# Workshop: Data science with R

## ZEW - Session #3

Obryan Poyser
2019-03-06

# Tidy Data and the Tidyverse

# `dplyr`: scoped variants

Until now we have seen the main function of `dplyr`. Each serves for specific tasks relating data wrangling. Nonetheless, each of them is accompanied with special variants called "scoped functions" that serve for even more explicit problems.

In this session we are going to work with the Global Financial Inclusion Database. According to the WB *"the world's most comprehensive data set on how adults save, borrow, make payments, and manage risk."* The DB consist on 140 different countries with representative surveys of 150K adults age 15+ for the year 2017.

The data below is a 10% sample without replacement of the whole DB.

```
findex <- readr::read_rds("datasets/global_findex/sample_findex.rds")

# There is a dictionary of the data as well.
dic <- readr::read_csv("datasets/global_findex/dictionary.csv")
```

## Scoped variants

- `*_all()`: affects every variable. Ex:
  - `mutate_all()`
  - `summarise_all()`
  - `filter_all()`
- `*_if()`: affects variables selected with a character vector or `vars()` Ex:
  - `mutate_if()`
  - `summarise_if()`-
  - `filter_if()`
- `*_at()`: affects variables selected with a predicate function (predicate are functions that return `TRUE` or `FALSE`) Ex:
  - `mutate_at()`
  - `summarise_at()`
  - `filter_at()`

# _all()

Example: `mutate_all()`

```
findex %>%
    mutate_all(.funs = ~as.character(.))
```

```
## # A tibble: 15,492 x 32
##    economy economycode regionwb pop_adult wpid_random wgt   female age
##    <chr>   <chr>       <chr>    <chr>     <chr>       <chr> <chr>  <chr>
##  1 Mozamb… MOZ         Sub-Sah… 15850773  141162711   0.78… Female 2
##  2 Ukraine UKR         Europe … 38149932  197500379   1.46… Female 26
##  3 Gabon   GAB         Sub-Sah… 1269789   146060394   0.30… Male   18
##  4 Venezu… VEN         Latin A… 22762362  147928189   0.55… Female 41
##  5 Rwanda  RWA         Sub-Sah… 7094419   179789279   1.65… Male   36
##  6 Monten… MNE         Europe … 509031.5… 202078829   0.32… Female 39
##  7 Morocco MAR         Middle … 25550170  13836       0.42… Female 15
##  8 Cambod… KHM         East As… 10814416  117109019   0.82… Male   36
##  9 Bolivia BOL         Latin A… 7400247   116466964   0.73… Female 12
## 10 Centra… CAF         Sub-Sah… 2595884   119279387   1.07… Male   8
## # … with 15,482 more rows, and 24 more variables: educ <chr>, inc_q <chr>,
## #   emp_in <chr>, fin2 <chr>, fin7 <chr>, fin8 <chr>, fin14a <chr>,
## #   fin14b <chr>, fin19 <chr>, fin26 <chr>, fin28 <chr>, fin32 <chr>,
## #   fin37 <chr>, mobileowner <chr>, account_fin <chr>, account_mob <chr>,
## #   account <chr>, saved <chr>, borrowed <chr>, receive_wages <chr>,
## #   receive_transfers <chr>, receive_pension <chr>,
## #   receive_agriculture <chr>, fin2_b <chr>
```

# _if()

Example: `mutate_if()`

```
findex %>%
    transmute_if(.predicate=is.numeric, .funs = list(log=~log(.)
                                                     , abs=~abs(.)))
```

```
## # A tibble: 15,492 x 16
##    pop_adult_log wpid_random_log wgt_log age_log receive_transfe…
##            <dbl>           <dbl>   <dbl>   <dbl>            <dbl>
##  1          16.6            18.8  -0.240    0.693            1.39
##  2          17.5            19.1   0.381    3.26             1.39
##  3          14.1            18.8  -1.20     2.89             1.39
##  4          16.9            18.8  -0.593    3.71             1.39
##  5          15.8            19.0   0.504    3.58             1.39
##  6          13.1            19.1  -1.12     3.66             0
##  7          17.1             9.54 -0.861    2.71             1.39
##  8          16.2            18.6  -0.193    3.58             1.39
##  9          15.8            18.6  -0.312    2.48             1.39
## 10          14.8            18.6   0.0692   2.08             1.39
## # … with 15,482 more rows, and 11 more variables:
## #   receive_pension_log <dbl>, receive_agriculture_log <dbl>,
## #   fin2_b_log <dbl>, pop_adult_abs <dbl>, wpid_random_abs <dbl>,
## #   wgt_abs <dbl>, age_abs <dbl>, receive_transfers_abs <dbl>,
## #   receive_pension_abs <dbl>, receive_agriculture_abs <dbl>,
## #   fin2_b_abs <dbl>
```

# `_at()`

Example: `mutate_at()`

```
findex %>%
    transmute_at(.vars = vars(contains("fin")), .funs = ~case_when(
        .=="no" ~ 0
        , .=="yes" ~ 1
    ))
```

```
## # A tibble: 15,492 x 12
##     fin2  fin7  fin8 fin14a fin14b fin19 fin26 fin28 fin32 fin37
##    <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     0     0    NA      0      0     0     1     0     0     0
##  2     1     0    NA      0      0     0     0     0     1     0
##  3     1     0    NA      1      1     0     1     1     1     0
##  4     1     1     1      1      1     1     1     1     1     0
##  5     0     0    NA      0      0     0     0     0     1     0
##  6     0     0    NA      0      0     1     0     1     1     1
##  7     1     0    NA      0      0     0     0     0     0     0
##  8     1     0    NA      0      0     0     1     0     0     0
##  9     1     0    NA      0      0     0     0     0     1     0
## 10     0     0    NA      0      0    NA     0     0     0     0
## # … with 15,482 more rows, and 2 more variables: account_fin <dbl>,
## #   fin2_b <dbl>
```

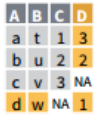
# Merging data

- Normally, one finds data distributed into several different files
- dplyr provides join functions to perform merging according to matching cells indetifiers

**left_join**(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)
Join matching values from y to x.

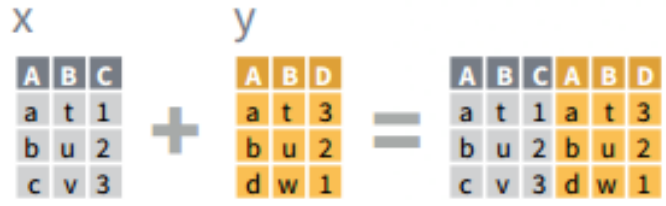
**right_join**(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)
Join matching values from x to y.

**inner_join**(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)
Join data. Retain only rows with matches.

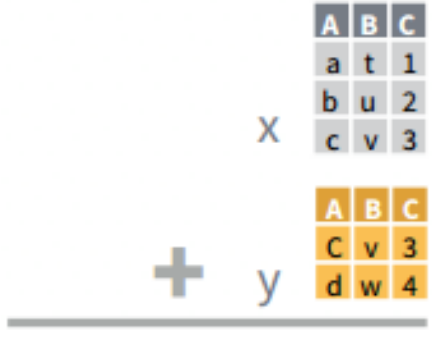


**full_join**(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)
Join data. Retain all values, all rows.

- Also it is possible to combine datasets when they share the same column names with `bind_cols()`




- Or rows with `bind_rows()`



# Merging data: example

```
(df1 <- tibble::tribble(
    ~col1, ~col2, ~col3,
      "a",     1,     3,
      "b",    NA,     5,
      "b",     5,     6
    ))
```

```
## # A tibble: 3 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
## 3 b         5     6
```

```
(df2 <- tibble::tribble(
           ~col1, ~col2, ~col4,
             "a",     1,  TRUE,
             "b",    NA, FALSE,
             "b",     6, FALSE,
             "c",    10, FALSE
           ))
```

```
## # A tibble: 4 x 3
##   col1   col2 col4
##   <chr> <dbl> <lgl>
## 1 a         1 TRUE
## 2 b        NA FALSE
## 3 b         6 FALSE
## 4 c        10 FALSE
```

```
(list_df <- list(e1=df1, e2=df1, e3=df2))
```

```
## $e1
## # A tibble: 3 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
## 3 b         5     6
##
## $e2
## # A tibble: 3 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
## 3 b         5     6
##
## $e3
## # A tibble: 4 x 3
##   col1   col2 col4
##   <chr> <dbl> <lgl>
## 1 a         1 TRUE
## 2 b        NA FALSE
## 3 b         6 FALSE
## 4 c        10 FALSE
```

# Merging data: left and right

```
df1
```

```
## # A tibble: 3 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
## 3 b         5     6
```

```
df2
```

```
## # A tibble: 4 x 3
##   col1   col2 col4
##   <chr> <dbl> <lgl>
## 1 a         1 TRUE
## 2 b        NA FALSE
## 3 b         6 FALSE
## 4 c        10 FALSE
```

```
# left_join(): matches values from df2 to df1
df1 %>%
    left_join(df2)
```

```
## Joining, by = c("col1", "col2")
```

```
## # A tibble: 3 x 4
##   col1   col2  col3 col4
##   <chr> <dbl> <dbl> <lgl>
## 1 a         1     3 TRUE
## 2 b        NA     5 FALSE
## 3 b         5     6 NA
```

```
# right_join(): matches valyes from df1 to df2
df1 %>%
    right_join(df2)
```

```
## Joining, by = c("col1", "col2")
```

```
## # A tibble: 4 x 4
##   col1   col2  col3 col4
##   <chr> <dbl> <dbl> <lgl>
## 1 a         1     3 TRUE
## 2 b        NA     5 FALSE
## 3 b         6    NA FALSE
## 4 c        10    NA FALSE
```

# Merging data: inner and full

```
df1
```

```
## # A tibble: 3 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
## 3 b         5     6
```

```
df2
```

```
## # A tibble: 4 x 3
##   col1   col2 col4
##   <chr> <dbl> <lgl>
## 1 a         1 TRUE
## 2 b        NA FALSE
## 3 b         6 FALSE
## 4 c        10 FALSE
```

```
# inner_join(): retain only the rows with matches
df1 %>%
    inner_join(df2)
```

```
## Joining, by = c("col1", "col2")

## # A tibble: 2 x 4
##   col1   col2  col3 col4
##   <chr> <dbl> <dbl> <lgl>
## 1 a         1     3 TRUE
## 2 b        NA     5 FALSE
```

```
# full_join(): retain all values, all rows
df1 %>%
    full_join(df2)
```

```
## Joining, by = c("col1", "col2")

## # A tibble: 5 x 4
##   col1   col2  col3 col4
##   <chr> <dbl> <dbl> <lgl>
## 1 a         1     3 TRUE
## 2 b        NA     5 FALSE
## 3 b         5     6 NA
## 4 b         6    NA FALSE
## 5 c        10    NA FALSE
```

# Merging data: semi and anti

```
df1
```

```
## # A tibble: 3 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
## 3 b         5     6
```

```
df2
```

```
## # A tibble: 4 x 3
##   col1   col2 col4
##   <chr> <dbl> <lgl>
## 1 a         1 TRUE
## 2 b        NA FALSE
## 3 b         6 FALSE
## 4 c        10 FALSE
```

```
# semi_join(): returns rows of df1 that have match in df2
df1 %>%
    semi_join(df2)
```

```
## Joining, by = c("col1", "col2")

## # A tibble: 2 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
```

```
# anti_join(): return rows of df1 that do not have a match in
df1 %>%
    anti_join(df2)
```

```
## Joining, by = c("col1", "col2")

## # A tibble: 1 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 b         5     6
```

# Merging data: by row/column

```
df1
```

```
## # A tibble: 3 x 3
##   col1   col2  col3
##   <chr> <dbl> <dbl>
## 1 a         1     3
## 2 b        NA     5
## 3 b         5     6
```

```
df2
```

```
## # A tibble: 4 x 3
##   col1   col2 col4
##   <chr> <dbl> <lgl>
## 1 a         1 TRUE
## 2 b        NA FALSE
## 3 b         6 FALSE
## 4 c        10 FALSE
```

```
# bind_rows: When row-binding, columns are matched by name,
# and any missing columns will be filled with NA.
df1 %>%
    bind_rows(df2)
```

```
## # A tibble: 7 x 4
##   col1   col2  col3 col4
##   <chr> <dbl> <dbl> <lgl>
## 1 a         1     3 NA
## 2 b        NA     5 NA
## 3 b         5     6 NA
## 4 a         1    NA TRUE
## 5 b        NA    NA FALSE
## 6 b         6    NA FALSE
## 7 c        10    NA FALSE
```

```
# bind_cols: rows are matched by position,
# so all data frames must have the same number of rows.
df1 %>%
    bind_cols(df1)
```

```
## # A tibble: 3 x 6
##   col1   col2  col3 col11 col21 col31
##   <chr> <dbl> <dbl> <chr> <dbl> <dbl>
## 1 a         1     3 a         1     3
## 2 b        NA     5 b        NA     5
## 3 b         5     6 b         5     6
```

# Merging data: advanced stuff

- Join functions have a `by=` argument to match specific columns.
- If two matching columns have different names one can specify such details by:

```
data1 %>%
    left_join(by = c("var1"="var2"))
```

Notice the difference between:

```
df1 %>%
    left_join(df2)
```

```
## Joining, by = c("col1", "col2")

## # A tibble: 3 x 4
##   col1   col2  col3 col4
##   <chr> <dbl> <dbl> <lgl>
## 1 a         1     3 TRUE
## 2 b        NA     5 FALSE
## 3 b         5     6 NA
```

and...

```
df1 %>%
    left_join(df2, by="col1")
```

```
## # A tibble: 5 x 5
##   col1  col2.x  col3 col2.y col4
##   <chr>  <dbl> <dbl>  <dbl> <lgl>
## 1 a          1     3      1 TRUE
## 2 b         NA     5     NA FALSE
## 3 b         NA     5      6 FALSE
## 4 b          5     6     NA FALSE
## 5 b          5     6      6 FALSE
```

# Merging data: advanced stuff

- What can we do if there are several separated datasets we want to:
  1. join?
  2. bind?
- According to the approach followed in the last slides it is necessary to call the join function per object, like:

```
df1 %>%
    left_join(df2) %>%
    left_join(df3) %>% ...
```

- However, there is a powerful and intuitive way to execute this task with the reduce function, which is also embedded in the package `purrr`. Therefore the problem could be solved only by specifying the list of dataframes and then applying the joining function recursively. Example:

```
list(df1, df1, df2) %>%
    reduce(.f = left_join, by="col1"
           , suffix=c("_iter1", "_iter2"))
```

```
## # A tibble: 9 x 7
##   col1  col2_iter1 col3_iter1 col2_iter2 col3_iter2  col2 col4
##   <chr>      <dbl>      <dbl>      <dbl>      <dbl> <dbl> <lgl>
## 1 a              1          3          1          3     1 TRUE
## 2 b             NA          5         NA          5    NA FALSE
## 3 b             NA          5         NA          5     6 FALSE
## 4 b             NA          5          5          6    NA FALSE
## 5 b             NA          5          5          6     6 FALSE
## 6 b              5          6         NA          5    NA FALSE
## 7 b              5          6         NA          5     6 FALSE
## 8 b              5          6          5          6    NA FALSE
## 9 b              5          6          5          6     6 FALSE
```

# `tidyr`

- The main purpose of `tidyr` is to create tidy data
- There are 2 main functions for converting long to wide data and viceversa, plus other 2 useful for separating tasks.
  - `gather()`: takes multiple columns, and gathers them into key-value pairs: it makes “wide” data longer.
  - `spread()`: takes two columns (key & value), and spreads into multiple columns: it makes “long” data wider.
  - `separate()`: pull apart columns that represent multiple variables
  - `extract()`: turns each group into a new column

# gather()

```
(df1 <- df1 %>%
    mutate(obs=1:3))
```

```
## # A tibble: 3 x 4
##   col1   col2  col3   obs
##   <chr> <dbl> <dbl> <int>
## 1 a         1     3     1
## 2 b        NA     5     2
## 3 b         5     6     3
```

```
df1 %>%
    gather(key = "key", value = "data")
```

```
## # A tibble: 12 x 2
##    key   data
##    <chr> <chr>
##  1 col1  a
##  2 col1  b
##  3 col1  b
##  4 col2  1
##  5 col2  <NA>
##  6 col2  5
##  7 col3  3
##  8 col3  5
##  9 col3  6
## 10 obs   1
## 11 obs   2
## 12 obs   3
```

```
(df1_long <- df1 %>%
    gather(key = "key", value = "data"
           , -obs)
 )
```

```
## # A tibble: 9 x 3
##     obs key   data
##   <int> <chr> <chr>
## 1     1 col1  a
## 2     2 col1  b
## 3     3 col1  b
## 4     1 col2  1
## 5     2 col2  <NA>
## 6     3 col2  5
## 7     1 col3  3
## 8     2 col3  5
## 9     3 col3  6
```

# spread()

```
(df1_long <- df1_long %>%
    mutate(new_var=paste0(data, "-", data)))
```

```
## # A tibble: 9 x 4
##     obs key   data  new_var
##   <int> <chr> <chr> <chr>
## 1     1 col1  a     a-a
## 2     2 col1  b     b-b
## 3     3 col1  b     b-b
## 4     1 col2  1     1-1
## 5     2 col2  <NA>  NA-NA
## 6     3 col2  5     5-5
## 7     1 col3  3     3-3
## 8     2 col3  5     5-5
## 9     3 col3  6     6-6
```

```
df1_long %>%
    spread(key = key, value = data)
```

```
## # A tibble: 9 x 5
##     obs new_var col1  col2  col3
##   <int> <chr>   <chr> <chr> <chr>
## 1     1 1-1     <NA>  1     <NA>
## 2     1 3-3     <NA>  <NA>  3
## 3     1 a-a     a     <NA>  <NA>
## 4     2 5-5     <NA>  <NA>  5
## 5     2 b-b     b     <NA>  <NA>
## 6     2 NA-NA   <NA>  <NA>  <NA>
## 7     3 5-5     <NA>  5     <NA>
## 8     3 6-6     <NA>  <NA>  6
## 9     3 b-b     b     <NA>  <NA>
```

# separate()

```
df1_long %>%
    separate(new_var, into = c("var1", "var2"), sep = "-")
```

```
## # A tibble: 9 x 5
##     obs key   data  var1  var2
##   <int> <chr> <chr> <chr> <chr>
## 1     1 col1  a     a     a
## 2     2 col1  b     b     b
## 3     3 col1  b     b     b
## 4     1 col2  1     1     1
## 5     2 col2  <NA>  NA    NA
## 6     3 col2  5     5     5
## 7     1 col3  3     3     3
## 8     2 col3  5     5     5
## 9     3 col3  6     6     6
```

```
sv <- tibble(id=1:3, q1=c("1", "1,2", "1,3,4"))
sv %>%
    separate_rows(q1)
```

```
## # A tibble: 6 x 2
##      id q1
##   <int> <chr>
## 1     1 1
## 2     2 1
## 3     2 2
## 4     3 1
## 5     3 3
## 6     3 4
```

# purrr

- Toolkit by providing a complete and consistent set of tools for working with functions and vectors.
- Allow you to replace many for loops with code that is both more succinct and easier to read.
- `purrr` has docenzs of functions, some could be overly complex. However, must of the time we will use the most basic function called `map()`
  - The `map_*()` functions transform their input by applying a function to each element and returning a vector the same length as the input.
  - `map()` is a powered `apply()`!

```
lapply(X = letters[1:3], FUN = function(x) return(x))
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "b"
##
## [[3]]
## [1] "c"
```

```
map(.x = letters[1:3], .f = function(x) return(x)) # explicit
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "b"
##
## [[3]]
## [1] "c"
```

```
map(.x = letters[1:3], .f = ~return(.)) # implicit functions
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "b"
##
## [[3]]
## [1] "c"
```

# `purrr`: unleashing the true power

```
(findex_model <- findex %>%
    group_by(economy, regionwb))
```

```
## # A tibble: 15,492 x 32
## # Groups:   economy, regionwb [144]
##    economy economycode regionwb pop_adult wpid_random   wgt female   age
##    <chr>   <chr>       <chr>        <dbl>       <dbl> <dbl> <fct>  <dbl>
##  1 Mozamb… MOZ         Sub-Sah… 15850773    141162711 0.786 Female     2
##  2 Ukraine UKR         Europe … 38149932    197500379 1.46  Female    26
##  3 Gabon   GAB         Sub-Sah…  1269789    146060394 0.301 Male      18
##  4 Venezu… VEN         Latin A… 22762362    147928189 0.553 Female    41
##  5 Rwanda  RWA         Sub-Sah…  7094419    179789279 1.65  Male      36
##  6 Monten… MNE         Europe …   509032.   202078829 0.326 Female    39
##  7 Morocco MAR         Middle … 25550170        13836 0.423 Female    15
##  8 Cambod… KHM         East As… 10814416    117109019 0.825 Male      36
##  9 Bolivia BOL         Latin A…  7400247    116466964 0.732 Female    12
## 10 Centra… CAF         Sub-Sah…  2595884    119279387 1.07  Male       8
## # … with 15,482 more rows, and 24 more variables: educ <fct>, inc_q <fct>,
## #   emp_in <fct>, fin2 <fct>, fin7 <fct>, fin8 <fct>, fin14a <fct>,
## #   fin14b <fct>, fin19 <fct>, fin26 <fct>, fin28 <fct>, fin32 <fct>,
## #   fin37 <fct>, mobileowner <fct>, account_fin <fct>, account_mob <fct>,
## #   account <fct>, saved <fct>, borrowed <fct>, receive_wages <fct>,
## #   receive_transfers <dbl>, receive_pension <dbl>,
## #   receive_agriculture <dbl>, fin2_b <dbl>
```

# `purrr`: unleashing the true power

```
(findex_model <- findex %>%
     group_by(economy, regionwb) %>%
     nest() # take every peace of data associated with the group
 )
```

```
## # A tibble: 144 x 3
##    economy             regionwb                                data
##    <chr>               <chr>                                   <list>
##  1 Mozambique          Sub-Saharan Africa (excluding high i… <tibble [93 ×…
##  2 Ukraine             Europe & Central Asia (excluding hig… <tibble [108 …
##  3 Gabon               Sub-Saharan Africa (excluding high i… <tibble [99 ×…
##  4 Venezuela, RB       Latin America & Caribbean (excluding… <tibble [111 …
##  5 Rwanda              Sub-Saharan Africa (excluding high i… <tibble [99 ×…
##  6 Montenegro          Europe & Central Asia (excluding hig… <tibble [90 ×…
##  7 Morocco             Middle East & North Africa (excludin… <tibble [579 …
##  8 Cambodia            East Asia & Pacific (excluding high … <tibble [154 …
##  9 Bolivia             Latin America & Caribbean (excluding… <tibble [99 ×…
## 10 Central African Re… Sub-Saharan Africa (excluding high i… <tibble [104 …
## # … with 134 more rows
```

# `purrr`: unleashing the true power

```
(findex_model <- findex %>%
    group_nest(economy, regionwb) %>% # Notice that there is a function that does the =
    mutate(model=map(data # loop over each element of the column data
                     , ~glm(fin2_b~pop_adult+female+age+educ # specification
                            , data = . # the dot is interpreted as every value of data
                            , family = binomial() # link function
                            )
                     )
           )
 )
```

```
## # A tibble: 144 x 4
##    economy    regionwb                                  data            model
##    <chr>      <chr>                                     <list>          <list>
##  1 Afghanist… South Asia                                <tibble [96 × … <S3: g…
##  2 Albania    Europe & Central Asia (excluding hig…     <tibble [98 × … <S3: g…
##  3 Algeria    Middle East & North Africa (excludin…     <tibble [90 × … <S3: g…
##  4 Argentina  Latin America & Caribbean (excluding…     <tibble [114 ×… <S3: g…
##  5 Armenia    Europe & Central Asia (excluding hig…     <tibble [106 ×… <S3: g…
##  6 Australia  High income: OECD                         <tibble [92 × … <S3: g…
##  7 Austria    High income: OECD                         <tibble [84 × … <S3: g…
##  8 Azerbaijan Europe & Central Asia (excluding hig…     <tibble [103 ×… <S3: g…
##  9 Bahrain    High income: nonOECD                      <tibble [123 ×… <S3: g…
## 10 Bangladesh South Asia                                <tibble [85 × … <S3: g…
## # … with 134 more rows
```

# `purrr`: unleashing the true power

```
(findex_model <- findex %>%
    group_nest(economy, regionwb) %>%
    mutate(model=map(data
                     , ~glm(fin2_b~female+age+educ
                            , data = .
                            , family = binomial()
                            )
                     )
           , coef=map(model, tidy) # coefficients as tibble
           , stats=map(model, glance) # summary statistics
           , fitted=map(model, augment) # adds info on each observation
           )
 )
```

```
## # A tibble: 144 x 7
##    economy   regionwb            data      model  coef    stats   fitted
##    <chr>     <chr>               <list>    <list> <list>  <list>  <list>
##  1 Afghanis… South Asia          <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  2 Albania   Europe & Central A… <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  3 Algeria   Middle East & Nort… <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  4 Argentina Latin America & Ca… <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  5 Armenia   Europe & Central A… <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  6 Australia High income: OECD   <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  7 Austria   High income: OECD   <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  8 Azerbaij… Europe & Central A… <tibble … <S3: … <tibbl… <tibbl… <tibble …
##  9 Bahrain   High income: nonOE… <tibble … <S3: … <tibbl… <tibbl… <tibble …
## 10 Banglade… South Asia          <tibble … <S3: … <tibbl… <tibbl… <tibble …
## # … with 134 more rows
```

# `purrr`: unleashing the true power

```
findex_model$coef[[1]]
```

```
## # A tibble: 5 x 5
##   term                              estimate std.error statistic p.value
##   <chr>                                <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)                     -2.66e+ 1    98716. -2.69e- 4   1.000
## 2 femaleFemale                    -2.97e-14    79568. -3.73e-19   1
## 3 age                              4.23e-16     3114.  1.36e-19   1
## 4 educsecondary                   -1.67e-14    93013. -1.80e-19   1
## 5 educcompleted tertiary or more -3.49e-14   216310. -1.62e-19   1
```

# `purrr`: unleashing the true power

Let's extract the log odds gender==female

```
(findex_model %>%
    mutate(female_coef=map(coef, ~.x %>%
                             dplyr::filter(term=="femaleFemale")
                           )
           )
 )
```

```
## # A tibble: 144 x 8
##    economy  regionwb       data    model  coef   stats  fitted  female_coef
##    <chr>    <chr>          <list>  <list> <list> <list> <list>  <list>
##  1 Afghani… South Asia     <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  2 Albania  Europe & Cent… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  3 Algeria  Middle East &… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  4 Argenti… Latin America… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  5 Armenia  Europe & Cent… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  6 Austral… High income: … <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  7 Austria  High income: … <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  8 Azerbai… Europe & Cent… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  9 Bahrain  High income: … <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
## 10 Banglad… South Asia     <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
## # … with 134 more rows
```

# `purrr`: unleashing the true power

Why not calculing confidence intervals? and extracting the relevant variables

```
(findex_model %>%
    mutate(female_coef=map(coef, ~.x %>%
                             dplyr::filter(term=="femaleFemale") %>%
                             mutate(min_int=estimate-1.96*std.error
                                    , max_int=estimate+1.96*std.error) %>%
                             dplyr::select(estimate, p.value, contains("int")
                                           )
                           )
           )
 )
```

```
## # A tibble: 144 x 8
##    economy  regionwb       data    model  coef   stats  fitted  female_coef
##    <chr>    <chr>          <list>  <list> <list> <list> <list>  <list>
##  1 Afghani… South Asia     <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  2 Albania  Europe & Cent… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  3 Algeria  Middle East &… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  4 Argenti… Latin America… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  5 Armenia  Europe & Cent… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  6 Austral… High income: … <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  7 Austria  High income: … <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  8 Azerbai… Europe & Cent… <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
##  9 Bahrain  High income: … <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
## 10 Banglad… South Asia     <tibbl… <S3: … <tibb… <tibb… <tibbl… <tibble [1…
## # … with 134 more rows
```

# `purrr`: unleashing the true power

Let's get rid of the columns that we are not interested

```
(findex_model %>%
    mutate(female_coef=map(coef, ~.x %>%
                           dplyr::filter(term=="femaleFemale") %>%
                           mutate(min_int=estimate-1.96*std.error
                                  , max_int=estimate+1.96*std.error) %>%
                           dplyr::select(estimate, p.value, contains("int")
                                         )
                          )
           ) %>%
     dplyr::select(economy, regionwb, female_coef)
 )
```

```
## # A tibble: 144 x 3
##    economy     regionwb                                        female_coef
##    <chr>       <chr>                                           <list>
##  1 Afghanistan South Asia                                      <tibble [1 × 4…
##  2 Albania     Europe & Central Asia (excluding high incom…    <tibble [1 × 4…
##  3 Algeria     Middle East & North Africa (excluding high …    <tibble [1 × 4…
##  4 Argentina   Latin America & Caribbean (excluding high i…    <tibble [1 × 4…
##  5 Armenia     Europe & Central Asia (excluding high incom…    <tibble [1 × 4…
##  6 Australia   High income: OECD                               <tibble [1 × 4…
##  7 Austria     High income: OECD                               <tibble [1 × 4…
##  8 Azerbaijan  Europe & Central Asia (excluding high incom…    <tibble [1 × 4…
##  9 Bahrain     High income: nonOECD                            <tibble [1 × 4…
## 10 Bangladesh  South Asia                                      <tibble [1 × 4…
## # … with 134 more rows
```

# `purrr`: unleashing the true power

What's inside female_coef?

```
(findex_model <- findex_model %>%
    mutate(female_coef=map(coef, ~.x %>%
                             dplyr::filter(term=="femaleFemale") %>%
                             mutate(min_int=estimate-1.96*std.error
                                    , max_int=estimate+1.96*std.error) %>%
                             dplyr::select(estimate, p.value, contains("int")
                                           )
                           )
           ) %>%
     dplyr::select(economy, regionwb, female_coef) %>%
     unnest(female_coef)
 )
```

```
## # A tibble: 144 x 6
##    economy   regionwb                  estimate p.value   min_int   max_int
##    <chr>     <chr>                        <dbl>   <dbl>     <dbl>     <dbl>
##  1 Afghanis… South Asia               -2.97e-14  1       -1.56e+5   1.56e+5
##  2 Albania   Europe & Central Asia (… -1.06e- 2  0.986   -1.19e+0   1.17e+0
##  3 Algeria   Middle East & North Afr… -6.69e- 1  0.238   -1.78e+0   4.41e-1
##  4 Argentina Latin America & Caribbe…  3.86e- 1  0.336   -4.00e-1   1.17e+0
##  5 Armenia   Europe & Central Asia (… -2.54e- 1  0.595   -1.19e+0   6.80e-1
##  6 Australia High income: OECD         3.09e+ 0  0.123   -8.38e-1   7.02e+0
##  7 Austria   High income: OECD        -2.26e- 1  0.865   -2.84e+0   2.39e+0
##  8 Azerbaij… Europe & Central Asia (…  8.16e- 1  0.118   -2.07e-1   1.84e+0
##  9 Bahrain   High income: nonOECD     -1.44e+ 0  0.0178  -2.64e+0  -2.50e-1
## 10 Banglade… South Asia               -1.83e+ 0  0.103   -4.03e+0   3.68e-1
## # … with 134 more rows
```

# `purrr`: unleashing the true power

Little exploration maybe?

```
findex_model %>% summary()
```

```
##    economy            regionwb            estimate        
##  Length:144         Length:144         Min.   :-18.3808  
##  Class :character   Class :character   1st Qu.: -0.7143  
##  Mode  :character   Mode  :character   Median : -0.2203  
##                                        Mean   : -0.3807  
##                                        3rd Qu.:  0.1082  
##                                        Max.   : 34.1079  
##     p.value            min_int              max_int       
##  Min.   :0.0000022   Min.   :-155953.92   Min.   :    -1.10  
##  1st Qu.:0.2057683   1st Qu.:     -2.16   1st Qu.:     0.48  
##  Median :0.4657282   Median :     -1.45   Median :     0.94  
##  Mean   :0.4920671   Mean   :  -3344.79   Mean   :  3344.03  
##  3rd Qu.:0.8000435   3rd Qu.:     -0.95   3rd Qu.:     1.47  
##  Max.   :1.0000000   Max.   :      0.16   Max.   :155953.92
```

# `purrr`: unleashing the true power

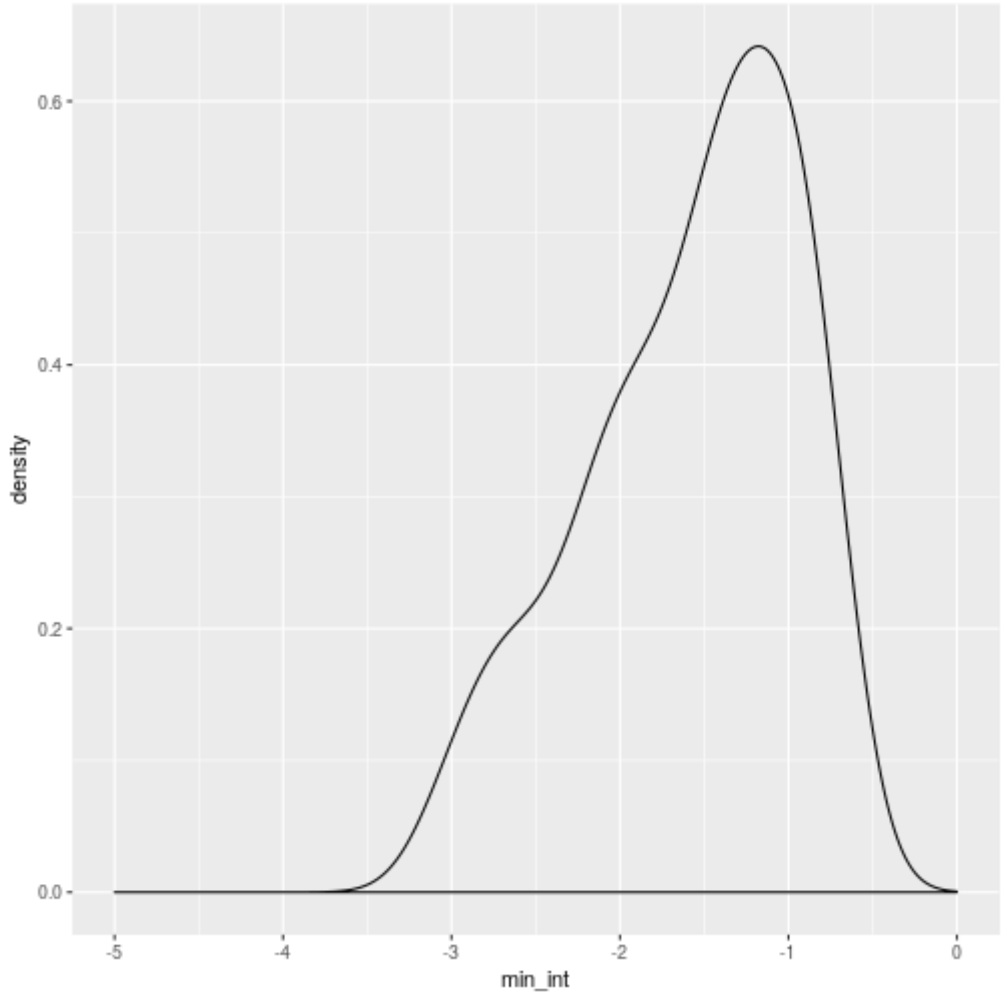
The natural question is "why", but for the sake of the exercise we are going to truncate the data and keep the record that lie between the 10% and 90% percentile of the lower confidence interval

```
(findex_model_coef <- findex_model %>%
    filter_at(.vars = vars(min_int)
              , .vars_predicate = any_vars(between(., quantile(., 0.1)
                                                   , quantile(., 0.9)
                                                   )
                                           )
              ) )
```

```
## # A tibble: 114 x 6
##    economy   regionwb                          estimate p.value min_int max_int
##    <chr>     <chr>                                <dbl>   <dbl>   <dbl>   <dbl>
##  1 Albania   Europe & Central Asia (exclu…  -0.0106  0.986   -1.19    1.17
##  2 Algeria   Middle East & North Africa (…  -0.669   0.238   -1.78    0.441
##  3 Armenia   Europe & Central Asia (exclu…  -0.254   0.595   -1.19    0.680
##  4 Australia High income: OECD               3.09    0.123   -0.838   7.02
##  5 Austria   High income: OECD              -0.226   0.865   -2.84    2.39
##  6 Bahrain   High income: nonOECD           -1.44    0.0178  -2.64   -0.250
##  7 Belarus   Europe & Central Asia (exclu…   0.0664  0.898   -0.947   1.08
##  8 Belgium   High income: OECD               0.0158  0.989   -2.16    2.19
##  9 Benin     Sub-Saharan Africa (excludin…  -0.0779  0.912   -1.46    1.30
## 10 Bolivia   Latin America & Caribbean (e…  -0.0690  0.888   -1.03    0.888
## # … with 104 more rows
```

# `purrr`: unleashing the true power

```
g0 <- findex_model_coef %>%
    qplot(min_int, geom = "density", data = ., ylab = "density
    scale_x_continuous(limits = c(-5, 0))
```



# purrr: unleashing the true power

```
g1 <- findex_model_coef %>%
    filter(grepl(regionwb, pattern = "High")) %>%
    ggplot() +
    geom_point(aes(economy, estimate, col=economy))+
    geom_linerange(aes(ymin=min_int, ymax=max_int, x=economy))
    geom_hline(yintercept = 0, linetype="dotted", col="red")+
    coord_flip()+
    facet_wrap(~regionwb)
```