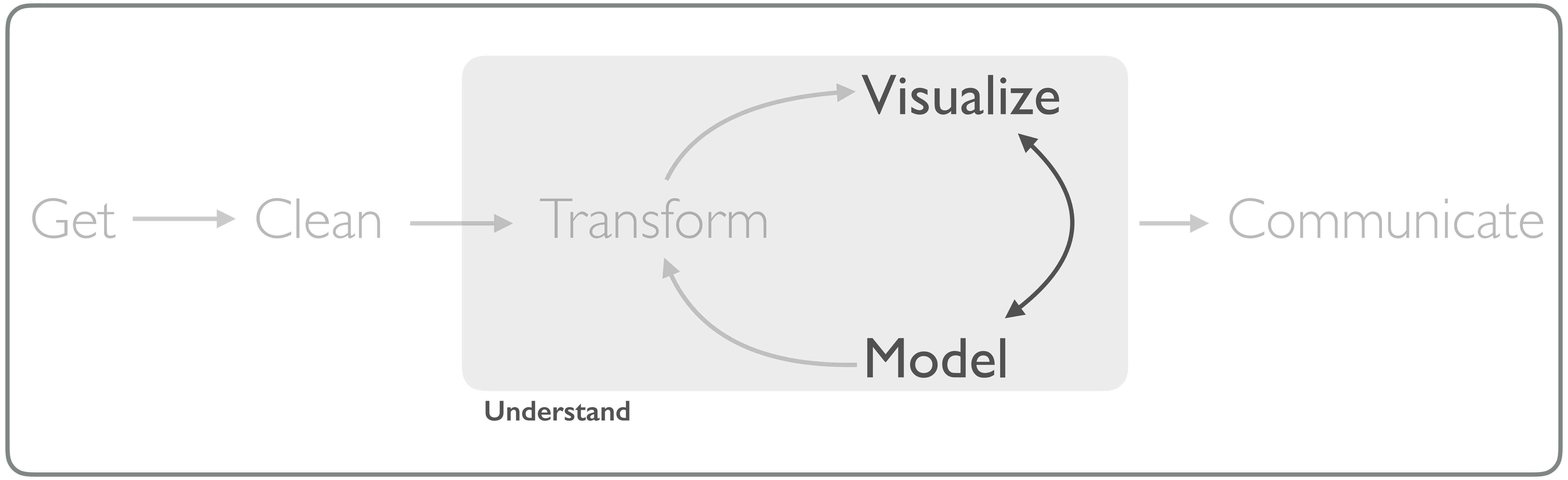
MANAGING MANY MODELS



†A modified version of Hadley Wickham's analytic process

PREREQUISITES

PREREQUISITES

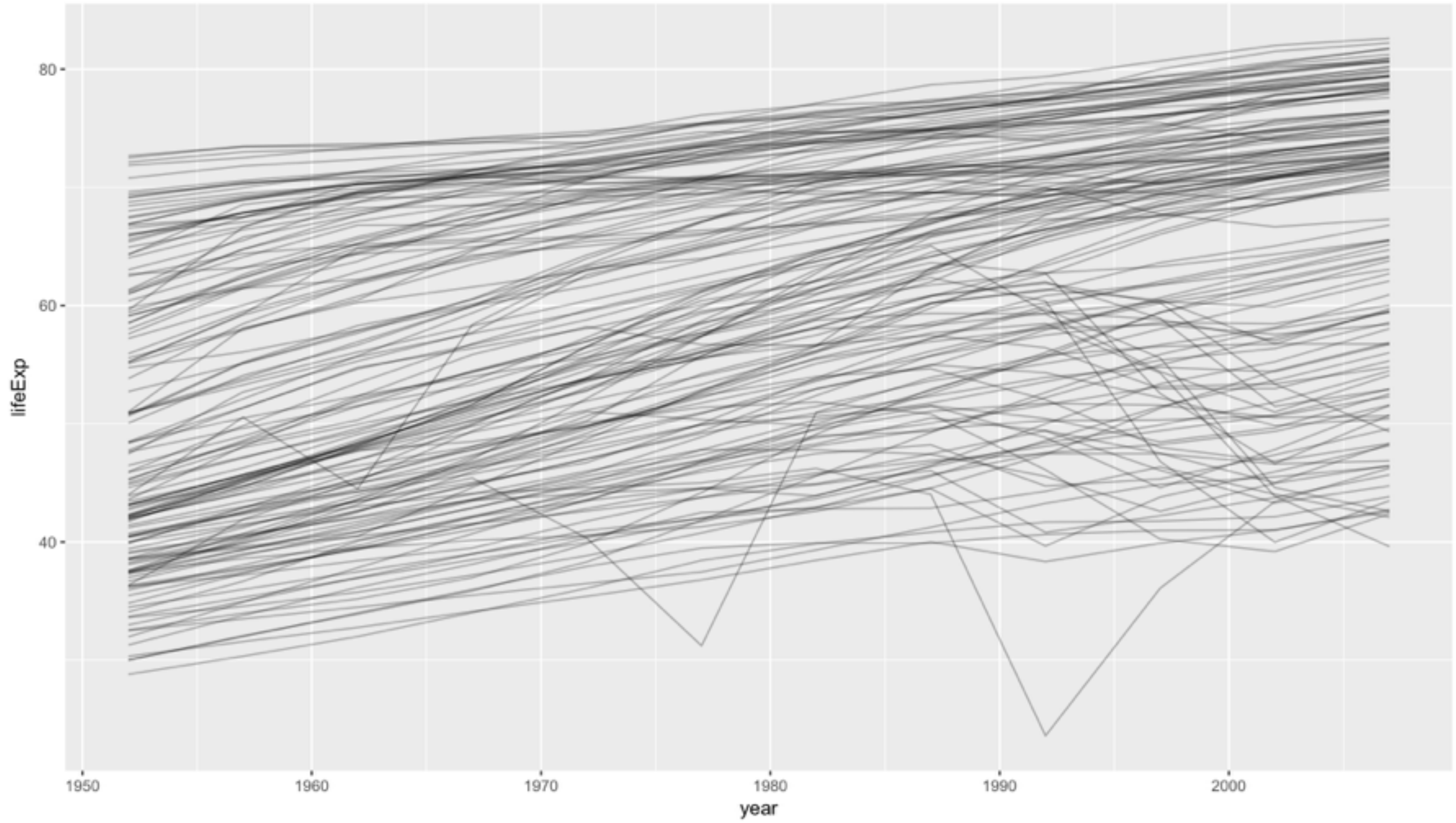
```
library(modelr)
library(tidyverse)
library(gapminder)

gapminder
# A tibble: 1,704 × 6
       country continent  year lifeExp      pop gdpPercap
        <fctr>    <fctr> <int>   <dbl>    <int>     <dbl>
1  Afghanistan      Asia  1952  28.801  8425333  779.4453
2  Afghanistan      Asia  1957  30.332  9240934  820.8530
3  Afghanistan      Asia  1962  31.997 10267083  853.1007
4  Afghanistan      Asia  1967  34.020 11537966  836.1971
5  Afghanistan      Asia  1972  36.088 13079460  739.9811
6  Afghanistan      Asia  1977  38.438 14880372  786.1134
7  Afghanistan      Asia  1982  39.854 12881816  978.0114
```

HOW DOES LIFE EXPECTANCY CHANGE OVER TIME?

A COMMON TREND…FOR THE MOST PART

```r
gapminder %>%
  ggplot(aes(year, lifeExp, group = country)) +
  geom_line(alpha = 1/3)
```



We see a fairly common trend across most countries

MODEL THE WHOLE THING

```
full_mod <- lm(lifeExp ~ year + country,
               data = gapminder)

summary(full_mod)
```

One approach to model this relationship over time is to use the following multivariate regression model

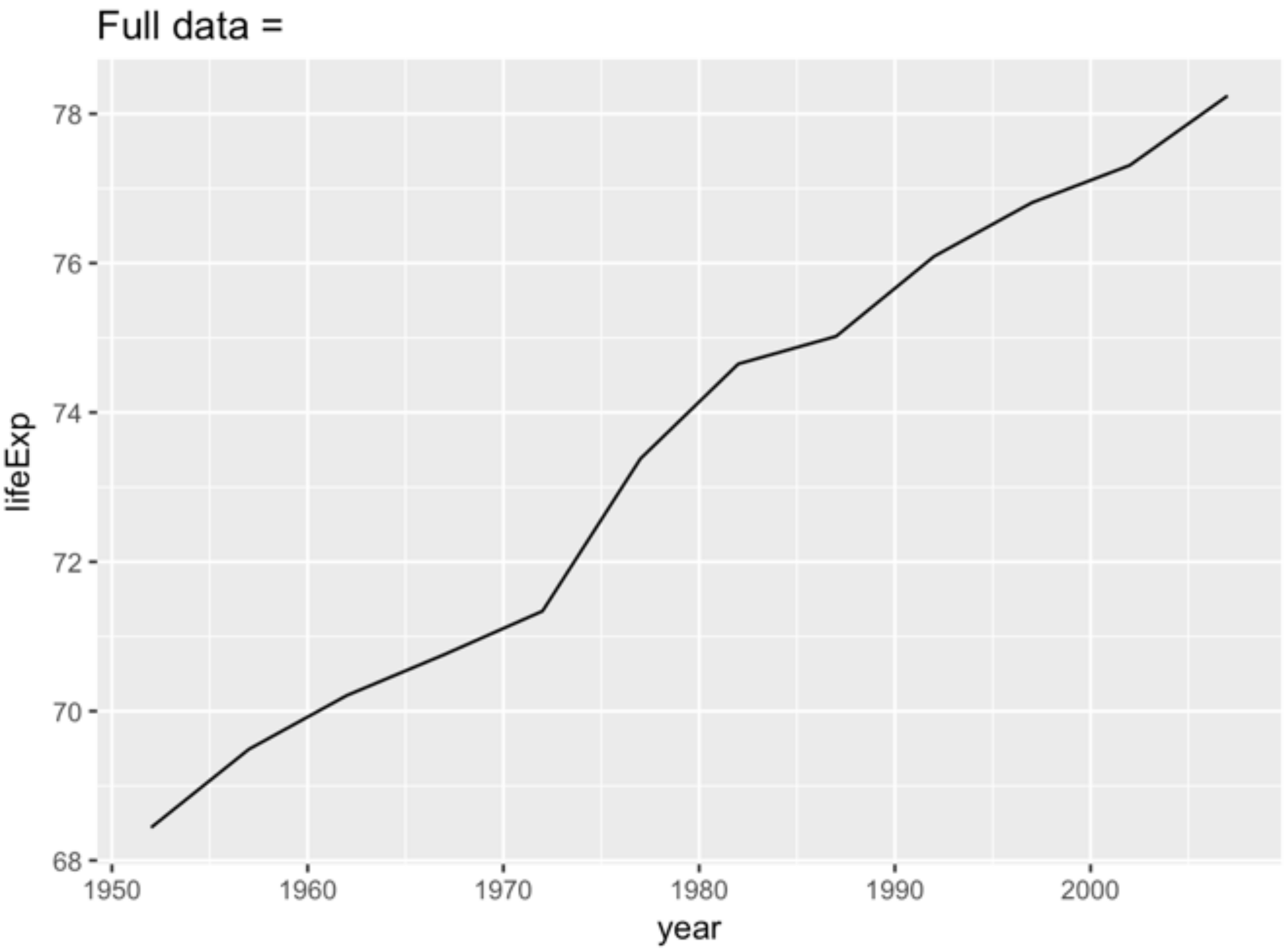
Numerically, it appears to fit very well but…

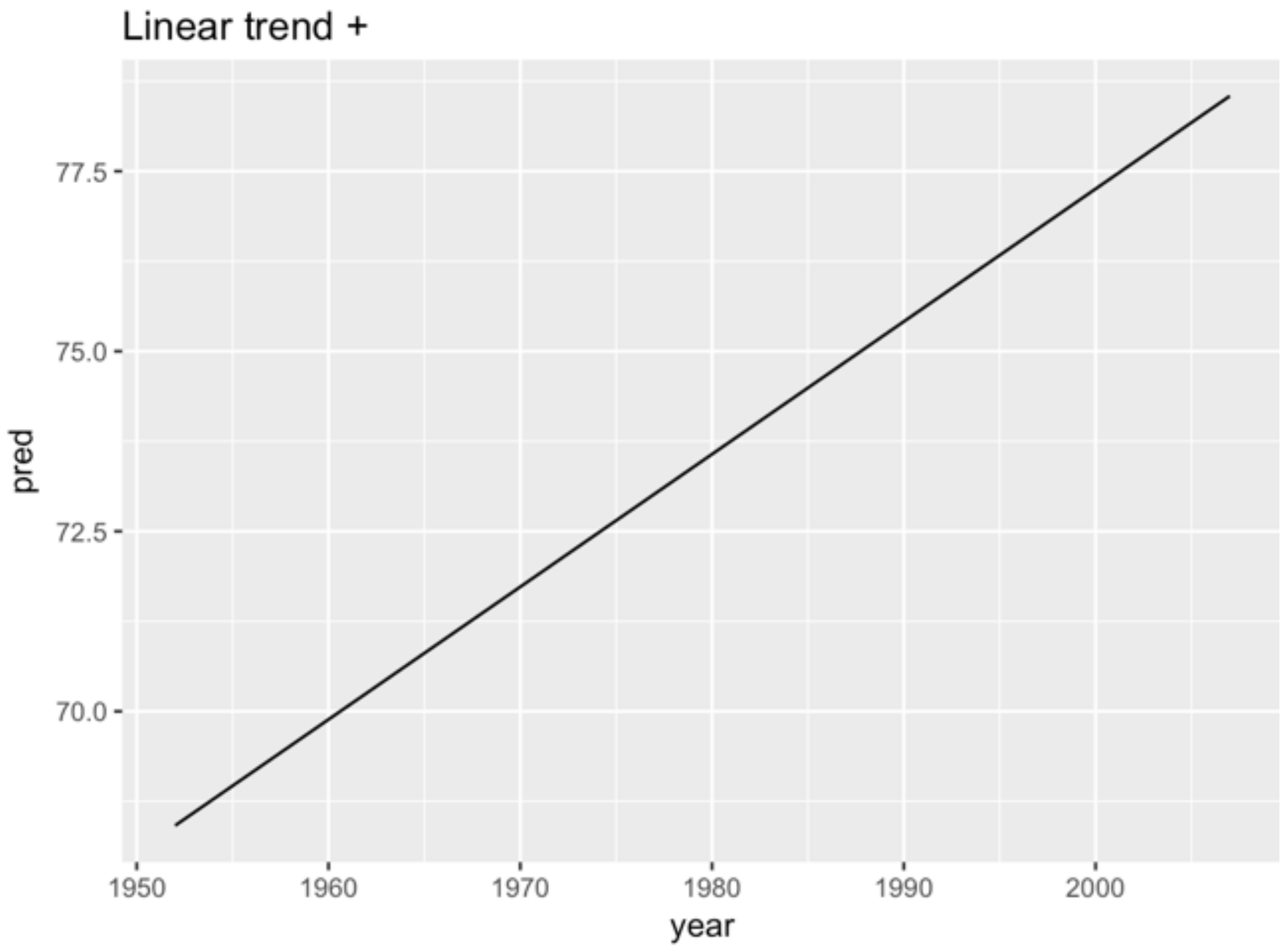
Visually, there are residual concerns in the latter year predictions

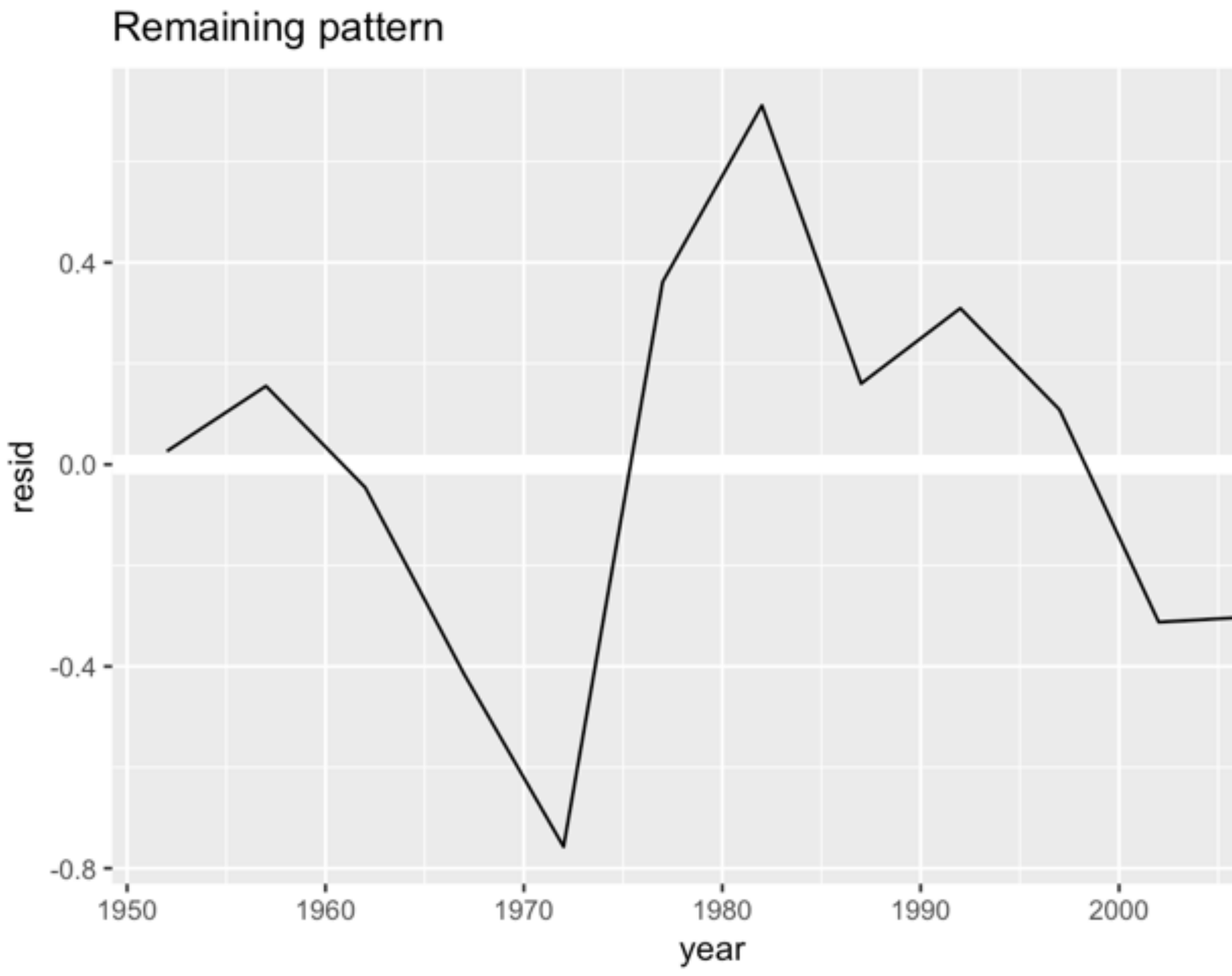
MODEL A SINGLE PART

```
usa <- filter(gapminder, country == "United States”)
usa_mod <- lm(lifeExp ~ year, data = usa)
```

Alternatively, we may want to just focus on a single country to see the particular relationship there







But what if we want to compare this country-level model across ***all*** *the countries?*

NESTED DATA

CONFOUNDING VARIABLE

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

by_country
# 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]>
```

Introducing a new data structure - the **nested** data frame

What is in each data column element? Can you figure out how to look at this data?

CONFOUNDING VARIABLE

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

by_country$data[[1]]
# A tibble: 12 × 4
    year lifeExp      pop gdpPercap
   <int>   <dbl>    <int>     <dbl>
1   1952  28.801  8425333  779.4453
2   1957  30.332  9240934  820.8530
3   1962  31.997 10267083  853.1007
4   1967  34.020 11537966  836.1971
5   1972  36.088 13079460  739.9811
6   1977  38.438 14880372  786.1134
7   1982  39.854 12881816  978.0114
```

Called: “`list-columns`”

- each element is a list
- interact with these elements just like you do a list

YOUR TURN!

Discuss with your neighbor how you could use this data structure (along with previously reviewed iterative functions) to apply a country-level model across each of the list-columns.

ITERATIVE MODEL APPLICATION

LET'S DEVELOP A MODEL FUNCTION

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

LET'S DEVELOP A MODEL FUNCTION

```
country_model <- function(df) {
  lm(lifeExp ~ year, data = df)
}
map(by_country$data, country_model)
[[1]]

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

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

Remember the **map** function?

We can apply this model over every element in the data column with **map**

How could we add this information to our nested data frame?

LET'S DEVELOP A MODEL FUNCTION

```
country_model <- function(df) {
  lm(lifeExp ~ year, data = df)
}
by_country <- by_country %>%
  mutate(model = map(data, country_model))

by_country
# A tibble: 142 × 4
      country continent               data    model
       <fctr>    <fctr>             <list>   <list>
1 Afghanistan      Asia <tibble [12 × 4]> <S3: lm>
2     Albania    Europe <tibble [12 × 4]> <S3: lm>
3     Algeria    Africa <tibble [12 × 4]> <S3: lm>
4      Angola    Africa <tibble [12 × 4]> <S3: lm>
```

Using **mutate** will save these regression results in a new list-column **model** variable

Now we have all our model results neatly packaged together with each country

UNNESTING

GETTING STUFF OUT OF OUR NEST

```
by_country %>%
  mutate(resids = map2(data, model, add_residuals))
       country continent              data     model               resids
        <fctr>    <fctr>            <list>    <list>               <list>
1  Afghanistan      Asia <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
2      Albania    Europe <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
3      Algeria    Africa <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
4       Angola    Africa <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
5    Argentina  Americas <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
6    Australia   Oceania <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
7      Austria    Europe <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
8      Bahrain      Asia <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
9   Bangladesh      Asia <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
10     Belgium    Europe <tibble [12 × 4]> <S3: lm> <tibble [12 × 5]>
# ... with 132 more rows
```

Using **map2**

- Similar to the **map** function
- But uses two argument inputs to map a function over (i.e. data, model)

Can you figure out what this code is doing?

GETTING STUFF OUT OF OUR NEST

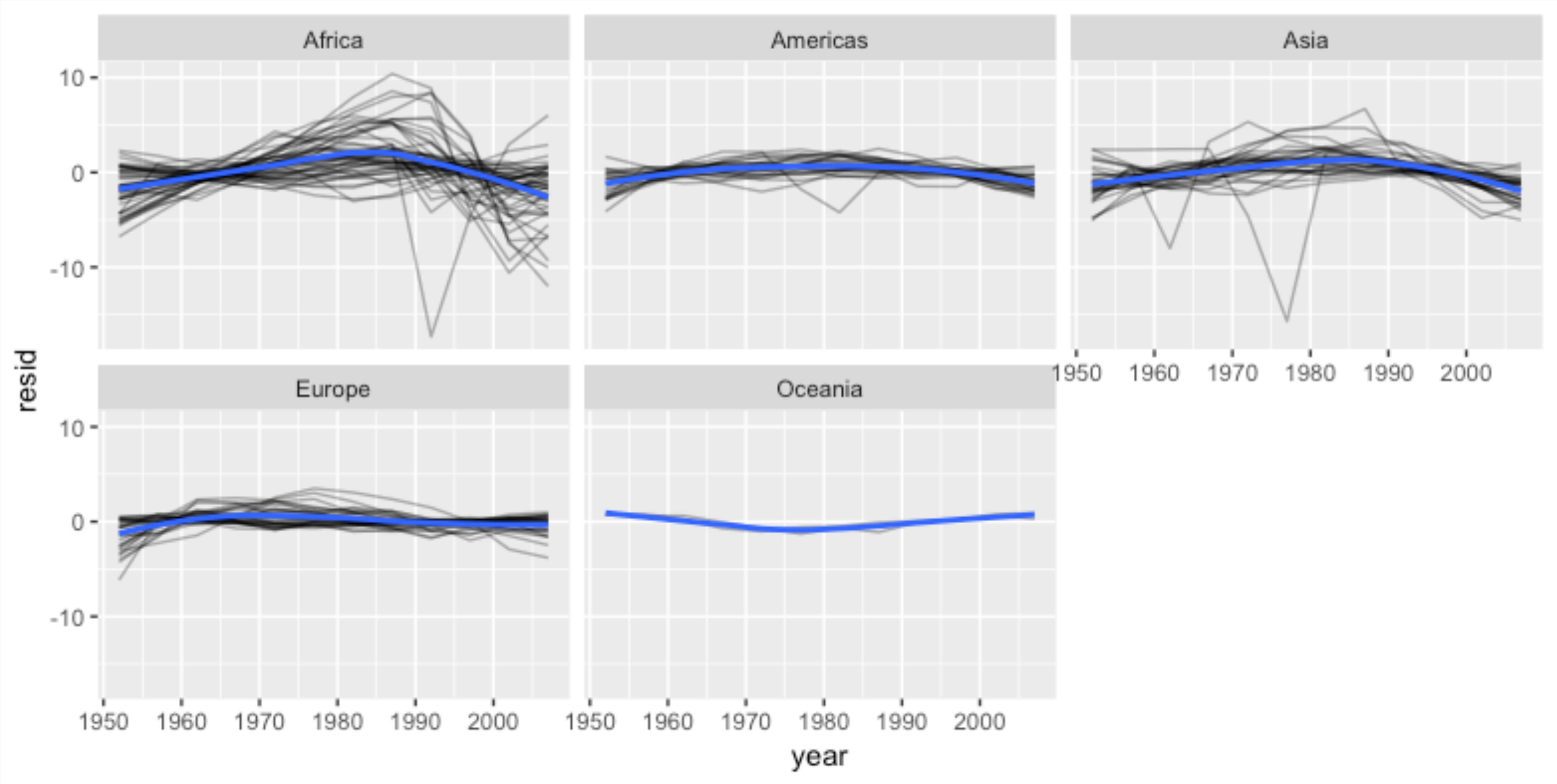
```
by_country %>%
  mutate(resids = map2(data, model, add_residuals)) %>%
  unnest(resids)
# A tibble: 1,704 × 7
       country continent  year lifeExp      pop gdpPercap       resid
        <fctr>    <fctr> <int>   <dbl>    <int>     <dbl>       <dbl>
1  Afghanistan      Asia  1952  28.801  8425333  779.4453 -1.10629487
2  Afghanistan      Asia  1957  30.332  9240934  820.8530 -0.95193823
3  Afghanistan      Asia  1962  31.997 10267083  853.1007 -0.66358159
4  Afghanistan      Asia  1967  34.020 11537966  836.1971 -0.01722494
5  Afghanistan      Asia  1972  36.088 13079460  739.9811  0.67413170
6  Afghanistan      Asia  1977  38.438 14880372  786.1134  1.64748834
7  Afghanistan      Asia  1982  39.854 12881816  978.0114  1.68684499
8  Afghanistan      Asia  1987  40.822 13867957  852.3959  1.27820163
9  Afghanistan      Asia  1992  41.674 16317921  649.3414  0.75355828
10 Afghanistan      Asia  1997  41.763 22227415  635.3414 -0.53408508
# ... with 1,694 more rows
```

…but our data is still nested

We can use `unnest` to extract the nested information of choice and convert back to a regular data frame

GETTING STUFF OUT OF OUR NEST

```
by_country %>%
  mutate(resids = map2(data, model, add_residuals)) %>%
  unnest(resids) %>%
  ggplot(aes(year, resid)) +
  geom_line(aes(group = country), alpha = 1 / 3) +
  geom_smooth(se = FALSE) +
  facet_wrap(~ continent)
```



This allows us to flow right into our normal visualization of residuals to compare across continents and countries

MODEL QUALITY

ASSESSING OUR MODEL(S)

```
usa <- filter(gapminder, country == "United States”)
usa_mod <- lm(lifeExp ~ year, data = usa)
```

Remember our single country model?

How would you normally assess model performance (numerically)?

ASSESSING OUR MODEL(S)

```
usa <- filter(gapminder, country == "United States”)
usa_mod <- lm(lifeExp ~ year, data = usa)
summary(usa_mod)
Call:
lm(formula = lifeExp ~ year, data = usa)

Residuals:
     Min       1Q   Median       3Q      Max
-0.75723 -0.30394  0.06735  0.19752  0.71108

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -291.08449   13.77740  -21.13 1.25e-09 ***
year           0.18417    0.00696   26.46 1.37e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Remember our single country model?

`summary()`

ASSESSING OUR MODEL(S)

```
usa <- filter(gapminder, country == "United States”)
usa_mod <- lm(lifeExp ~ year, data = usa)
broom::glance(usa_mod)
# A tibble: 1 × 11
  r.squared adj.r.squared     sigma statistic
      <dbl>         <dbl>     <dbl>     <dbl>
1 0.9859202     0.9845122 0.4161339  700.2351
# ... with 7 more variables: p.value <dbl>, df <int>,
#   logLik <dbl>, AIC <dbl>, BIC <dbl>,
#   deviance <dbl>, df.residual <int>
```

An alternative approach is with **`broom::glance`**

Creates a tidy one-row data frame with useful model results

ASSESSING OUR MODEL(S)

```
by_country
# A tibble: 142 × 4
        country continent                   data     model
         <fctr>    <fctr>                 <list>    <list>
1   Afghanistan      Asia <tibble [12 × 4]> <S3: lm>
2       Albania    Europe <tibble [12 × 4]> <S3: lm>
3       Algeria    Africa <tibble [12 × 4]> <S3: lm>
4        Angola    Africa <tibble [12 × 4]> <S3: lm>
5     Argentina  Americas <tibble [12 × 4]> <S3: lm>
6     Australia   Oceania <tibble [12 × 4]> <S3: lm>
7       Austria    Europe <tibble [12 × 4]> <S3: lm>
8       Bahrain      Asia <tibble [12 × 4]> <S3: lm>
9    Bangladesh      Asia <tibble [12 × 4]> <S3: lm>
10      Belgium    Europe <tibble [12 × 4]> <S3: lm>
# ... with 132 more rows
```

How could we use this with our many models approach?

ASSESSING OUR MODEL(S)

```
by_country %>%
  mutate(glance = map(model, broom::glance))
# A tibble: 142 × 5
       country continent              data    model                   glance
        <fctr>    <fctr>            <list>   <list>                   <list>
1  Afghanistan      Asia <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
2      Albania    Europe <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
3      Algeria    Africa <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
4       Angola    Africa <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
5    Argentina  Americas <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
6    Australia   Oceania <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
7      Austria    Europe <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
8      Bahrain      Asia <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
9   Bangladesh      Asia <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
10     Belgium    Europe <tibble [12 × 4]> <S3: lm> <data.frame [1 × 11]>
```

ASSESSING OUR MODEL(S)

```
by_country %>%
  mutate(glance = map(model, broom::glance)) %>%
  unnest(glance, .drop = TRUE)
# A tibble: 142 × 13
       country continent r.squared adj.r.squared     sigma  statistic      p.value
        <fctr>    <fctr>     <dbl>         <dbl>     <dbl>      <dbl>        <dbl>
1  Afghanistan      Asia 0.9477123     0.9424835 1.2227880  181.24941 9.835213e-08
2      Albania    Europe 0.9105778     0.9016355 1.9830615  101.82901 1.462763e-06
3      Algeria    Africa 0.9851172     0.9836289 1.3230064  661.91709 1.808143e-10
4       Angola    Africa 0.8878146     0.8765961 1.4070091   79.13818 4.593498e-06
```

This allows us to quickly assess and compare the performance of many models!

YOUR TURN!

Using the unnested glance data:

1. *Can you find the country models with the highest adjusted R^2? What about the lowest?*

2. *Plot the adjusted R^2 against each continent? What do you find?*

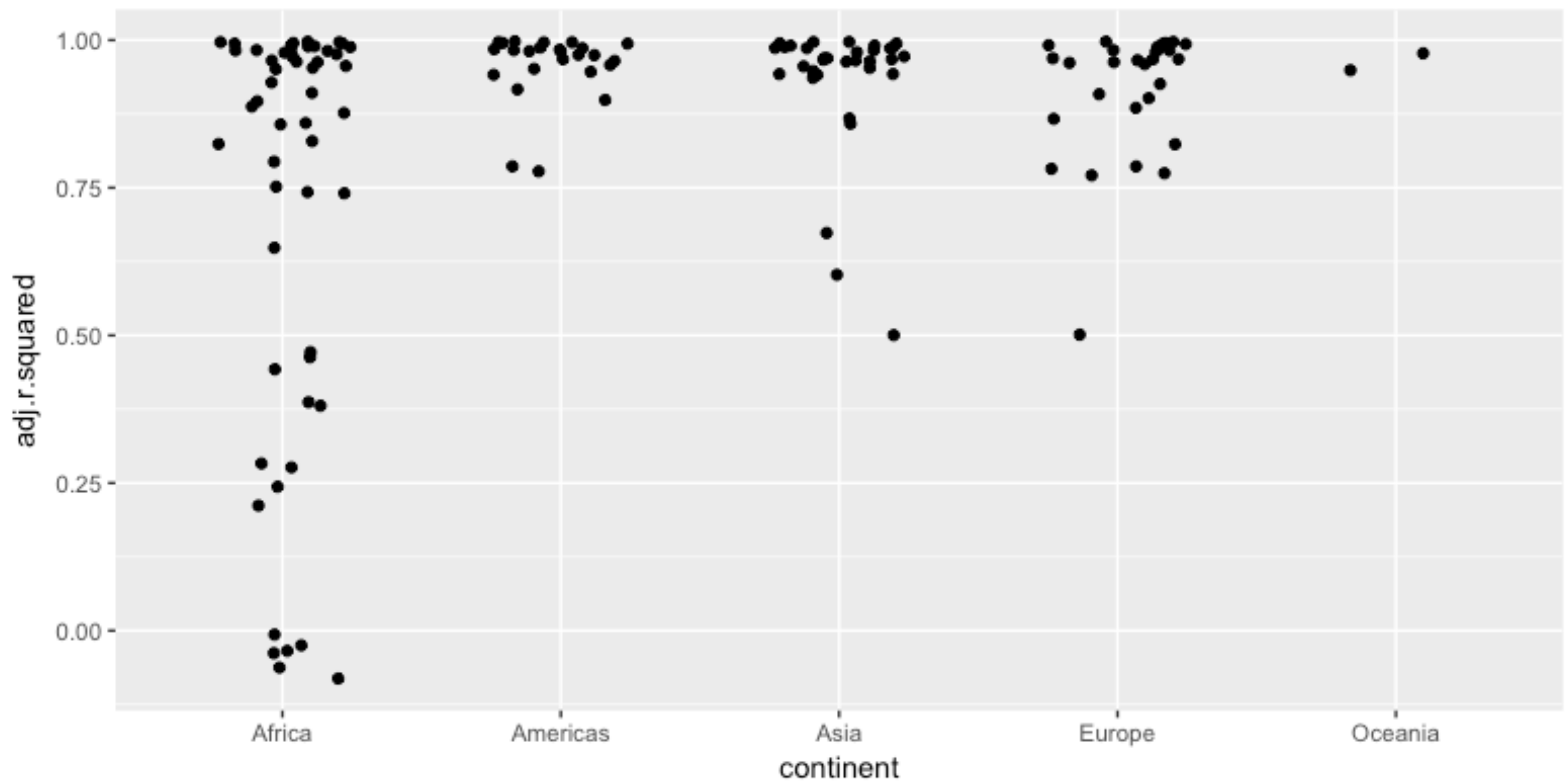
3. *Filter for adjusted R2 < 0.25. What countries do you find? What do you think is driving this bad fit? (Hint: plot the life expectancy over time for these countries)*

SOLUTION

```
# Can you find the country models with the highest adjusted R²?  What about the lowest?
by_country %>%
  mutate(glance = map(model, broom::glance)) %>%
  unnest(glance, .drop = TRUE) %>%
  arrange(adj.r.squared)
# A tibble: 142 × 13
         country continent  r.squared adj.r.squared    sigma statistic    p.value
          <fctr>    <fctr>      <dbl>         <dbl>    <dbl>     <dbl>      <dbl>
1         Rwanda    Africa 0.01715964  -0.081124401 6.558269 0.1745923 0.68489269
2       Botswana    Africa 0.03402340  -0.062574259 6.112177 0.3522177 0.56604135
3       Zimbabwe    Africa 0.05623196  -0.038144842 7.205431 0.5958240 0.45802898
4         Zambia    Africa 0.05983644  -0.034179918 4.528713 0.6364471 0.44353178
5      Swaziland    Africa 0.06821087  -0.024968046 6.644091 0.7320419 0.41225300
6        Lesotho    Africa 0.08485635  -0.006658011 5.933934 0.9272463 0.35828637
7  Cote d'Ivoire    Africa 0.28337240   0.211709644 3.925590 3.9542491 0.07480350
```

SOLUTION

```
# Plot the adjusted R² against each continent? What do you find?
by_country %>%
  mutate(glance = map(model, broom::glance)) %>%
  unnest(glance, .drop = TRUE) %>%
  ggplot(aes(continent, adj.r.squared)) +
  geom_jitter(width = 0.25)
```



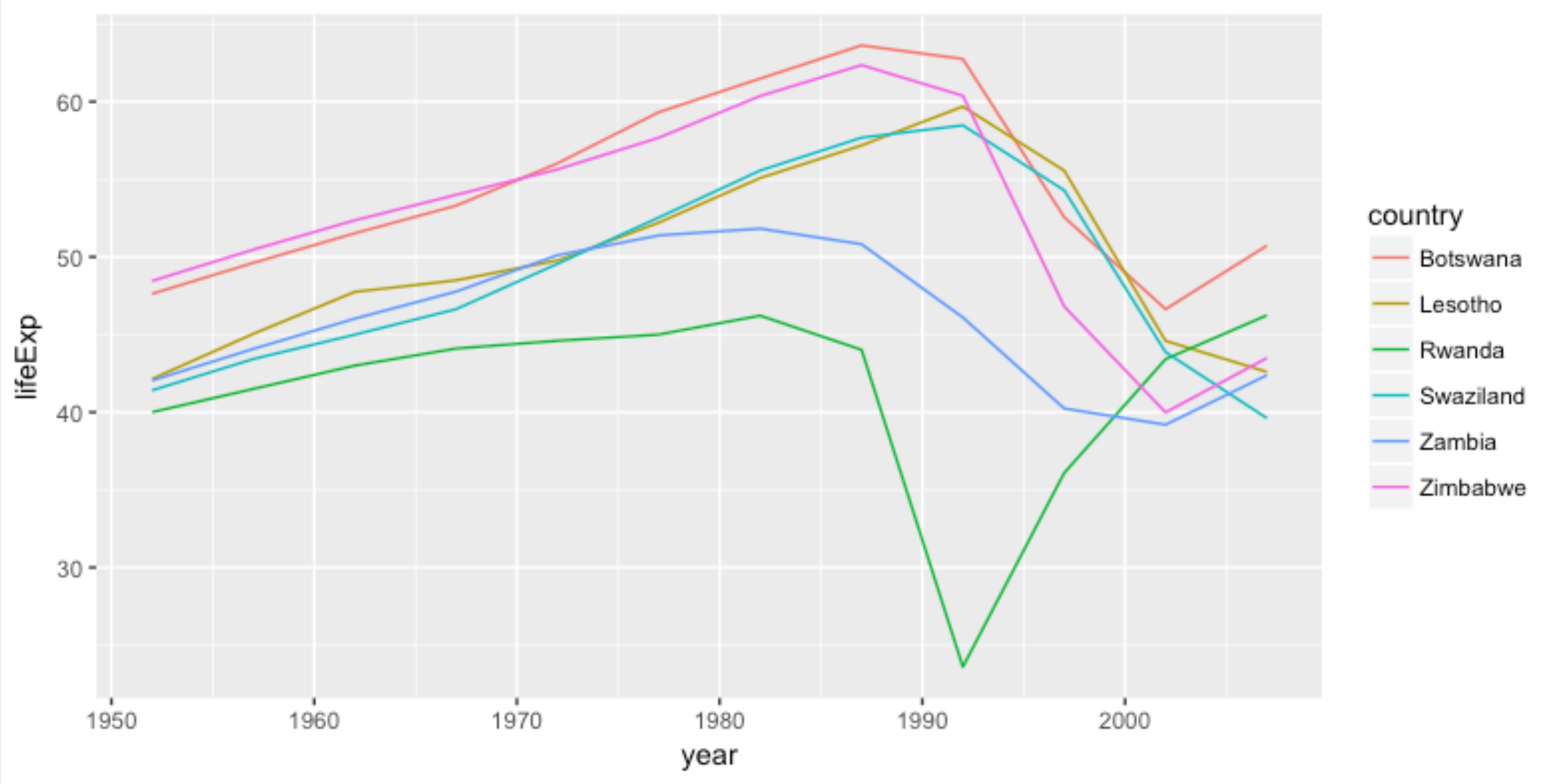
SOLUTION

```
# Filter for adjusted R2 < 0.25.  What countries do you find?
bad_fit <- by_country %>%
  mutate(glance = map(model, broom::glance)) %>%
  unnest(glance, .drop = TRUE) %>%
  filter(r.squared < 0.25)

bad_fit
# A tibble: 6 × 13
    country continent  r.squared adj.r.squared    sigma statistic   p.value    df
     <fctr>    <fctr>      <dbl>         <dbl>    <dbl>     <dbl>     <dbl> <int>
1  Botswana    Africa 0.03402340  -0.062574259 6.112177 0.3522177 0.5660414     2
2   Lesotho    Africa 0.08485635  -0.006658011 5.933934 0.9272463 0.3582864     2
3    Rwanda    Africa 0.01715964  -0.081124401 6.558269 0.1745923 0.6848927     2
4 Swaziland    Africa 0.06821087  -0.024968046 6.644091 0.7320419 0.4122530     2
5    Zambia    Africa 0.05983644  -0.034179918 4.528713 0.6364471 0.4435318     2
```

SOLUTION

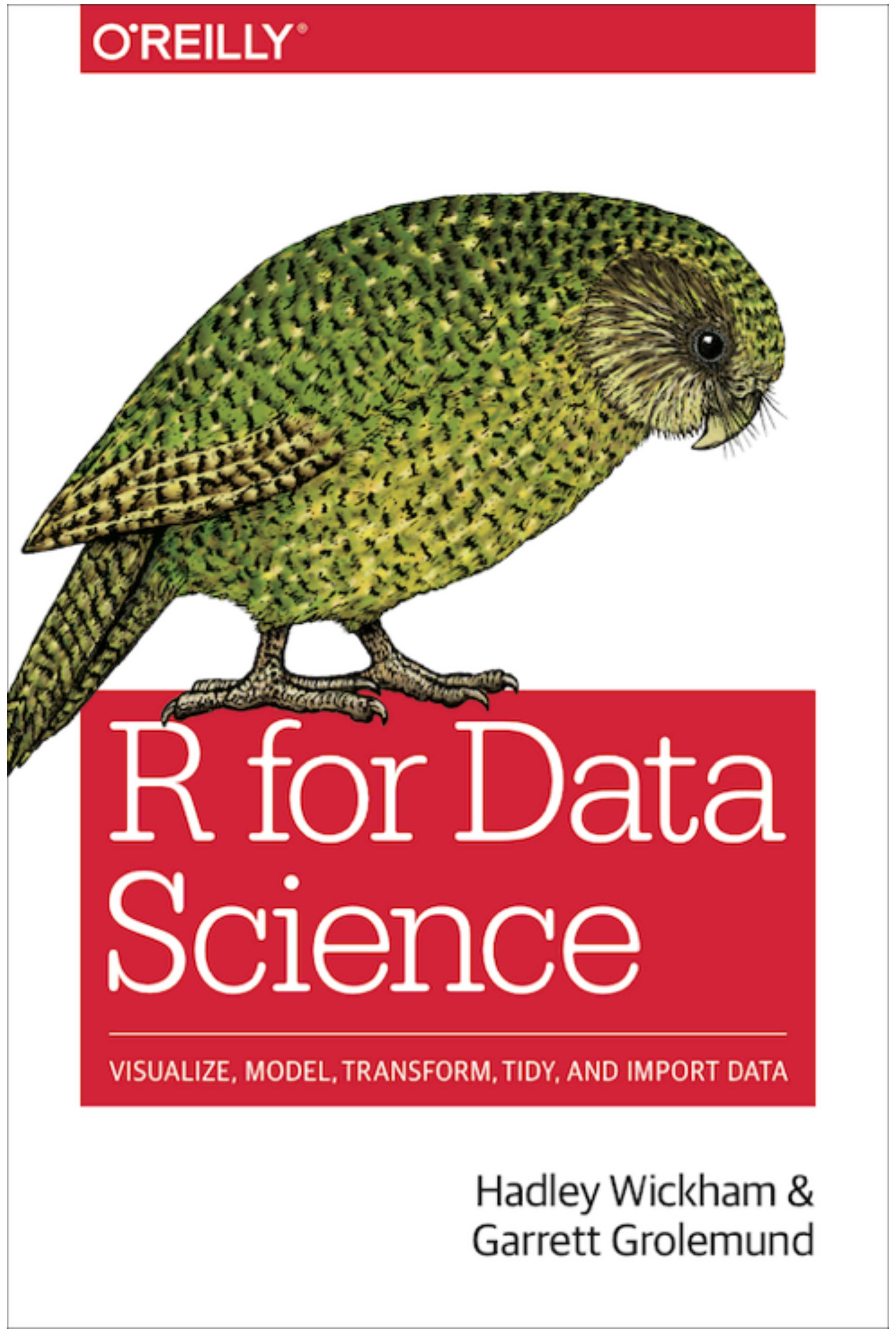
```
# plot the life expectancy over time for these countries
gapminder %>%
  semi_join(bad_fit, by = "country") %>%
  ggplot(aes(year, lifeExp, colour = country)) +
  geom_line()
```

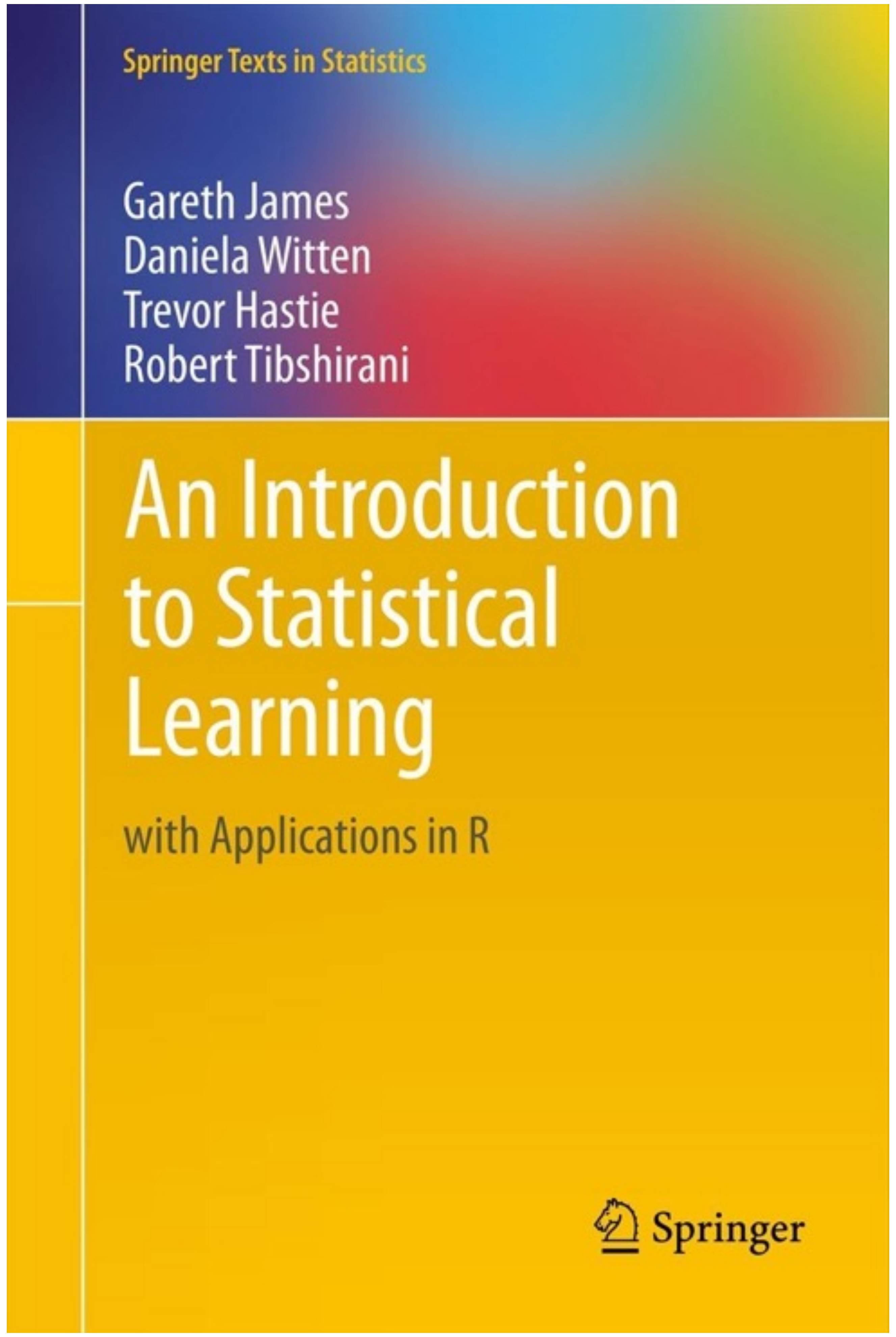


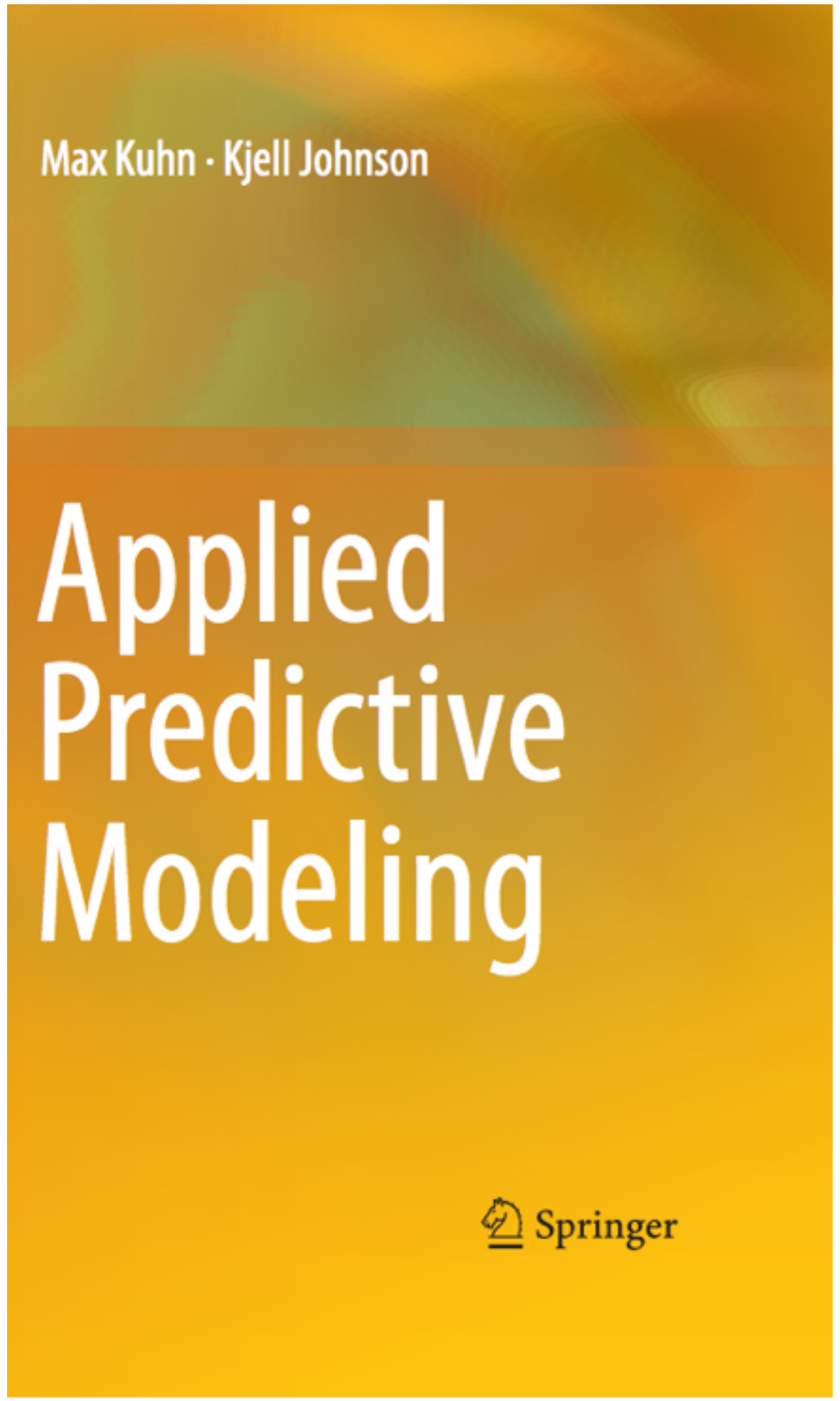
likely the tragedies of the HIV/AIDS epidemic and the Rwandan genocide

SO LITTLE TIME!

LEARN MORE


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham &
Garrett Grolemund


Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer


Max Kuhn · Kjell Johnson
Applied Predictive Modeling
Springer

WHAT TO REMEMBER

FUNCTIONS TO REMEMBER

Operator/Function	Description
`nest`	Create a nested data frame with list-columns
`map2`	Similar to the map but will map a specified function over two data inputs
`unnest`	Unnest our data
`broom::glance`	Extract model quality metrics into a tidy data structure