Lesson 3 of 5
Modeling and Analysis

Intro to R workshop, LU Skills School

Instructor:

Christopher Swader
LU Sociology (Assoc. Prof) and LMU (Munich, Researcher)

Teaching Assistant: Maximilian Hornung (MS programme in Social Scientific Data Analysis, LU)

Slides adapted from and inspired by:
Irina Vartanova, Institute for Future Studies, Stockholm

Today’s agenda

- Aggregating Data
- Grouping Cases
- Descriptives
- Bivariate Analyses
- Multivariate Analyses
- Quadratic Terms
- Interaction Terms

- Use the link to download the files we will be using today: https://github.com/ChristopherSwader/R_introduction
- Download the Day3 folder.

Aggregate

- Clear your environment
- Set your working directory to Day 3.
- Load the moral_issues data from day 2!
    - Either navigate through Rstudio’s import menu and paste the code
    - Or if you know the code but just need the path, you can get it through the files pane → settings

summarise()

Compute table of summaries.

Alternative for one summary stat:

aggregate(moral_issues$n, by=list(issue=moral_issues$issue), sum)

```
moral_issues %>%
  group_by(issue) %>%
  summarise(total = sum(n), max = max(n))
```

moral_issues

issue	year	n	prop
fehome	1990	890	0.8213483
libath	1990	881	0.6912599
marblk	1990	940	0.1670213
polescap	1990	854	0.2236534
spkrac	1990	892	0.6423767
fehire	1996	1236	0.6480583

→

```
  issue    total  max
  <chr>    <dbl> <dbl>
1 abany    36794  1939
2 abdefect 44152  1926
3 abhlth   44360  1934
4 abnomore 44072  1925
5 abpoor   44023  1928
6 abrape   43922  1934
7 absingle 44020  1938
8 advfront  8559  1806
9 affrmact 18944  1904
10 busing  22886  1859
```

Your Turn

Alter the last code to extract the rows where ***issue == "abany"****. Then use* ***summarise()*** *and* ***mean(), min(),*** *and* ***max()*** *to find:*

1. *The average public opinion for the issue over all time points it was measured.*
2. *The first and the last* ***years*** *the issue appeared in the survey.*

```
moral_issues %>%
  filter(issue == "abany") %>%
  summarise(mean_prop = mean(prop),
            first = min(year),
            last = max(year))

# A tibble: 1 × 3
  mean_prop first  last
      <dbl> <dbl> <dbl>
1     0.414  1977  2018
```

n()

The number of rows in a dataset/group

```
moral_issues %>% summarise(n = n())
```

moral_issues

issue	year	n	prop
fehome	1990	890	0.8213483
libath	1990	881	0.6912599
marblk	1990	940	0.1670213
polescap	1990	854	0.2236534
spkrac	1990	892	0.6423767
fehire	1996	1236	0.6480583

→

n
1651

A more simple alternative:

nrow(moral_issues)

n_distinct()

The number of distinct values in a variable

```
moral_issues %>% summarise(n = n(),
                           n_issue = n_distinct(issue))
```

moral_issues

issue	year	n	prop
fehome	1990	890	0.8213483
libath	1990	881	0.6912599
marblk	1990	940	0.1670213
polescap	1990	854	0.2236534
spkrac	1990	892	0.6423767
fehire	1996	1236	0.6480583

n	n_issue
1651	81

A base r alternative:

length(unique(moral_issues$issue))

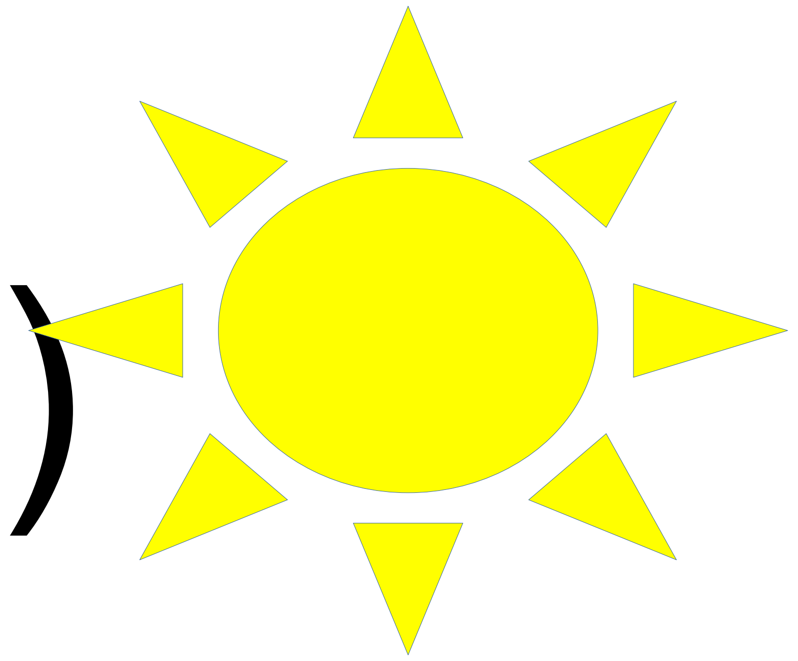
More on grouping cases

Here: grouping by just one variable (city)

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
18,5	37	2
19,0	38	2
88,5	177	2

group_by() + summarise()



group_by()

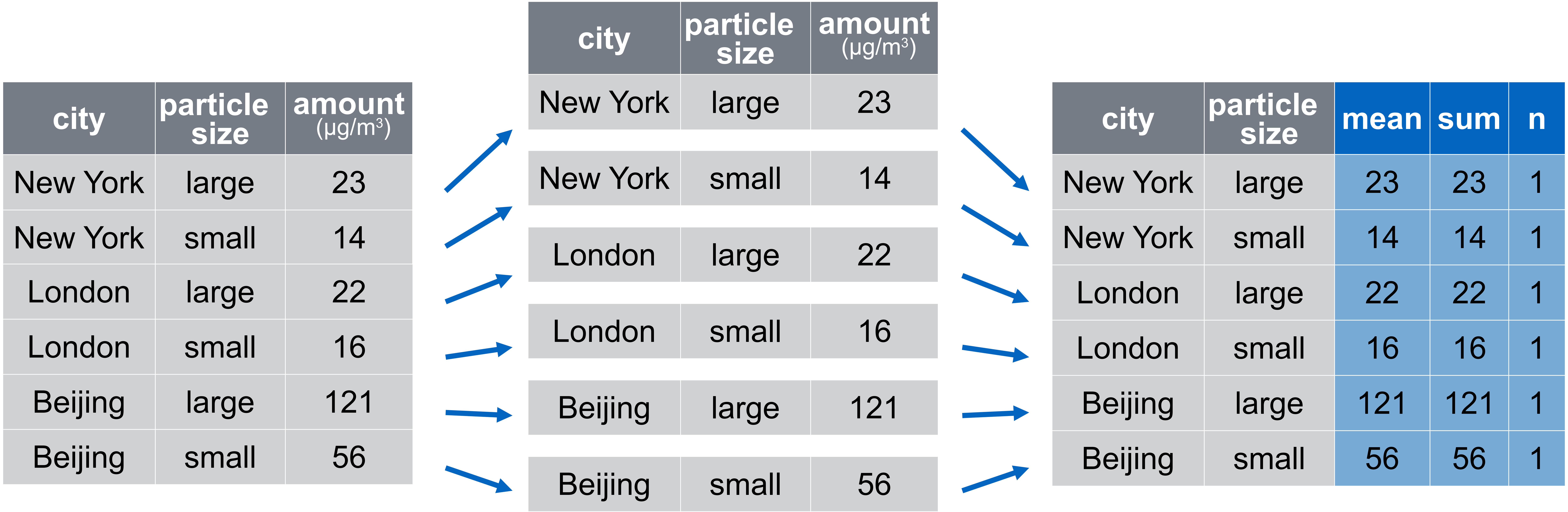
Groups cases by common values of one **or more** columns.

```
pollution %>%
  group_by(city)
```

```
# A tibble: 6 x 3
# Groups:   city [3]
  city     size  amount
  <chr>    <chr>  <dbl>
1 New York large     23
2 New York small     14
3 London   large     22
```

They are grouped, but no operations are yet performed on the cases as groups.

group_by() with multiple grouping variables



city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

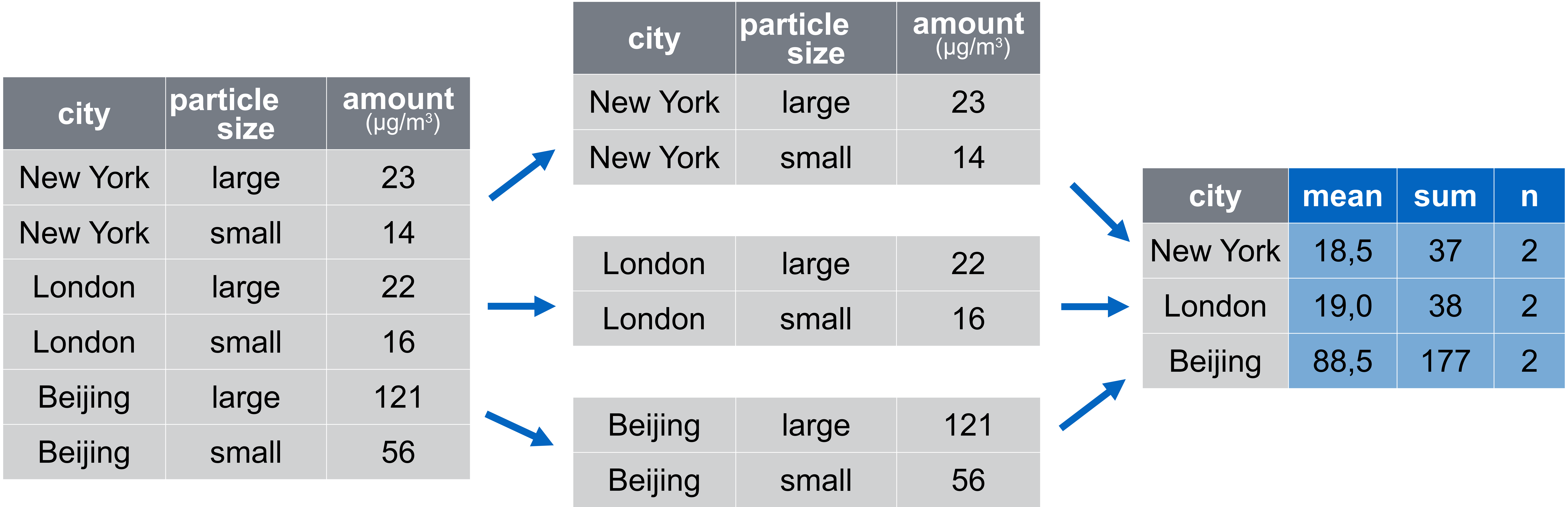
city	particle size	mean	sum	n
New York	large	23	23	1
New York	small	14	14	1
London	large	22	22	1
London	small	16	16	1
Beijing	large	121	121	1
Beijing	small	56	56	1

```
pollution %>%
  group_by(city, size) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

Your Turn

Use group_by() and summarize to get the mean() the sum() and the n() of each city in the toy pollution dataset.

group_by()



city	particle size	amount (μg/m^3)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount (μg/m^3)
New York	large	23
New York	small	14

London	large	22
London	small	16

Beijing	large	121
Beijing	small	56

city	mean	sum	n
New York	18,5	37	2
London	19,0	38	2
Beijing	88,5	177	2

```
pollution %>%
  group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

ungroup()

<u>Removes</u> grouping criteria from a data frame.

```
pollution %>%
  group_by(city) %>%
  ungroup() %>%
  summarise(sum = sum(amount))
```

```
pollution %>%
  group_by(city) %>%
  summarise(sum = sum(amount))
```

sum
252

city	sum
New York	37
London	38
Beijing	177

Your Turn

*With moral_issues, use **group_by()**, **summarise()**, and **arrange()** to display the **issues** with the highest average public opinion.*

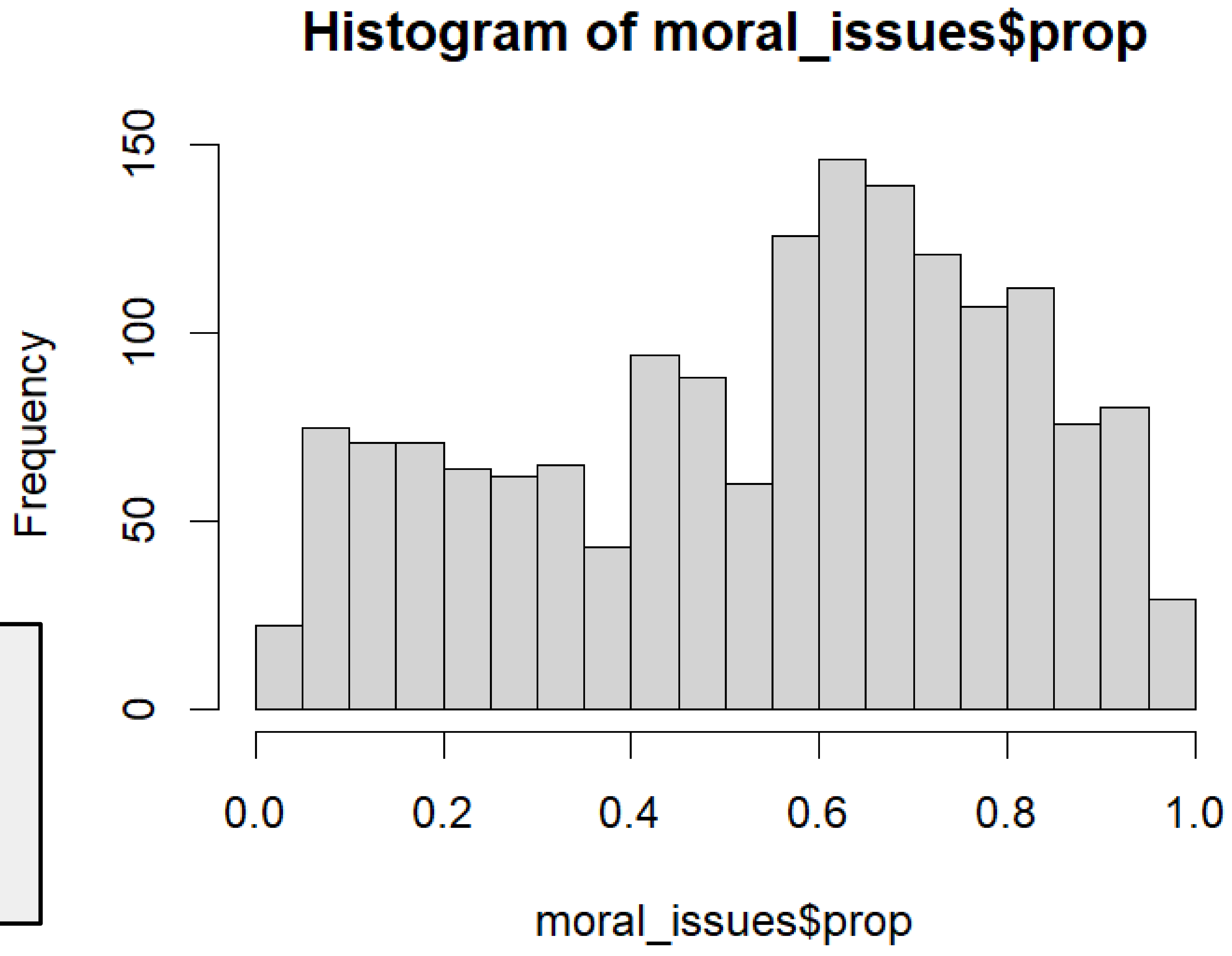
```
moral_issues %>%
  group_by(issue) %>%
  summarise(mean_prop = mean(prop)) %>%
  arrange(desc(mean_prop))


#   issue     mean_prop
# 1 hitmarch      0.967
# 2 racfew        0.952
# 3 marwht        0.943
# 4 hitdrunk      0.915
# 5 polmurdr      0.901
# 6 abhlth        0.896
# 7 polabuse      0.886
# … with 71 more rows
```

Descriptives

Descriptives of one (numerical) variable

```
summary(moral_issues$prop)
mean(moral_issues$prop)
min(moral_issues$prop)
max(moral_issues$prop)
sd(moral_issues$prop)

hist(moral_issues$prop, breaks=20)
```

This histogram version is the basic one. GGplot2 can produce much more fancy versions. Day 4



Descriptives of one (categorical) variable

```
count(moral_issues, issue) #tidyverse way
table(moral_issues$issue) #base R way
```

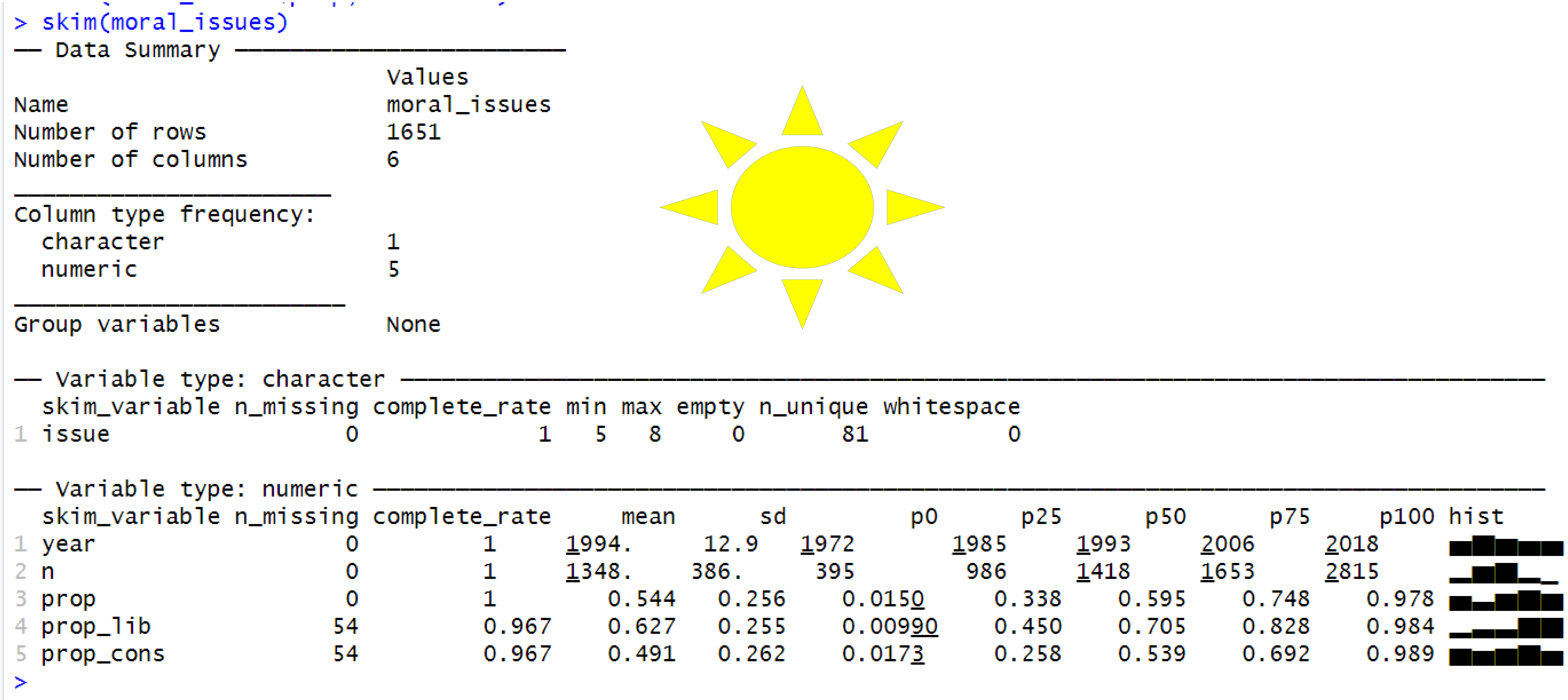
Descriptives of all variables:

Can i just get a quick snapshot?

Your Turn

Install the package ”skimr”. Load it. Then run the function skim() on moral_issues.

skim() output: Look at whole dataset in a glance



```
> skim(moral_issues)
── Data Summary ────────────────────────
                           Values
Name                       moral_issues
Number of rows             1651
Number of columns          6
_______________________
Column type frequency:
  character                1
  numeric                  5
________________________
Group variables            None

── Variable type: character ──────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate min max empty n_unique whitespace
1 issue                 0             1   5   8     0       81          0

── Variable type: numeric ────────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate     mean      sd        p0      p25      p50      p75     p100 hist
1 year                  0         1     1994.     12.9   1972       1985     1993     2006     2018     ▆▇▇▆▆
2 n                     0         1     1348.    386.     395        986     1418     1653     2815     ▂▇▇▂▁
3 prop                  0         1        0.544   0.256    0.0150     0.338    0.595    0.748    0.978 ▆▃▇▇▇
4 prop_lib             54         0.967    0.627   0.255    0.00990    0.450    0.705    0.828    0.984 ▁▂▂▇▇
5 prop_cons            54         0.967    0.491   0.262    0.0173     0.258    0.539    0.692    0.989 ▇▆▇▇▆
>
```

Bivariate relationships

Correlation Tables

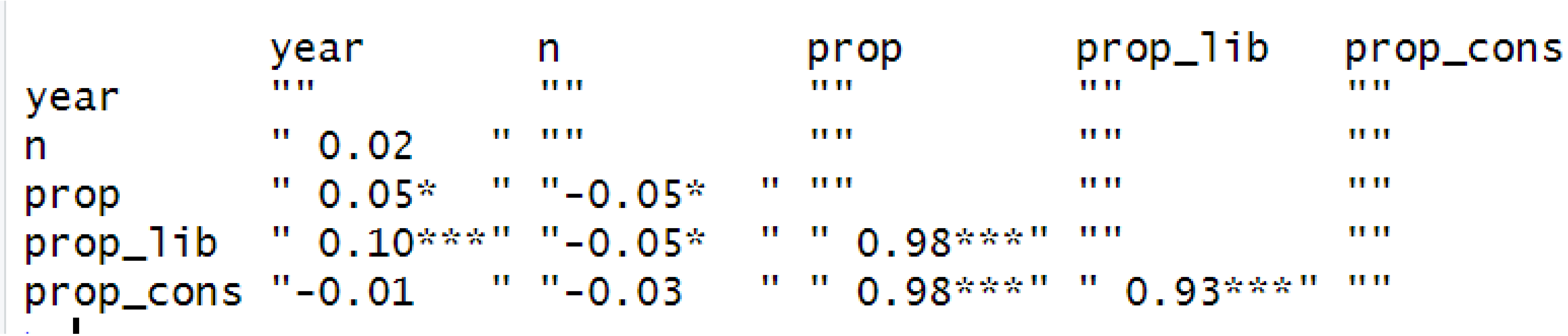
```
cor(na.omit(moral_issues)[,-1]) #
```

Can you explain my code?

You can specify method=”spearman” for ordinal data

Correlation Tables with signif levels

```
install.packages("corrtable")

library(corrtable)

correlation_matrix(moral_issues, digits = 2 , use = "lower",
replace_diagonal = T)
```

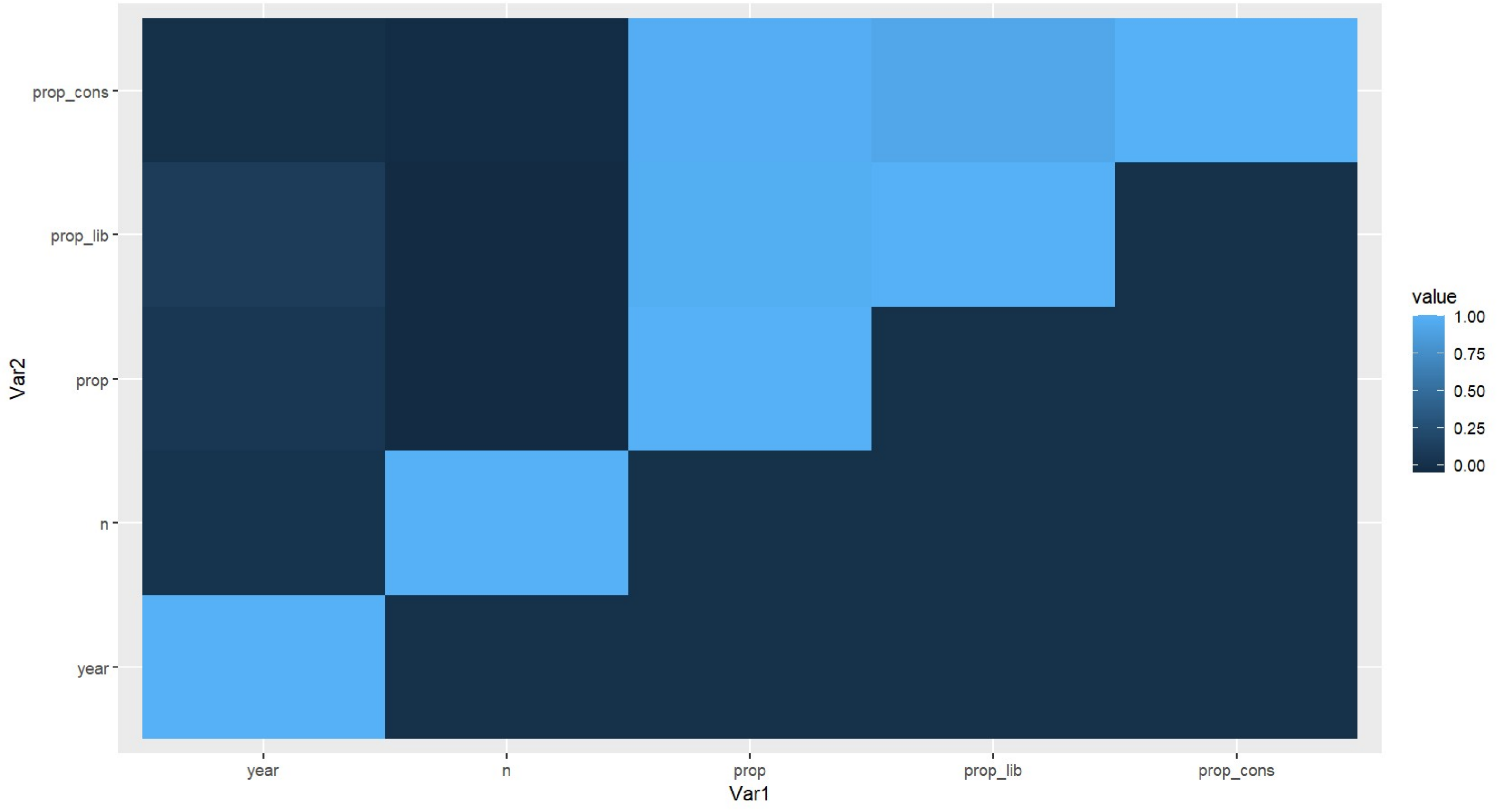


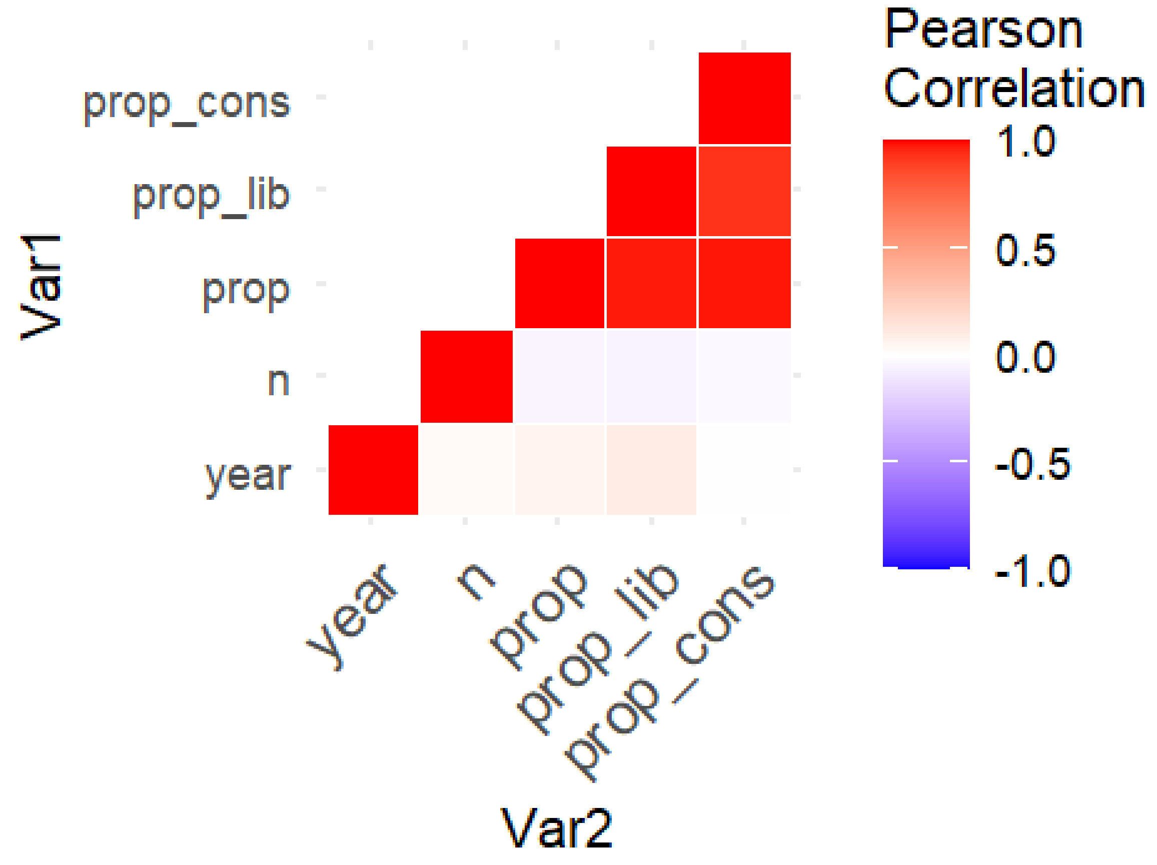
```
          year       n          prop       prop_lib   prop_cons
year      ""         ""         ""         ""         ""
n         " 0.02   " ""         ""         ""         ""
prop      " 0.05*  " "-0.05*  " ""         ""         ""
prop_lib  " 0.10***" "-0.05*  " " 0.98***" ""         ""
prop_cons "-0.01   " "-0.03   " " 0.98***" " 0.93***" ""
```

Correlation Heatmaps in R

Try https://r-graph-gallery.com/ for a huge range of R resources

```
library(reshape2)
upper_tri <-my_corr_matrix*upper.tri(my_corr_matrix, diag = T)

my_corr_matrix <- cor(na.omit(moral_issues)[,-1])

melted_matrix <- melt(upper_tri)
library(ggplot2)

ggplot(data = melted_matrix, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()

ggplot(data = melted_matrix, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
```





Bi and Multivariate Modeling

Some traditional modelling functions in R

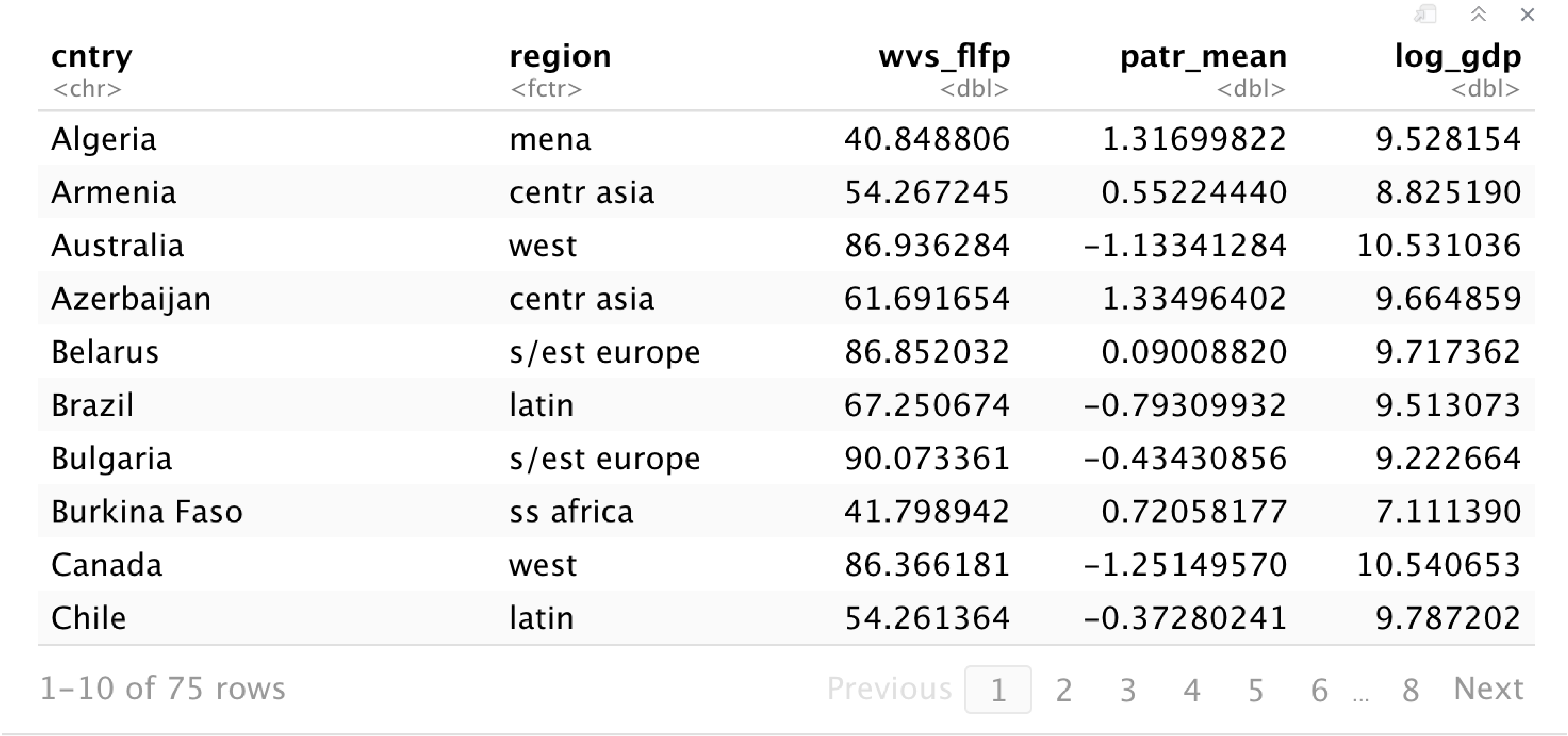
function	package	fits
lm()	**stats**	**linear models**
glm()	**stats**	**generalized linear models**
gam()	mgcv	generalized additive models
rlm()	MASS	robust linear models
lmer()	lme4	linear mixed-effects models

Also consider machine learning methods, all fully accessible via R: random forest, artificial neural nets, topic modeling, etc.

(Popular) modelling functions in R

function	package	fits
lm()	stats	linear models
glm()	stats	generalized linear models
gam()	mgcv	generalized additive models
rlm()	MASS	robust linear models
lmer()	lme4	linear mixed-effects models

```
flfp #female labor force participation
```



cntry <chr>	region <fctr>	wvs_flfp <dbl>	patr_mean <dbl>	log_gdp <dbl>
Algeria	mena	40.848806	1.31699822	9.528154
Armenia	centr asia	54.267245	0.55224440	8.825190
Australia	west	86.936284	-1.13341284	10.531036
Azerbaijan	centr asia	61.691654	1.33496402	9.664859
Belarus	s/est europe	86.852032	0.09008820	9.717362
Brazil	latin	67.250674	-0.79309932	9.513073
Bulgaria	s/est europe	90.073361	-0.43430856	9.222664
Burkina Faso	ss africa	41.798942	0.72058177	7.111390
Canada	west	86.366181	-1.25149570	10.540653
Chile	latin	54.261364	-0.37280241	9.787202

1-10 of 75 rows

skim()

Display summary statistics

```r
library(skimr)
skim(flfp)
```

The data set

```
library(skimr)
skim(flfp)
```

```
── Data Summary ────────────────────────
                           Values
Name                       flfp
Number of rows             44670
Number of columns          15
_______________________
Column type frequency:
  character                1
  factor                   7
  numeric                  7
________________________
Group variables            None

── Variable type: character ──────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate min max empty n_unique whitespace
1 year                  0             1   4   4     0       10          0

── Variable type: factor ─────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique top_counts
1 cntry                 0         1     FALSE         78 Sou: 1483, Ind: 1312, Ira: 1174, Jap: 1052
2 denom              1411         0.968 FALSE          4 Chr: 18632, Mus: 12567, Non: 7131, Oth: 4929
3 age_gr              125         0.997 FALSE          6 26-: 11959, 36-: 10847, 46-: 8435, 18-: 7086
4 edu                 705         0.984 FALSE          3 Mid: 21769, Low: 14547, Hig: 7649
5 marit                94         0.998 FALSE          3 Mar: 30668, Sin: 8257, Div: 5651
6 children            608         0.986 FALSE          4 2-3: 18755, No : 9784, 1 c: 8253, 4 a: 7270
7 region                0         1     FALSE          7 Cen: 7123, Eas: 6981, Sou: 6754, MEN: 6428

── Variable type: numeric ────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate    mean    sd    p0    p25     p50   p75 p100 hist
1 wgt                   0         1     0.981  0.380 0.0574 0.877  1      1     5    ▇▁▁▁▁
2 lfp                   0         1     0.657  0.475  0      0     1      1     1    ▅▁▁▁▇
3 patr_values        2836         0.937 -0.0171 0.984 -3.20  -0.796 -0.0752 0.648 5.56 ▁▇▅▁▁
4 religious             7         1.00  0.431  0.495  0      0     0      1     1    ▇▁▁▁▇
5 patr_mean           621         0.986 0.112  0.984 -2.08  -0.793 0.120  0.879 1.86 ▂▅▇▅▆
6 log_gdp             998         0.978 9.41   1.03   6.70   8.72  9.51   10.3  11.7 ▁▃▇▅▁
7 muslim_cntry          0         1     0.303  0.460  0      0     0      1     1    ▇▁▁▁▃
```

A bit wierd that year is stored as a character variable?

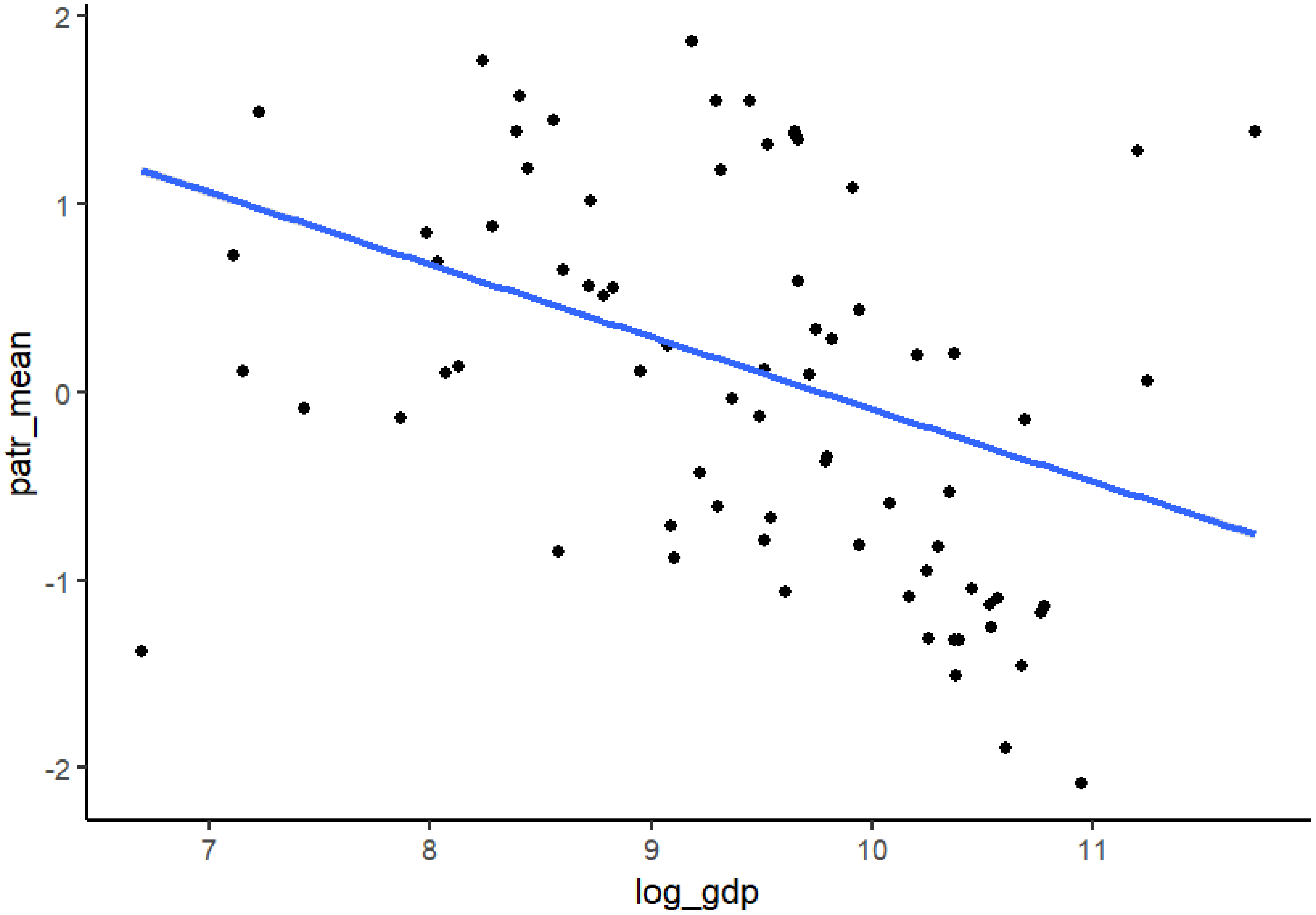
Convert it to numeric or integer!

```
flfp <- type_convert(flfp)
#or
flfp$year <- as.integer(flfp$year)
```

Notice type_convert will give different results from type.convert

lm()

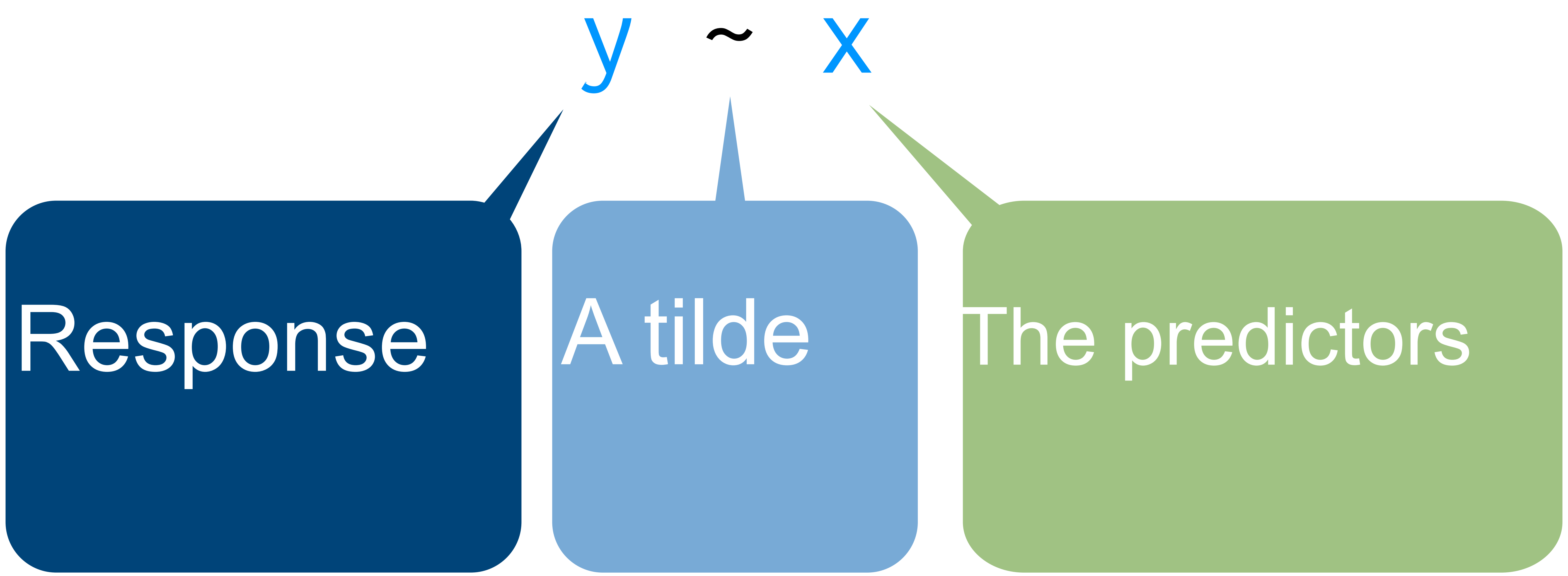
```
ggplot(flfp_agg, aes(x = log_gdp, y = patr_mean)) +
  geom_point()+
  geom_smooth(method = "lm" )
```



formulas

Formula only needs to include the response/dependent variable and predictors/(independent variables & controls)

$$y = \alpha + \beta x + \epsilon$$



y ~ x

Response

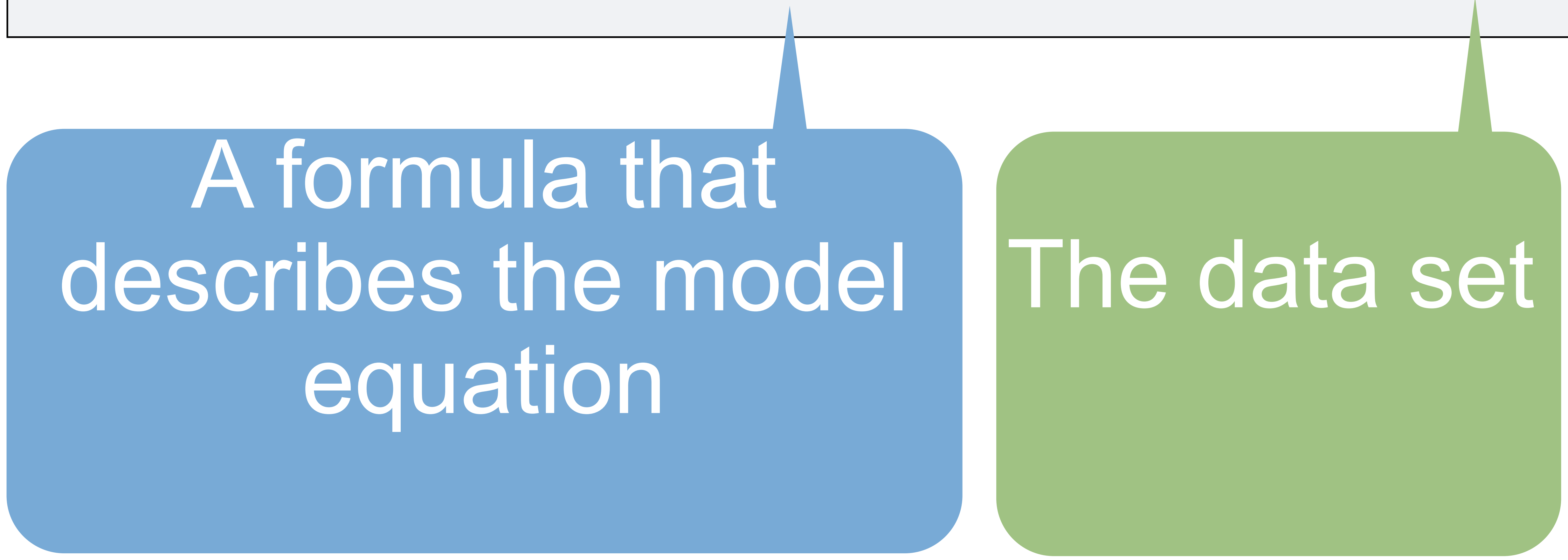
A tilde

The predictors

lm()

Fit a linear model to data

```
my_model <- lm(patr_mean ~ log_gdp, data = flfp_agg)
```



A formula that describes the model equation

The data set

Your Turn

Fit the model below and then examine the output. What does it look like?

```
my_model <- lm(patr_mean ~ log_gdp, data = flfp_agg)
```

```
my_model
Call:
lm(formula = patr_mean ~ log_gdp, data =
flfp_agg)

Coefficients:
(Intercept)      log_gdp
     3.4488      -0.3621
```

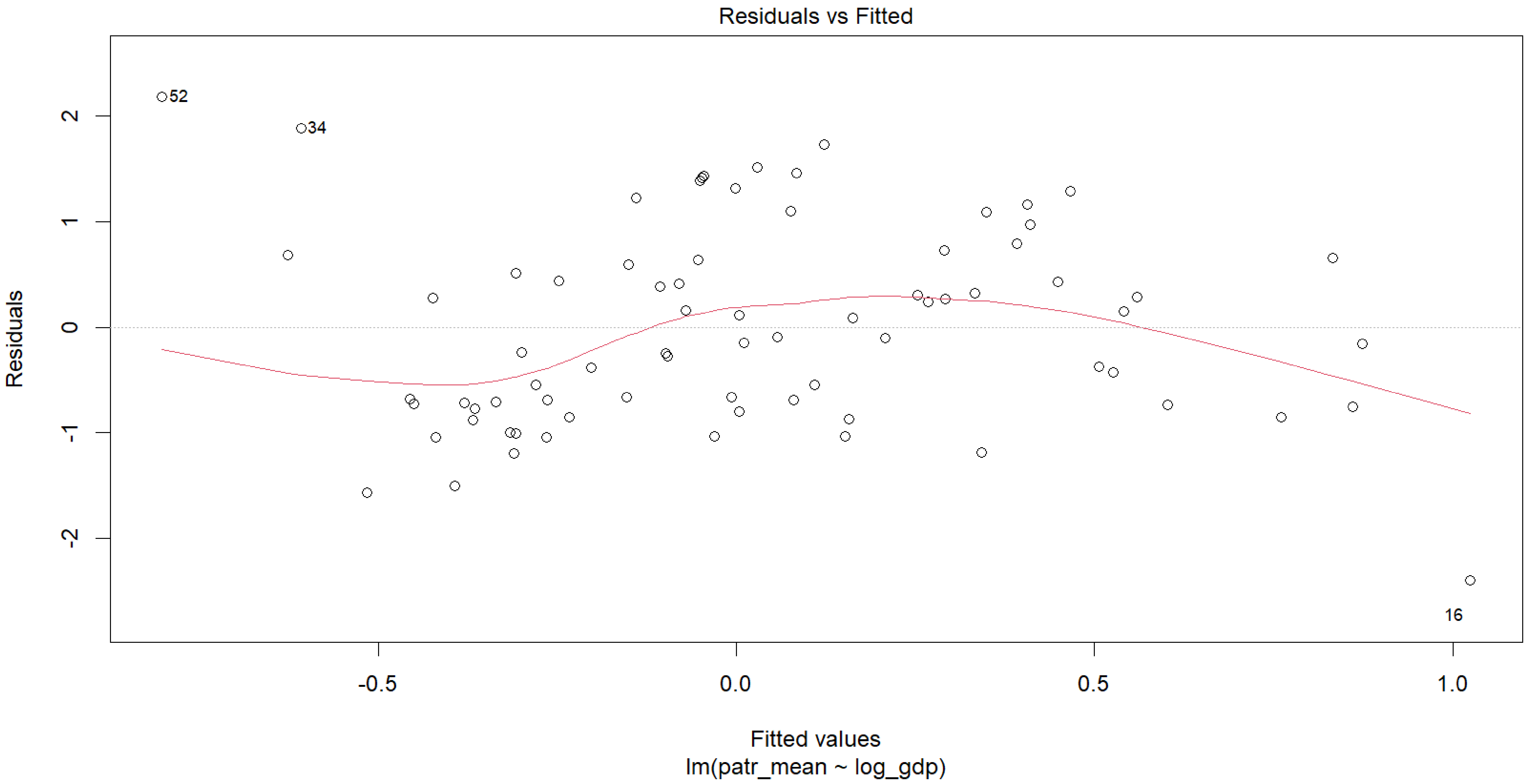
Summary!

```
summary(my_model)
```

```
Call:
lm(formula = patr_mean ~ log_gdp, data = flfp_agg)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.4011 -0.7323 -0.1004  0.6470  2.1837 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   3.4488     0.9818   3.513 0.000765 ***
log_gdp      -0.3621     0.1034  -3.504 0.000788 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9492 on 73 degrees of freedom
Multiple R-squared:  0.144,	Adjusted R-squared:  0.1322 
F-statistic: 12.28 on 1 and 73 DF,  p-value: 0.0007875
```

Model diagnostics

```
plot(my_model, which = 1)
```



Regression Assumptions/Issues

Assumption	When to check	How to test	Violating this assumption implies	How to Respond
1) Linear relationship between Y and the X terms	Check this first, during your bivariate checks.	Scatterplot with regression line. Correlation coefficient (Pearson's R).	Biased coefficients	Transformation of an X into ordinal form. Adding an X-squared term. Transform your Y variable. (This assumption is not problematic if the residuals are homoscedastic and normally distributed (points 4 and 5)).
2) Low multicollinearity	Can do initial check in bivariate analysis if you like, (correlations) but def. check this in the coefficient table for your multiple regression.	Low Pearson's R in X correlations (.8 or smaller) You want VIF less than 3. 5 is potential problem. 10 is a violation. (Ignore this in interaction terms)	Biased p-values (inflated confidence intervals and signficance)	Remove a variable or combine two into an index. (If you are only predicting (e.g. Machine learning), do nothing.)
3) Normal distribution of errors (residuals)	Check this when you find a model you are beginning to like.	P-P plot & residuals histogram (regression-plots window) Or Q-Q plot (more tricky)	Biased p-values	**In large samples, not a worry**
4) Homoscedasticity (constant variance) of errors (residuals) across all values of X	Check this when you find a model you are beginning to like.	Scatterplot of residuals by predicted value (Y: zresid, X:zpred), (regression-plots window) Avoid 'funneling'	Biased p-values	**In large samples, not necessarily a worry.** Redefine variables with large ranges (e.g. raw income) Y transformation (e.g. box-cox transformation, log transformation) Switch to a different kind of regression (read Williams et al.)
5) Errors should be independent from one another	Check this when you find a model you are beginning to like.	Durbin-Watson (1.5 to 2.5 is normal. Close to 0 or 4 is problematic) (regression-statistics window)	Biased p-values	Depends on type of dependence (e.g. time series model or multi-level regression. There is some major grouping variable you are probably missing, e.g. country)
6) Outliers	Check this when you find a model you are beginning to like.	Cook's distance (you want smaller than .5, definitely smaller than 1) (regression-save window) Casewise diagnostics(regression-	Biased coefficients	Doublecheck data accuracy. **In a big sample, usually do nothing.**

See Williams et. al 2013

broom

Turns model output into data frames

```
# install.packages("tidyverse")
library(broom)
```

broom

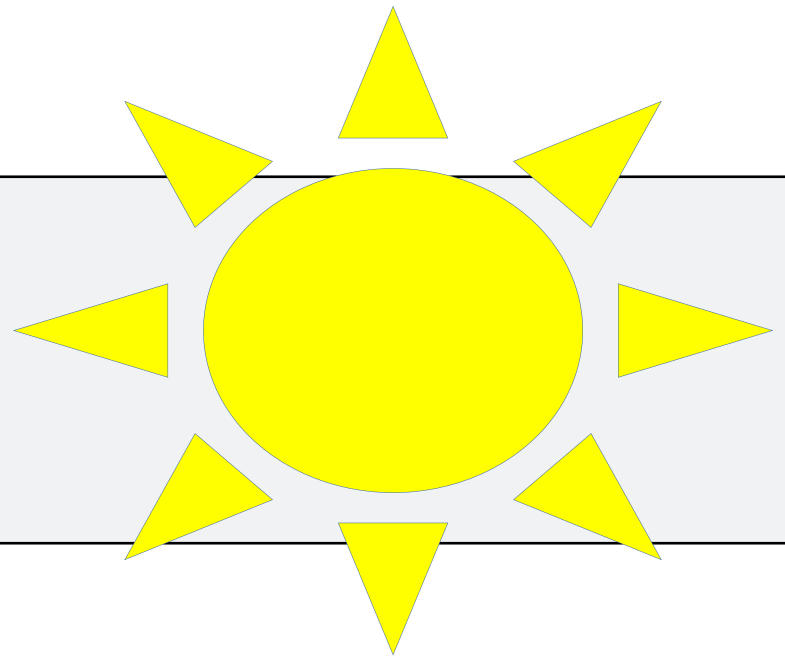
Broom includes three functions which work for most types of models (and can be extended to more):

1. **tidy()** - returns model coefficients, stats
2. **glance()** - returns model diagnostics
3. **augment()** - returns predictions, residuals, and other raw values

tidy()

Returns useful **model output** as a data frame (can be handy for prepping for publication)

```
tidy(my_model)
```



```
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    3.45      0.982      3.51 0.000765
2 log_gdp       -0.362     0.103     -3.50 0.000788
```

glance()

Returns common **model diagnostics** as a data frame

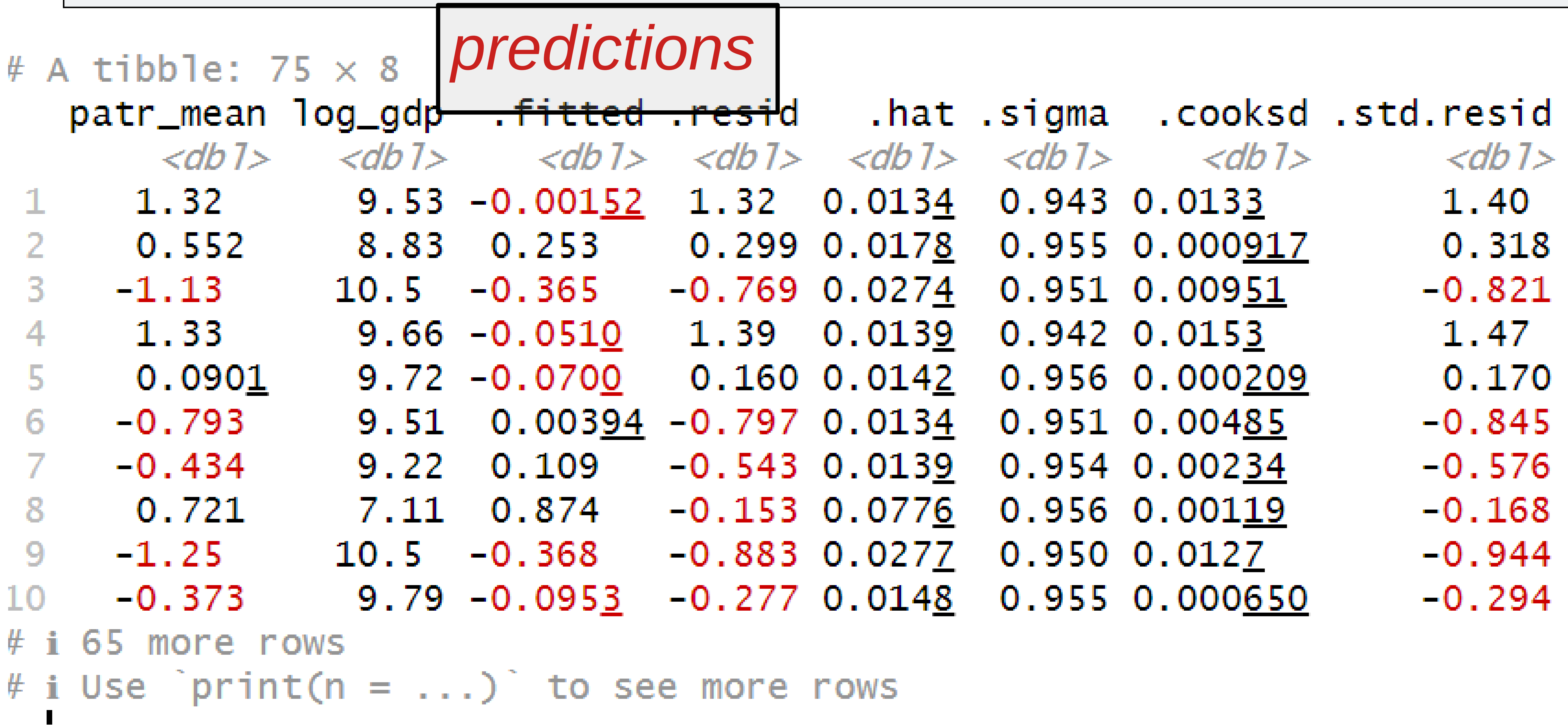
```
glance(my_model)
```

```
# A tibble: 1 × 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.144         0.132 0.949      12.3 0.000788     1  -102.  209.  216.
```

augment()

Returns data frame of **model output related to original data points**

```
augment(my_model)
```



predictions

```
# A tibble: 75 × 8
   patr_mean log_gdp  .fitted .resid   .hat .sigma  .cooksd .std.resid
       <dbl>   <dbl>    <dbl>  <dbl>  <dbl>  <dbl>    <dbl>      <dbl>
 1    1.32      9.53 -0.00152  1.32  0.0134  0.943 0.0133        1.40 
 2    0.552     8.83  0.253    0.299 0.0178  0.955 0.000917      0.318
 3   -1.13     10.5  -0.365   -0.769 0.0274  0.951 0.00951      -0.821
 4    1.33      9.66 -0.0510   1.39  0.0139  0.942 0.0153        1.47 
 5    0.0901    9.72 -0.0700   0.160 0.0142  0.956 0.000209      0.170
 6   -0.793     9.51  0.00394 -0.797 0.0134  0.951 0.00485      -0.845
 7   -0.434     9.22  0.109   -0.543 0.0139  0.954 0.00234      -0.576
 8    0.721     7.11  0.874   -0.153 0.0776  0.956 0.00119      -0.168
 9   -1.25     10.5  -0.368   -0.883 0.0277  0.950 0.0127       -0.944
10   -0.373     9.79 -0.0953  -0.277 0.0148  0.955 0.000650     -0.294
# ℹ 65 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Your turn

- Using moral_issues, predict prop_lib (as Y) with prop_cons (X) across all issues and years. Use lm() to model this. Then use tidy, glance, and plot functions to:

  1. Extract the estimated regression coefficients.

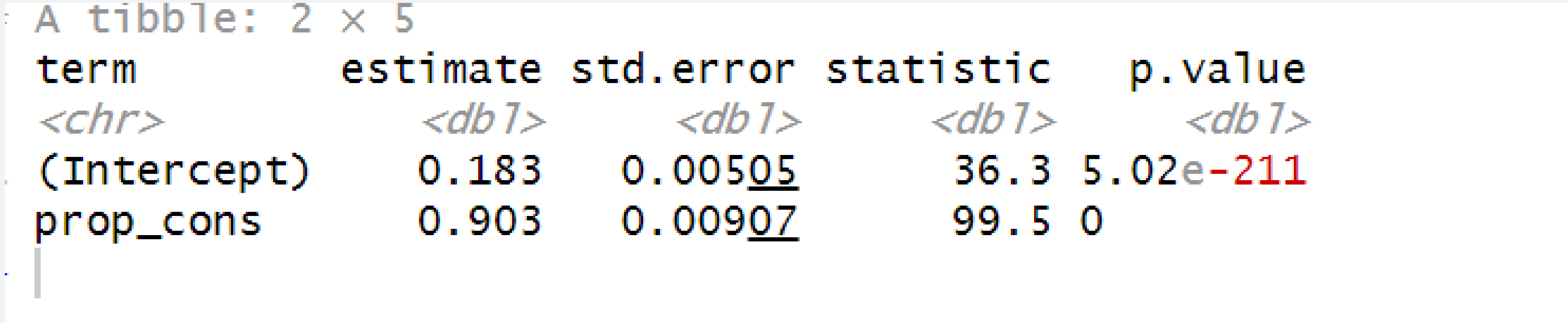
  2. Look at the estimates of model fit.

  3. Make diagnostics plots.

- Extra: Use select to extract only adj.r.squared and BIC from the model fit data frame.

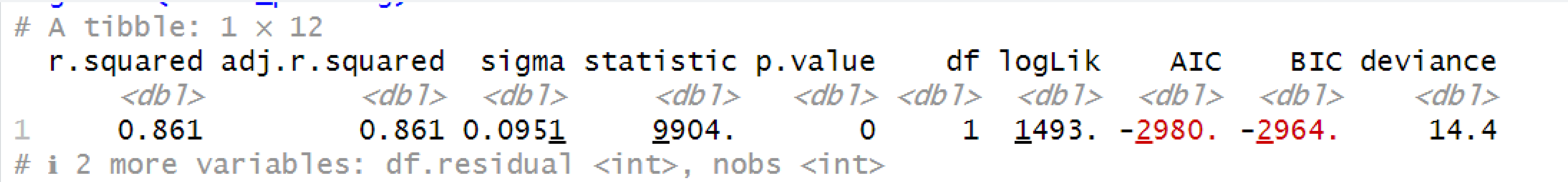
```
issue_pairing <-  lm(prop_lib ~ prop_cons,
data=moral_issues )
```

```
tidy(issue_pairing)
```

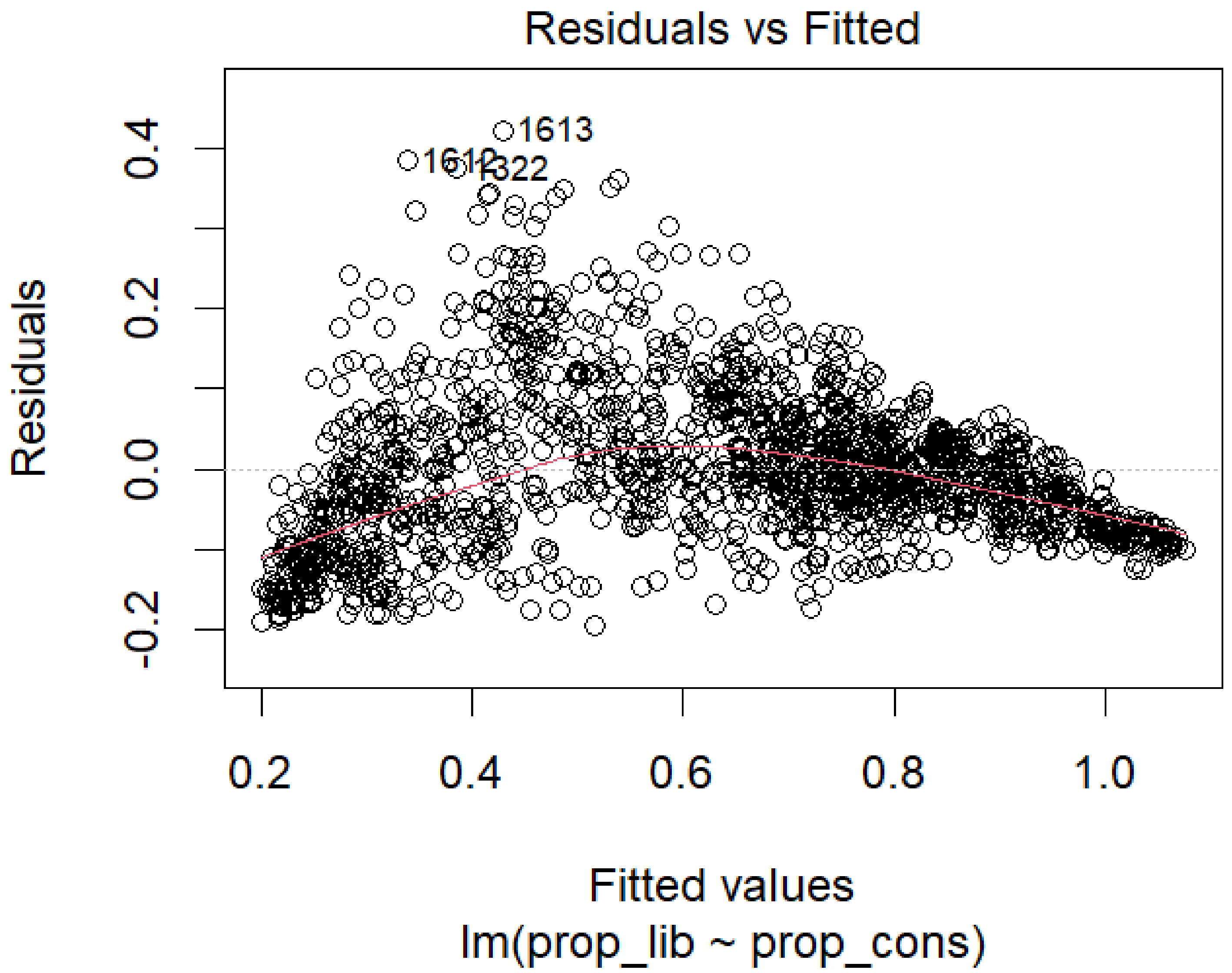


```
 A tibble: 2 × 5
 term        estimate std.error statistic   p.value
 <chr>          <dbl>     <dbl>     <dbl>     <dbl>
 (Intercept)    0.183   0.00505      36.3 5.02e-211
 prop_cons      0.903   0.00907      99.5 0
```

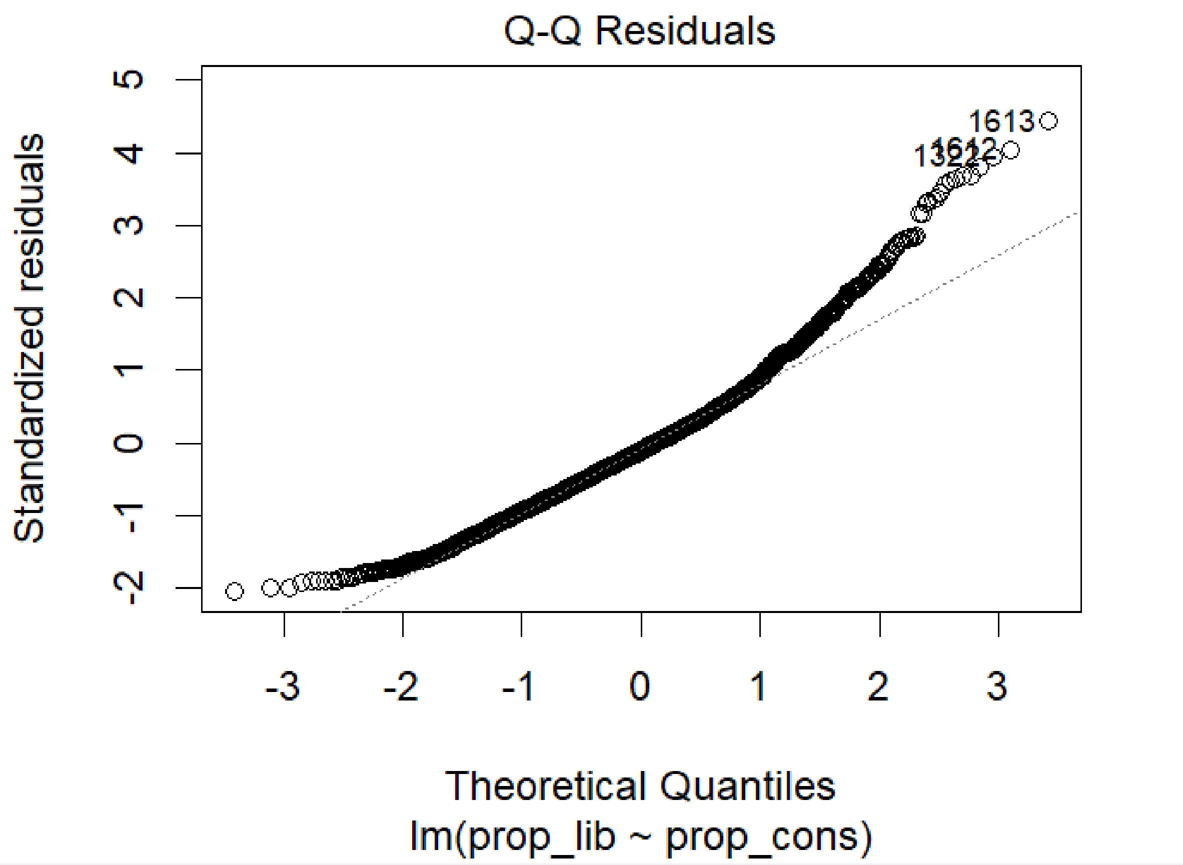
```
glance(mod_gdp)
```



```
# A tibble: 1 × 12
  r.squared adj.r.squared  sigma statistic p.value    df logLik    AIC    BIC deviance
      <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>    <dbl>
1     0.861         0.861 0.0951     9904.       0     1  1493. -2980. -2964.     14.4
# ℹ 2 more variables: df.residual <int>, nobs <int>
```



plot(issue_pairing
, which = 1)



plot(issue_pairing,
which = 2)

```
glance(issue_pairing) %>%
     select(adj.r.squared, BIC)

# A tibble: 1 × 2
  adj.r.squared    BIC
          <dbl>  <dbl>
1         0.861 -2964.
```

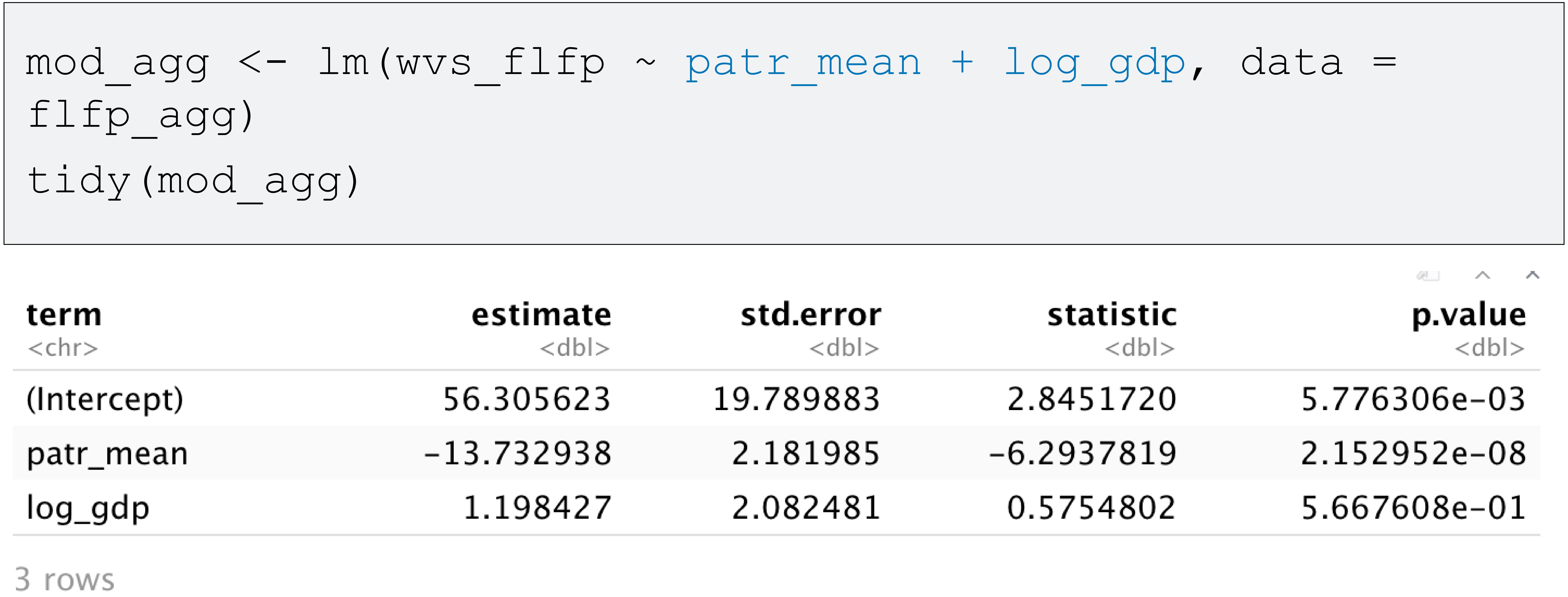
Multivariate regression

To fit multiple predictors,
Simply add multiple variables to the formula
with a + sign entered in the lm() function:

```
wvs_flfp ~ patr_mean + log_gdp
```

Your turn

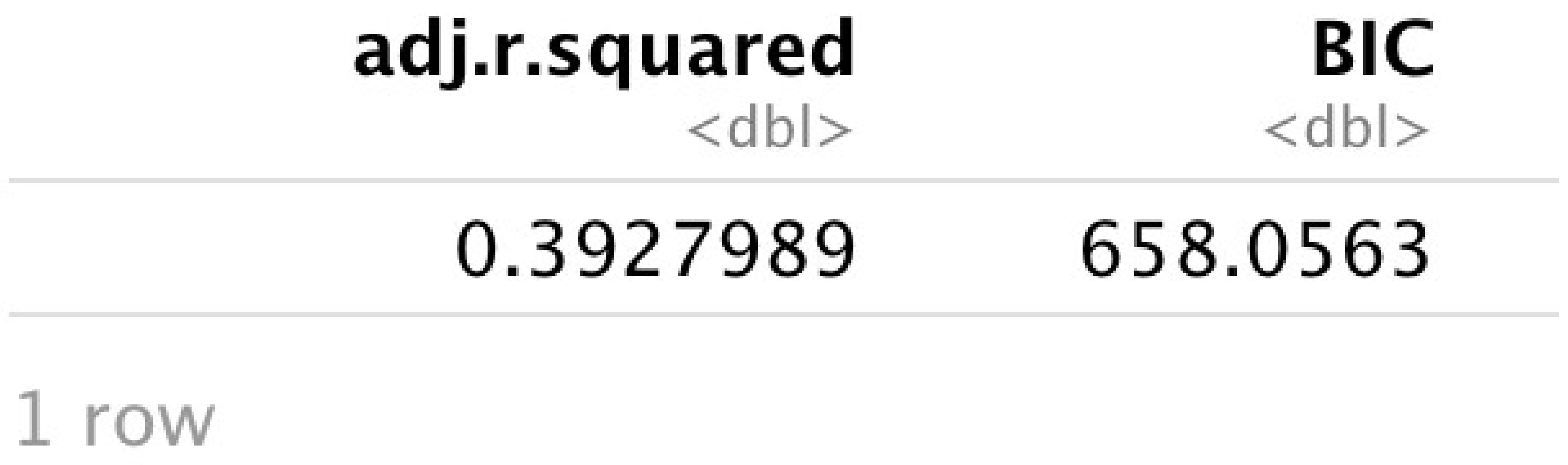
- Predict wvs_flfp using both patr_mean and log_gdp.
- Call up the coefficients for the model using tidy() and the adjusted R squared and BIC using glance()



```r
mod_agg <- lm(wvs_flfp ~ patr_mean + log_gdp, data =
flfp_agg)
tidy(mod_agg)
```

```
term            estimate    std.error    statistic      p.value
<chr>              <dbl>        <dbl>        <dbl>        <dbl>
(Intercept)    56.305623    19.789883    2.8451720 5.776306e-03
patr_mean     -13.732938     2.181985   -6.2937819 2.152952e-08
log_gdp         1.198427     2.082481    0.5754802 5.667608e-01
3 rows
```

```r
glance(mod_agg) %>%
   select(adj.r.squared, BIC)
```



adj.r.squared <dbl>	BIC <dbl>
0.3927989	658.0563

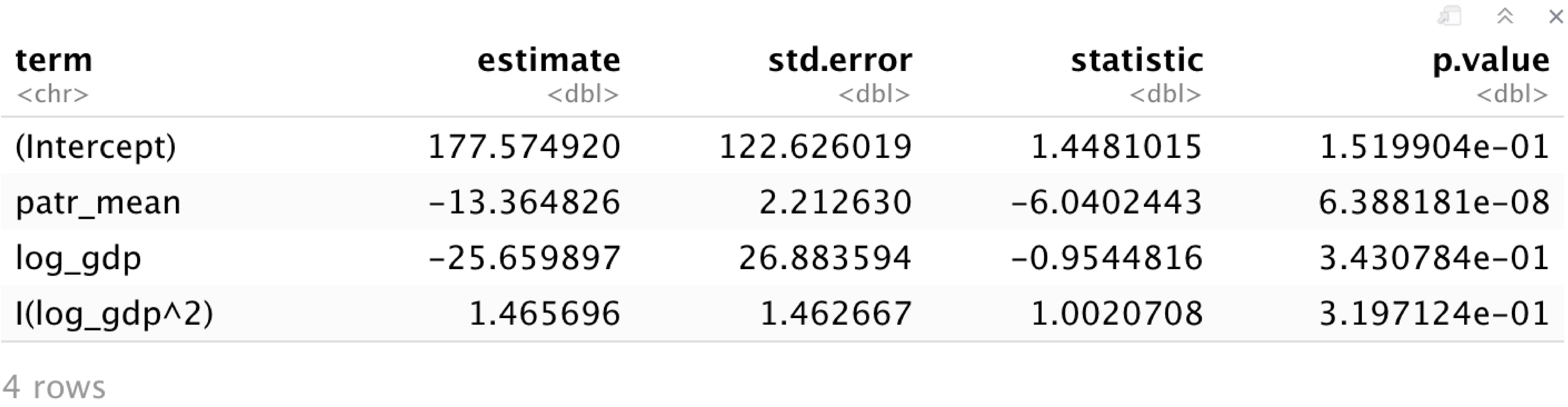
1 row

Quadratic terms

Quadratic regression

```
mod_agg_quadratic <- lm(wvs_flfp ~ patr_mean + log_gdp +
I(log_gdp^2), data = flfp_agg)
tidy(mod_agg_quadratic)
```

Keep both!

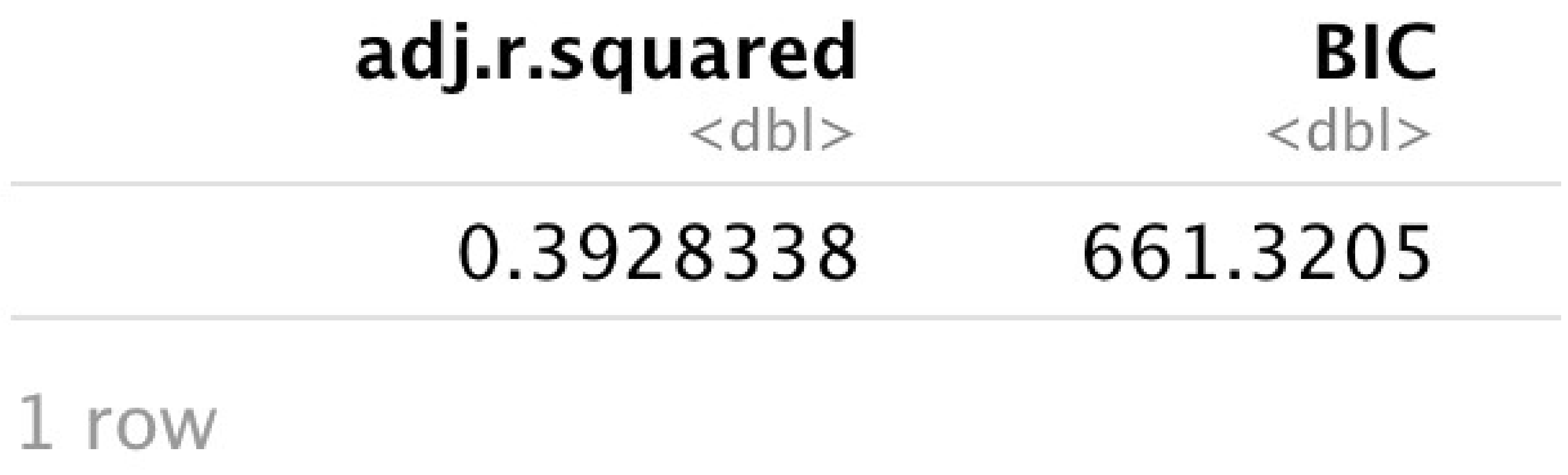


term	estimate	std.error	statistic	p.value
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
(Intercept)	177.574920	122.626019	1.4481015	1.519904e-01
patr_mean	-13.364826	2.212630	-6.0402443	6.388181e-08
log_gdp	-25.659897	26.883594	-0.9544816	3.430784e-01
I(log_gdp^2)	1.465696	1.462667	1.0020708	3.197124e-01

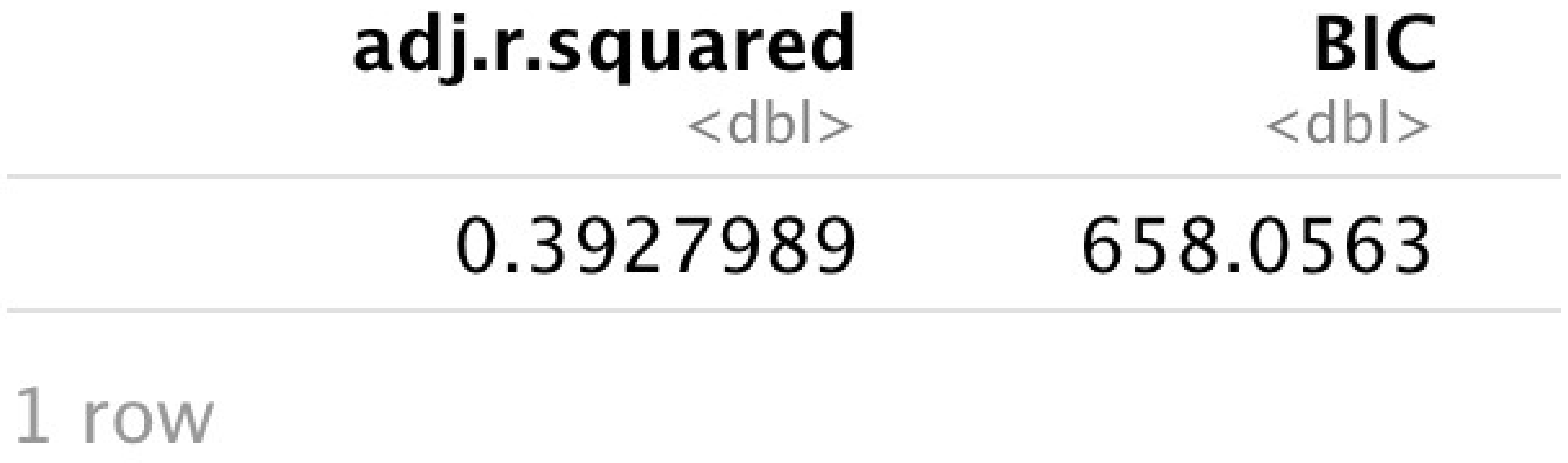
4 rows

```
glance(mod_agg) %>%
  select(adj.r.squared, BIC)
```

```
glance(mod_agg_quadratic) %>%
  select(adj.r.squared, BIC)
```



```
adj.r.squared        BIC
        <dbl>      <dbl>
    0.3928338   661.3205
1 row
```

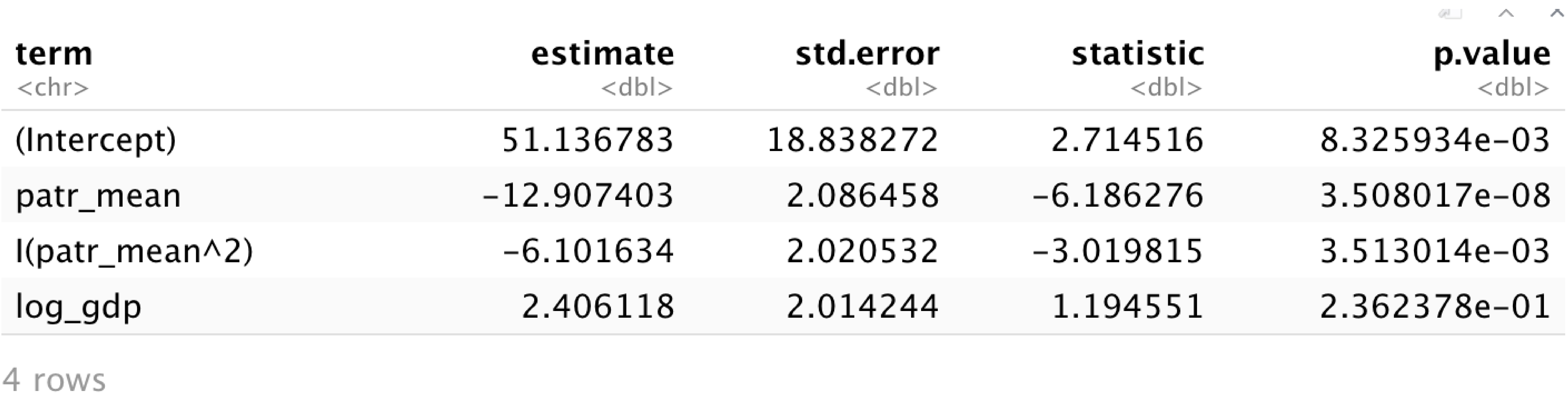


```
adj.r.squared        BIC
        <dbl>      <dbl>
    0.3927989   658.0563
1 row
```

Your turn

- Model wvs_flfp against patr_mean using a quadratic term in the regression. Keep log_gdp as a control variable with a linear effect. Does the quadratic term of patriarchal values improve the model fit?

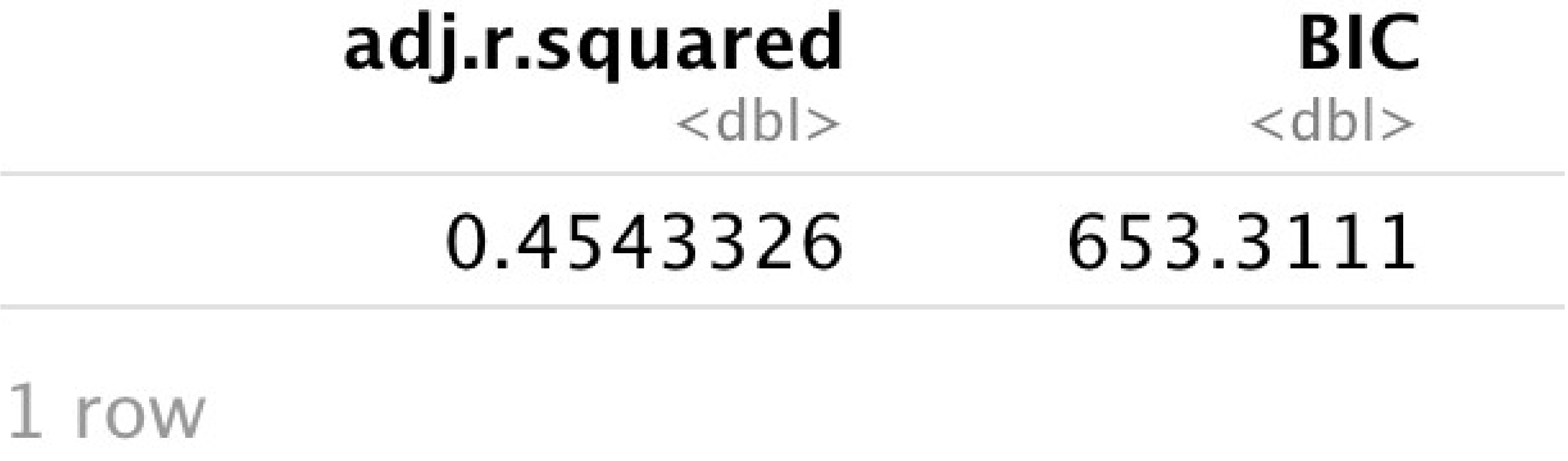
```r
mod_agg_patr2 <- lm(wvs_flfp ~ patr_mean + I(patr_mean^2) + log_gdp,
             data = flfp)
tidy(mod_agg_patr2)
```



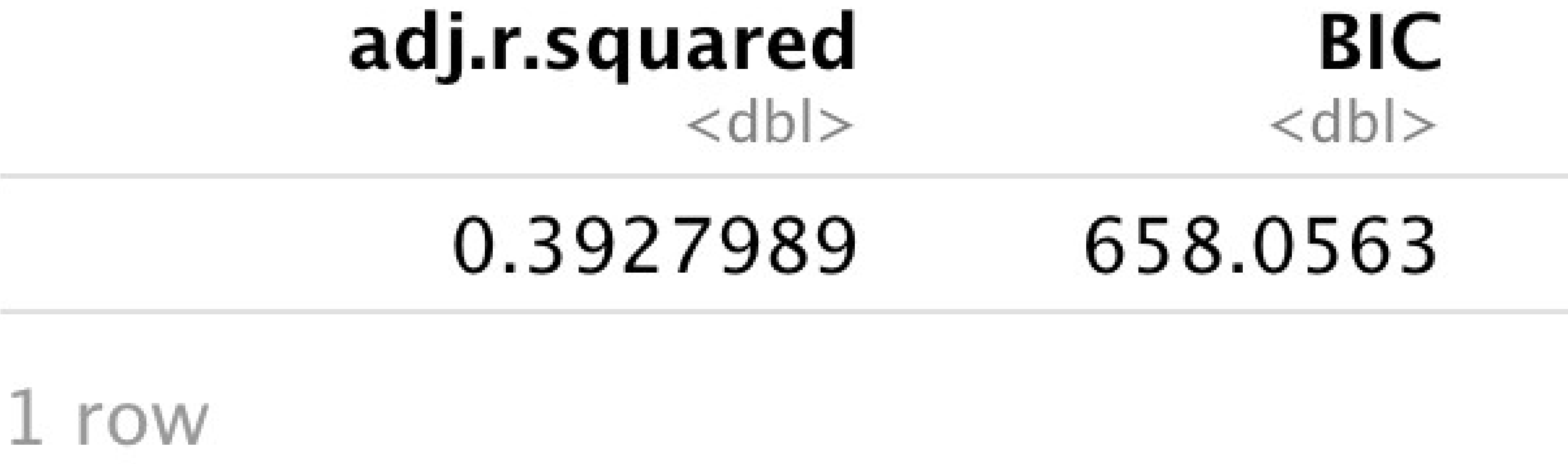
```
term                   estimate   std.error   statistic        p.value
<chr>                     <dbl>       <dbl>       <dbl>          <dbl>
(Intercept)           51.136783   18.838272    2.714516   8.325934e-03
patr_mean            -12.907403    2.086458   -6.186276   3.508017e-08
I(patr_mean^2)        -6.101634    2.020532   -3.019815   3.513014e-03
log_gdp                2.406118    2.014244    1.194551   2.362378e-01
4 rows
```

```
glance(mod_agg_patr2) %>%
  select(adj.r.squared, BIC)
```

```
glance(mod_agg_quadratic) %>
%
  select(adj.r.squared, BIC)
```



```
adj.r.squared         BIC
        <dbl>       <dbl>
    0.4543326    653.3111
1 row
```

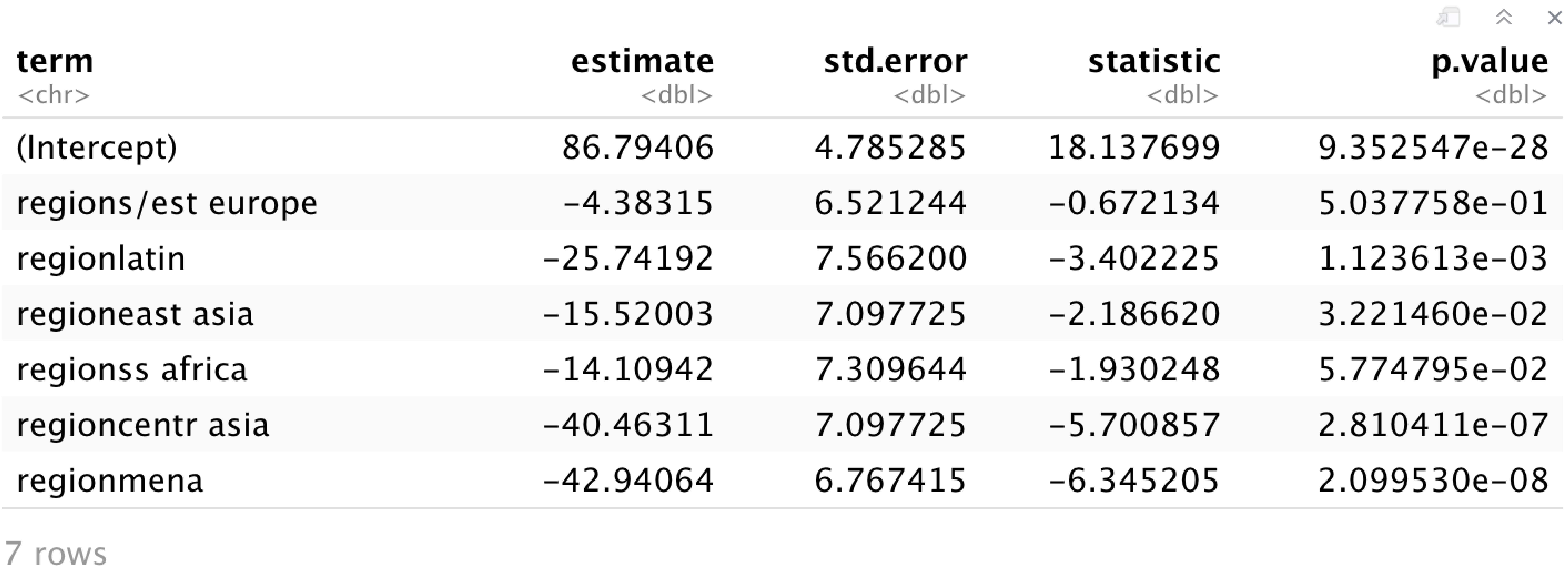


```
adj.r.squared         BIC
        <dbl>       <dbl>
    0.3927989    658.0563
1 row
```

Categorical predictors

Regional differences in FLFP

```r
mod_reg <- lm(wvs_flfp ~ region, data = flfp)
tidy(mod_reg)
```



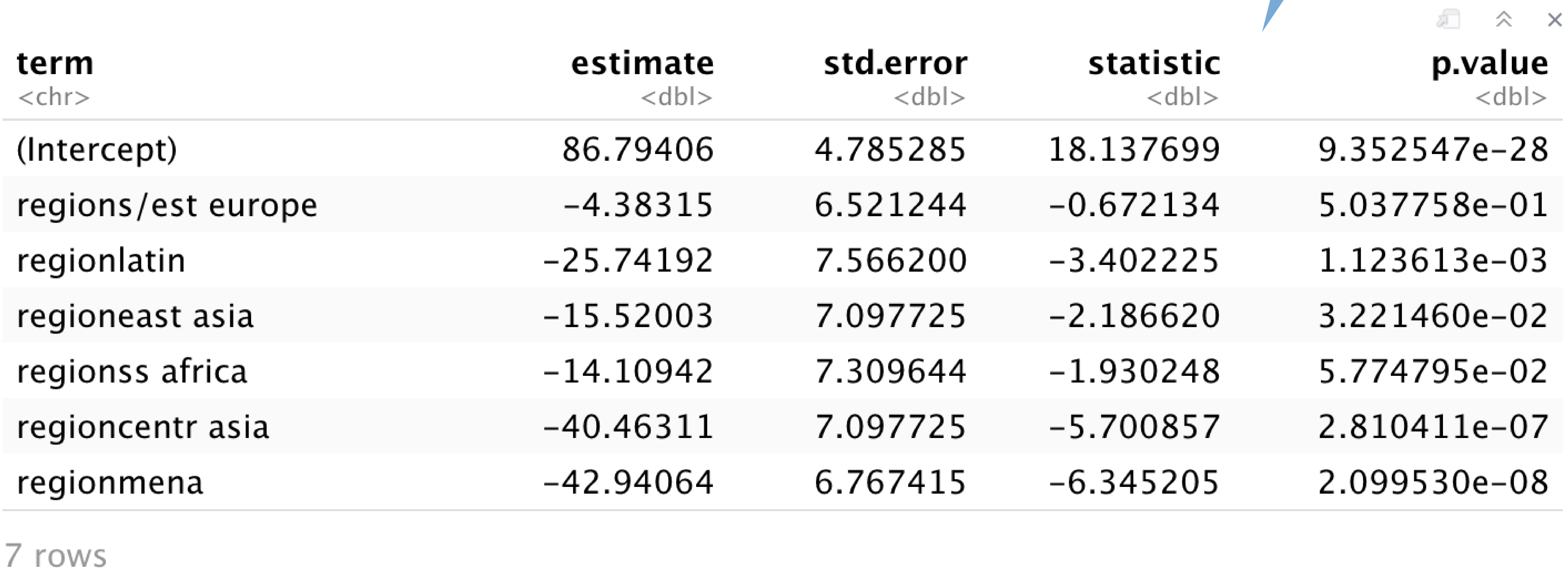
term	estimate	std.error	statistic	p.value
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
(Intercept)	86.79406	4.785285	18.137699	9.352547e-28
regions/est europe	-4.38315	6.521244	-0.672134	5.037758e-01
regionlatin	-25.74192	7.566200	-3.402225	1.123613e-03
regioneast asia	-15.52003	7.097725	-2.186620	3.221460e-02
regionss africa	-14.10942	7.309644	-1.930248	5.774795e-02
regioncentr asia	-40.46311	7.097725	-5.700857	2.810411e-07
regionmena	-42.94064	6.767415	-6.345205	2.099530e-08

7 rows

Regional differences in FLFP

```
mod_reg <- lm(wvs_flfp ~ region, data = flfp)
tidy(mod_reg)
```

Where is the West?



term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	86.79406	4.785285	18.137699	9.352547e-28
regions/est europe	-4.38315	6.521244	-0.672134	5.037758e-01
regionlatin	-25.74192	7.566200	-3.402225	1.123613e-03
regioneast asia	-15.52003	7.097725	-2.186620	3.221460e-02
regionss africa	-14.10942	7.309644	-1.930248	5.774795e-02
regioncentr asia	-40.46311	7.097725	-5.700857	2.810411e-07
regionmena	-42.94064	6.767415	-6.345205	2.099530e-08

7 rows

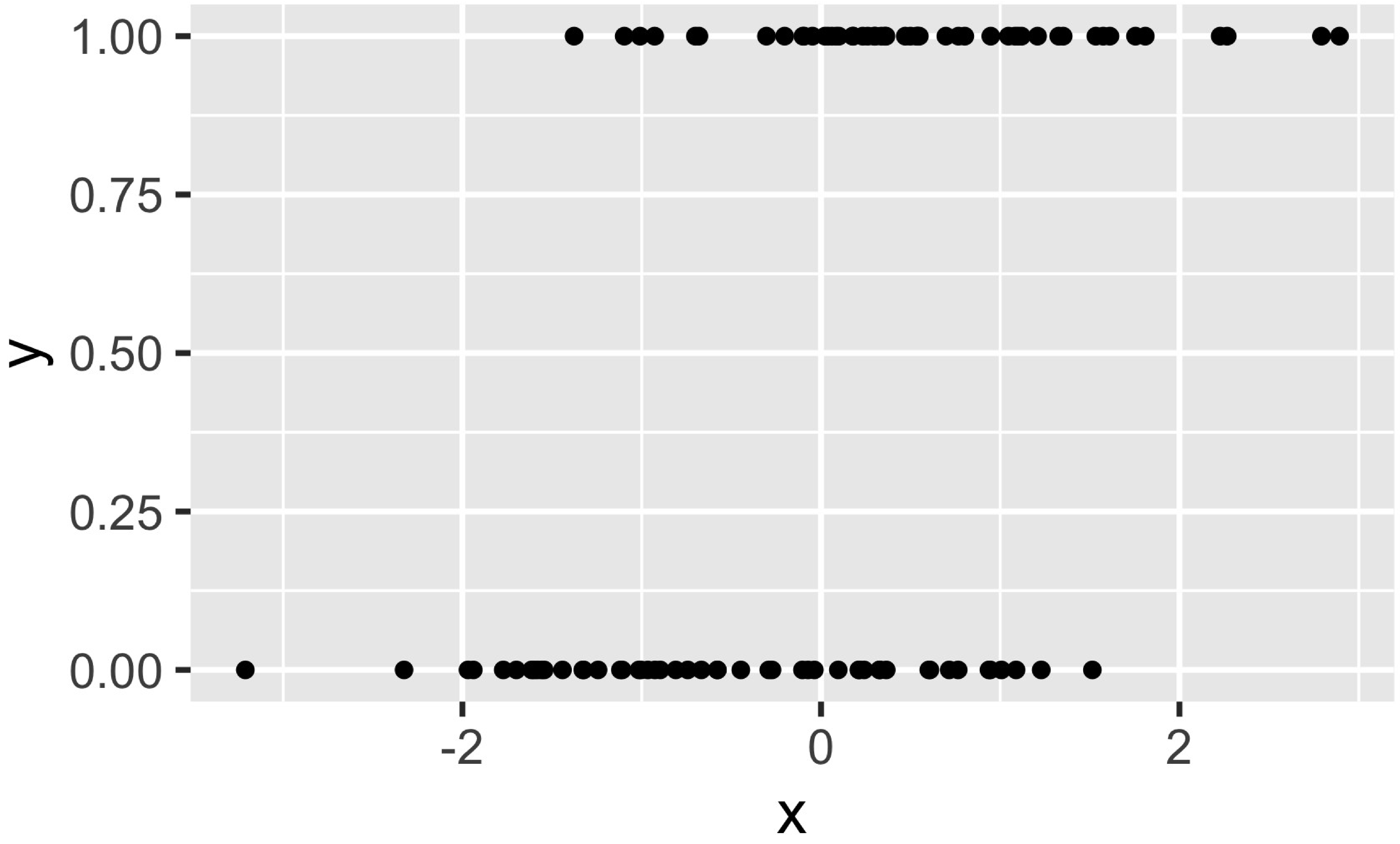
Interaction terms

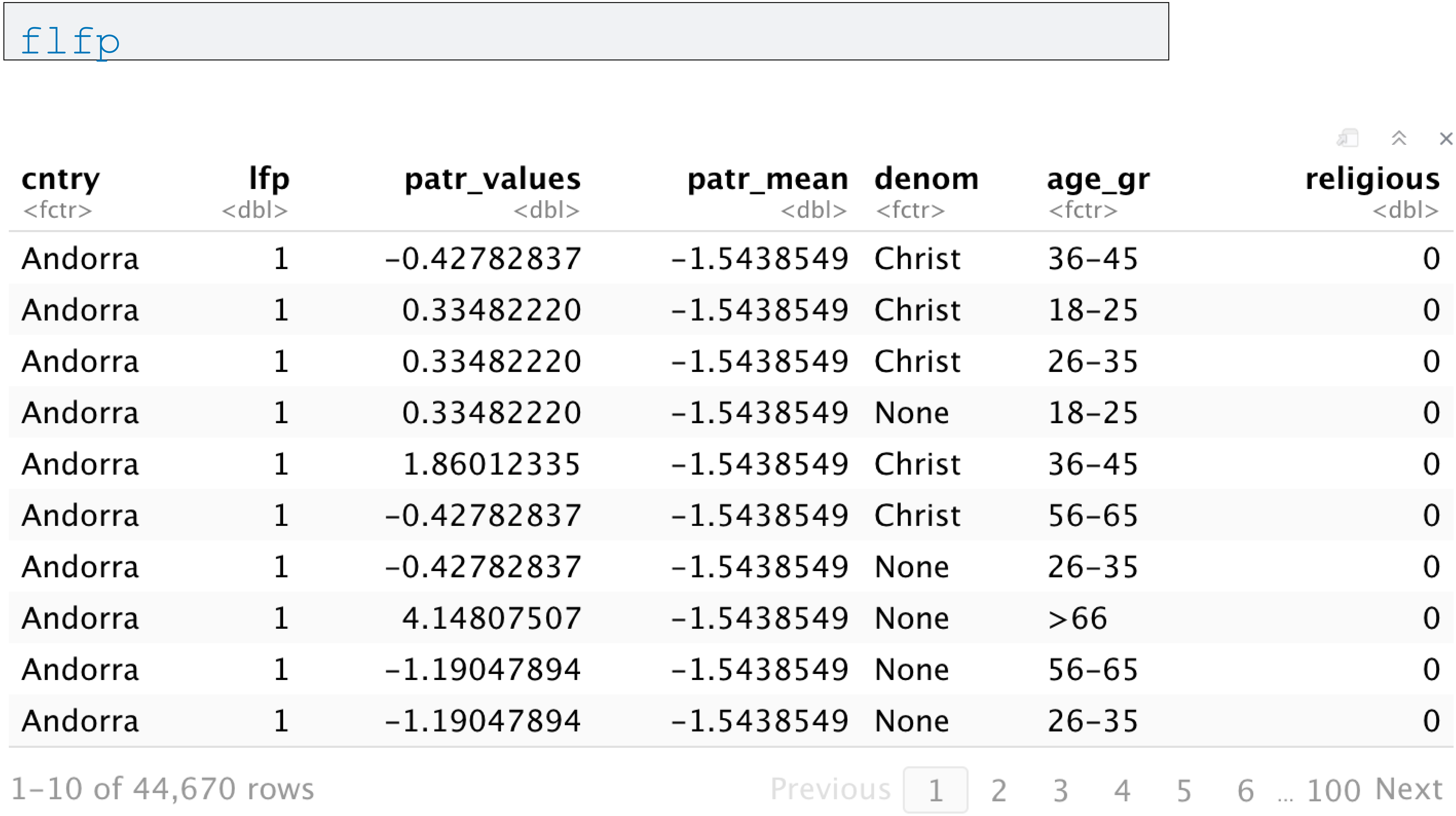
Interaction model

```
mod_int <- lm(wvs_flfp ~ patr_mean + muslim + patr_mean:muslim,
data = flfp_agg)

tidy(mod_int)


  term                   estimate std.error statistic  p.value
  <chr>                     <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)               73.6       2.65    27.7   8.47e-40
2 patr_mean                 -6.14      2.96    -2.07  4.18e- 2
3 muslimTRUE                 6.72     10.1      0.665 5.09e- 1
4 patr_mean:muslimTRUE   -23.2        8.43    -2.75  7.61e- 3
```

glm()

Binary outcome

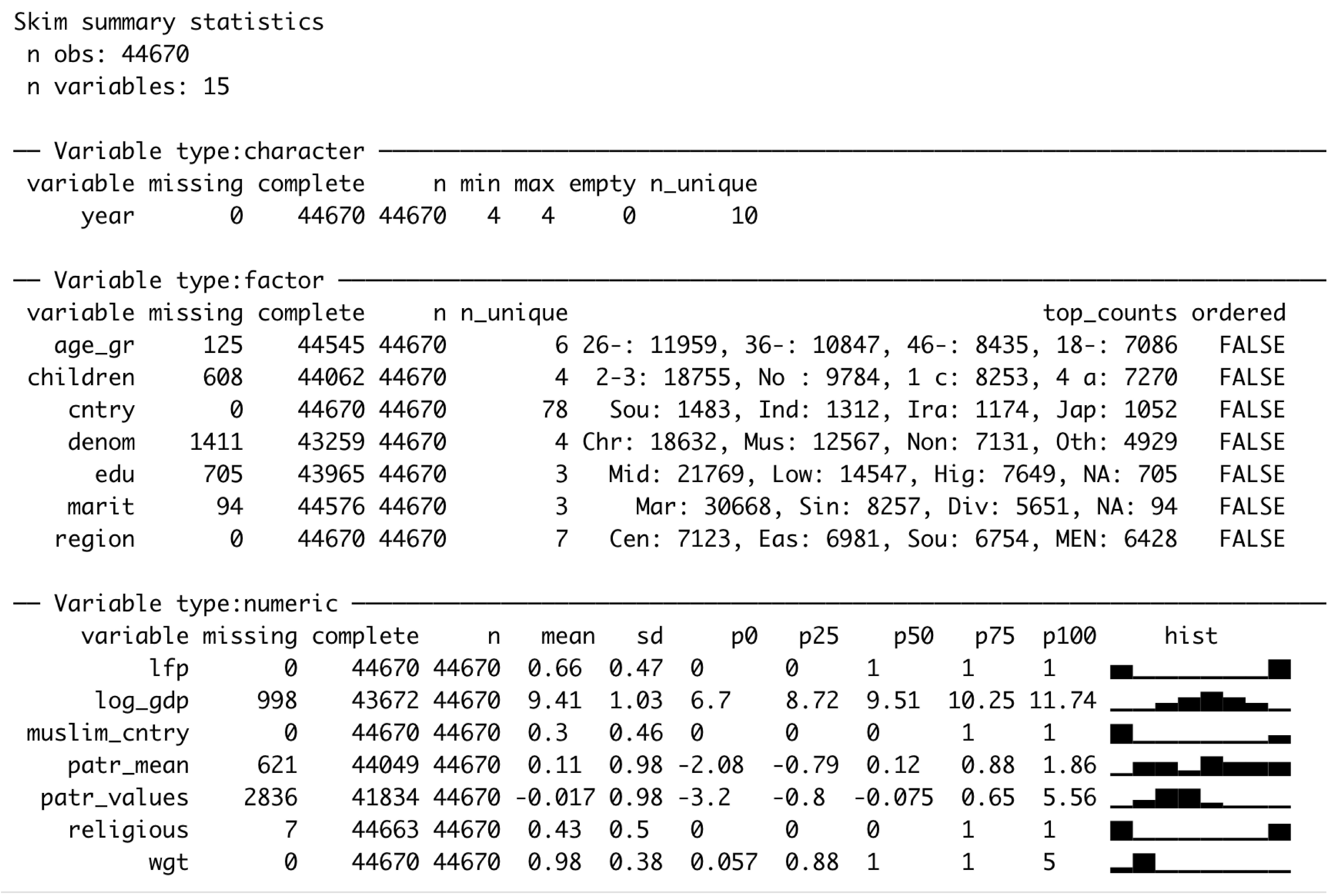




```
flfp
```

cntry <fctr>	lfp <dbl>	patr_values <dbl>	patr_mean <dbl>	denom <fctr>	age_gr <fctr>	religious <dbl>
Andorra	1	-0.42782837	-1.5438549	Christ	36-45	0
Andorra	1	0.33482220	-1.5438549	Christ	18-25	0
Andorra	1	0.33482220	-1.5438549	Christ	26-35	0
Andorra	1	0.33482220	-1.5438549	None	18-25	0
Andorra	1	1.86012335	-1.5438549	Christ	36-45	0
Andorra	1	-0.42782837	-1.5438549	Christ	56-65	0
Andorra	1	-0.42782837	-1.5438549	None	26-35	0
Andorra	1	4.14807507	-1.5438549	None	>66	0
Andorra	1	-1.19047894	-1.5438549	None	56-65	0
Andorra	1	-1.19047894	-1.5438549	None	26-35	0

1-10 of 44,670 rows



```
Skim summary statistics
 n obs: 44670
 n variables: 15

── Variable type:character ───────────────────────────────────────────────────────────────────────
 variable missing complete     n min max empty n_unique
     year       0    44670 44670   4   4     0       10

── Variable type:factor ──────────────────────────────────────────────────────────────────────────
 variable missing complete     n n_unique                                   top_counts ordered
   age_gr     125    44545 44670        6 26-: 11959, 36-: 10847, 46-: 8435, 18-: 7086   FALSE
 children     608    44062 44670        4  2-3: 18755, No : 9784, 1 c: 8253, 4 a: 7270   FALSE
    cntry       0    44670 44670       78   Sou: 1483, Ind: 1312, Ira: 1174, Jap: 1052   FALSE
    denom    1411    43259 44670        4 Chr: 18632, Mus: 12567, Non: 7131, Oth: 4929   FALSE
      edu     705    43965 44670        3   Mid: 21769, Low: 14547, Hig: 7649, NA: 705   FALSE
    marit      94    44576 44670        3     Mar: 30668, Sin: 8257, Div: 5651, NA: 94   FALSE
   region       0    44670 44670        7   Cen: 7123, Eas: 6981, Sou: 6754, MEN: 6428   FALSE

── Variable type:numeric ─────────────────────────────────────────────────────────────────────────
     variable missing complete     n   mean   sd     p0   p25    p50   p75  p100     hist
          lfp       0    44670 44670  0.66  0.47  0      0     1      1     1    ▅▁▁▁▁▁▁▇
      log_gdp     998    43672 44670  9.41  1.03  6.7    8.72  9.51  10.25 11.74 ▁▁▃▅▇▅▃▁
 muslim_cntry       0    44670 44670  0.3   0.46  0      0     0      1     1    ▇▁▁▁▁▁▁▃
    patr_mean     621    44049 44670  0.11  0.98 -2.08  -0.79  0.12   0.88  1.86 ▁▅▅▂▇▅▅▅
  patr_values    2836    41834 44670 -0.017 0.98 -3.2   -0.8  -0.075  0.65  5.56 ▁▃▇▇▂▁▁▁
    religious       7    44663 44670  0.43  0.5   0      0     0      1     1    ▇▁▁▁▁▁▁▆
          wgt       0    44670 44670  0.98  0.38  0.057  0.88  1      1     5    ▂▇▁▁▁▁▁▁
```

glm()

Fits a generalised linear model to data

```
model_val <- glm(lfp ~ patr_values + cntry,
                 family = binomial(link = "logit"),
                 data = flfp_ind)
```

Modelled distribution

Link function

Coefficients

```
glm(lfp ~ patr_values + cntry,
    family = binomial(link = "logit"),
    data = flfp_ind)

tidy(mod_val) %>% filter(!str_detect(term, "cntry"))

## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   -0.333    0.110      -3.02 2.57e-  3
## 2 patr_values   -0.289    0.0119    -24.2  1.13e-129
```

Avg. Change in log odds.

Odds ratios

```
tidy(mod_val, exponentiate = TRUE) %>%
  filter(!str_detect(term, "cntry"))

## # A tibble: 2 x 5
##   term          estimate std.error statistic   p.value
##   <chr>            <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)      0.717    0.110      -3.02 2.57e-  3
## 2 patr_values      0.749    0.0119    -24.2  1.13e-129
```

Odds ratio compared to referens category.

Your turn

- Read the individual level data.
- Add the variable patr_mean to the model we looked at last and remove cntry.
- Which values have stronger effect: the individual or the country mean?

```
mod_val_cntr <- glm(lfp ~ patr_values + patr_mean ,
                    family = binomial(link = "logit"),
                    data = flfp)

tidy(mod_val_cntr, exponentiate = TRUE)


  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)    2.31     0.0117      71.3 0
2 patr_values    0.776    0.0110     -23.0 6.91e-117
3 patr_mean      0.488    0.0119     -60.1 0
```

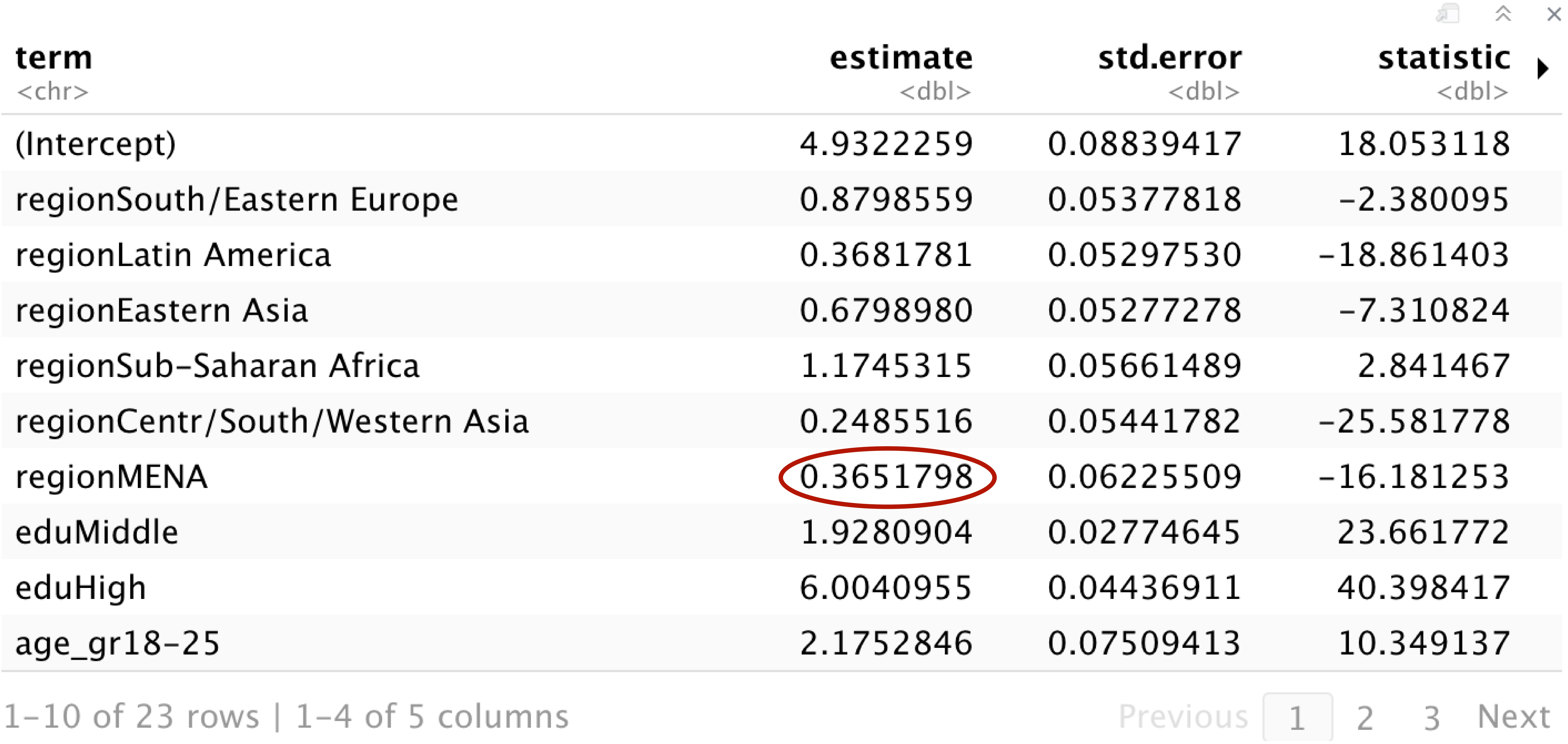
Categorical variables

```
mod_val_edu <- glm(lfp ~ patr_values + edu + cntry,
                family = binomial(link = "logit"),
                data = flfp

tidy(mod_val_edu, exponentiate = TRUE) %>%
  filter(!str_detect(term, "cntry"))

## # A tibble: 4 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    0.452    0.116      -6.84 7.80e- 12
## 2 patr_values    0.827    0.0125    -15.2  5.78e- 52
## 3 eduMiddle      2.49     0.0300     30.4  6.54e-203
## 4 eduHigh        7.58     0.0458     44.2  0.
```

Your turn

- Estimate lfp against country region (s) on the individual level. Add edu, age_gr, marit, children, religious, and denom as control variables, but do not include cntry. Compare the odds of a female from a MENA country to be employed to those of a female from the West?

```r
mod_reg <- glm(lfp ~ region + edu + age_gr + marit +
                  children + religious + denom,
               family = binomial(link = "logit"),
               data = flfp)

tidy(mod_reg, exponentiate = TRUE)
```



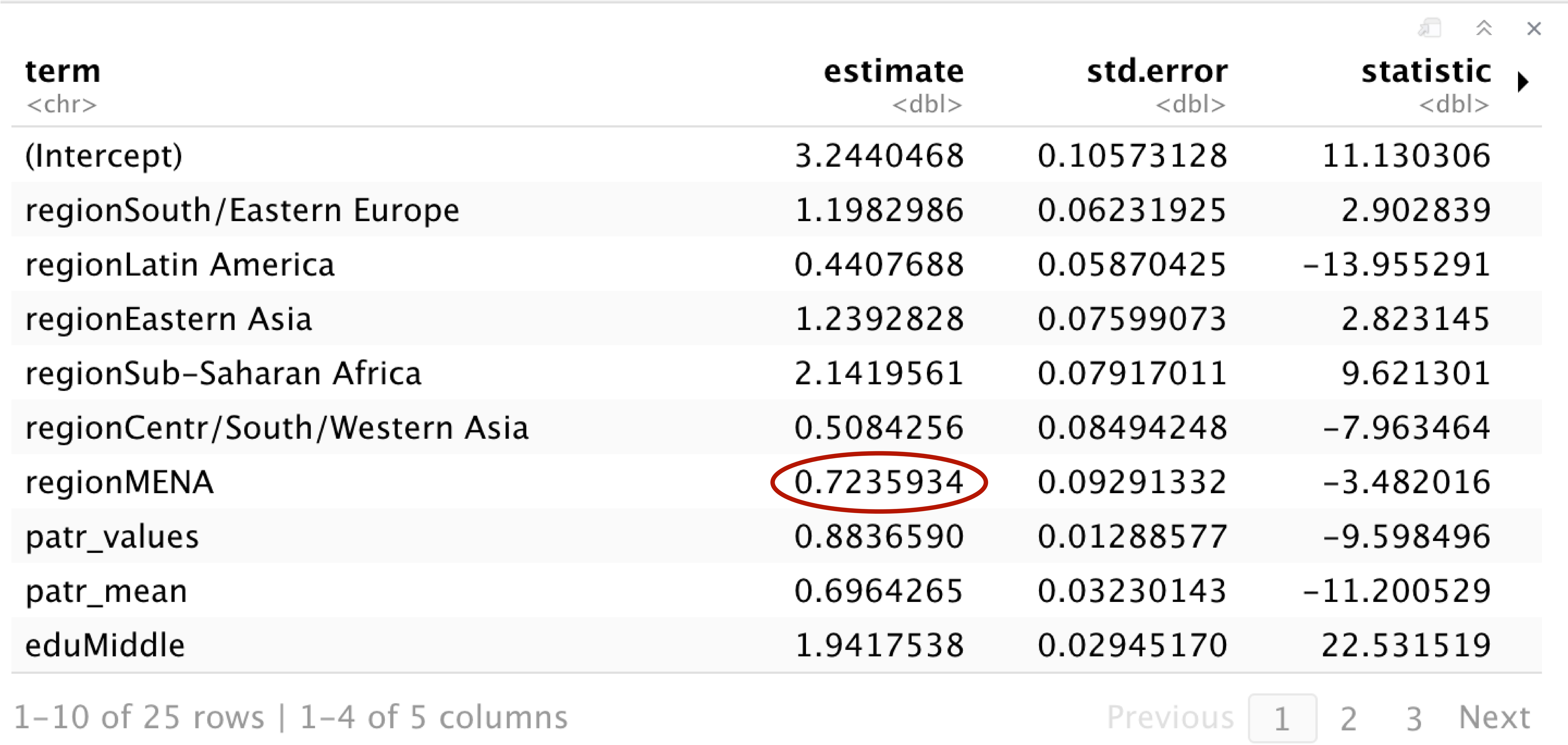
term	estimate	std.error	statistic
<chr>	<dbl>	<dbl>	<dbl>
(Intercept)	4.9322259	0.08839417	18.053118
regionSouth/Eastern Europe	0.8798559	0.05377818	-2.380095
regionLatin America	0.3681781	0.05297530	-18.861403
regionEastern Asia	0.6798980	0.05277278	-7.310824
regionSub-Saharan Africa	1.1745315	0.05661489	2.841467
regionCentr/South/Western Asia	0.2485516	0.05441782	-25.581778
regionMENA	0.3651798	0.06225509	-16.181253
eduMiddle	1.9280904	0.02774645	23.661772
eduHigh	6.0040955	0.04436911	40.398417
age_gr18-25	2.1752846	0.07509413	10.349137

1-10 of 23 rows | 1-4 of 5 columns

Your turn

- Add patr_values and patr_mean to the model we just fit. How does the regional differences change?

```
mod_reg_val <- glm(lfp ~ region + patr_values + patr_mean +
                      edu + age_gr + marit + children +
                     religious + denom,
                   family = binomial(link = "logit"),
                   data = flfp)

tidy(mod_reg_val, exponentiate = TRUE)
```



term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>
(Intercept)	3.2440468	0.10573128	11.130306
regionSouth/Eastern Europe	1.1982986	0.06231925	2.902839
regionLatin America	0.4407688	0.05870425	-13.955291
regionEastern Asia	1.2392828	0.07599073	2.823145
regionSub-Saharan Africa	2.1419561	0.07917011	9.621301
regionCentr/South/Western Asia	0.5084256	0.08494248	-7.963464
regionMENA	0.7235934	0.09291332	-3.482016
patr_values	0.8836590	0.01288577	-9.598496
patr_mean	0.6964265	0.03230143	-11.200529
eduMiddle	1.9417538	0.02945170	22.531519

1-10 of 25 rows | 1-4 of 5 columns

Generalised linear models

- Use different link functions to connect variety of outcomes to the linear predictor.

```
glm(y ~ x, family = poisson(link = "log"))
```

- Check the full list with ?family.