How to keep it togetheR

List-columns in a dataframe

Marc A.T. Teunis, Ph.D. - Hogeschool Utrecht, Lab. Innovative Testing

Last update: 2019-01-27 21:21:29

Demo for the Utrecht University R Cafe, 28 January 2019

What is R to me?

- Every day go-to for analytics of very different types of data
- Statistical analysis (`{lme}`, `{lme4}`, `{nlme}`)
- Genomics (`{DESeq2}`, `{edgeR}`, `{limma}`, `{Glimma}`)
- Microbiome analysis (`Bioconductor` & `qiime2`)
- Interactivity with `{Shiny}` and `{flexdashboards}`
- Reproducibility (`{rmarkdown}`, `{bookdown}`, `{blogdown}`)
- Tutorials and teaching (`{learnr}`, `{reticulate}` (combine Python & R))
- Getting data (`{plumber}`, `{getGEO}`, `{rentrez}`)
- Text mining (`{tidytext}`, `{igraph}`, `{ggiraph}`)
- Visualizations (`{ggplot2}`, `{tidygraph}`, `{gganimate}`, `{tmap}`)

Content for today

PART I: Starting with List-columns

There is more in the tutorial than we can cover today

PART II: MORE LIST-COLUMNS (do-it-yourself)

Getting access and materials

Clone the repository to your RStudio Environment from:

`https://github.com/uashogeschoolutrecht/rcafe` or login

`http://rserverkcgdl.hudatascience.nl` with

login: … passwd: …

Packages

The packages used in this tutorial

```r
library(tidyverse)
library(modelr)
library(lubridate)
library(broom)
library(purrr)
library(repurrrsive)
```

Dataframes and lists are recursive vectors

```
table1 <- tribble(
  ~ a, ~b, ~c, ~d,
  "x", 1, TRUE, 1.45,
  "y", 2, FALSE, 3.88,
  "z", 3, TRUE, 33.5
  )
table1
```

```
## # A tibble: 3 x 4
##   a         b c         d
##   <chr> <dbl> <lgl> <dbl>
## 1 x         1 TRUE   1.45
## 2 y         2 FALSE  3.88
## 3 z         3 TRUE  33.5
```

```
is.atomic(table1$a)
```

```
## [1] TRUE
```

Column containing a list (in a dataframe)

```
table2 <- tribble(
  ~ a,      ~b,      ~c,       ~d,     ~e,
  "x",      1,       TRUE,     1.45,   1:10,
  "y",      2,       FALSE,    3.88,   c(TRUE, FALSE),
  "z",      3,       TRUE,     33.5,   "Utrecht"
  )

is.list(table2$e)
```

```
## [1] TRUE
```

```
is.vector(table2$e)
```

```
## [1] TRUE
```

Iterate over a dataframe

```
map(table1, class)
```

```
## $a
## [1] "character"
##
## $b
## [1] "numeric"
##
## $c
## [1] "logical"
##
## $d
## [1] "numeric"
```

Iterate over a list-column

```
map(table2$e, nchar)
```

```
## [[1]]
##  [1] 1 1 1 1 1 1 1 1 1 2
##
## [[2]]
## [1] 4 5
##
## [[3]]
## [1] 7
```

Case data

Let's switch to RStudio and open the file:

`demo.Rmd`

Data origin

Whooping cough outbreaks from The World Health Organization

http://data.euro.who.int/cisid/?TabID=463987

See for variable and data collection also:

http://ecdc.europa.eu/sites/portal/files/documents/Pertussis%20AER.pdf

for more details see file: "load_data.R"

Load the tidy version of the dataset

The code for cleaning and tidying the data is in the file "./load_data.R"

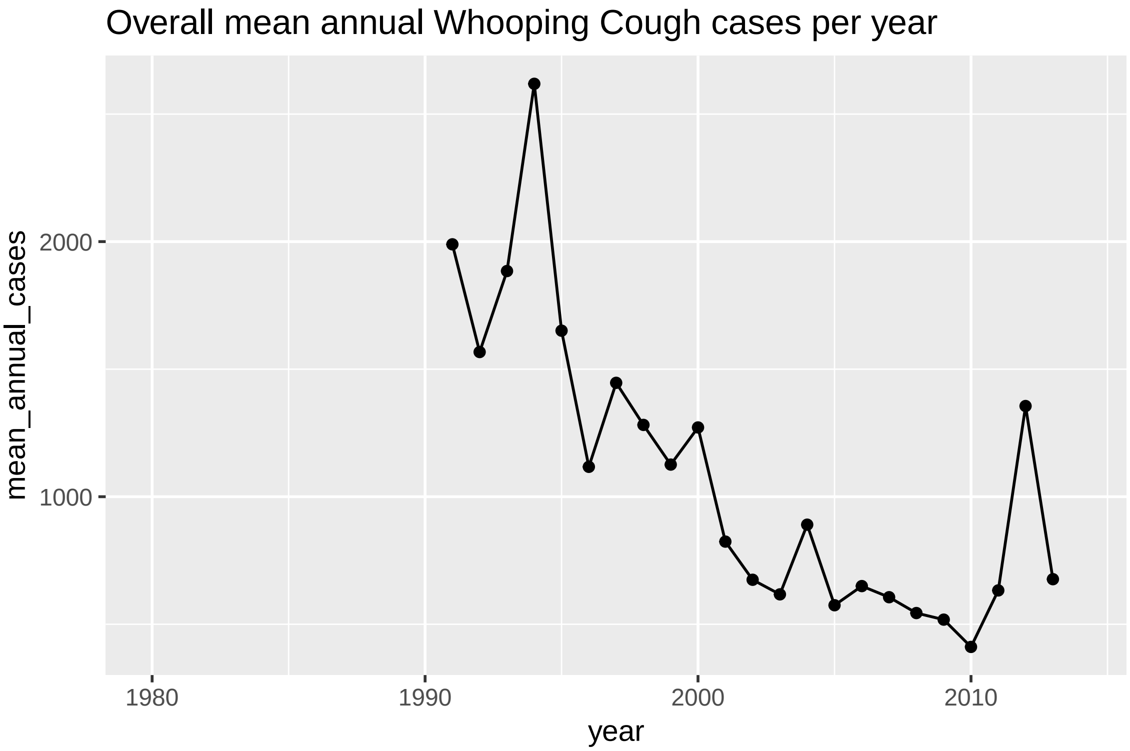
```
source(file = file.path(root,
                        "load_data.R"))

head(pertussis_data_tidy, n = 2)
```

```
## # A tibble: 2 x 4
##     key country year       annual_pertussis_cases
##   <dbl> <chr>   <date>                      <int>
## 1     2 Albania 1980-01-01                     NA
## 2     5 Andorra 1980-01-01                     NA
```

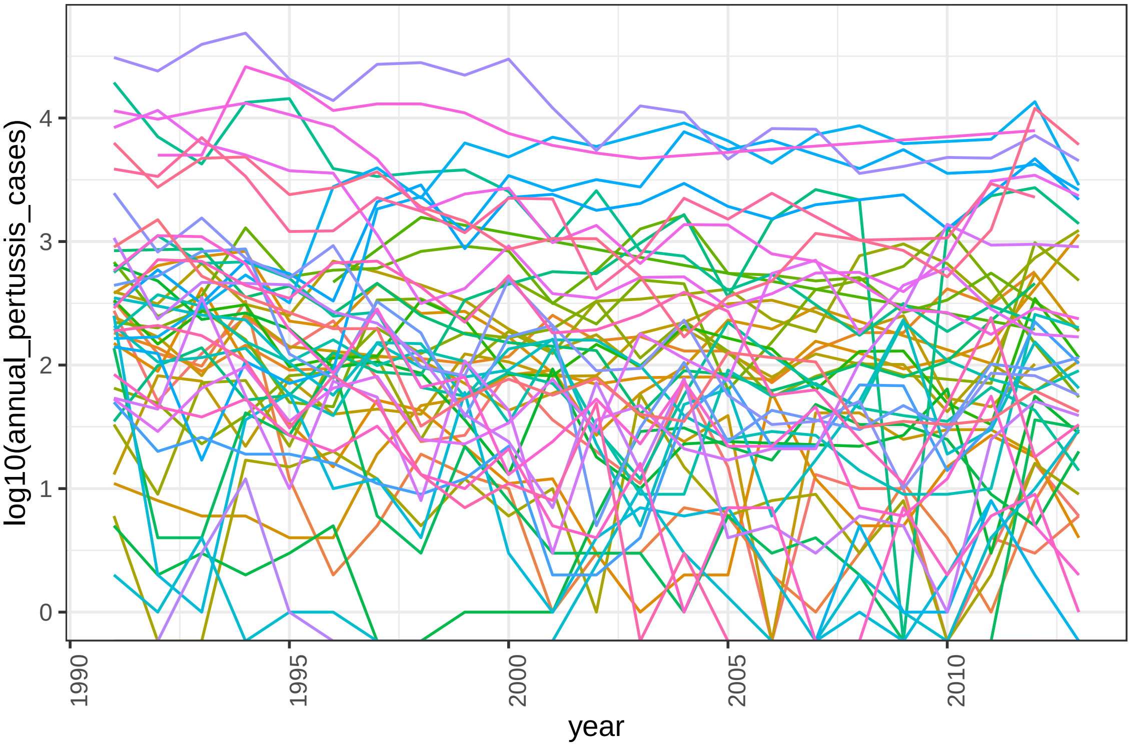
```
names(pertussis_data_tidy)
```

```
## [1] "key"                    "country"
## [3] "year"                   "annual_pertussis_cases"
```

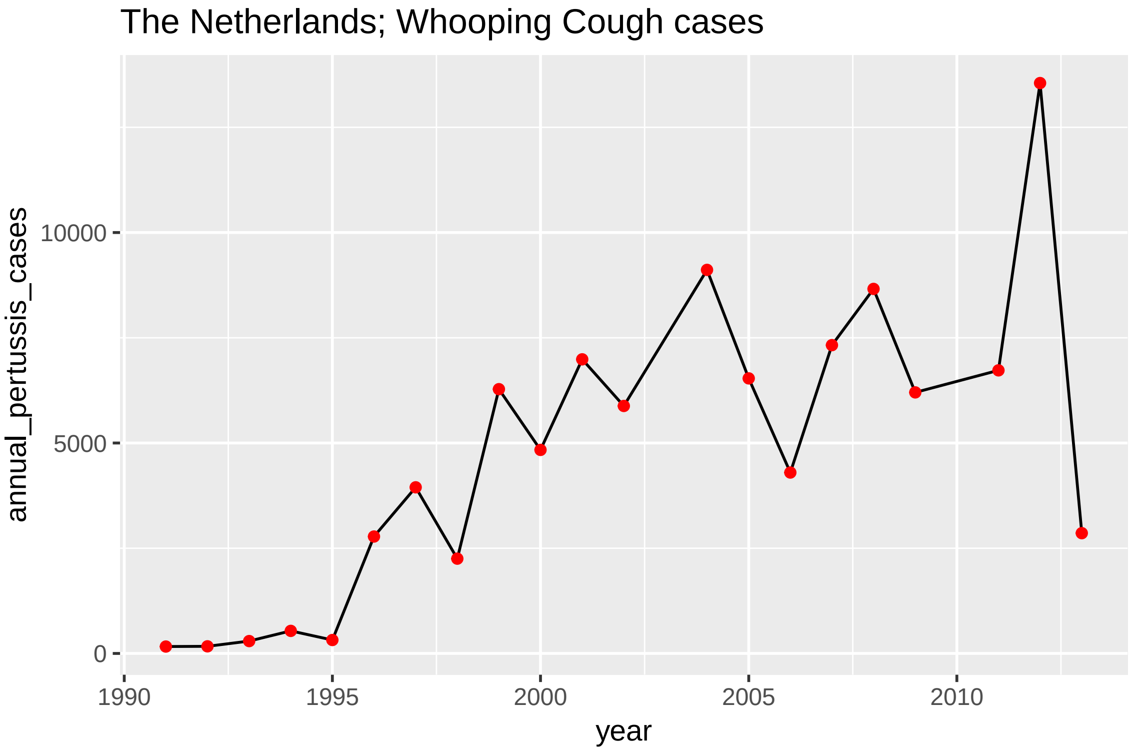
Overall trend



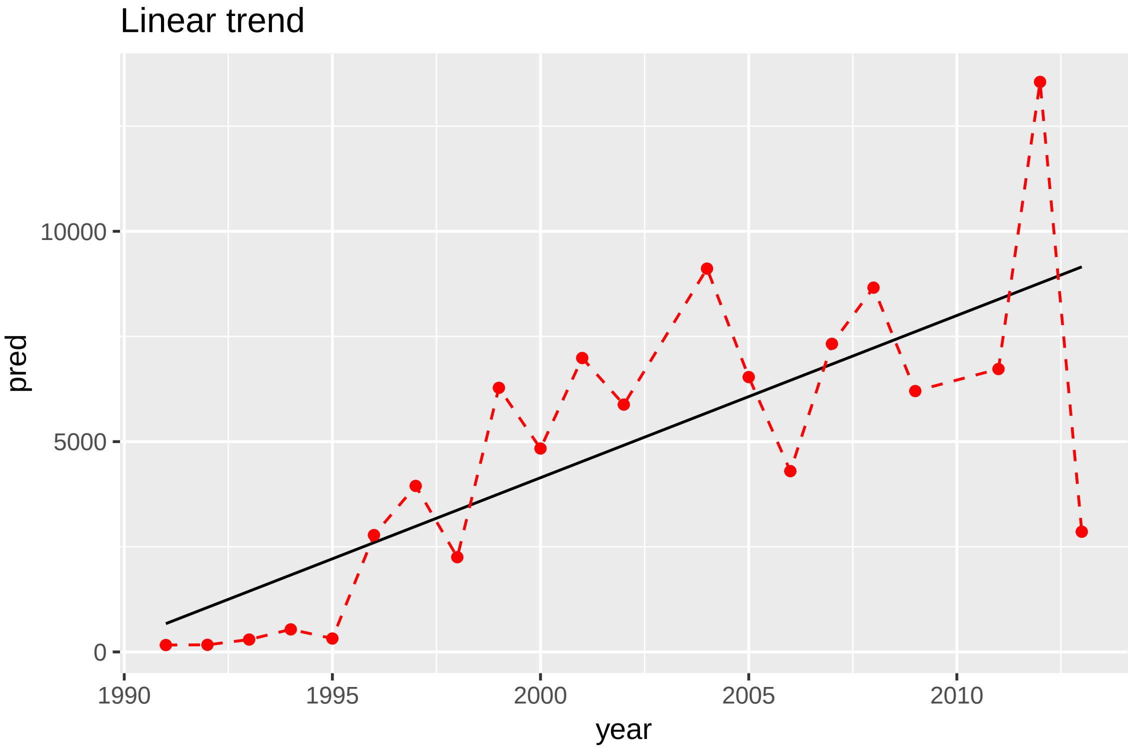
Data for individual countries, over time



Data for only The Netherlands



Plot linear model for NL



How can we apply this to every country

without doing the above 53 times.

Split the data by country and apply the model or graphing code to each subset

- In fact, data for each country is a subset of the full dataset
- We can subset the original dataframe into seperate dataframes for each country

List-columns to track your results and models

```
nested_pertussis <- pertussis_data_tidy %>%
  na.omit() %>%
  dplyr::select(country, year, annual_pertussis_cases) %>%
  group_by(country) %>%
    nest()
```

Inspecting the nested dataframes

```
head(nested_pertussis, 2) ## you see the grouping by country
```

```
## # A tibble: 2 x 2
##   country data
##   <chr>   <list>
## 1 Albania <tibble [22 x 2]>
## 2 Armenia <tibble [23 x 2]>
```

```
head(nested_pertussis$data[[1]], 2) ## you get the individual country df
```

```
## # A tibble: 2 x 2
##   year       annual_pertussis_cases
##   <date>                      <int>
## 1 1991-01-01                    275
## 2 1992-01-01                     51
```

Label (name) the idividual elements of the list column

```
names(nested_pertussis$data) <- nested_pertussis$country
head(nested_pertussis$data, 1)
```

Linear model for each country

First we write a function that creates the linear model for one country

```
country_model_lm <- function(df){
  model <- lm(
    annual_pertussis_cases ~ year,
    data = df)
  return(model)
}
```

Iterate the model function over nested `$data` with `purrr::map()`

```
models <- map(
  nested_pertussis$data, country_model_lm
  )
head(models, 2)
```

```
## [[1]]
##
## Call:
## lm(formula = annual_pertussis_cases ~ year, data = df)
##
## Coefficients:
## (Intercept)         year
##   356.23325     -0.02502
##
##
## [[2]]
##
## Call:
## lm(formula = annual_pertussis_cases ~ year, data = df)
##
## Coefficients:
## (Intercept)         year
##   297.70414     -0.02168
```

Keep it togetheR

- We have the models now
- Better to store them together with the data and the group ('country') info
- By using `dplyr::mutate()` in conjunction with `map()`

[`map()` vs. `lapply()`]https://stackoverflow.com/questions/45101045/why-use-purrrmap-instead-of-lapply)

Create an additional list-column on the basis of an existing one

```
nested_pertussis <- nested_pertussis %>%
  mutate(models_lm = map(data, country_model_lm))
head(nested_pertussis, 2)
```

```
## # A tibble: 2 x 3
##   country data              models_lm
##   <chr>   <list>            <list>
## 1 Albania <tibble [22 x 2]> <S3: lm>
## 2 Armenia <tibble [23 x 2]> <S3: lm>
```

Add model summaries as a list-column

```
nested_pertussis <- nested_pertussis %>%
  mutate(models_lm_summary = map(models_lm, summary))
head(nested_pertussis, 2)
```

```
## # A tibble: 2 x 4
##   country data              models_lm models_lm_summary
##   <chr>   <list>            <list>    <list>
## 1 Albania <tibble [22 x 2]> <S3: lm>  <S3: summary.lm>
## 2 Armenia <tibble [23 x 2]> <S3: lm>  <S3: summary.lm>
```

Extracting information from a list-column of models; `glance()` & `pluck()`

```
nested_pertussis <- nested_pertussis %>%
  mutate(params_lm =  map(models_lm, broom::glance)) %>%
  mutate(p_value = map(params_lm, pluck, "p.value"))

nested_pertussis$params_lm[[1]]
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl> <dbl>
## 1     0.529         0.505  58.8      22.4 1.27e-4     2  -120.  246.  249.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
nested_pertussis$p_value[[1]] %>% round(6)
```

```
## [1] 0.000127
```

Adding a list of plots in a column

A function that creates a graph for a single country

```
plot_line <- function(df, key){

  model <- lm(
  annual_pertussis_cases ~ year,
  data = df %>%
    na.omit()
)
## plot model for NL

plot <- df %>%
  na.omit() %>%
  add_predictions(model) %>%
  ggplot(aes(x = year,
             y = pred)) +
  geom_line() +
  geom_point(
    data = df,
    aes(x = year,
    y = annual_pertussis_cases),
    colour = "red") +
  geom_line(
    data = df %>% na.omit, ## note the pipe to remove NA
    aes(x = year,
        y = annual_pertussis_cases),
    colour = "red",
    linetype = "dashed"
) +
  ggtitle(paste("Linear model for", key %>% as.character()))

  return(plot)
}
```

Iterate plot function over nested data

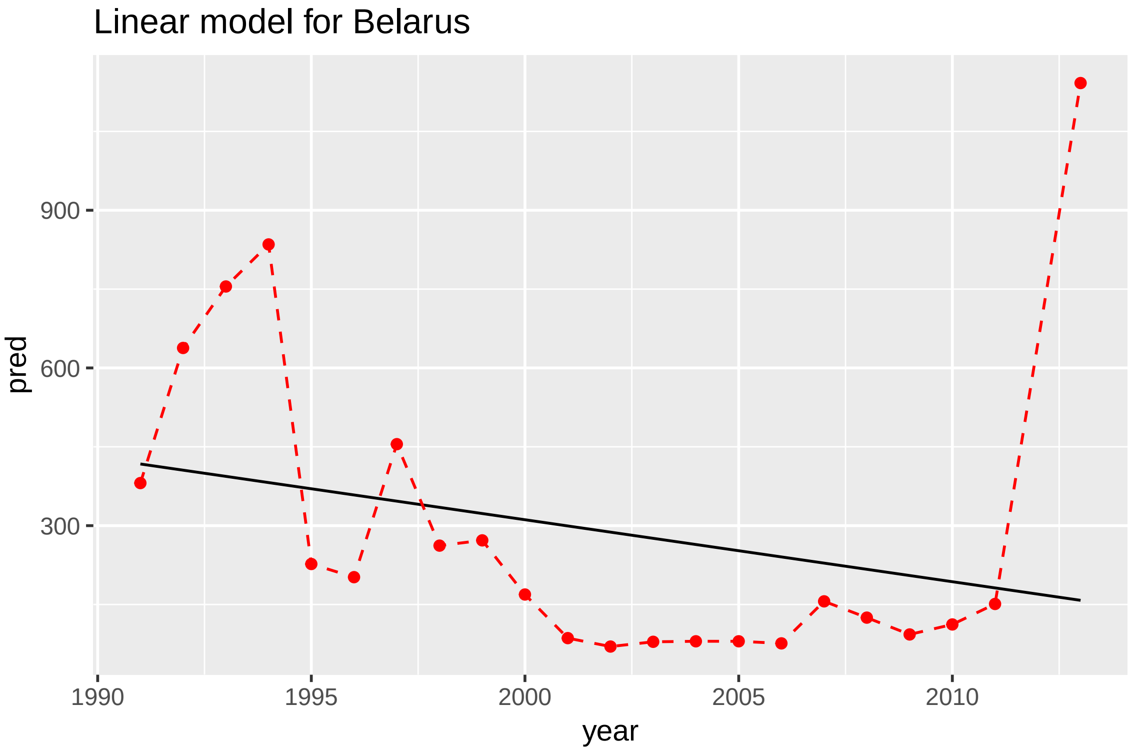
```
nested_pertussis <- nested_pertussis %>%
  mutate(
    plots_lm = map2(data, country, plot_line)
  )
nested_pertussis
```

```
## # A tibble: 52 x 7
##    country   data    models_lm models_lm_summa~ params_lm  p_value plots_lm
##    <chr>     <list>  <list>    <list>           <list>     <list>  <list>
##  1 Albania   <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  2 Armenia   <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  3 Austria   <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  4 Azerbaij~ <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  5 Belarus   <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  6 Belgium   <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  7 Bosnia a~ <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  8 Bulgaria  <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
##  9 Croatia   <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
## 10 Cyprus    <tibbl~ <S3: lm>  <S3: summary.lm> <tibble [~ <dbl [~ <S3: gg>
## # ... with 42 more rows
```

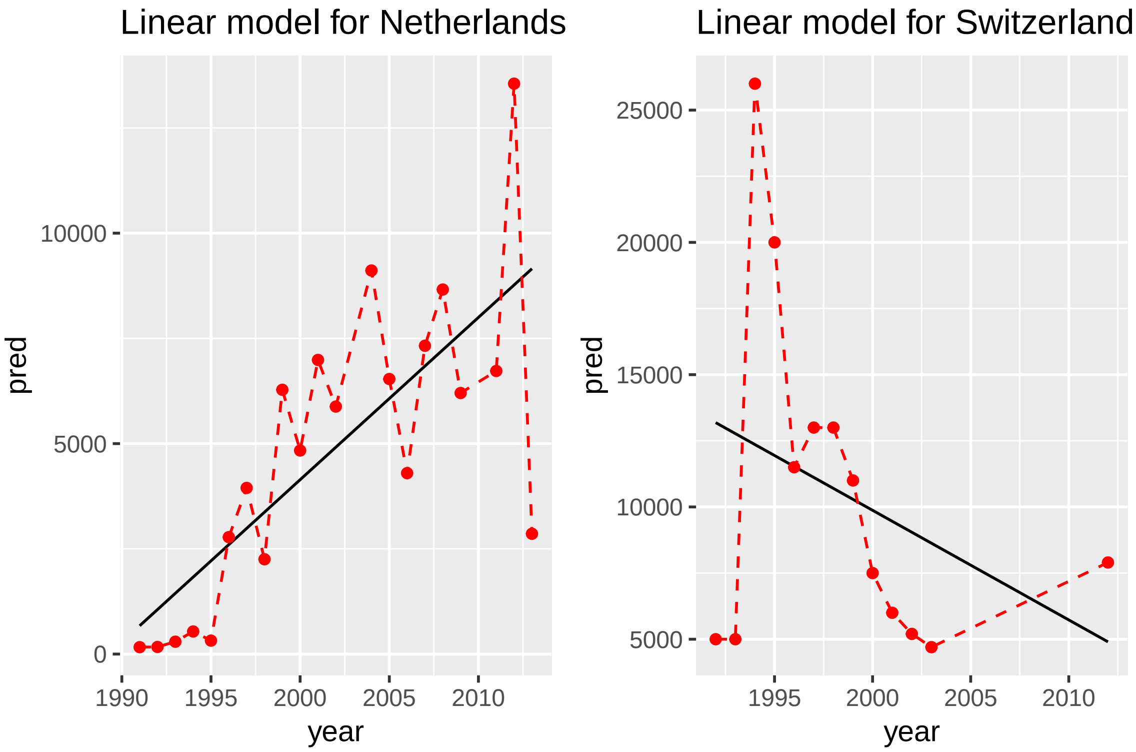
Name elements in the list-column

```
names(nested_pertussis$plots_lm) <- nested_pertussis$country
```

Show a plot



Panel of plots



Consider this

Imagine you are writing/using a function to loop over data or models in a list-(column) with `map()` or `lapply`, but it throws an ERROR half way through the list, stopping the loop

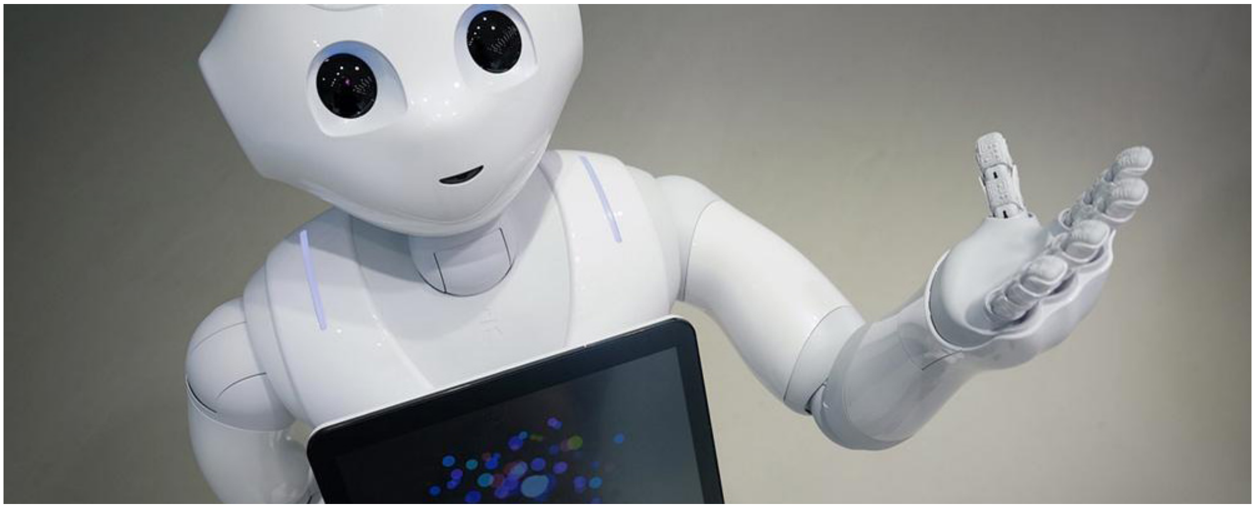
How would you solve this?

The answer is PART II below

Learn more?

'Managing many models with R' by Hadley Wickham - Lecture https://www.youtube.com/watch?v=rz3_FDVt9eg

'R for Data Science' by Garret Grolemund & Hadley Wickham https://r4ds.had.co.nz/ Especially chapters: 21 - https://r4ds.had.co.nz/iteration.html 25 - https://r4ds.had.co.nz/many-models.html



PART II; Extracting more information from a list column

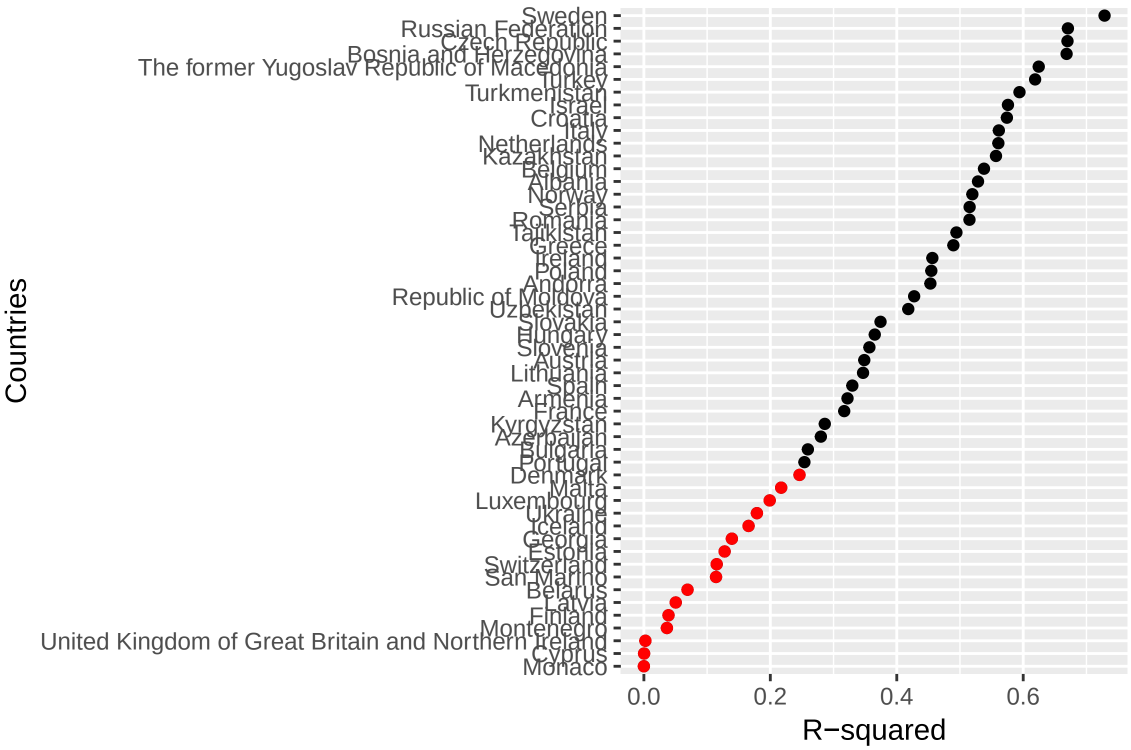
ADVANCED

For another day...

Looking at quantative statistical measures for model quality

```
r_squared <- nested_pertussis %>%
  mutate(glance = map(models_lm, broom::glance)) %>%
  unnest(glance, drop = TRUE) %>%
  select(country, r.squared, AIC, BIC) %>%
  arrange(r.squared)
```

Plotting r.sqared values



Plotting pertussis cases for countries with low r.squared over time

1. Filter countries with `r.squared` <= 0.25
2. Put countries in vector
3. Plot data

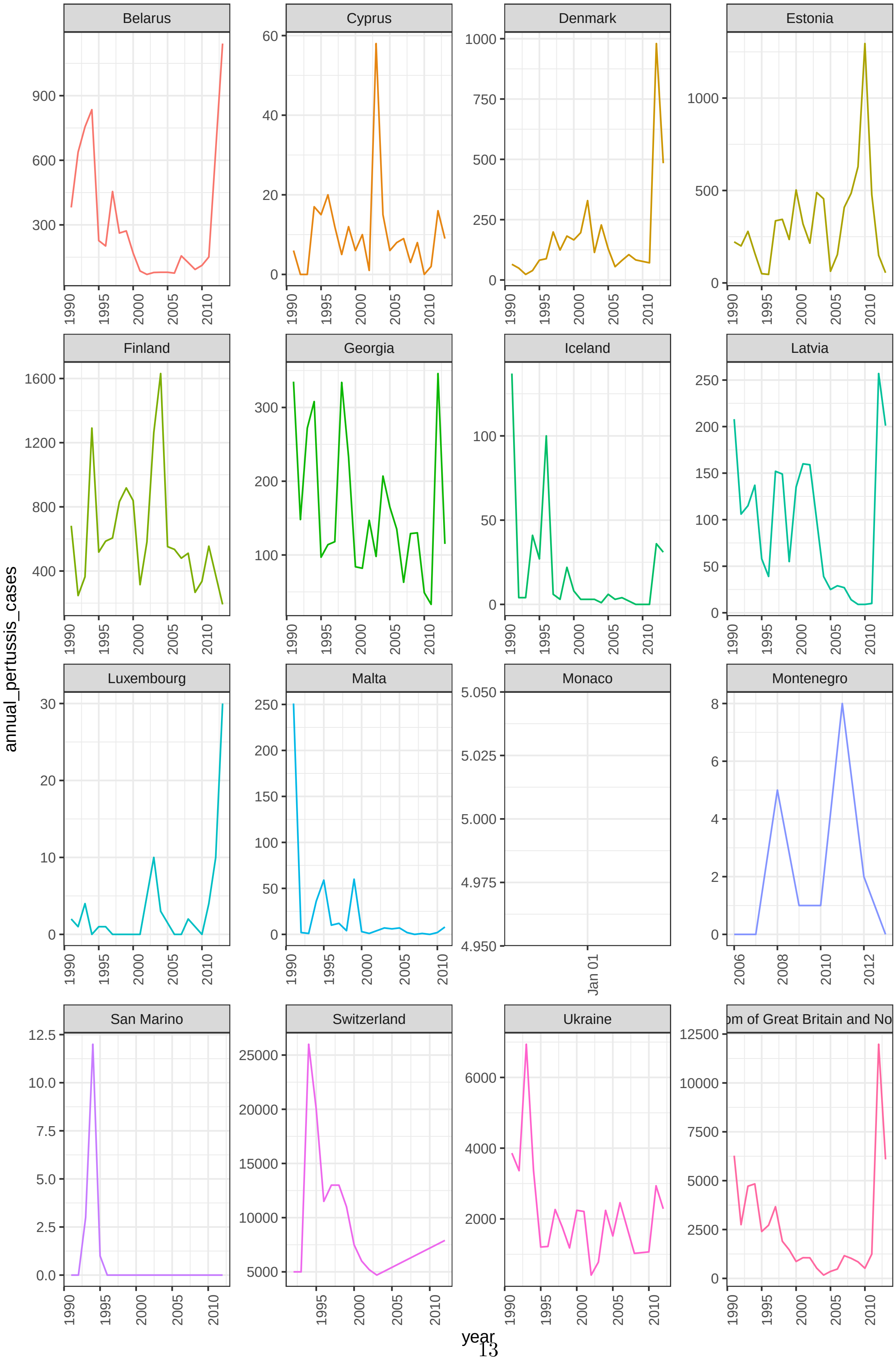
Step 1

```
low_r_squared <- r_squared %>%
  dplyr::filter(r.squared <= 0.25) %>%
  dplyr::select(country)
low_r_squared <- low_r_squared$country
```

Step 2

```
low_r_squared_nested <- nested_pertussis %>%
  dplyr::filter(country %in% low_r_squared) %>%
  select(country, data) %>%
  unnest()
```

Step 3


Belarus
Cyprus
Denmark
Estonia
Finland
Georgia
Iceland
Latvia
Luxembourg
Malta
Monaco
Montenegro
San Marino
Switzerland
Ukraine
om of Great Britain and No
annual_pertussis_cases
year
300
600
900
0
20
40
60
0
250
500
750
1000
0
500
1000
400
800
1200
1600
100
200
300
0
50
100
0
50
100
150
200
250
0
10
20
30
0
50
100
150
200
250
4.950
4.975
5.000
5.025
5.050
0
2
4
6
8
0.0
2.5
5.0
7.5
10.0
12.5
5000
10000
15000
20000
25000
2000
4000
6000
0
2500
5000
7500
10000
12500
1990
1995
2000
2005
2010
Jan 01
2006
2008
2010
2012

What is happening with the pertussis vaccination grade over the past 8 years?

Store ggplot2 objects in a list-column

1. Create a function that makes the plot
2. Test function on single dataframe
3. Apply the function using `mutate()` and `map()` to all dataframes or models

```
## isolate one dataframe to test function
df <- nested_pertussis$data[[1]]
plot_country <- function(df){

  df %>%
    ggplot(aes(x = year,
           y = annual_pertussis_cases)) +
    geom_line() +
    geom_smooth() +
    ylab("Annual cases")

}

## test function
# plot_country(df = df)
```

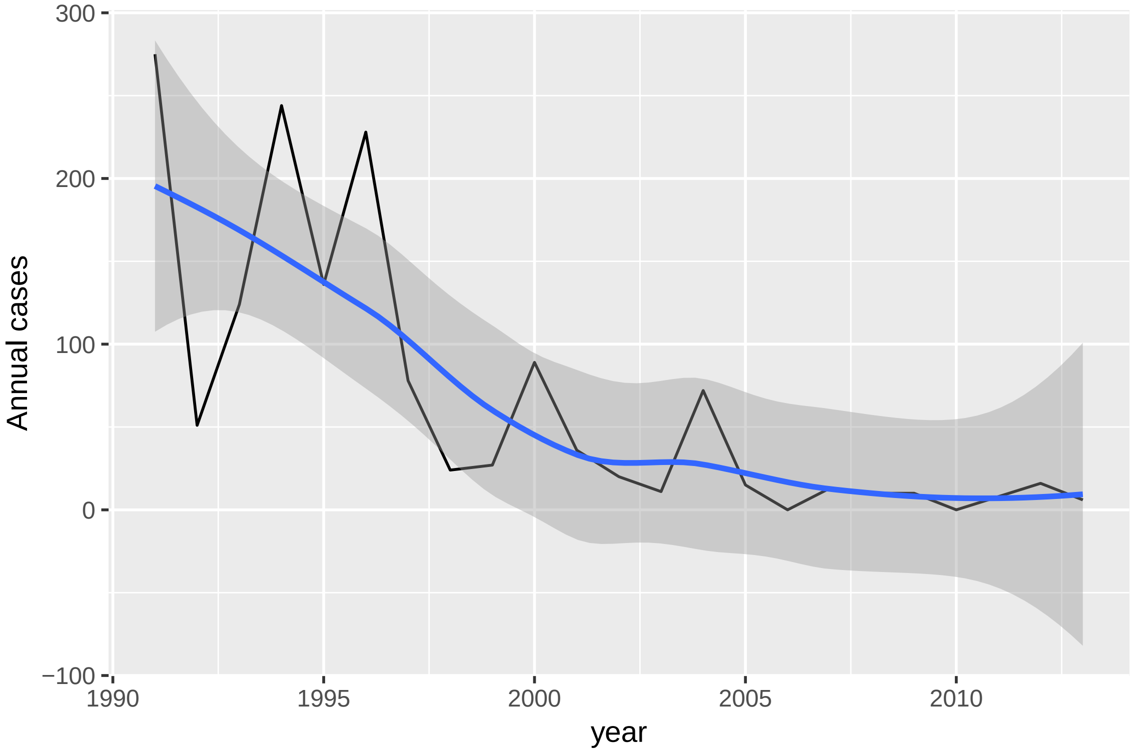
Apply plotting function to nested data

```
nested_pertussis <- nested_pertussis %>%
  mutate(plots_cases_over_time = map(data,
                                     plot_country))
```

Add countries as names to nest-column

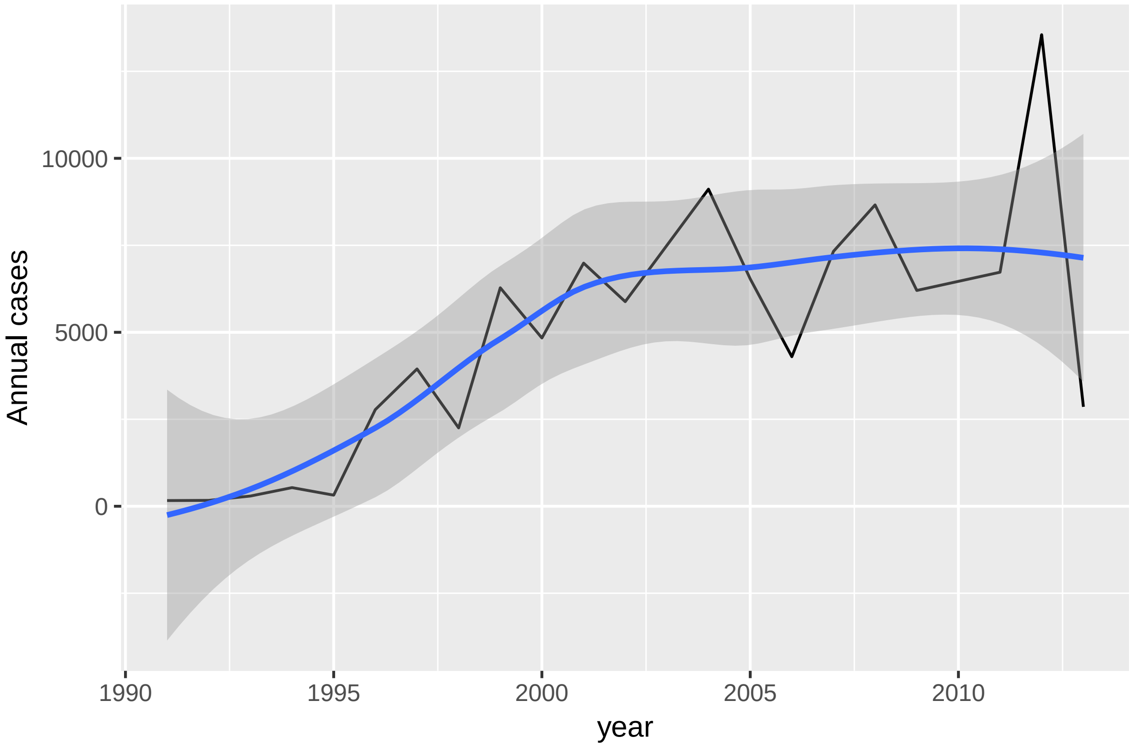
```
names(nested_pertussis$plots_cases_over_time) <-
  c(nested_pertussis$country)

nested_pertussis$plots_cases_over_time[1]
```

```
## $Albania
```

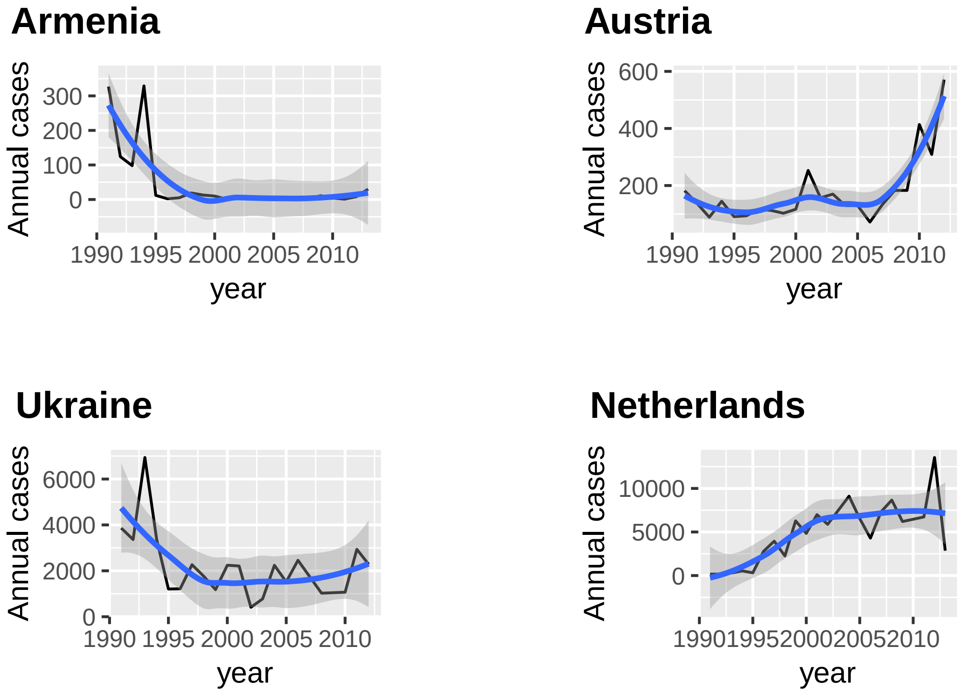


Pull out "The Netherlands"

```
nested_pertussis$plots_cases_over_time$Netherlands
```



Plotting a panel of 4 graphs



Literature

For some background on the pattern we are seeing

- https://www.scientificamerican.com/article/why-whooping-cough-vaccines-are-wearing-off/
- http://outbreaknewstoday.com/pertussis-cases-up-significantly-in-the-eu-netherlands-and-uk-worst-hit-55315/

Exploring many more models

Let's add a quadratic model in the mix. Assume we want to explore non-linear relationships in this dataset

```
non_linear_model <- function(df, model_params){

  nl_model <- lm(
    annual_pertussis_cases ~ poly(year,
                                  model_params),
                data = df)

  return(nl_model)
}
```

Creating a safe version of this function

```
safe_non_linear <- purrr::safely(non_linear_model)
## apply test:
df = nested_pertussis$data[[1]]
```

```
test_non_linear <-  df %>% non_linear_model(df = .,
                                            model_params = 2)
```

Test function on one country

```
test_non_linear %>% broom::glance()
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl> <dbl>
## 1     0.607         0.566  55.0      14.7 1.39e-4     3  -118.  244.  248.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

Apply model to all countries

We rerun the steps above to add this new model and new graphs to the nested dataframe

Add new model to the nested table

```
nested_pertussis <- nested_pertussis %>%
  mutate(models_nl_2 = map(data, safe_non_linear,
                          model_params = 2))

nested_pertussis$models_nl_2 <- transpose(nested_pertussis$models_nl_2)
```

Set names to elements in the list-column

To be able to `pluck()` by name later

```
names(nested_pertussis$models_nl_2$result) <- nested_pertussis$country
```

Pluck results in new list-column

```
nested_pertussis$models_nl_2$result[[1]] %>% summary
```

```
## 
## Call:
## lm(formula = annual_pertussis_cases ~ poly(year, model_params), 
##     data = df)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -135.782  -19.621   -3.885    7.188  114.303 
## 
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                     67.50      11.73   5.753 1.52e-05 ***
## poly(year, model_params)1     -278.33      55.03  -5.057 7.00e-05 ***
## poly(year, model_params)2      107.53      55.03   1.954   0.0656 .  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.03 on 19 degrees of freedom
## Multiple R-squared:  0.6074, Adjusted R-squared:  0.5661
## F-statistic:  14.7 on 2 and 19 DF,  p-value: 0.0001389
```

```
nested_pertussis <- nested_pertussis %>%
  mutate(statistics_nl = pluck(models_nl_2, "result"))

nested_pertussis$statistics_nl[[1]] %>% summary
```

```
##
## Call:
## lm(formula = annual_pertussis_cases ~ poly(year, model_params),
##     data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -135.782  -19.621   -3.885    7.188  114.303
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   67.50      11.73   5.753 1.52e-05 ***
## poly(year, model_params)1   -278.33      55.03  -5.057 7.00e-05 ***
## poly(year, model_params)2    107.53      55.03   1.954   0.0656 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.03 on 19 degrees of freedom
## Multiple R-squared:  0.6074, Adjusted R-squared:  0.5661
## F-statistic:  14.7 on 2 and 19 DF,  p-value: 0.0001389
```

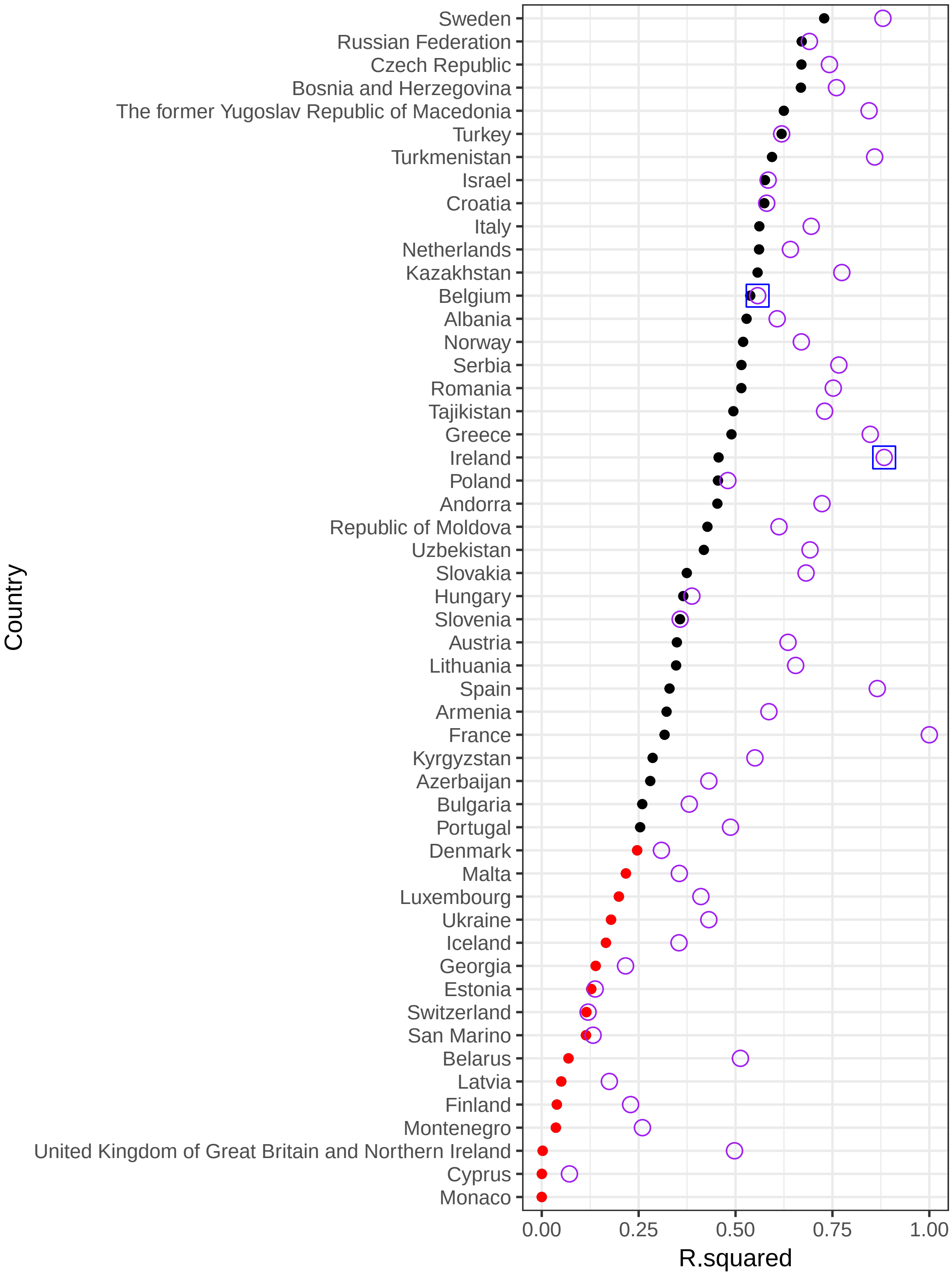
Tidy the list-column with {broom}

```
nested_pertussis <- nested_pertussis %>%
  mutate(parameters_nl = map(statistics_nl, glance))
```

Looking at quantative statistical measures for model quality

```
r_squared_nl <- nested_pertussis %>%
  select(country, parameters_nl) %>%
  unnest()
```

Plotting r.sqared values



Let’s examine two models for two countries where the non-linear did and did not not improve the R.squared: Ireland (improved) and Belgium (not-improved)

```
x <- nested_pertussis %>%
  select(country,
         data,
         models_lm,
         statistics_nl) %>%
  gather(models_lm:statistics_nl, key = "models", value = "model_params") %>%
 print()
```

```
## # A tibble: 104 x 4
##    country                data              models    model_params
##    <chr>                  <list>            <chr>     <list>
##  1 Albania                <tibble [22 x 2]> models_lm <S3: lm>
##  2 Armenia                <tibble [23 x 2]> models_lm <S3: lm>
##  3 Austria                <tibble [22 x 2]> models_lm <S3: lm>
##  4 Azerbaijan             <tibble [23 x 2]> models_lm <S3: lm>
##  5 Belarus                <tibble [22 x 2]> models_lm <S3: lm>
##  6 Belgium                <tibble [21 x 2]> models_lm <S3: lm>
##  7 Bosnia and Herzegovina <tibble [19 x 2]> models_lm <S3: lm>
##  8 Bulgaria               <tibble [22 x 2]> models_lm <S3: lm>
##  9 Croatia                <tibble [23 x 2]> models_lm <S3: lm>
## 10 Cyprus                 <tibble [23 x 2]> models_lm <S3: lm>
## # ... with 94 more rows
```

Remove ‘empty model’

```
ind <- x$model_params == "NULL"
#ind <- x$data == "NULL"
x <- x[!ind, ]
```

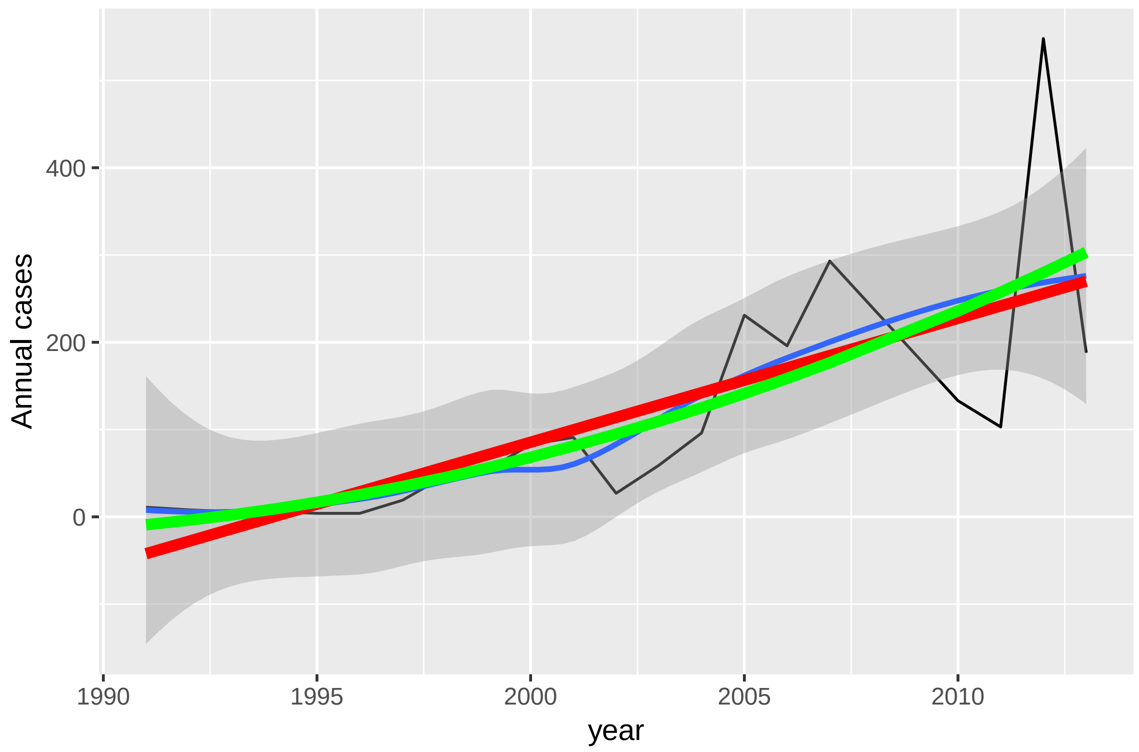
Add prediction-list column

```
predictions <- x %>%
#  filter(country == "Czech Republic") %>%
  mutate(predictions = map2(data, model_params, add_predictions,
         var = "annual_pertussis_cases")) %>%
  filter(country == "Ireland" |
         country == "Belgium") %>%
  select(country, data, predictions)
```

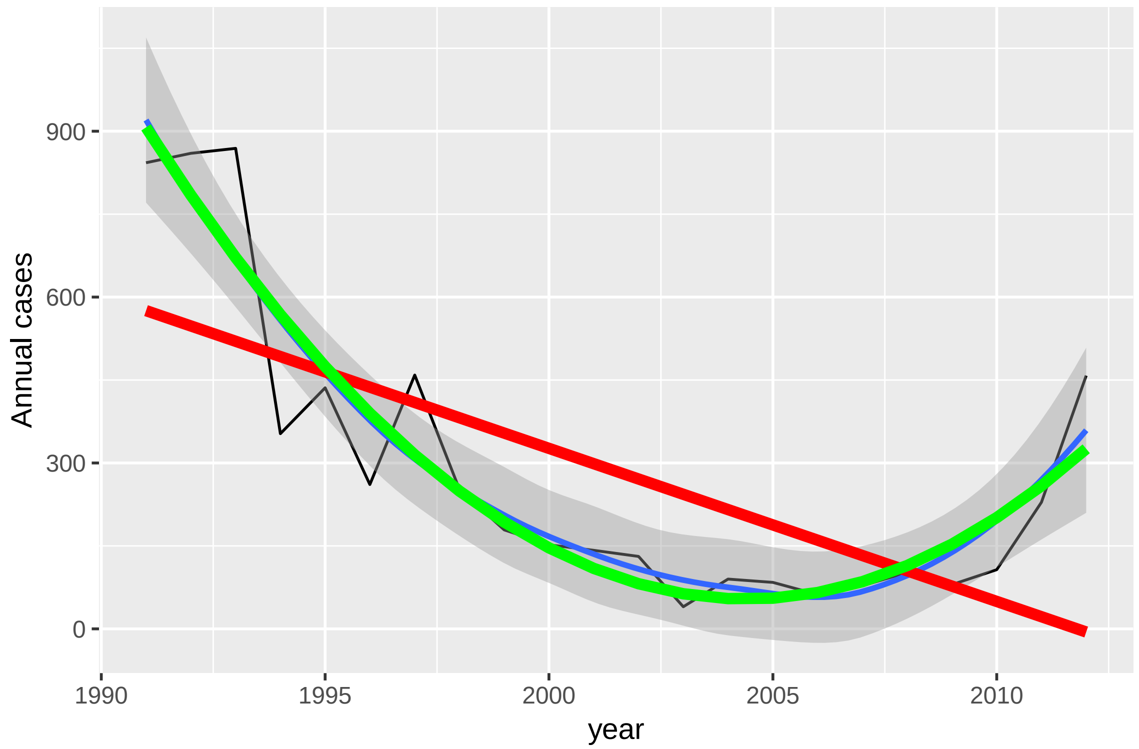
Set names

```
names(predictions$predictions) <- predictions$country
```

Belgium



Ireland



Learn more?

To practice with more examples have a look at

```
# repurrrsive
```

Disclaimer & Licence

The work presented here may be shared, remixed or adapted as long as the original references and the authors of this document are mentioned in the redistribution: LICENCE: CC BY-SA

Credits

Much of this material has been derived in one way or the other from the work of Hadley Wickham and Garret Grolemund and many others. For a more elaborate reference list see the resources.Rmd file in the project root.

Thanks to Hadley & Garret for writing the book "R for Data Science" http://r4ds.had.co.nz/ and for their work in general to innovate the R world.

Work on integration of Git/Github with R/RStudio is thoroughly and wit-fully documented by Jenny Brian. I also very much appreciate her work on the use of RMarkdown and thanks for pointing me into the direction of using the `rprojroot` package (CSAMA Course 2016). See also:

https://github.com/jennybc/happy-git-with-r & http://stat545.com/block007_first-use-rmarkdown.html