

# Case study - gapminder

tidy models with **broom**



A. Ginolhac | rworkshop | 2021-09-10

# Learning objectives

## You will learn

- Use `dplyr` / `purrr` for efficient data manipulation
- Tidying multiple linear models using `broom`
- Managing related things together in `one tibble`
- Summarise findings in one `ggplot` using relevant aesthetics

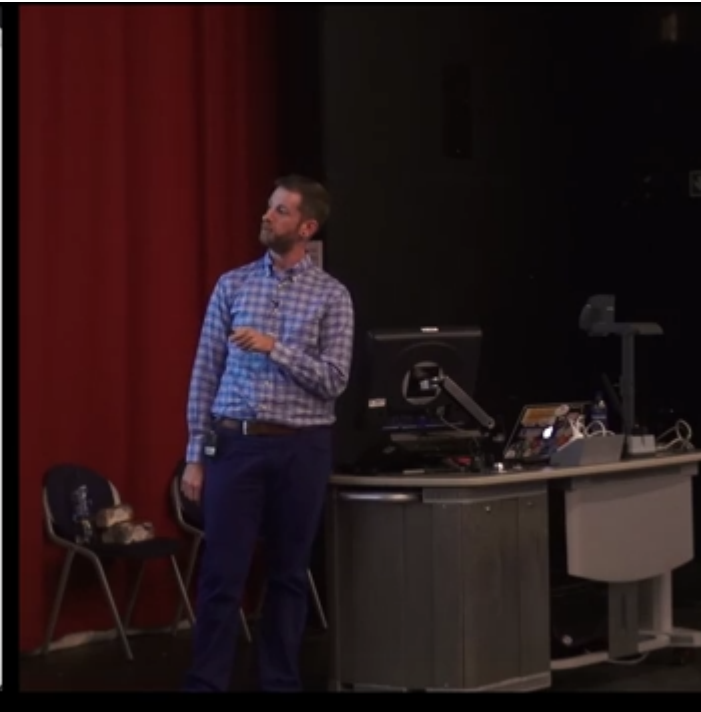
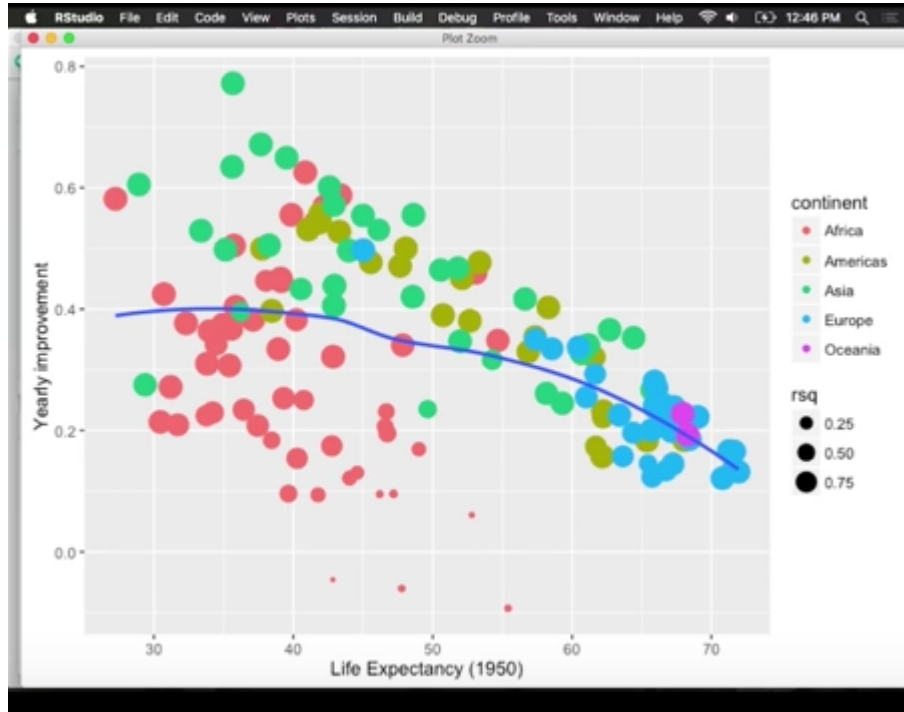


## Guided practical

Interactive session

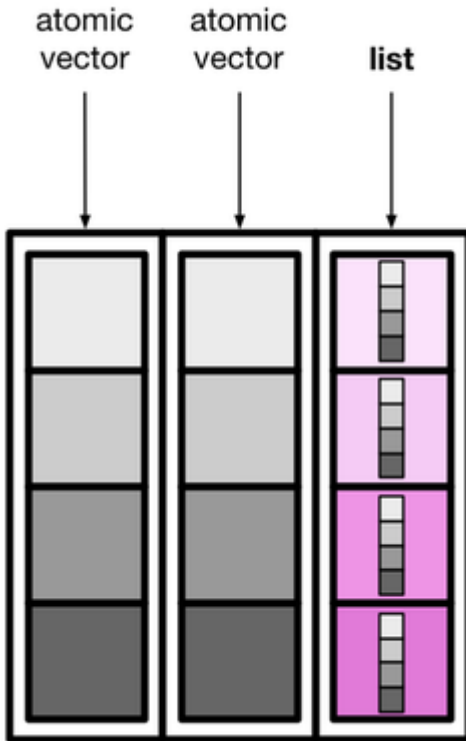
# Managing multiple models

Tutorial based on the great conference by [Hadley Wickham](#)

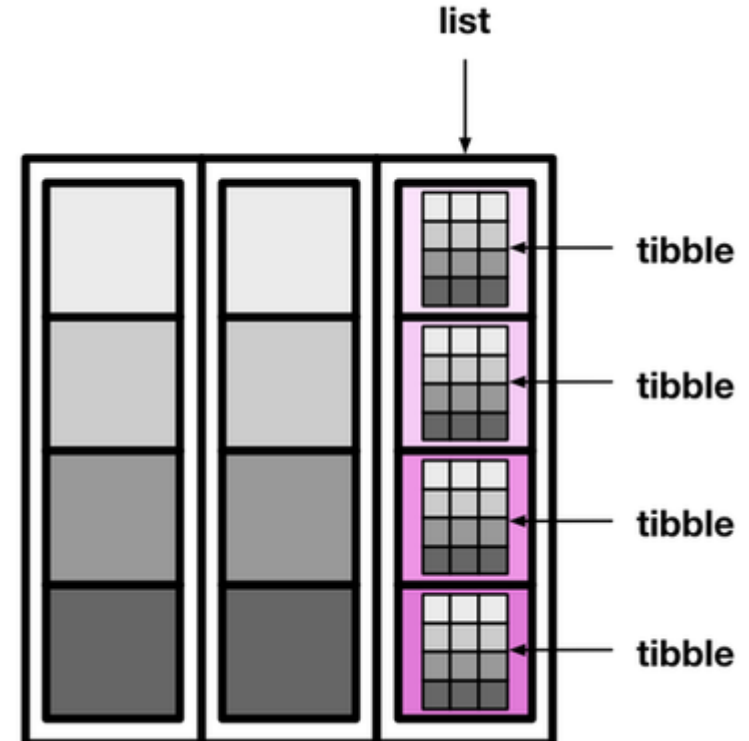


# List-column cheatsheet, reminder

## list in a data.frame/tibble



## list of tibbles in a data.frame/tibble



Functional Programming by S. Altman, B. Behrman, H. Wickham (CC4 Licence)

# Keep all analyses together

## Steps: cols, per group: rows

### Workflow from last time

```
palmerpenguins::penguins %>%  
  nest_by(island, species) %>%  
  mutate(model = list(lm(bill_depth_mm ~ bill_length_mm,  
                        data = data)),  
         summary = list(summary(model)),  
         r_squared = pluck(summary, "r.squared"))
```

### Remember

`nest_by()` is roughly the equivalent to:

```
penguins %>%  
  group_by(island, species) %>%  
  summarise(data = list(cur_data())) %>%  
  rowwise()
```

- `rowwise()` ensures that operations are performed on each row

```
# A tibble: 5 × 6  
# Rowwise: island, species  
  island species data model summary r_sq  
  <fct>   <fct>   <list<tibble[,6]>> <list> <list>  
1 Biscoe  Adelie      [44 × 6] <lm> <smmry.lm> 0  
2 Biscoe  Gentoo     [124 × 6] <lm> <smmry.lm> 0  
3 Dream   Adelie      [56 × 6] <lm> <smmry.lm> 0  
4 Dream   Chinstrap  [68 × 6] <lm> <smmry.lm> 0  
5 Torgersen Adelie     [52 × 6] <lm> <smmry.lm> 0
```

# Gapminder is a fact tank

## Dataset

- From an independent Swedish foundation
- [R package](#) by [Jenny Bryan!](#)
- Install from CRAN: `gapminder`



## Hans Rosling

- Fundamentally optimistic
- Great [talk](#)



# Guided practical, explore gapminder

## Questions ?

- Install the `gapminder` package
- Load `gapminder` and `tidyverse` packages
- Use the pipe `%>%` to pass `gapminder` to `ggplot()`
- **Plot** the `life expectancy` (`lifeExp` in `y`) ~ `year` (`x`)
- Use `geom_line()`

## Warning!

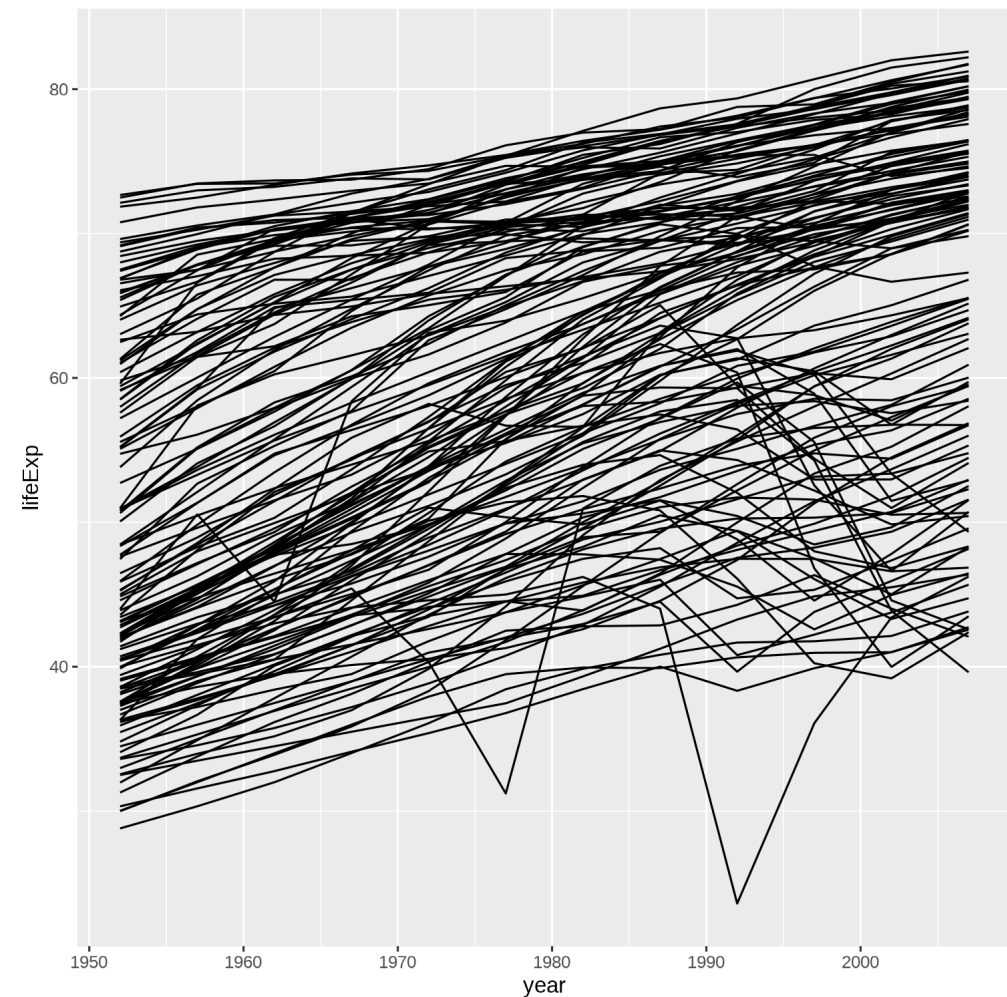
mind the  
grouping!



06:00

# Gapminder

```
# install.packages("gapminder")  
library(gapminder)  
gapminder %>%  
  ggplot(aes(x = year, y = lifeExp,  
             group = country)) +  
  geom_line()
```





# Keep related things together using list-column

## Questions ?

- Add a column using `mutate()` named `year1950` which is: `year - 1950`
- Nest with `nest_by()` the tibble by `country` and `continent`
- How many rows will you get? **Expectations help catching mistakes**
- Save the object as `by_country`

04:00

# Keep related things together

## Nest *per country*

```
by_country <- gapminder %>%  
  mutate(year1950 = year - 1950) %>%  
  nest_by(continent, country)  
by_country
```

## Helpers

- By default the list column is named `data`
- `year1950` will help to get meaningful intercepts
- Add `continent` to keep it along with `country`
- Note that `nest_by()` contains `rowwise()`

```
# A tibble: 142 × 3  
# Rowwise: continent, country  
  continent country data  
  <fct>      <fct> <list<tibble[,5]>>  
1 Africa    Algeria [12 × 5]  
2 Africa    Angola  [12 × 5]  
3 Africa    Benin   [12 × 5]  
4 Africa    Botswana [12 × 5]  
5 Africa    Burkina Faso [12 × 5]  
6 Africa    Burundi  [12 × 5]  
7 Africa    Cameroon [12 × 5]  
8 Africa    Central African Republic [12 × 5]  
9 Africa    Chad     [12 × 5]  
10 Africa   Comoros  [12 × 5]  
# ... with 132 more rows
```

# One country example: Germany

## From original tibble

```
gapminder %>%  
  filter(country == "Germany") %>%  
  select(-country, -continent)
```

```
# A tibble: 12 × 4  
  year lifeExp      pop gdpPerCap  
  <int>   <dbl>   <int>   <dbl>  
1  1952    67.5 69145952    7144.  
2  1957    69.1 71019069   10188.  
3  1962    70.3 73739117   12902.  
4  1967    70.8 76368453   14746.  
5  1972    71   78717088   18016.  
6  1977    72.5 78160773   20513.  
7  1982    73.8 78335266   22032.  
8  1987    74.8 77718298   24639.  
9  1992    76.1 80597764   26505.  
10 1997    77.3 82011073   27789.  
11 2002    78.7 82350671   30036.  
12 2007    79.4 82400996   32170.
```

## Nested tibble

```
by_country %>%  
  filter(country == "Germany")
```

```
# A tibble: 1 × 3  
# Rowwise: continent, country  
  continent country      data  
  <fct>      <fct>   <list<tibble[,5]>>  
1 Europe    Germany [12 × 5]
```

```
by_country %>%  
  filter(country == "Germany") %>%  
  unnest(data) # also summarise(data)
```

```
# A tibble: 12 × 7  
# Groups:   continent, country [1]  
  continent country year lifeExp      pop gdpPerCap year1950  
  <fct>      <fct>   <int>   <dbl>   <int>   <dbl>   <dbl>  
1 Europe    Germany  1952    67.5 69145952    7144.     2  
2 Europe    Germany  1957    69.1 71019069   10188.     7  
3 Europe    Germany  1962    70.3 73739117   12902.    12  
4 Europe    Germany  1967    70.8 76368453   14746.    17  
5 Europe    Germany  1972    71   78717088   18016.    22  
6 Europe    Germany  1977    72.5 78160773   20513.    27  
7 Europe    Germany  1982    73.8 78335266   22032.    32  
8 Europe    Germany  1987    74.8 77718298   24639.    37  
9 Europe    Germany  1992    76.1 80597764   26505.    42  
10 Europe    Germany  1997    77.3 82011073   27789.    47  
11 Europe    Germany  2002    78.7 82350671   30036.    52
```

# What happens in the `data.frame`, STAYS in the `data.frame`



# Las Vegas principle, add linear models

## Questions ?

- Using `by_country`
- Add a new column `model` with linear regressions of `lifeExp` on `year1950`
- Save as `by_country_lm`

## Ask yourself

If you see `add column`, do you use `mutate` or `summarise`?



06:00

# Linear models

## Linear model *per country*

```
by_country_lm <- by_country %>%  
  mutate(model = list(lm(lifeExp ~ year1950, data = data)))  
by_country_lm
```

```
# A tibble: 142 × 4  
# Rowwise: continent, country  
  continent country data model  
  <fct>      <fct>      <list<tibble[,5]>> <list>  
1 Africa    Algeria    [12 × 5] <lm>  
2 Africa    Angola      [12 × 5] <lm>  
3 Africa    Benin       [12 × 5] <lm>  
4 Africa    Botswana    [12 × 5] <lm>  
5 Africa    Burkina Faso [12 × 5] <lm>  
6 Africa    Burundi     [12 × 5] <lm>  
7 Africa    Cameroon    [12 × 5] <lm>  
8 Africa    Central African Republic [12 × 5] <lm>  
9 Africa    Chad        [12 × 5] <lm>  
10 Africa   Comoros     [12 × 5] <lm>  
# ... with 132 more rows
```

# Explore a list column

## Questions ?

- Count # rows per country using the `data` column
- Does any country have less data than others?

## Reminder

- A **list column** is a list, you need to iterate through elements
- **But** remember the `tibble` is still **rowwise** grouped (good).
- `distinct()` will help to find unique values
- **But** remember the `tibble` is still **rowwise** grouped (**bad**).

04:00

# Explore a list column

```
by_country_lm %>%  
  mutate(n = nrow(data)) %>%  
  select(continent, country, n)
```

```
# A tibble: 142 × 3  
# Rowwise: continent, country  
  continent country      n  
  <fct>    <fct>    <int>  
1 Africa  Algeria      12  
2 Africa  Angola       12  
3 Africa  Benin        12  
4 Africa  Botswana     12  
5 Africa  Burkina Faso  12  
6 Africa  Burundi      12  
7 Africa  Cameroon     12  
8 Africa  Central African Republic 12  
9 Africa  Chad         12  
10 Africa Comoros      12  
# ... with 132 more rows
```

```
by_country_lm %>%  
  mutate(n = nrow(data)) %>%  
  ungroup() %>%  
  distinct(n)
```

```
# A tibble: 1 × 1  
  n  
  <int>  
1  12
```



# Explore a list column, plotting

## Questions ?

- Plot `lifeExp ~ year1950` for **Bulgaria** by **unnesting** data

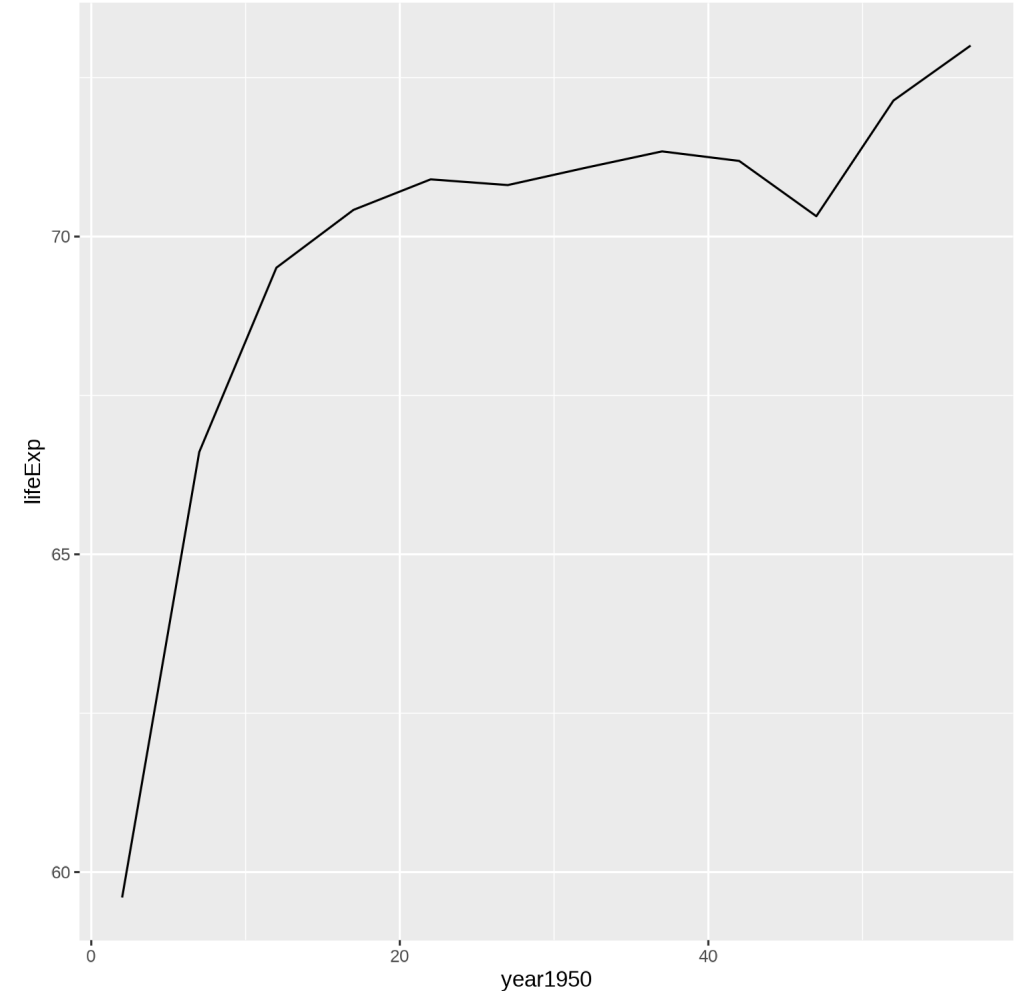
## reminder

- `filter()` for the desired country
- `unnest()` raw data
- Pipe to `ggplot()`

03:00

# Explore a list column, plotting

```
by_country_lm %>%  
  filter(country == "Bulgaria") %>%  
  unnest(data) %>%  
  ggplot(aes(x = year1950, y = lifeExp)) +  
  geom_line()
```



# Explore nested tibble

## Questions ?

- Display the `summary` for the linear model of **Rwanda**
- How do you interpret the  $r^2$  for this particular model?

## reminder

- `filter()` for the desired country
- Use `list()` to run `summary()` on the linear model
- To extract the named `"r.squared"`, use the `pluck(summary, "r.squared")` `purrr` syntax

05:00

# Linear model for Rwanda

```
by_country_lm %>%  
  filter(country == "Rwanda") %>%  
  mutate(summary = list(summary(model)),  
          r2 = pluck(summary, "r.squared")) %>%  
  select(country, r2)
```

Adding missing grouping variables: `continent`

```
# A tibble: 1 × 3  
# Rowwise: continent, country  
  continent country    r2  
  <fct>      <fct>    <dbl>  
1 Africa    Rwanda  0.0172
```

- $r^2$  is close to 0, linearity sounds broken
- **broom** will cleanup linear model elements into tibbles

# Cleanup using broom



## broom::tidy()

```
# A tibble: 4 x 5
  term          estimate std.error statistic  p.value
<chr>         <dbl>    <dbl>    <dbl>    <dbl>
1 (Intercept)    70.9      3.11     22.8 1.14e-25
2 Education     -0.428     0.260    -1.64 1.08e- 1
3 Catholic       0.184     0.0545     3.37 1.58e- 3
4 Education:Catholic -0.00938 0.00590    -1.59 1.19e- 1
```

```
Call:
lm(formula = Fertility ~ Education * Catholic, data = swiss)
```

```
Residuals:
    Min       1Q   Median       3Q      Max
-11.953  -6.319  -1.368   6.380  14.297
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  70.937553   3.106471   22.835 < 2e-16 **
Education    -0.427637   0.260176   -1.644  0.10754
Catholic      0.184003   0.054539    3.374  0.00158 **
Education:Catholic -0.009380  0.005904   -1.589  0.11942
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 8.191 on 43 degrees of freedom
Multiple R-squared:  0.5981,    Adjusted R-squared:  0.5701
F-statistic: 21.33 on 3 and 43 DF, p-value: 1.286e-08
```

## broom::glance()

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC    BIC deviance df.residual  nobs
  <dbl>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
1   0.598      0.570    8.19     21.3 0.0000000129     3  -163.  337.  346.   2885.        43    47
```

## broom::augment()

```
# A tibble: 47 x 10
  .rownames Fertility Education Catholic .fitted .resid .hat .sigma .cooksd .std.resid
  <chr>      <dbl>    <int>    <dbl>    <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
1 Courtelary    80.2      12     9.96    66.5  13.7  0.0358  8.00  0.0269    1.70
2 Delemont     83.1       9    84.8    75.5   7.56  0.0512  8.20  0.0121    0.948
3 Franches-Mnt  92.5       5    93.4    81.6  10.9  0.0649  8.10  0.0328    1.38
4 Moutier      85.8       7    33.8    71.9  13.9  0.0279  8.00  0.0211    1.72
5 Neuveville   76.9      15     5.16    64.7  12.2  0.0577  8.06  0.0358    1.53
6 Porrentruy   76.1       7    90.6    78.7  -2.56  0.0531  8.28  0.00145   -0.321
7 Broye        83.8       7    92.8    78.9   4.87  0.0558  8.25  0.00553    0.612
8 Glane        92.4       8    97.2    78.1  14.3  0.0629  7.97  0.0545    1.80
9 Gruyere      82.4       7    97.7    79.5   2.90  0.0622  8.27  0.00221    0.365
10 Sarine       82.9      13    91.4    71.0  11.9  0.112   8.06  0.0744    1.54
# ... with 37 more rows
```

# Tidying models



## Questions ?

- Install **broom** from **CRAN**
- Using `by_country_lm`, add 4 new columns:
  - `glance`, using the broom function on the `model` column
  - `tidy`, using the broom function on the `model` column
  - `augment`, using the broom function on the `model` column
  - `rsq` from the `glance` column
- Save as `models`
- Why extracting the  $r^2$  in the main tibble is useful?

## Reminder

- Use `list()` when dealing with a list column `rowwise` grouped

05:00

# Tidying models

## Useful info

- Coefficients estimates:
  - **slope**
  - **intercept**
- $r^2$
- Residuals

```
library(broom)
models <- by_country_lm %>%
  mutate(glance = list(glance(model)),
         tidy    = list(tidy(model)),
         augment = list(augment(model)),
         rsq     = pluck(glance, "r.squared"))
```

## Extracting $r^2$ in main tibble

Why? no need to **unnest** for sorting / filtering.

continent <fct>	country <fct>
Africa	Algeria
Africa	Angola
Africa	Benin
Africa	Botswana
Africa	Burkina Faso
Africa	Burundi
Africa	Cameroon
Africa	Central African Republic
Africa	Chad
Africa	Comoros

1-10 of 142 rows | 1-... Previous **1** 2 3 4 5 6 ... 15 Next

# Exploratory plots



# Plotting $r^2$ for countries

## Questions ?

- Plot `country ~ rsq`
- Color points per continent
- **Reorder** country levels by  $r^2$  (`rsq`): *snake plot*
- Which continent shows most of the low  $r^2$  values?

## Reminder

to reorder the discrete values of `country`:

- Use the `forcats` package
- `fct_reorder(country, rsq)`  
to reorder based on the `rsq` continuous variable

05:00

# Do linear models fit all countries?

## Snake plot

```
library(forcats)
models %>%
  ggplot(aes(x = rsq,
             y = fct_reorder(country,
                             rsq))) +
  geom_point(aes(colour = continent),
            alpha = 0.7, size = 2) +
  theme_classic(18) +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = c(0.25, 0.75)) +
  guides(color = guide_legend(
    override.aes = list(alpha = 1))) +
  labs(x = "r square",
       y = "Country")
```

# Display the real data for countries with a low $r^2$

## Questions ?

- Focus on non-linear trends
- Filter the 20 countries with the lowest  $r^2$
- `unnest` column `data`
- Plot `lifeExp ~ year` with lines
- Colour per continent
- Facet per country
- Same questions for the **top 20**  $r^2$

## Reminder

- You **must** `ungroup()` as we work currently **by row**
- `slice_min(col, n = 5)`  
returns the 5 minimal values of `col`

04:00

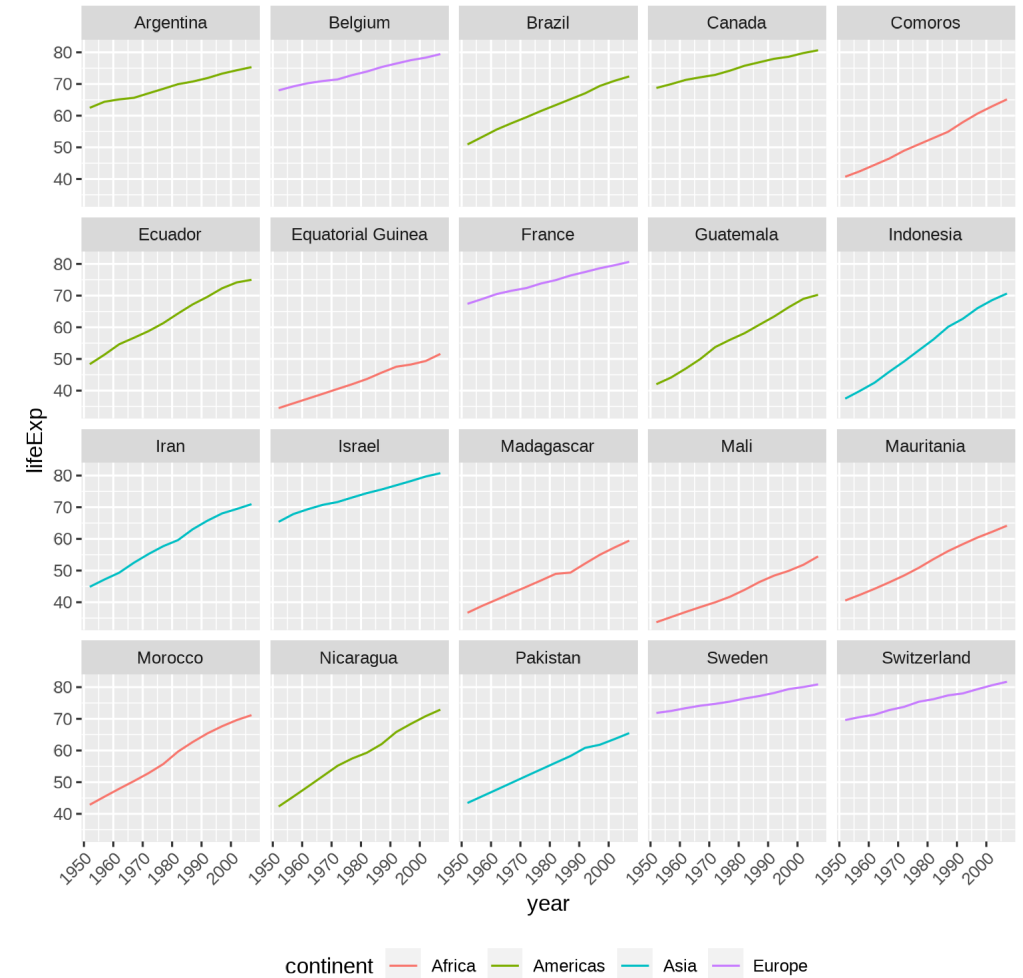
# Focus on non-linear trends

```
models %>%  
  ungroup() %>%  
  slice_min(rsq, n = 20) %>%  
  unnest(data) %>%  
  ggplot(aes(x = year, y = lifeExp)) +  
  geom_line(aes(colour = continent)) +  
  facet_wrap(~ country) +  
  theme(axis.text.x = element_text(angle = 45,  
                                     hjust = 1),  
        legend.position = "bottom")
```



# Focus on best linear trends

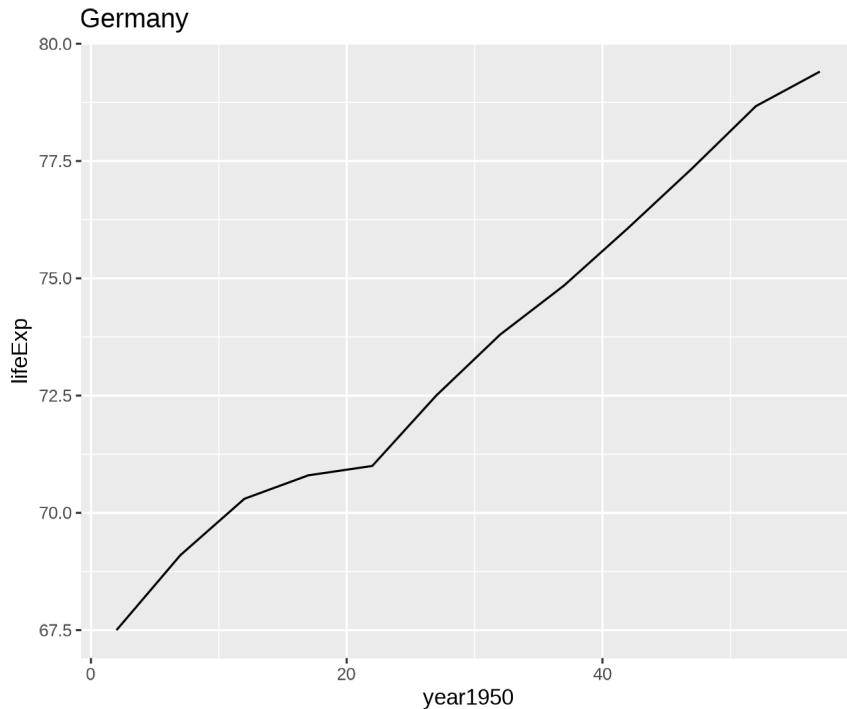
```
models %>%  
  ungroup() %>%  
  slice_max(rsq, n = 20) %>%  
  unnest(data) %>%  
  ggplot(aes(x = year, y = lifeExp)) +  
  geom_line(aes(colour = continent)) +  
  facet_wrap(~ country) +  
  theme(axis.text.x = element_text(angle = 45,  
                                     hjust = 1),  
        legend.position = "bottom")
```



# Interpreting the linear model

## Regression

- What represents the **intercept**?
  - Using **year1950**?
  - Using **year**?
  - Justify Hadley choice
- What represents the **slope**?



- Coefficients with predictor **year** - 1950

```
filter(models, country == "Germany") %>%  
  unnest(tidy) %>%  
  select(continent, country, estimate)
```

```
# A tibble: 2 × 3  
# Groups:   continent, country [1]  
  continent country estimate  
  <fct>      <fct>      <dbl>  
1 Europe    Germany    67.1  
2 Europe    Germany    0.214
```

- Compare with model and original years

```
gapminder %>%  
  filter(country == "Germany") %>%  
  lm(lifeExp ~ year, data = .) %>%  
  tidy() %>%  
  select(term, estimate)
```

```
# A tibble: 2 × 2  
  term          estimate  
  <chr>         <dbl>  
1 (Intercept) -350.  
2 year         0.214
```

# Summarise on one plot

by Hadley Wickham

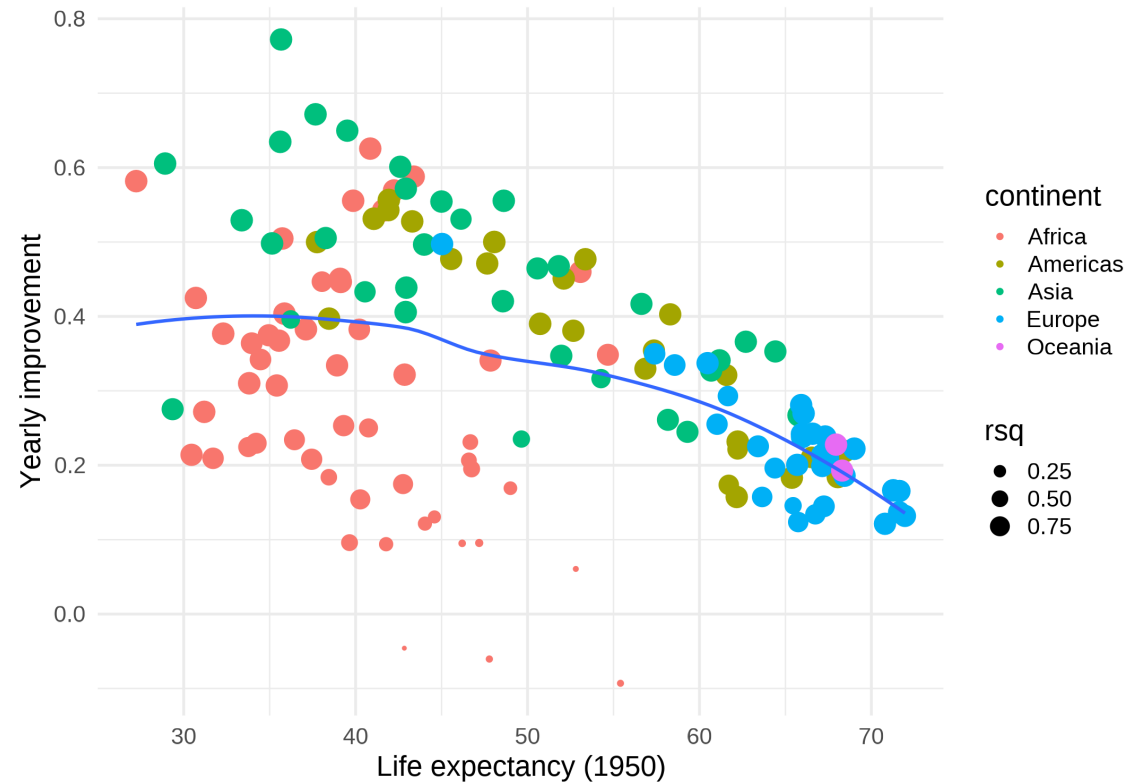
## Questions ?

- Unnest coefficients (`tidy` column)
  - Mind to keep the `continent`, `country` and `rsq` columns
- Put `intercept` and `slope` in their own columns
  - In `wide` format, only one value can be used.
  - Discard unused columns.
- Plot `slope ~ intercept` (watch out the `(Intercept)` name which needs to be called between backticks ``')
- Colour per continent
- Size per  $r^2$  (use for `scale_size_area()` for lisibility)
- Add tendency with `geom_smooth(method = "loess")`

04:00

# Full pipeline

```
gapminder %>%
  mutate(year1950 = year - 1950) %>%
  nest_by(continent, country) %>%
  mutate(model = list(lm(lifeExp ~ year1950,
                        data = data)),
         glance = list(glance(model)),
         tidy = list(tidy(model)),
         rsq = pluck(glance, "r.squared")) %>%
  unnest(tidy) %>%
  select(continent, country, rsq, term, estimate) %>%
  pivot_wider(names_from = term,
              values_from = estimate) %>%
  ggplot(aes(x = `(Intercept)`, y = year1950)) +
  geom_point(aes(colour = continent,
                 size = rsq)) +
  geom_smooth(se = FALSE,
            method = "loess",
            formula = "y ~ x") +
  scale_size_area() +
  labs(x = "Life expectancy (1950)",
       y = "Yearly improvement") +
  theme_minimal(18)
```

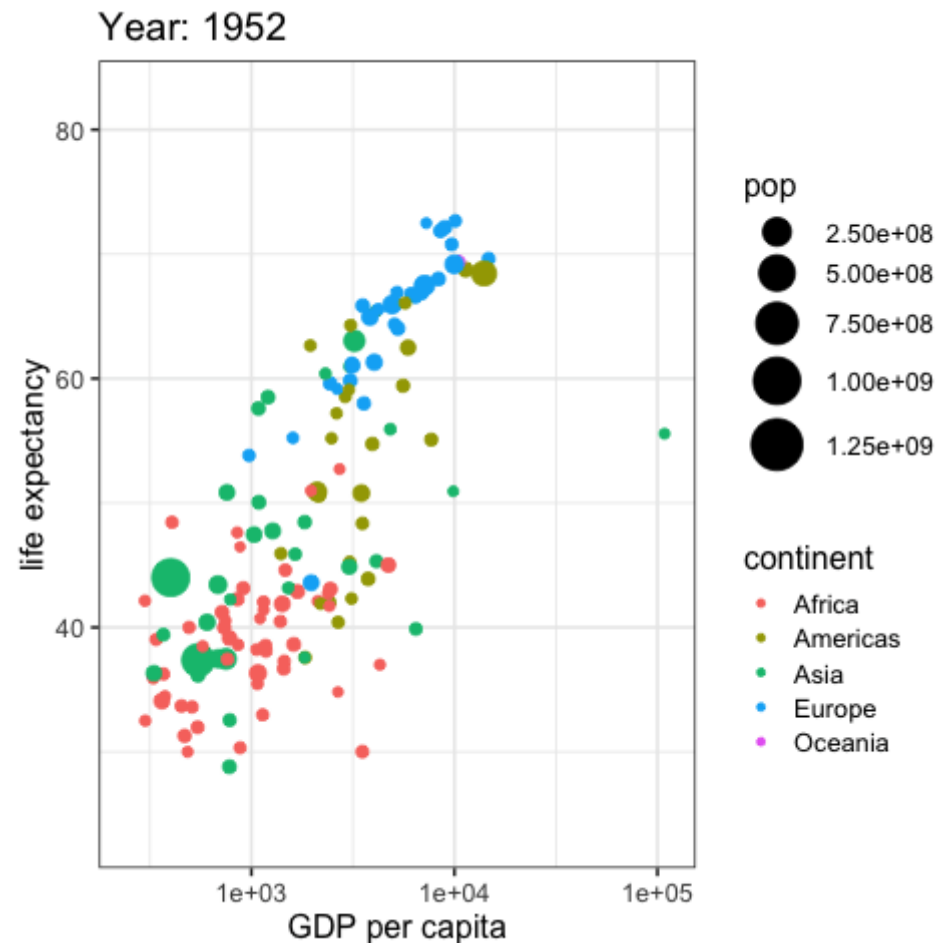




# Animation with gganimate

Takes ~ 5 minutes due to easing (**linear** since time in years)

```
library(gganimate)
gapminder %>%
  ggplot(aes(x = gdpPercap,
             y = lifeExp,
             size = pop,
             color = continent)) +
  transition_time(year) +
  ease_aes("linear") +
  scale_size(range = c(2, 12)) +
  geom_point() +
  theme_bw(16) +
  labs(title = "Year: {frame_time}",
       x = "GDP per capita",
       y = "life expectancy") +
  scale_x_log10() -> p
animate(p)
anim_save("gapminder2.gif")
```



# Before we stop

## You learned to:

- Keep related things together:
  - Input data
  - Meaningful grouping ids
  - Perform modelling
  - Extract relevant model components
  - Explore visually your findings



## Acknowledgments ☐ ☐

- Hadley Wickham
- Jennifer Bryan
- David Robinson
- Thomas Pedersen
- Eric Koncina
- Roland Krause

**Thank you for your attention!**