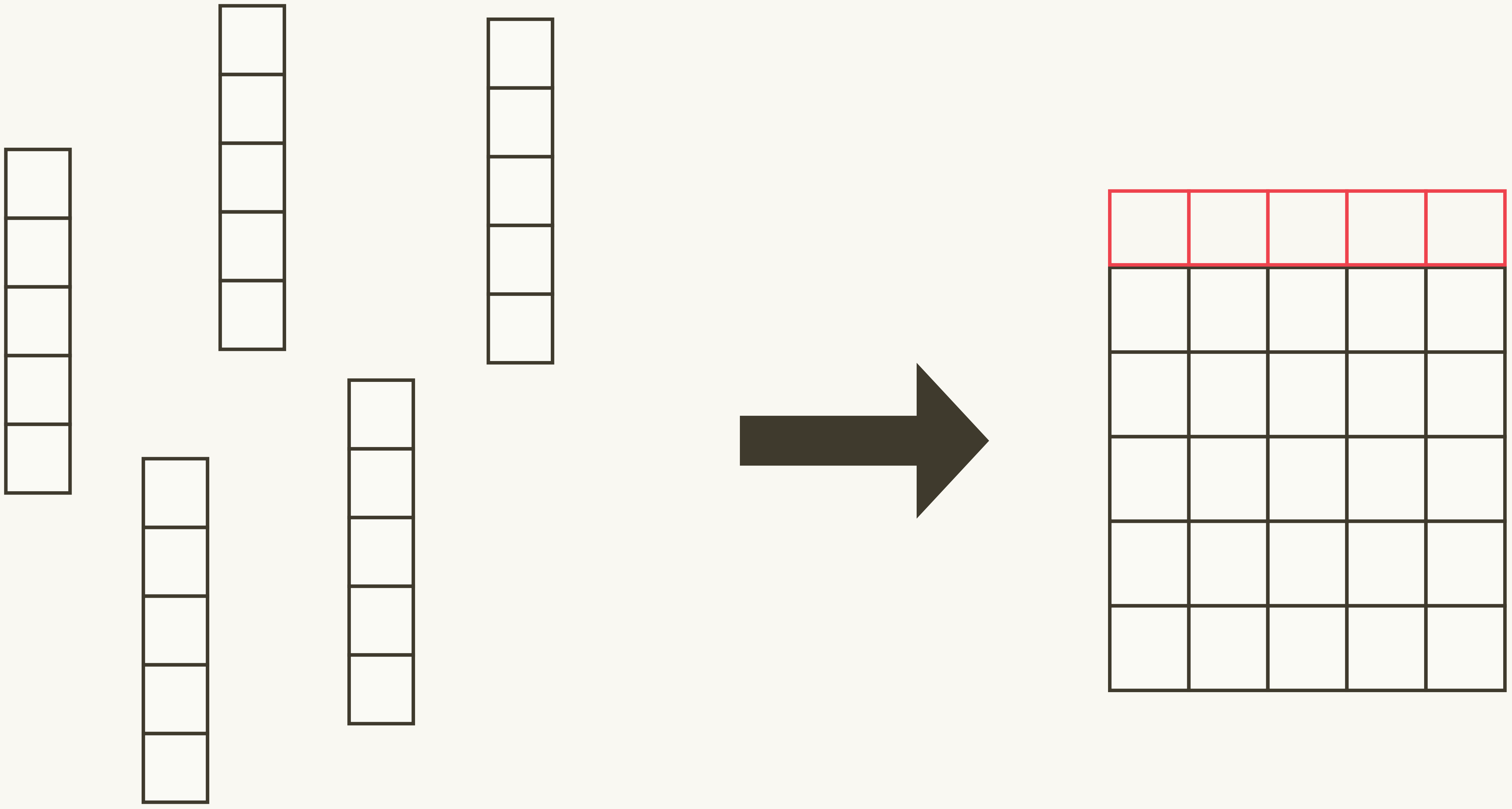
```
lm(lifeExp ~ year1950, data = afghanistan)
```

```
lm(lifeExp1950 ~ year, data = albania)
```

Why not store that in a column too?

Country	Data	Model
Afghanistan	<df>	<lm>
Albania	<df>	<lm>
Algeria	<df>	<lm>
...	...	...

List-columns keep related things together



Anything can go in a list & a list can go in a data frame

In R:

```
library(dplyr)
library(purrr)

country_model <- function(df) {
  lm(lifeExp ~ year1950, data = df)
}

models <- by_country %>%
  mutate(
    mod  = map(data, country_model)
  )
```

Functional programming

Or, why for loops are “bad”

Motivated by baking cupcakes

Vanilla cupcakes

The hummingbird bakery cookbook

1 cup flour	
a scant ¾ cup sugar	
1 ½ t baking powder	
3 T unsalted butter	
½ cup whole milk	
1 egg	
¼ t pure vanilla extract	

Preheat oven to 350°F.

Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Chocolate cupcakes

The hummingbird bakery cookbook

¾ cup + 2T flour
2 ½ T cocoa powder
a scant ¾ cup sugar
1 ½ t baking powder
3 T unsalted butter
½ cup whole milk
1 egg
¼ t pure vanilla extract

Preheat oven to 350°F.

Put the flour, cocoa, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Chocolate cupcakes

The hummingbird bakery cookbook

¾ cup + 2T flour
2 ½ T cocoa powder
a scant ¾ cup sugar
1 ½ t baking powder
3 T unsalted butter
½ cup whole milk
1 egg
¼ t pure vanilla extract

Preheat oven to 350°F.

Put the flour, cocoa, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Vanilla cupcakes

The hummingbird bakery cookbook

1 cup flour
a scant ¾ cup sugar
1 ½ t baking powder
3 T unsalted butter
½ cup whole milk
1 egg
¼ t pure vanilla extract

Preheat oven to 350°F.

Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Vanilla cupcakes

The hummingbird bakery cookbook

120g flour	Preheat oven to 170°C.
140g sugar	Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.
1.5 t baking powder	Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.
40g unsalted butter	Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.
120ml milk	Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.
1 egg	
0.25 t pure vanilla extract	

1. Convert units

Vanilla cupcakes

The hummingbird bakery cookbook

120g flour	Beat flour, sugar, baking powder, salt, and butter until sandy.
140g sugar	Whisk milk, egg, and vanilla. Mix half into flour mixture until smooth (use high speed). Beat in remaining half. Mix until smooth.
1.5 t baking powder	Bake 20-25 min at 170°C.
40g butter	
120ml milk	
1 egg	
0.25 t vanilla	

2. Rely on domain knowledge

Vanilla cupcakes

The hummingbird bakery cookbook

120g flour	Beat dry ingredients + butter until sandy.
140g sugar	Whisk together wet ingredients. Mix half into dry until smooth (use high speed). Beat in remaining half. Mix until smooth.
1.5 t baking powder	Bake 20-25 min at 170°C.
40g butter	
120ml milk	
1 egg	
0.25 t vanilla	

3. Use variables

Cupcakes

Beat dry ingredients + butter until sandy.

Whisk together wet ingredients. Mix half into dry until smooth (use high speed). Beat in remaining half. Mix until smooth.

Bake 20-25 min at 170°C.

Vanilla	Chocolate
120g flour	100g flour
	20g cocoa
140g sugar	140g sugar
1.5t baking powder	1.5t baking powder
40g butter	40g butter
120ml milk	120ml milk
1 egg	1 egg
0.25 t vanilla	0.25 t vanilla

4. Extract out common code

Cupcakes

	Flour	Baking powder	Sugar	Butter	Egg	Extra
Vanilla	120	1.5	140	40	1	0.25t vanilla
Chocolate	100	1.5	140	40	1	20g cocoa • 0.25t vanilla
Lemon	120	1.5	140	40	1	2T lemon zest
Red velvet	150	0	150	60	1	10g cocoa • 20ml red colouring • 1.5t vinegar • 0.5 t baking soda

4. Convert to data

For loops emphasise the objects

```r
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```

For loops emphasise the objects

```
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```

Not the actions

```r
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```

Functional programming emphasises the actions

```r
library(purrr)
means <- mtcars %>% map_dbl(mean)
medians <- mtcars %>% map_dbl(median)
```

What does map_dbl() look like?

```
map_dbl <- function(x, f, ...) {
  out <- vector("double", length(x))
  for (i in seq_along(out)) {
    out[i] <- f(x[[i]], ...)
  }
  out
}
```

Actual implementation a little different

There are many variants:

```r
map_int <- function(x, f, ...) {
  out <- vector("integer", length(x))
  for (i in seq_along(out)) {
    out[i] <- f(x[[i]], ...)
  }
  out
}
```

Some vary the output

```
map <- function(x, f, ...) {
  out <- vector("list", length(x))
  for (i in seq_along(out)) {
    out[[i]] <- f(x[[i]], ...)
  }
  out
}
```

This is the same as lapply()!

Others vary the input

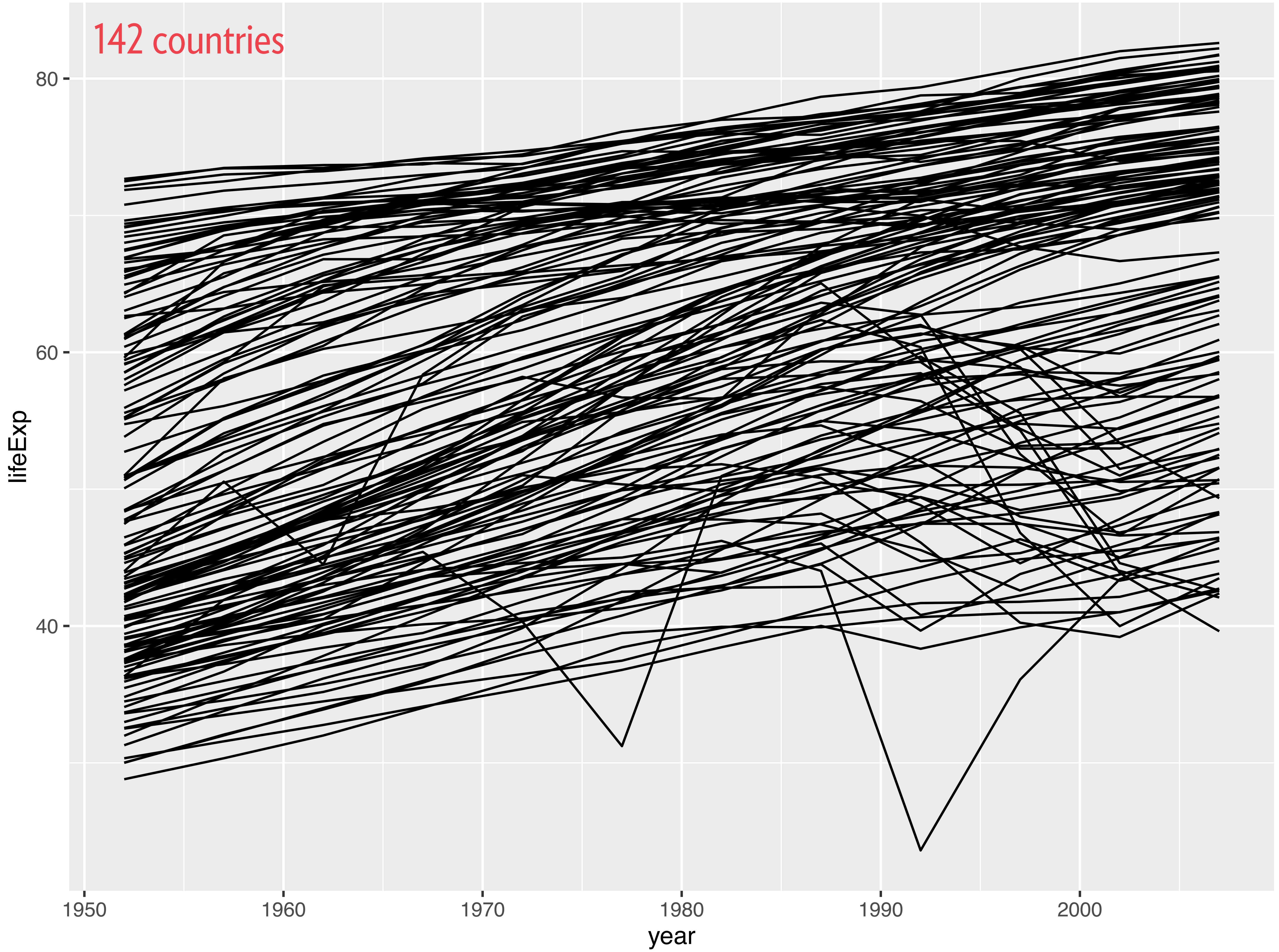
```
map2 <- function(x, y, f, ...) {
  out <- vector("list", length(x))
  for (i in seq_along(out)) {
    out[[i]] <- f(x[[i]], y[[i]], ...)
  }
  out
}
```

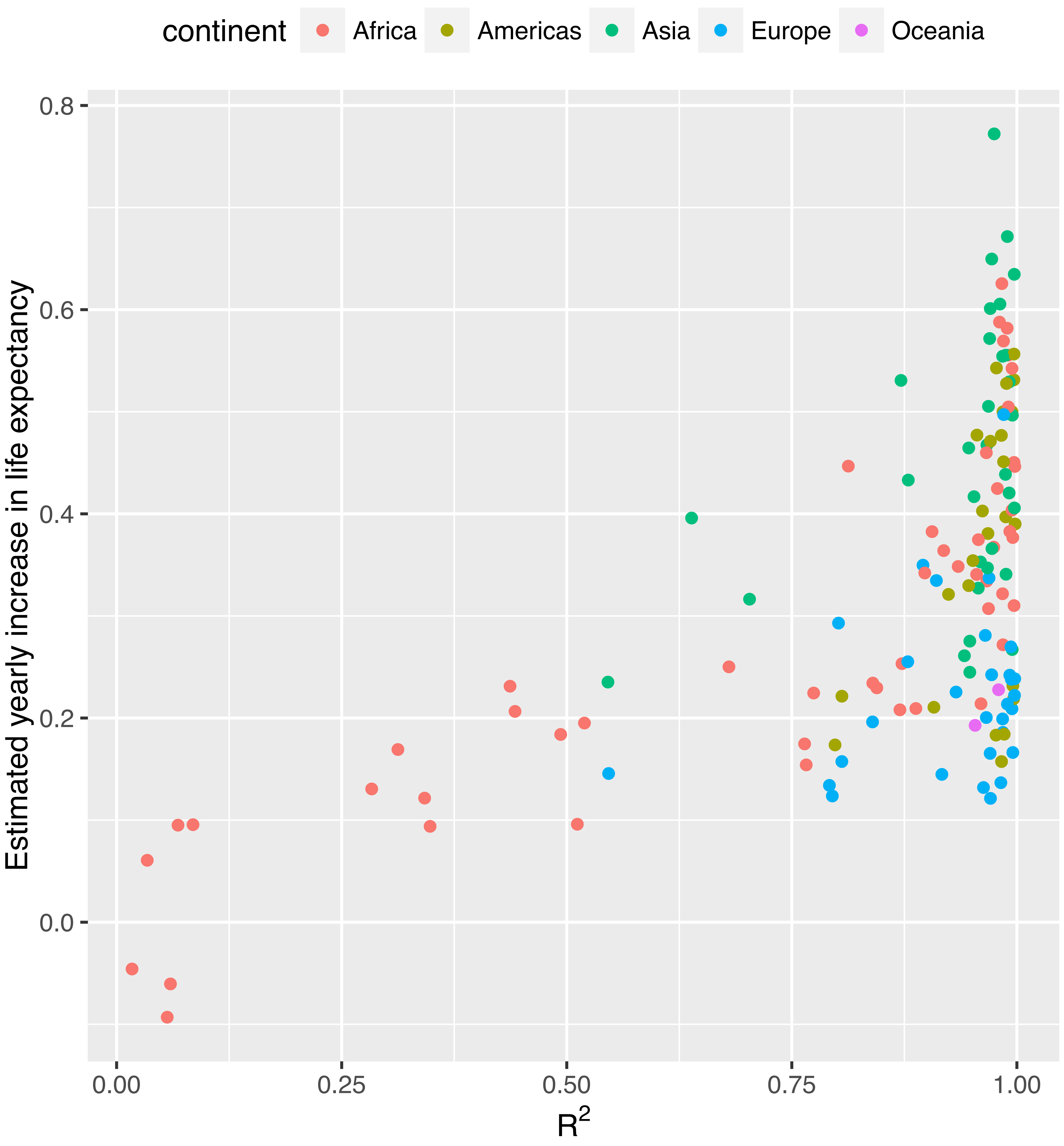
We can even think of functions as data

```r
funs <- list(mean, median, sd)

funs %>%
  map(~ mtcars %>% map_dbl(.x))
```

Back to gapminder


142 countries
80
60
40
lifeExp
1950
1960
1970
1980
1990
2000
year


continent
Africa
Americas
Asia
Europe
Oceania
Estimated yearly increase in life expectancy
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
R^2

We nested the data to get a list of data frames

Country	Year	LifeEx
Afghanistan	1952	28.9
Afghanistan	1957	30.3
Afghanistan	...	...
Albania	1952	55.2
Albania	1957	59.3
Albania	...	...
Algeria	...	...
...	...	...

$\xrightarrow{\texttt{nest()}}$

Country	Data
Afghanistan	<df>
Albania	<df>
Algeria	<df>
...	...

Then we fitted a model to each country

```
library(dplyr)
library(tidyr)
library(purrr)

country_model <- function(df) {
  lm(lifeExp ~ year1950, data = df)
}

gapminder %>%
  group_by(continent, country) %>%
  nest() %>%
  mutate(
    mod  = data %>% map(country_model)
  )
```

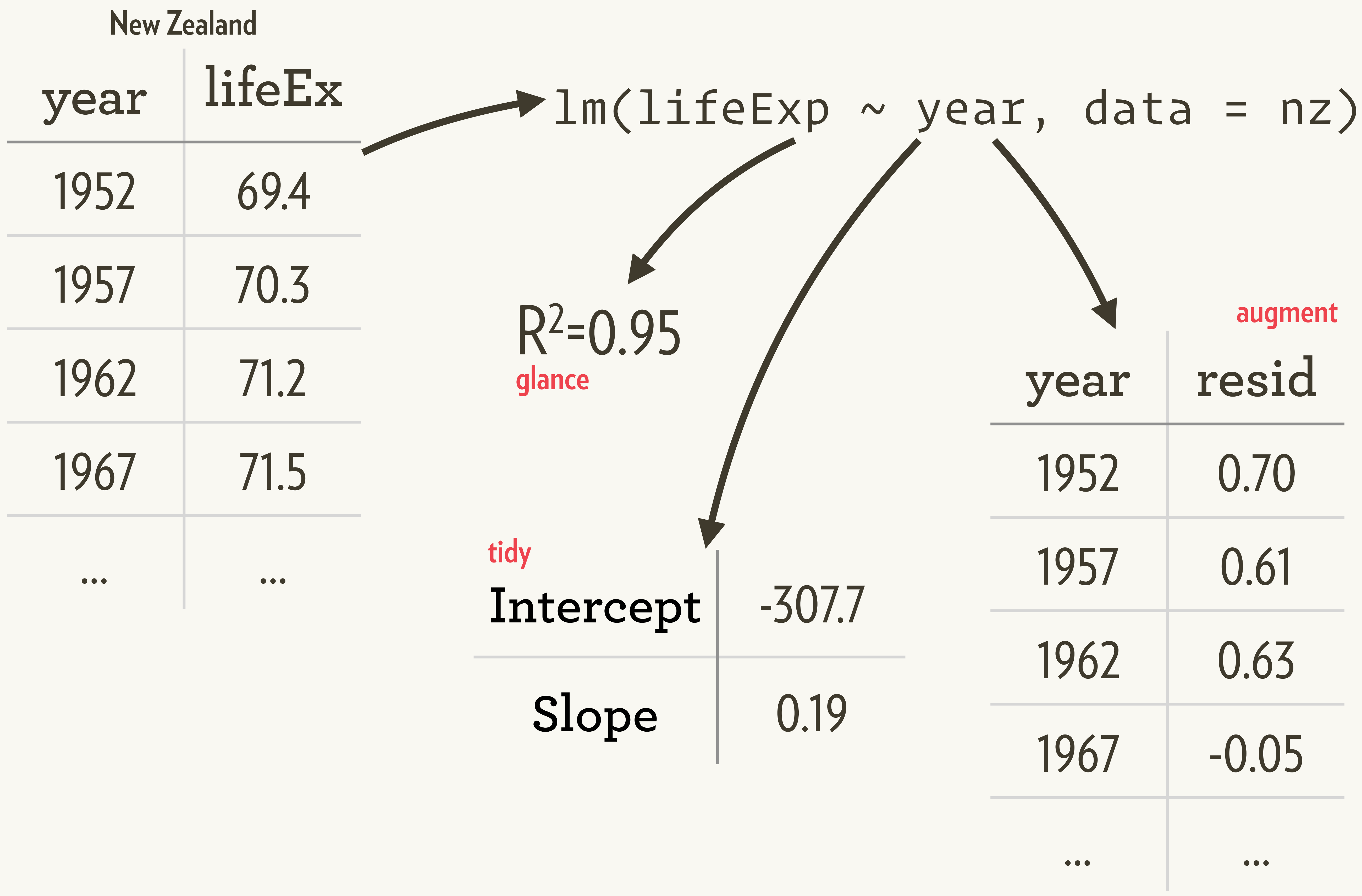
What can we do with a list of models?

Country	Data	Model
Afghanistan	<data>	<lm>
Albania	<data>	<lm>
Algeria	<data>	<lm>
...	<data>	<lm>

Models → tidy data

With broom, by David Robinson

What data can we extract from a model?



New Zealand

year	lifeEx
1952	69.4
1957	70.3
1962	71.2
1967	71.5
...	...

```
lm(lifeExp ~ year, data = nz)
```

R^2=0.95

glance

tidy

Intercept	-307.7
Slope	0.19

augment

year	resid
1952	0.70
1957	0.61
1962	0.63
1967	-0.05
...	...

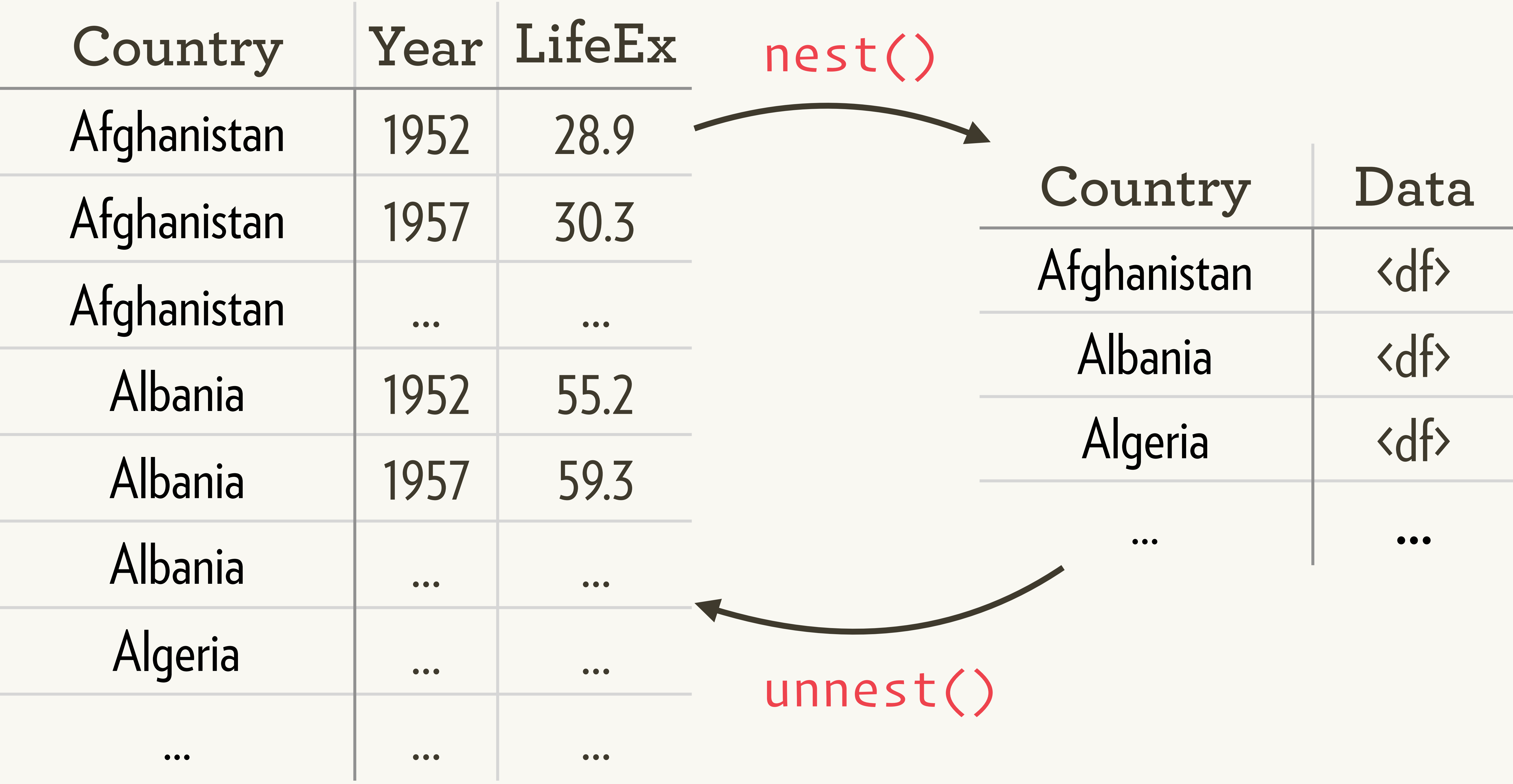
We need to do that for each model

```
models <- models %>%
  mutate(
    tidy    = map(model, broom::tidy),
    glance  = map(model, broom::glance),
    augment = map(model, broom::augment)
  )
```

Which gives us:

Country	Data	Model	Glance	Tidy	Augment
Afghanistan	<df>	<lm>	<df>	<df>	<df>
Albania	<df>	<lm>	<df>	<df>	<df>
Algeria	<df>	<lm>	<df>	<df>	<df>
...	...	...	...	...	...

Unnest lets us go back to a regular data frame



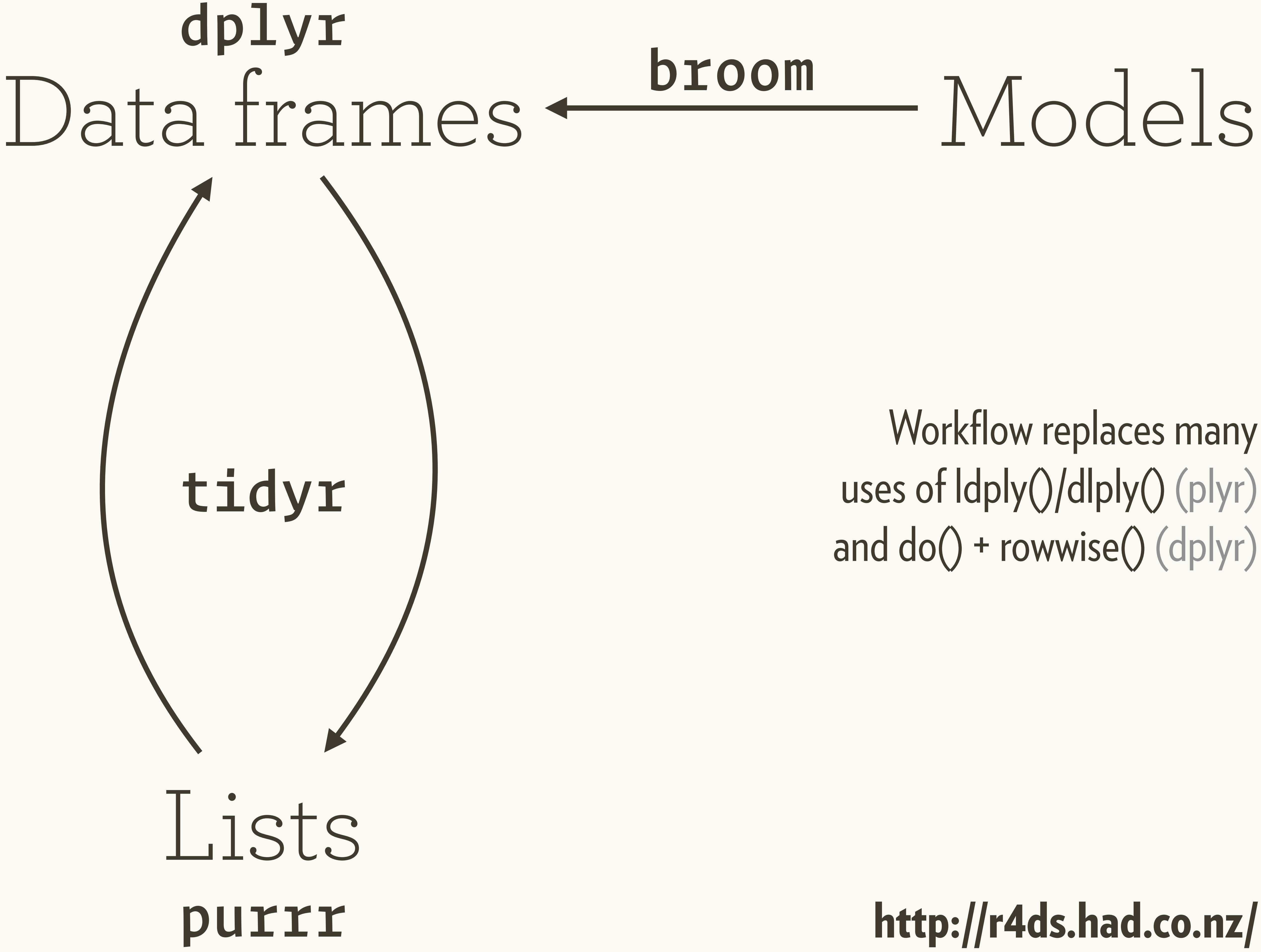
Country	Year	LifeEx
Afghanistan	1952	28.9
Afghanistan	1957	30.3
Afghanistan	...	...
Albania	1952	55.2
Albania	1957	59.3
Albania	...	...
Algeria	...	...
...	...	...

Country	Data
Afghanistan	<df>
Albania	<df>
Algeria	<df>
...	...

Demo

Conclusion

1. Store related objects in **list-columns**.
2. Learn **FP** so you can focus on verbs, not objects.
3. Use **broom** to convert models to tidy data.



Workflow replaces many
uses of ldply()/dlply() (plyr)
and do() + rowwise() (dplyr)

http://r4ds.had.co.nz/