L16 Many Models

Data Science I (STAT 301-1)

Shay Lebovitz

Contents

Overview

The goal of this lab is for students to explore and develop three powerful ideas/techniques:

1. Using many simple models to better understand complex data.
2. Using list-columns to store arbitrary data structures.
3. Using the `broom` package to turn models into tidy data.

These skills and ideas will be essential as we proceed to data science work that is beyond EDA.

Datasets

We will be utilizing the `gapminder` dataset from the `gapminder` package (you may need to install this package). See `?gapminder` for details concerning the dataset.

Exercises

Please complete the following exercises. Be sure that your solutions are clearly indicated and that the document is neatly formatted.

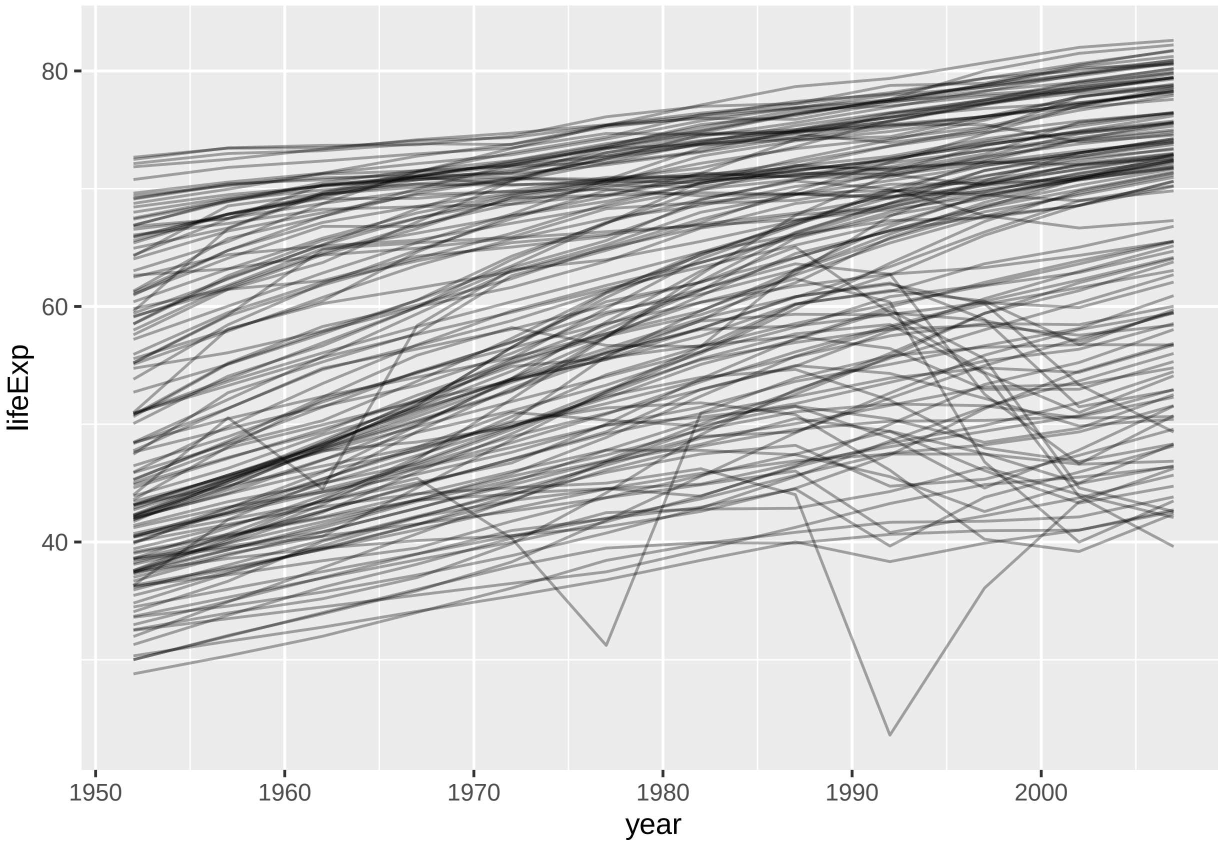
```
# Loading package(s)
library(modelr)
library(tidyverse)
library(gapminder)
library(purrr)
```

Load Packages

Exercise 1

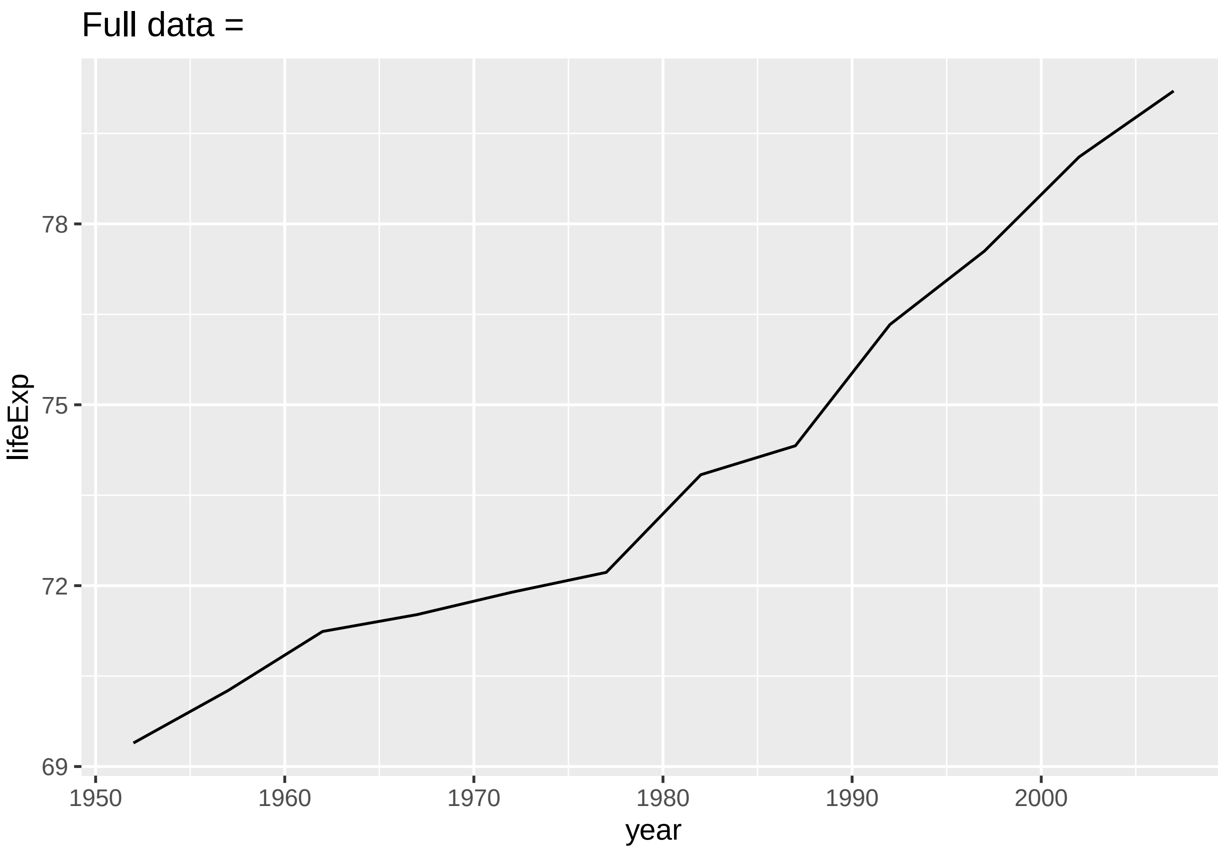
Work through and transcribe (do not copy and paste – rewrite) the work detailed in section 25.2 gapminder. Consider adding your own comments/notes and using your own naming conventions.

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

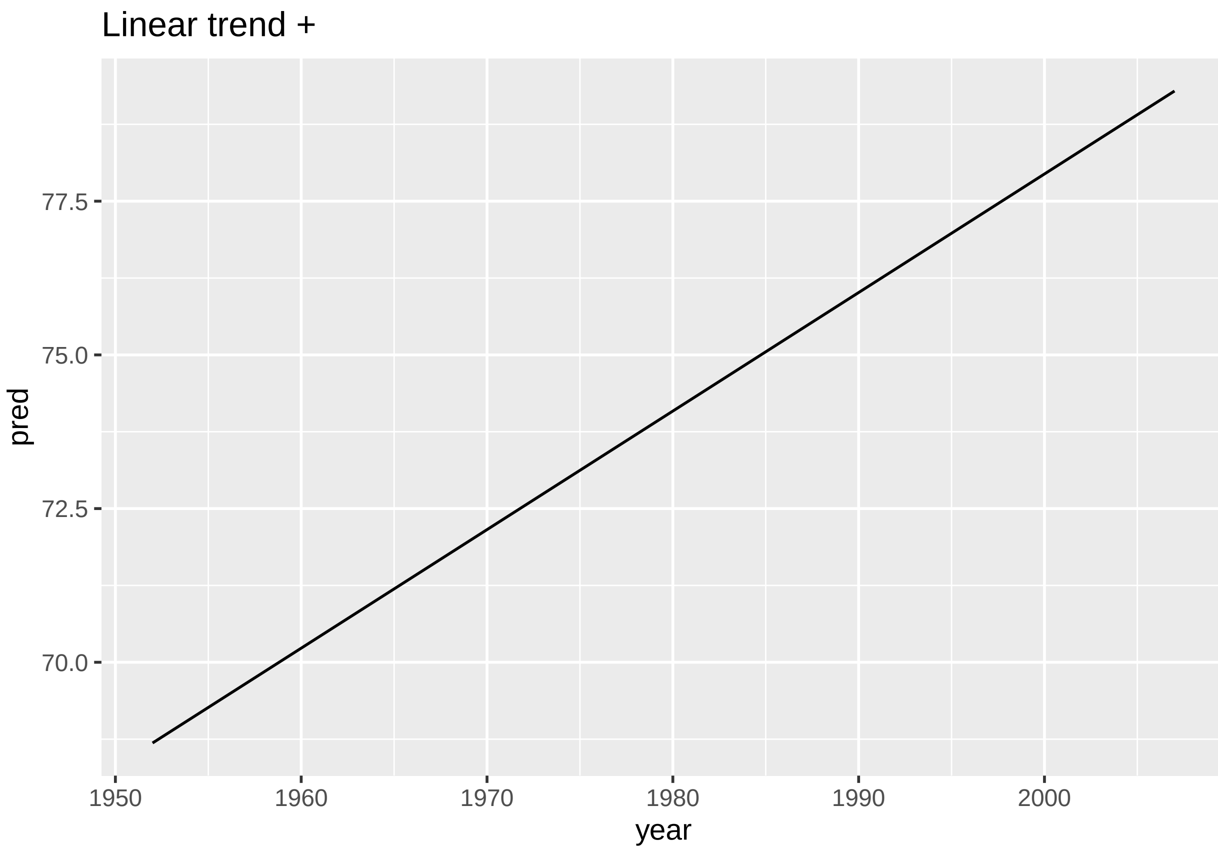


Overall we see life expectancy improving over the years, but some countries have some weird things happening. We want to explore that further. This is simple if we only have one country.

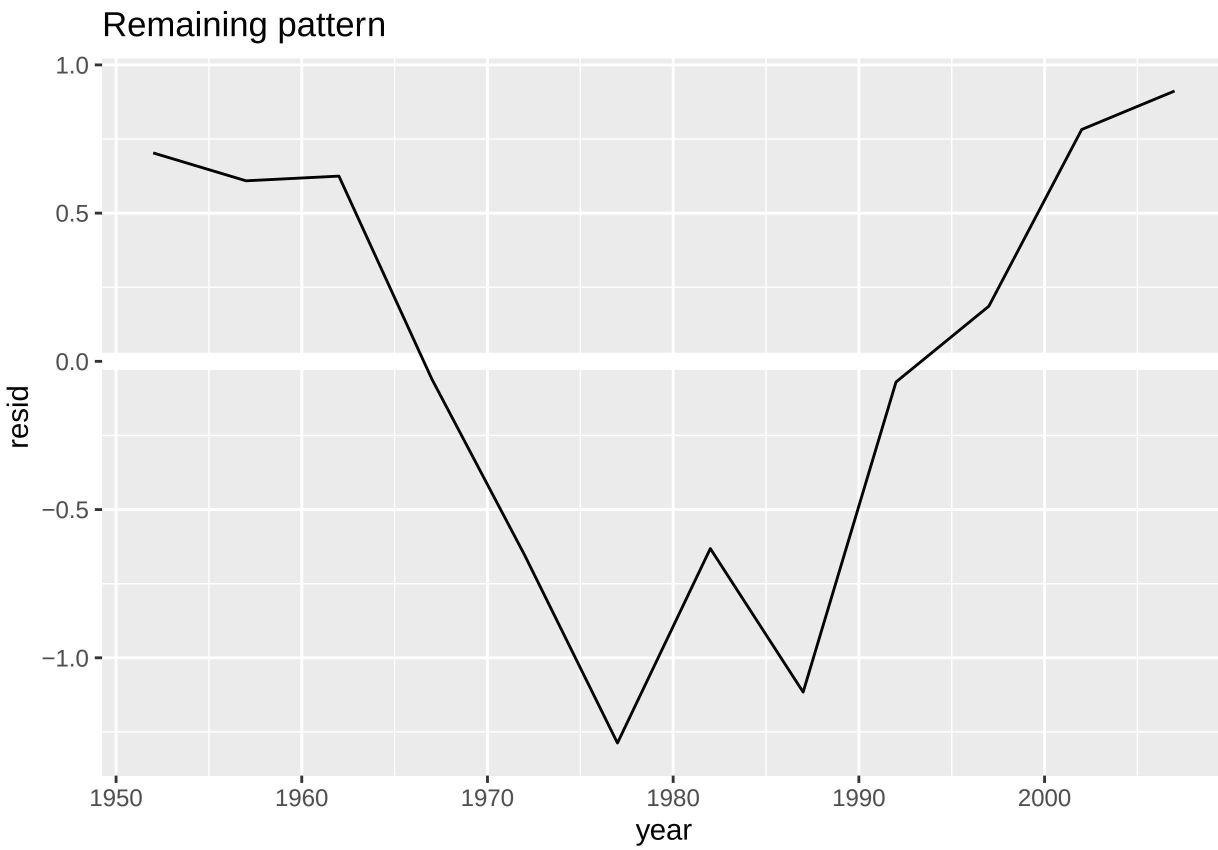
```r
nz <- filter(gapminder, country == "New Zealand")
nz %>%
  ggplot(aes(year, lifeExp)) +
  geom_line() +
  ggtitle("Full data = ")
```




```
nz_mod <- lm(lifeExp ~ year, data = nz)
nz %>%
  add_predictions(nz_mod) %>%
  ggplot(aes(year, pred)) +
  geom_line() +
  ggtitle("Linear trend + ")
```




```
nz %>%
  add_residuals(nz_mod) %>%
  ggplot(aes(year, resid)) +
  geom_hline(yintercept = 0, colour = "white", size = 3) +
  geom_line() +
  ggtitle("Remaining pattern")
```



To do so for every country, we need a nested data frame.

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

by_country
```

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

We see that every country has a row, and the `data` column are nested data frames containing that country's observations.

We can now create a function that will create a model:

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

Apply it to the `by_country` data frame, and add it as an additional column.

```r
models <- map(by_country$data, country_model)
by_country <- by_country %>%
  mutate(model = map(data, country_model))
by_country
```

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

Next, we can compute the residuals of each model and add it to the data frame.

```r
by_country <- by_country %>%
  mutate(
    resids = map2(data, model, add_residuals)
  )
by_country
```

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

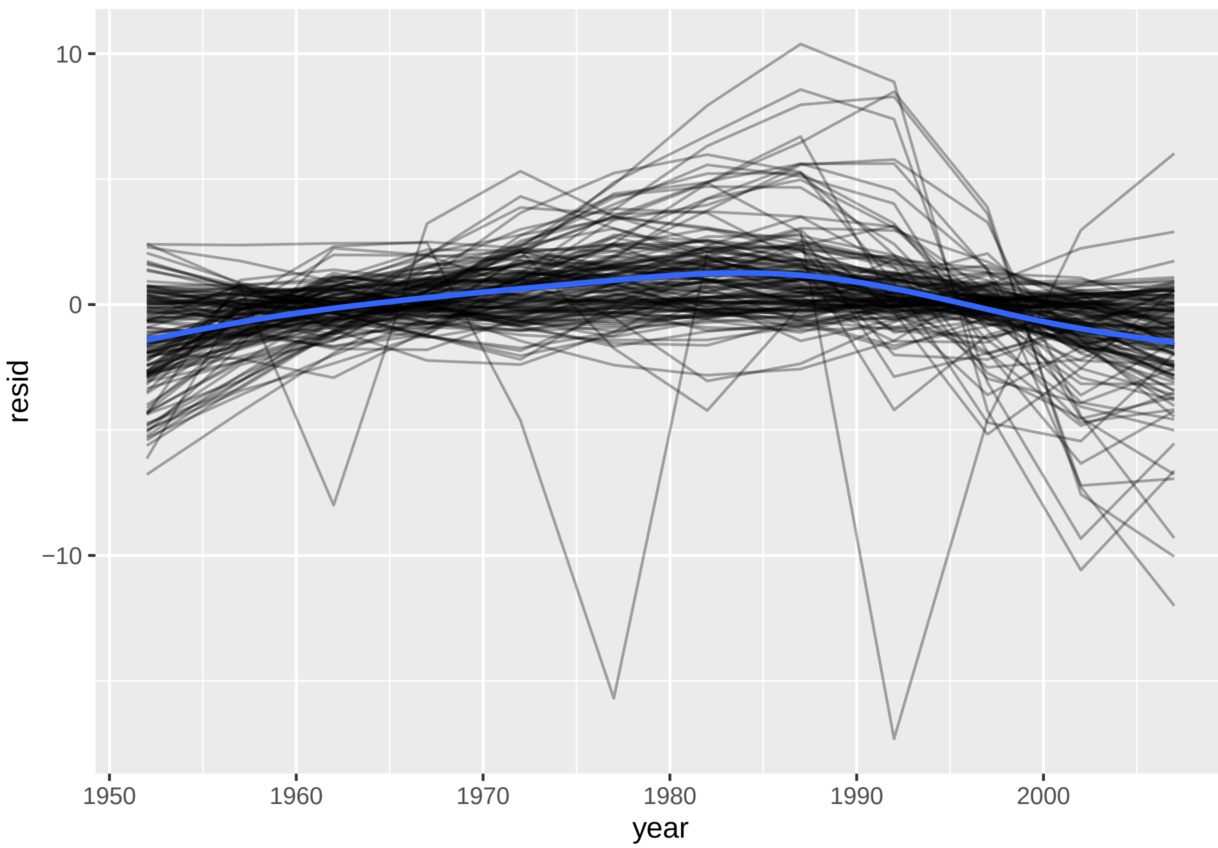
To plot the data, we need to undo the nesting with `unnest()`.

```r
resids <- unnest(by_country, resids)
resids
```

```
## # A tibble: 1,704 x 9
```

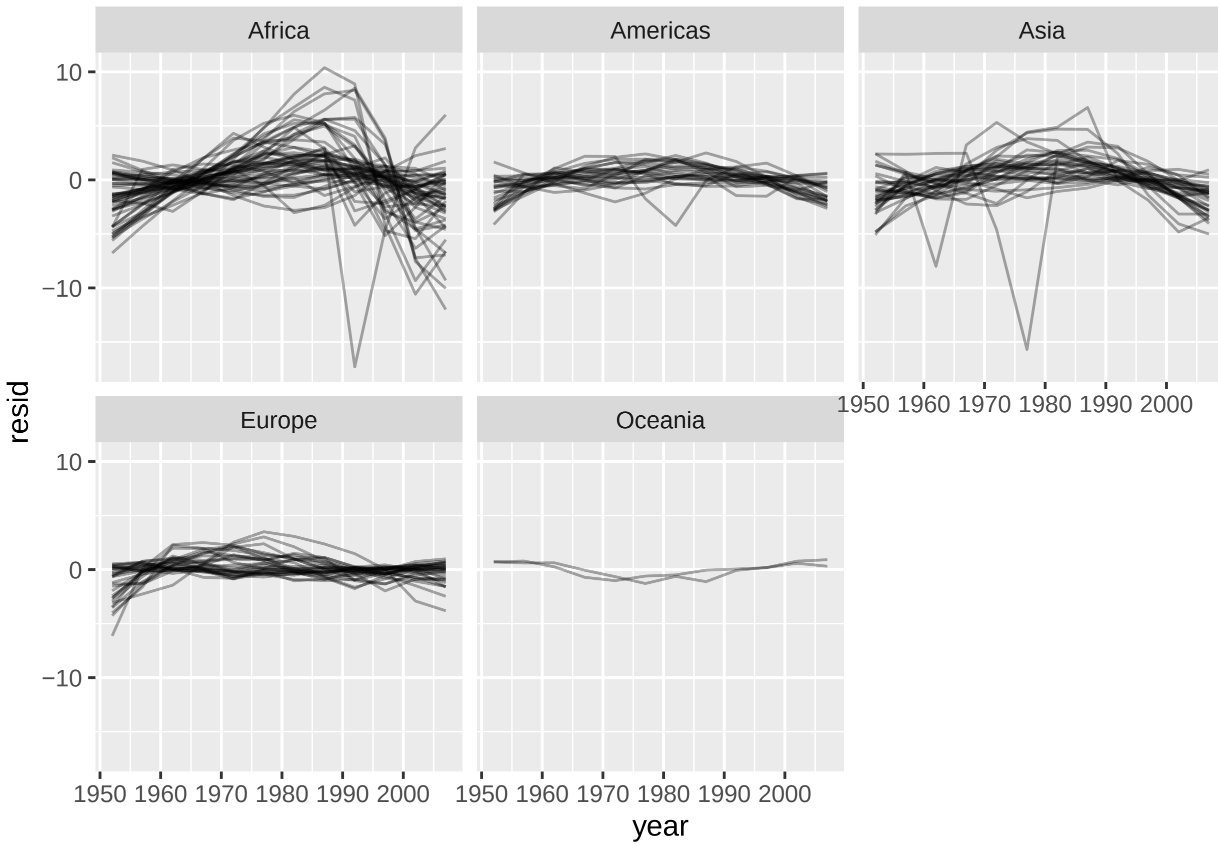
```
## # Groups:   country, continent [142]
##    country     continent data         model   year lifeExp     pop gdpPercap   resid
##    <fct>       <fct>     <list>       <list> <int>   <dbl>   <int>     <dbl>   <dbl>
##  1 Afghanis~ Asia      <tibble [1~ <lm>     1952    28.8 8.43e6      779. -1.11
##  2 Afghanis~ Asia      <tibble [1~ <lm>     1957    30.3 9.24e6      821. -0.952
##  3 Afghanis~ Asia      <tibble [1~ <lm>     1962    32.0 1.03e7      853. -0.664
##  4 Afghanis~ Asia      <tibble [1~ <lm>     1967    34.0 1.15e7      836. -0.0172
##  5 Afghanis~ Asia      <tibble [1~ <lm>     1972    36.1 1.31e7      740.  0.674
##  6 Afghanis~ Asia      <tibble [1~ <lm>     1977    38.4 1.49e7      786.  1.65
##  7 Afghanis~ Asia      <tibble [1~ <lm>     1982    39.9 1.29e7      978.  1.69
##  8 Afghanis~ Asia      <tibble [1~ <lm>     1987    40.8 1.39e7      852.  1.28
##  9 Afghanis~ Asia      <tibble [1~ <lm>     1992    41.7 1.63e7      649.  0.754
## 10 Afghanis~ Asia      <tibble [1~ <lm>     1997    41.8 2.22e7      635. -0.534
## # ... with 1,694 more rows
```

```
resids %>%
  ggplot(aes(year, resid)) +
  geom_line(aes(group = country), alpha = 1 / 3) +
  geom_smooth(se = FALSE)
```



We can split this graph up by continents.

```
resids %>%
  ggplot(aes(year, resid, group = country)) +
  geom_line(alpha = 1 / 3) +
  facet_wrap(~continent)
```



Unsurprisingly, we see large residuals in Africa and Asia, but small residuals in the Americas and Europe. Not much can be said about Oceania, as there are only two countries. To get information on the quality of the models, we can use the `broom` package:

```
glance <- by_country %>%
  mutate(glance = map(model, broom::glance)) %>%
  unnest(glance, .drop = TRUE)
glance
```

```
## # A tibble: 142 x 17
## # Groups:   country, continent [142]
##    country continent data  model resids r.squared adj.r.squared sigma statistic
##    <fct>   <fct>     <lis> <lis> <list>     <dbl>         <dbl> <dbl>     <dbl>
##  1 Afghan~ Asia      <tib~ <lm>  <tibb~     0.948         0.942 1.22      181.
##  2 Albania Europe    <tib~ <lm>  <tibb~     0.911         0.902 1.98      102.
##  3 Algeria Africa    <tib~ <lm>  <tibb~     0.985         0.984 1.32      662.
##  4 Angola  Africa    <tib~ <lm>  <tibb~     0.888         0.877 1.41       79.1
##  5 Argent~ Americas  <tib~ <lm>  <tibb~     0.996         0.995 0.292    2246.
##  6 Austra~ Oceania   <tib~ <lm>  <tibb~     0.980         0.978 0.621     481.
##  7 Austria Europe    <tib~ <lm>  <tibb~     0.992         0.991 0.407    1261.
##  8 Bahrain Asia      <tib~ <lm>  <tibb~     0.967         0.963 1.64      291.
##  9 Bangla~ Asia      <tib~ <lm>  <tibb~     0.989         0.988 0.977     930.
## 10 Belgium Europe    <tib~ <lm>  <tibb~     0.995         0.994 0.293    1822.
## # ... with 132 more rows, and 8 more variables: p.value <dbl>, df <dbl>,
## #   logLik <dbl>, AIC <dbl>, BIC <dbl>, deviance <dbl>, df.residual <int>,
## #   nobs <int>
```

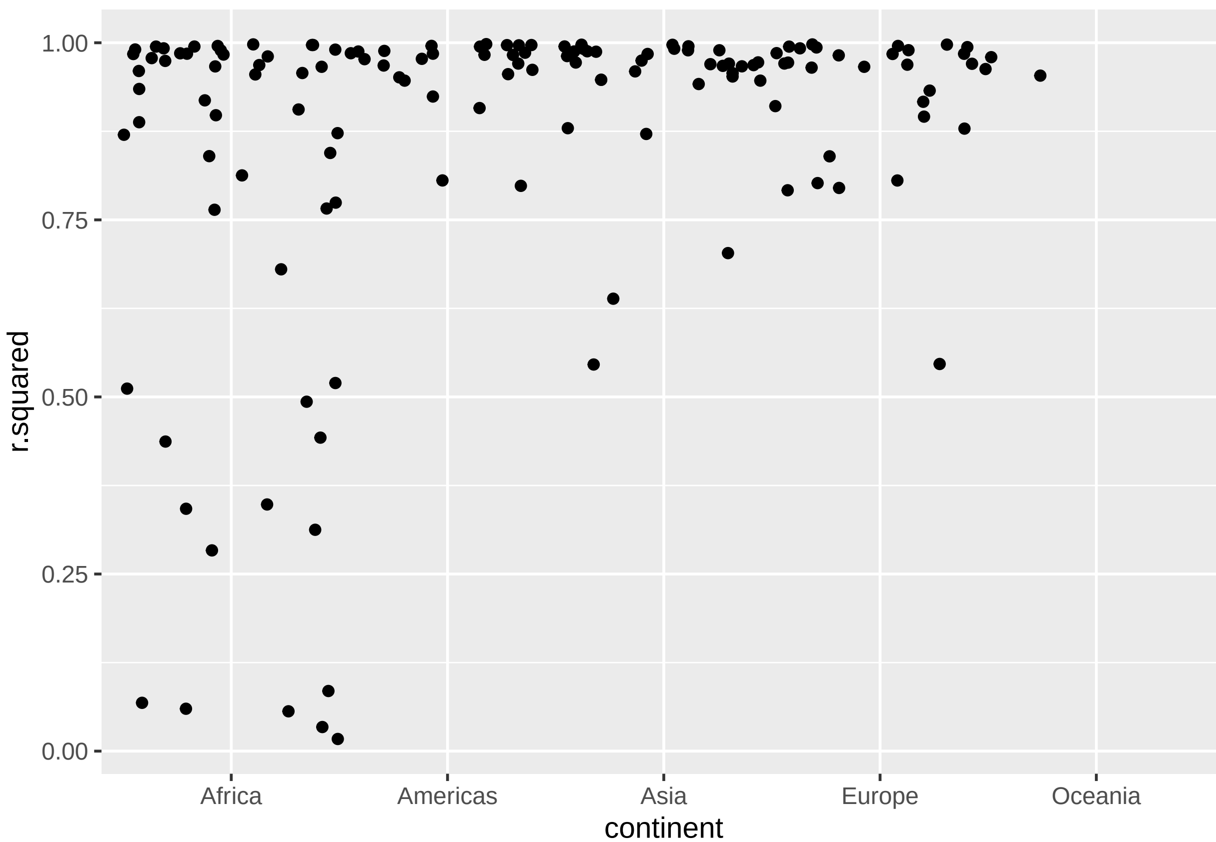
R-squared is a goodness-of-fit measure, so to see the poorest quality models, we can arrange by R-squared.

```
glance %>%
  arrange(r.squared)
```

```
## # A tibble: 142 x 17
## # Groups:   country, continent [142]
##    country continent data  model resids r.squared adj.r.squared sigma statistic
##    <fct>   <fct>     <lis> <lis> <list>     <dbl>         <dbl> <dbl>     <dbl>
##  1 Rwanda  Africa    <tib~ <lm>  <tibb~    0.0172      -0.0811   6.56     0.175
##  2 Botswa~ Africa    <tib~ <lm>  <tibb~    0.0340      -0.0626   6.11     0.352
##  3 Zimbab~ Africa    <tib~ <lm>  <tibb~    0.0562      -0.0381   7.21     0.596
##  4 Zambia  Africa    <tib~ <lm>  <tibb~    0.0598      -0.0342   4.53     0.636
##  5 Swazil~ Africa    <tib~ <lm>  <tibb~    0.0682      -0.0250   6.64     0.732
##  6 Lesotho Africa    <tib~ <lm>  <tibb~    0.0849      -0.00666  5.93     0.927
##  7 Cote d~ Africa    <tib~ <lm>  <tibb~    0.283        0.212    3.93     3.95
##  8 South ~ Africa    <tib~ <lm>  <tibb~    0.312        0.244    4.74     4.54
##  9 Uganda  Africa    <tib~ <lm>  <tibb~    0.342        0.276    3.19     5.20
## 10 Congo,~ Africa    <tib~ <lm>  <tibb~    0.348        0.283    2.43     5.34
## # ... with 132 more rows, and 8 more variables: p.value <dbl>, df <dbl>,
## #   logLik <dbl>, AIC <dbl>, BIC <dbl>, deviance <dbl>, df.residual <int>,
## #   nobs <int>
```

All the worst models are in Africa, so lets plot and see:

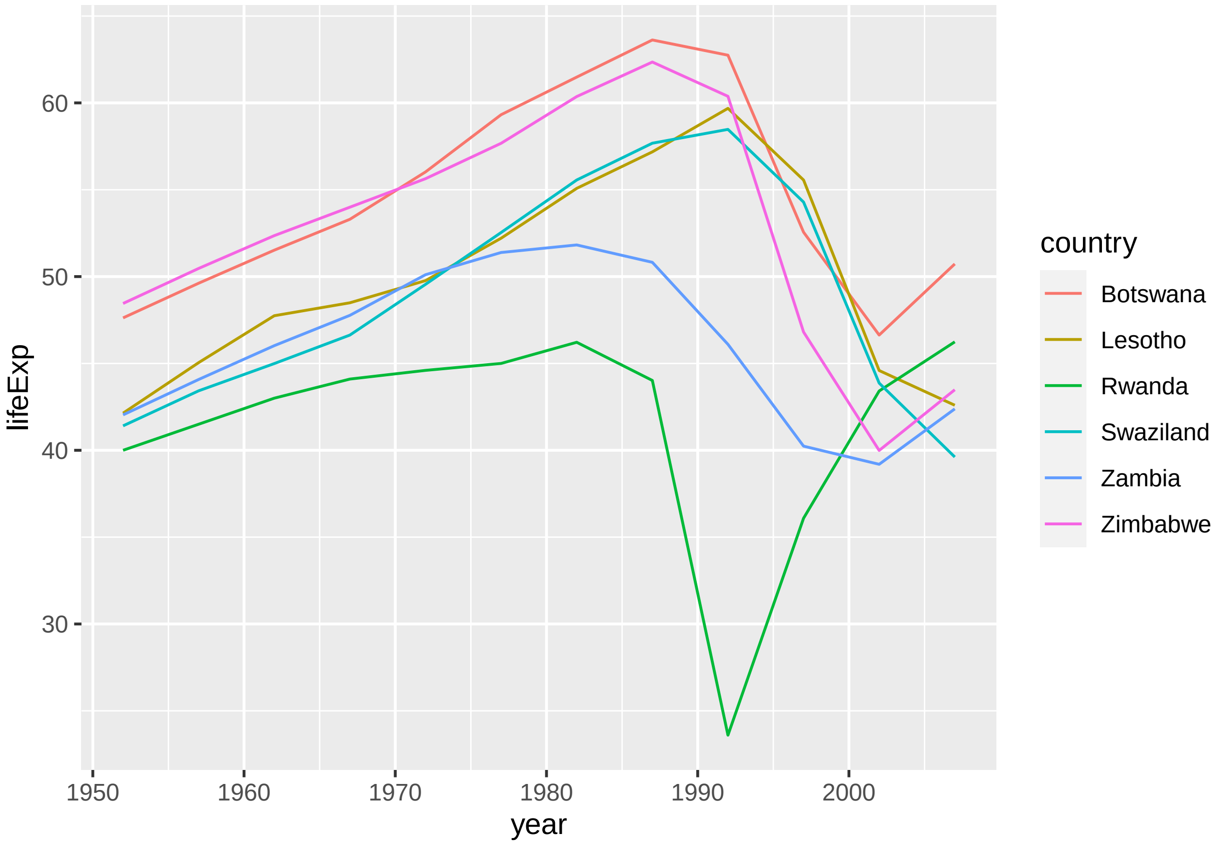
```
glance %>%
  ggplot(aes(continent, r.squared)) +
  geom_jitter(width = 0.5)
```



Finally, lets pull out the worst models and plot them:

```r
bad_fit <- filter(glance, r.squared < 0.25)

gapminder %>%
  semi_join(bad_fit, by = "country") %>%
  ggplot(aes(year, lifeExp, colour = country)) +
  geom_line()
```



The HIV/AIDS epidemic and the Rwandan genocide explain a lot of these trends.

Exercise 2 (Website: 25.2.5 Ex. 3)

To create the last plot in section 25.2 gapminder (showing the data for the countries with the worst model fits according to R^2), we needed two steps; we created a data frame with one row per country and then semi-joined it to the original dataset. It's possible to avoid using this join if we use `unnest()` instead of `unnest(.drop = TRUE)` when creating the `glance` tibble. Demonstrate how this could be done.

(Hint: You might have to use another `unnest()` call. Also note that you won't need the `bad_fit` tibble, since you will be able to pipe directly into the `filter()`.)

```r
gapminder %>%
  group_by(country, continent) %>%
  nest() %>%
  mutate(model = map(data, ~lm(lifeExp ~ year, .))) %>%
  mutate(glance = map(model, broom::glance)) %>%
  unnest(data) %>%
  unnest(glance) %>%
  filter(r.squared < 0.25) %>%
```

```r
ggplot(mapping = aes(x = year, y = lifeExp)) +
geom_line(mapping = aes(color = country))
```