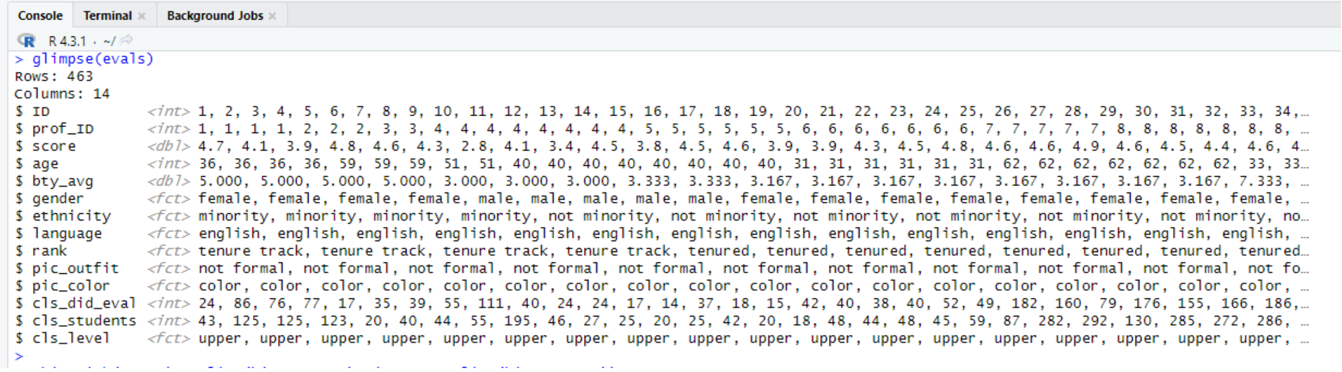
(LC5.1) Conduct a new exploratory data analysis with the same outcome variable y being score but with age as the new explanatory variable x. Remember, this involves three things:

a. Looking at the raw data values.
b. Computing summary statistics.
c. Creating data visualizations.

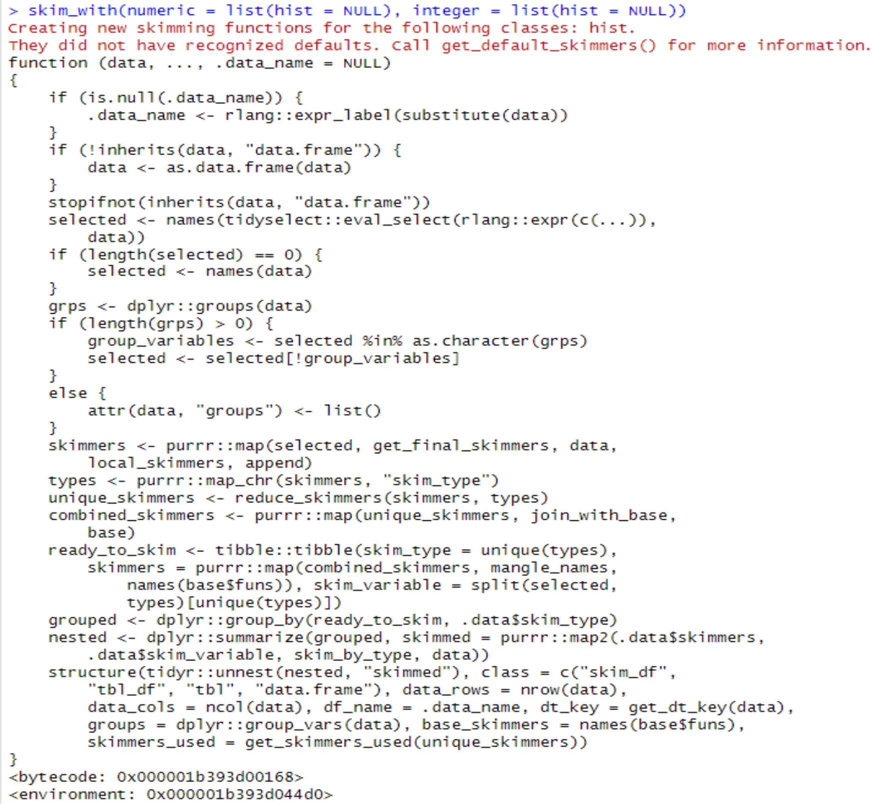
What can you say about the relationship between age and teaching scores based on this exploration?

glimpse(evals)



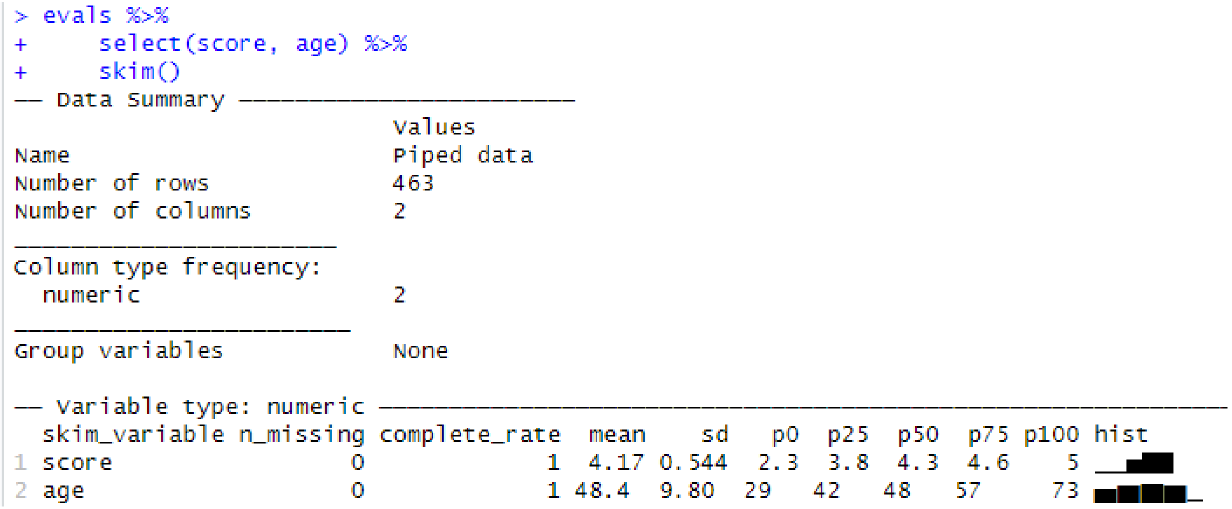
```
> glimpse(evals)
Rows: 463
Columns: 14
$ ID           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,…
$ prof_ID      <int> 1, 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, …
$ score        <dbl> 4.7, 4.1, 3.9, 4.8, 4.6, 4.3, 2.8, 4.1, 3.4, 4.5, 3.8, 4.5, 4.6, 3.9, 3.9, 4.3, 4.5, 4.8, 4.6, 4.6, 4.9, 4.6, 4.5, 4.4, 4.6, 4…
$ age          <int> 36, 36, 36, 36, 59, 59, 59, 51, 51, 40, 40, 40, 40, 40, 40, 40, 40, 31, 31, 31, 31, 31, 31, 62, 62, 62, 62, 62, 62, 62, 33, 33…
$ bty_avg      <dbl> 5.000, 5.000, 5.000, 5.000, 3.000, 3.000, 3.000, 3.333, 3.333, 3.167, 3.167, 3.167, 3.167, 3.167, 3.167, 3.167, 3.167, 7.333, …
$ gender       <fct> female, female, female, female, male, male, male, male, male, female, female, female, female, female, female, female, female, …
$ ethnicity    <fct> minority, minority, minority, minority, not minority, not minority, not minority, not minority, not minority, not minority, no…
$ language     <fct> english, english, english, english, english, english, english, english, english, english, english, english, english, english, …
$ rank         <fct> tenure track, tenure track, tenure track, tenure track, tenured, tenured, tenured, tenured, tenured, tenured, tenured, tenured…
$ pic_outfit   <fct> not formal, not formal, not formal, not formal, not formal, not formal, not formal, not formal, not formal, not formal, not fo…
$ pic_color    <fct> color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, …
$ cls_did_eval <int> 24, 86, 76, 77, 17, 35, 39, 55, 111, 40, 24, 24, 17, 14, 37, 18, 15, 42, 40, 38, 40, 52, 49, 182, 160, 79, 176, 155, 166, 186,…
$ cls_students <int> 43, 125, 125, 123, 20, 40, 44, 55, 195, 46, 27, 25, 20, 25, 42, 20, 18, 48, 44, 48, 45, 59, 87, 282, 292, 130, 285, 272, 286, …
$ cls_level    <fct> upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, …
>
```

skim_with(numeric = list(hist = NULL), integer = list(hist = NULL))



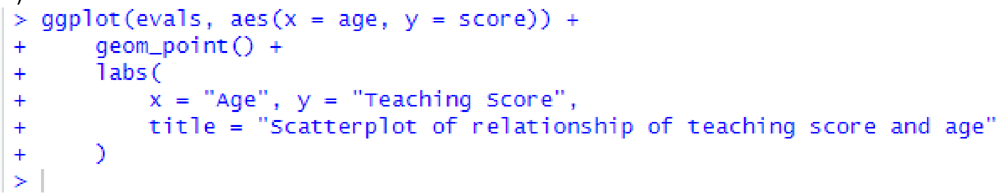
```
> skim_with(numeric = list(hist = NULL), integer = list(hist = NULL))
Creating new skimming functions for the following classes: hist.
They did not have recognized defaults. Call get_default_skimmers() for more information.
function (data, ..., .data_name = NULL)
{
    if (is.null(.data_name)) {
        .data_name <- rlang::expr_label(substitute(data))
    }
    if (!inherits(data, "data.frame")) {
        data <- as.data.frame(data)
    }
    stopifnot(inherits(data, "data.frame"))
    selected <- names(tidyselect::eval_select(rlang::expr(c(...)),
        data))
    if (length(selected) == 0) {
        selected <- names(data)
    }
    grps <- dplyr::groups(data)
    if (length(grps) > 0) {
        group_variables <- selected %in% as.character(grps)
        selected <- selected[!group_variables]
    }
    else {
        attr(data, "groups") <- list()
    }
    skimmers <- purrr::map(selected, get_final_skimmers, data,
        local_skimmers, append)
    types <- purrr::map_chr(skimmers, "skim_type")
    unique_skimmers <- reduce_skimmers(skimmers, types)
    combined_skimmers <- purrr::map(unique_skimmers, join_with_base,
        base)
    ready_to_skim <- tibble::tibble(skim_type = unique(types),
        skimmers = purrr::map(combined_skimmers, mangle_names,
            names(base$funs)), skim_variable = split(selected,
            types)[unique(types)])
    grouped <- dplyr::group_by(ready_to_skim, .data$skim_type)
    nested <- dplyr::summarize(grouped, skimmed = purrr::map2(.data$skimmers,
        .data$skim_variable, skim_by_type, data))
    structure(tidyr::unnest(nested, "skimmed"), class = c("skim_df",
        "tbl_df", "tbl", "data.frame"), data_rows = nrow(data),
        data_cols = ncol(data), df_name = .data_name, dt_key = get_dt_key(data),
        groups = dplyr::group_vars(data), base_skimmers = names(base$funs),
        skimmers_used = get_skimmers_used(unique_skimmers))
}
<bytecode: 0x000001b393d00168>
<environment: 0x000001b393d044d0>
```

evals %>%
select(score, age) %>%
skim()

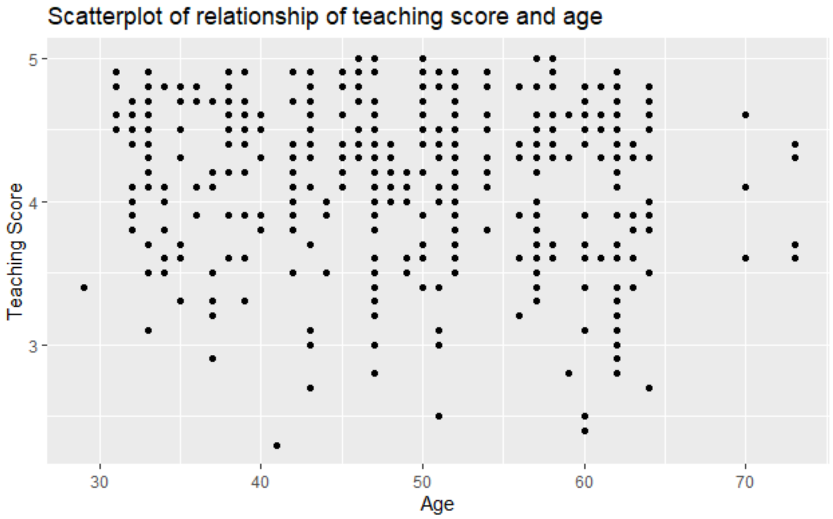


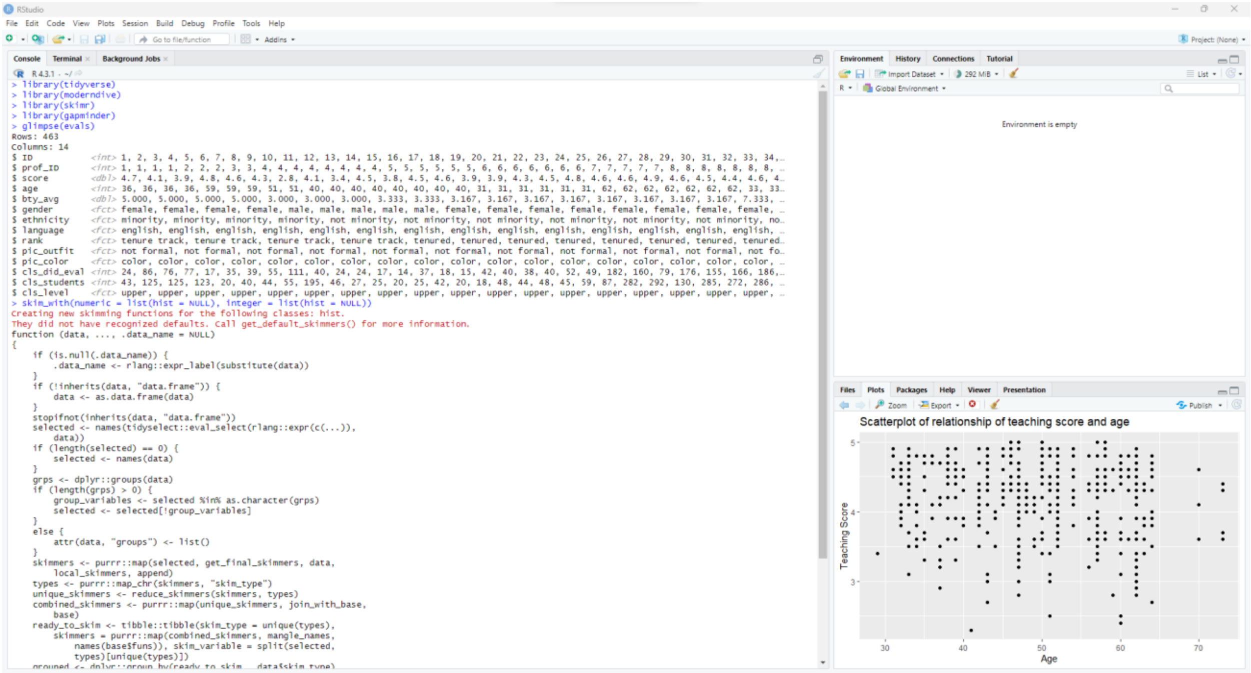
```
> evals %>%
+     select(score, age) %>%
+     skim()
── Data Summary ────────────────────────
                           Values
Name                       Piped data
Number of rows             463
Number of columns          2
_______________________
Column type frequency:
  numeric                  2
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd   p0  p25  p50  p75 p100 hist
1 score                 0             1  4.17 0.544  2.3  3.8  4.3  4.6    5 ▁▁▃▇▇
2 age                   0             1 48.4  9.80  29   42   48   57     73 ▅▆▇▆▁
```

ggplot(evals, aes(x = age, y = score)) +
 geom_point() +
 labs(
  x = "Age", y = "Teaching Score",
  title = "Scatterplot of relationship of teaching score and age"
)

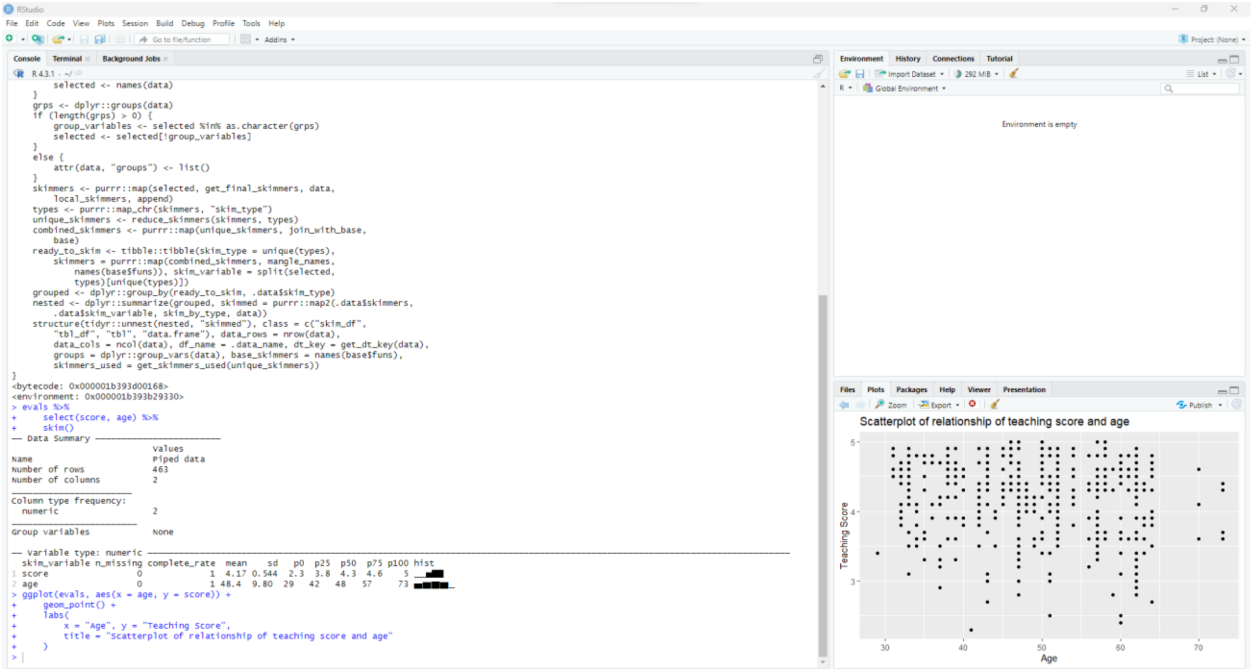


```
> ggplot(evals, aes(x = age, y = score)) +
+     geom_point() +
+     labs(
+         x = "Age", y = "Teaching Score",
+         title = "Scatterplot of relationship of teaching score and age"
+     )
> |
```





```
> library(tidyverse)
> library(moderndive)
> library(skimr)
> library(gapminder)
> glimpse(evals)
Rows: 463
Columns: 14
$ ID           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,…
$ prof_ID      <int> 1, 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, …
$ score        <dbl> 4.7, 4.1, 3.9, 4.8, 4.6, 4.3, 2.8, 4.1, 3.4, 4.5, 3.8, 4.5, 4.6, 3.9, 3.9, 4.3, 4.5, 4.8, 4.6, 4.6, 4.9, 4.6, 4.5, 4.4, 4.6, 4…
$ age          <int> 36, 36, 36, 36, 59, 59, 59, 51, 51, 40, 40, 40, 40, 40, 40, 40, 40, 31, 31, 31, 31, 31, 31, 62, 62, 62, 62, 62, 62, 62, 33, 33…
$ bty_avg      <dbl> 5.000, 5.000, 5.000, 5.000, 3.000, 3.000, 3.000, 3.333, 3.333, 3.167, 3.167, 3.167, 3.167, 3.167, 3.167, 3.167, 3.167, 7.333, …
$ gender       <fct> female, female, female, female, male, male, male, male, male, female, female, female, female, female, female, female, female, …
$ ethnicity    <fct> minority, minority, minority, minority, not minority, not minority, not minority, not minority, not minority, not minority, no…
$ language     <fct> english, english, english, english, english, english, english, english, english, english, english, english, english, english, …
$ rank         <fct> tenure track, tenure track, tenure track, tenure track, tenured, tenured, tenured, tenured, tenured, tenured, tenured, tenured…
$ pic_outfit   <fct> not formal, not formal, not formal, not formal, not formal, not formal, not formal, not formal, not formal, not formal, not fo…
$ pic_color    <fct> color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, color, …
$ cls_did_eval <int> 24, 86, 76, 77, 17, 35, 39, 55, 111, 40, 24, 24, 17, 14, 37, 18, 15, 42, 40, 38, 40, 52, 49, 182, 160, 79, 176, 155, 166, 186,…
$ cls_students <int> 43, 125, 125, 123, 20, 40, 44, 55, 195, 46, 27, 25, 20, 25, 42, 20, 18, 48, 44, 48, 45, 59, 87, 282, 292, 130, 285, 272, 286, …
$ cls_level    <fct> upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, upper, …
> skim_with(numeric = list(hist = NULL), integer = list(hist = NULL))
Creating new skimming functions for the following classes: hist.
They did not have recognized defaults. Call get_default_skimmers() for more information.
function (data, ..., .data_name = NULL)
{
    if (is.null(.data_name)) {
        .data_name <- rlang::expr_label(substitute(data))
    }
    if (!inherits(data, "data.frame")) {
        data <- as.data.frame(data)
    }
    stopifnot(inherits(data, "data.frame"))
    selected <- names(tidyselect::eval_select(rlang::expr(c(...)),
        data))
    if (length(selected) == 0) {
        selected <- names(data)
    }
    grps <- dplyr::groups(data)
    if (length(grps) > 0) {
        group_variables <- selected %in% as.character(grps)
        selected <- selected[!group_variables]
    }
    else {
        attr(data, "groups") <- list()
    }
    skimmers <- purrr::map(selected, get_final_skimmers, data,
        local_skimmers, append)
    types <- purrr::map_chr(skimmers, "skim_type")
    unique_skimmers <- reduce_skimmers(skimmers, types)
    combined_skimmers <- purrr::map(unique_skimmers, join_with_base,
        base)
    ready_to_skim <- tibble::tibble(skim_type = unique(types),
        skimmers = purrr::map(combined_skimmers, mangle_names,
            names(base$funs)), skim_variable = split(selected,
            types)[unique(types)])
    grouped <- dplyr::group_by(ready_to_skim, .data$skim_type)
```

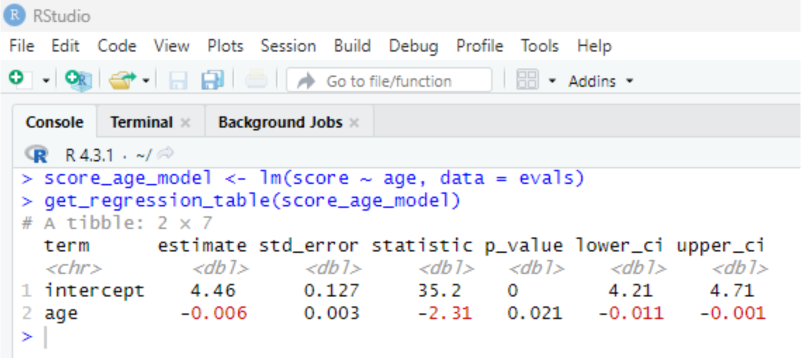


```
        selected <- names(data)
    }
    grps <- dplyr::groups(data)
    if (length(grps) > 0) {
        group_variables <- selected %in% as.character(grps)
        selected <- selected[!group_variables]
    }
    else {
        attr(data, "groups") <- list()
    }
    skimmers <- purrr::map(selected, get_final_skimmers, data,
        local_skimmers, append)
    types <- purrr::map_chr(skimmers, "skim_type")
    unique_skimmers <- reduce_skimmers(skimmers, types)
    combined_skimmers <- purrr::map(unique_skimmers, join_with_base,
        base)
    ready_to_skim <- tibble::tibble(skim_type = unique(types),
        skimmers = purrr::map(combined_skimmers, mangle_names,
            names(base$funs)), skim_variable = split(selected,
            types)[unique(types)])
    grouped <- dplyr::group_by(ready_to_skim, .data$skim_type)
    nested <- dplyr::summarize(grouped, skimmed = purrr::map2(.data$skimmers,
        .data$skim_variable, skim_by_type, data))
    structure(tidyr::unnest(nested, "skimmed"), class = c("skim_df",
        "tbl_df", "tbl", "data.frame"), data_rows = nrow(data),
        data_cols = ncol(data), df_name = .data_name, dt_key = get_dt_key(data),
        groups = dplyr::group_vars(data), base_skimmers = names(base$funs),
        skimmers_used = get_skimmers_used(unique_skimmers))
}
<bytecode: 0x000001b393d00168>
<environment: 0x000001b393b29330>
> evals %>%
+     select(score, age) %>%
+     skim()
── Data Summary ────────────────────────
                           Values
Name                       Piped data
Number of rows             463
Number of columns          2
_______________________
Column type frequency:
  numeric                  2
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd   p0  p25  p50  p75 p100 hist
1 score                 0             1  4.17 0.544  2.3  3.8  4.3  4.6    5 ▁▁▃▇
2 age                   0             1  48.4  9.80  29   42   48   57    73 ▇▇▇▇▁
> ggplot(evals, aes(x = age, y = score)) +
+     geom_point() +
+     labs(
+         x = "Age", y = "Teaching Score",
+         title = "Scatterplot of relationship of teaching score and age"
+     )
> |
```

(LC5.2) Fit a new simple linear regression using lm(score ~ age, data = evals_ch5) where age is the new explanatory variable x. Get information about the "best-fitting" line from the regression table by applying the get_regression_table() function. How do the regression results match up with the results from your earlier exploratory data analysis?

score_age_model <- lm(score ~ age, data = evals)

get_regression_table(score_age_model)

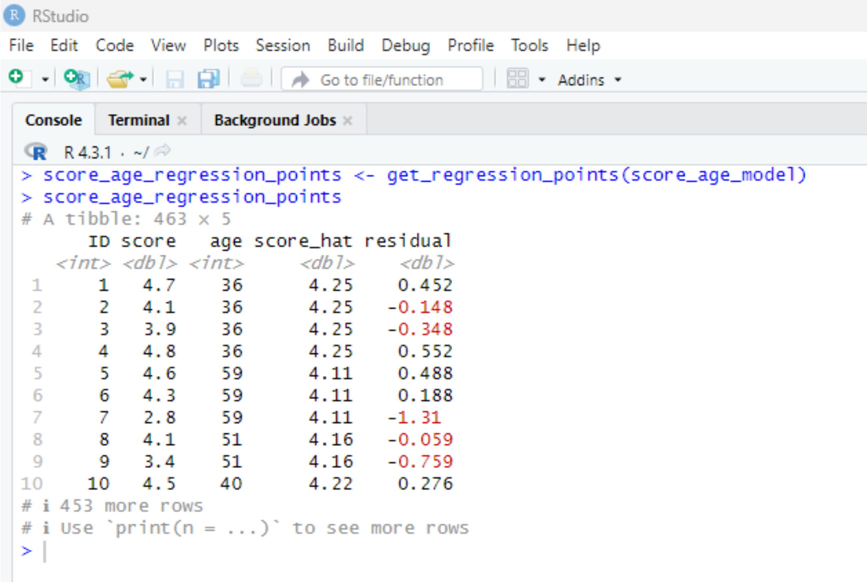


```
> score_age_model <- lm(score ~ age, data = evals)
> get_regression_table(score_age_model)
# A tibble: 2 × 7
  term      estimate std_error statistic p_value lower_ci upper_ci
  <chr>        <dbl>     <dbl>     <dbl>   <dbl>    <dbl>    <dbl>
1 intercept    4.46      0.127     35.2    0        4.21     4.71
2 age         -0.006     0.003     -2.31   0.021   -0.011   -0.001
> |
```

Every unit increase in age is related with a 0.006 unit drop in score. It corresponds to the findings of our previous exploratory data study.

(LC5.3) Generate a data frame of the residuals of the model where you used age as the explanatory x variable.

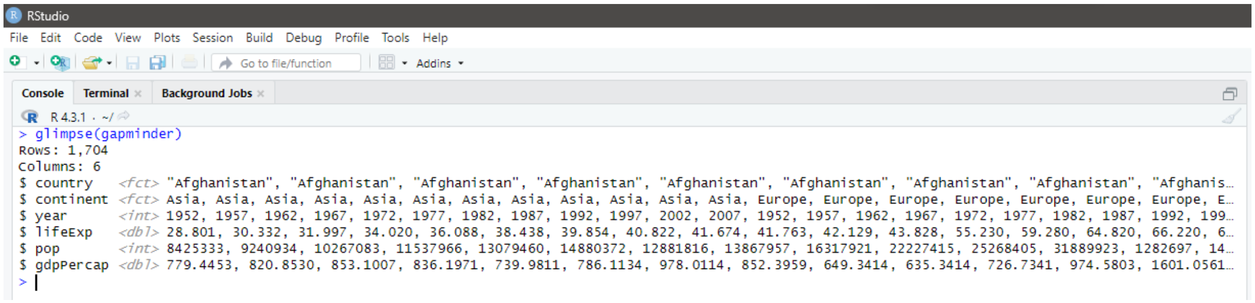
score_age_regression_points <- get_regression_points(score_age_model)
score_age_regression_points



```
> score_age_regression_points <- get_regression_points(score_age_model)
> score_age_regression_points
# A tibble: 463 × 5
      ID score   age score_hat residual
   <int> <dbl> <int>     <dbl>    <dbl>
 1     1   4.7    36      4.25    0.452
 2     2   4.1    36      4.25   -0.148
 3     3   3.9    36      4.25   -0.348
 4     4   4.8    36      4.25    0.552
 5     5   4.6    59      4.11    0.488
 6     6   4.3    59      4.11    0.188
 7     7   2.8    59      4.11   -1.31
 8     8   4.1    51      4.16   -0.059
 9     9   3.4    51      4.16   -0.759
10    10   4.5    40      4.22    0.276
# ℹ 453 more rows
# ℹ Use `print(n = ...)` to see more rows
> |
```

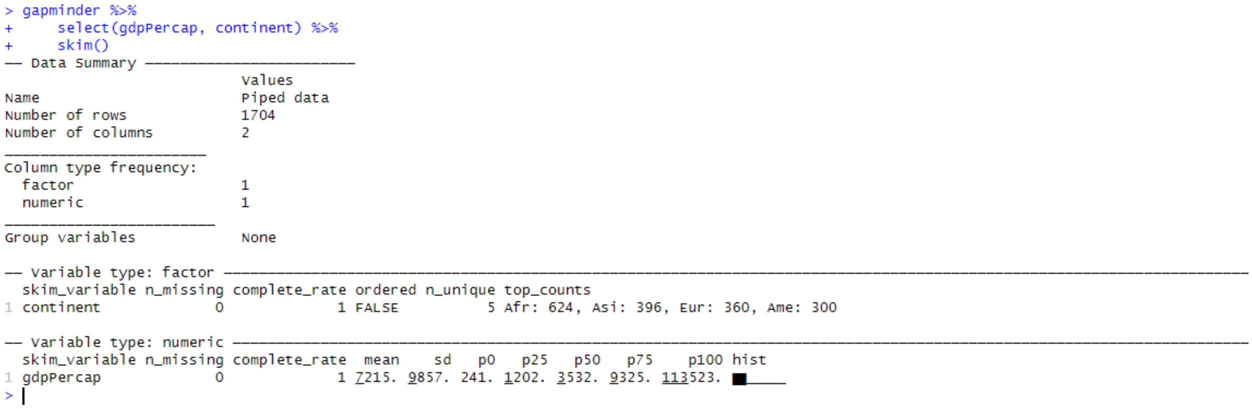
(LC5.4) Conduct a new exploratory data analysis with the same explanatory variable x being continent but with gdpPercap as the new outcome variable y. What can you say about the differences in GDP per capita between continents based on this exploration?

glimpse(gapminder)



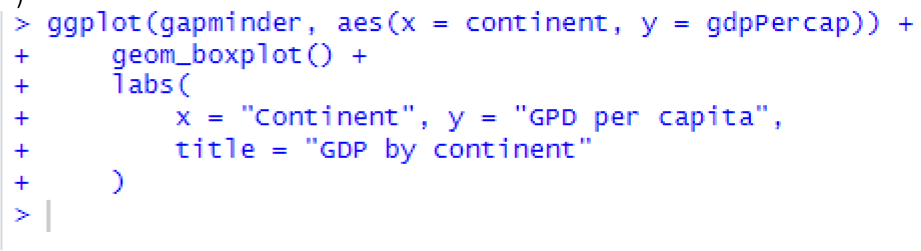
```
> glimpse(gapminder)
Rows: 1,704
Columns: 6
$ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis…
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Europe, Europe, Europe, Europe, Europe, Europe, Europe, E…
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002, 2007, 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 199…
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.822, 41.674, 41.763, 42.129, 43.828, 55.230, 59.280, 64.820, 66.220, 6…
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12881816, 13867957, 16317921, 22227415, 25268405, 31889923, 1282697, 14…
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, 978.0114, 852.3959, 649.3414, 635.3414, 726.7341, 974.5803, 1601.0561…
> |
```

gapminder %>%
select(gdpPercap, continent) %>%
skim()

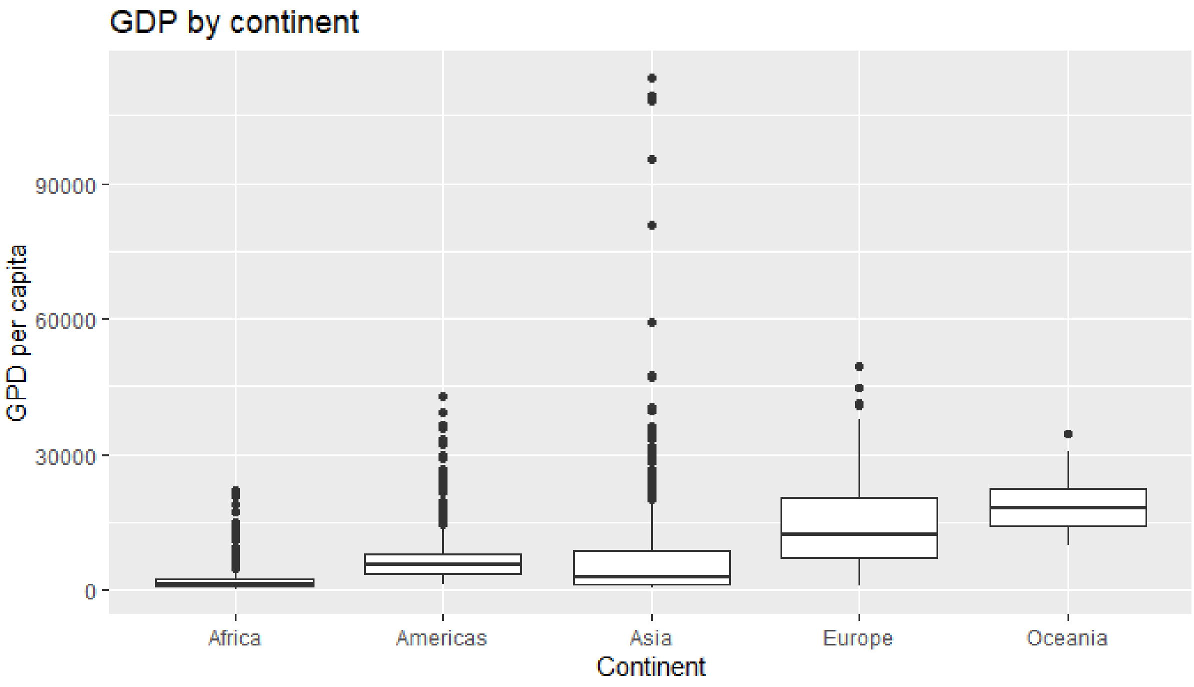


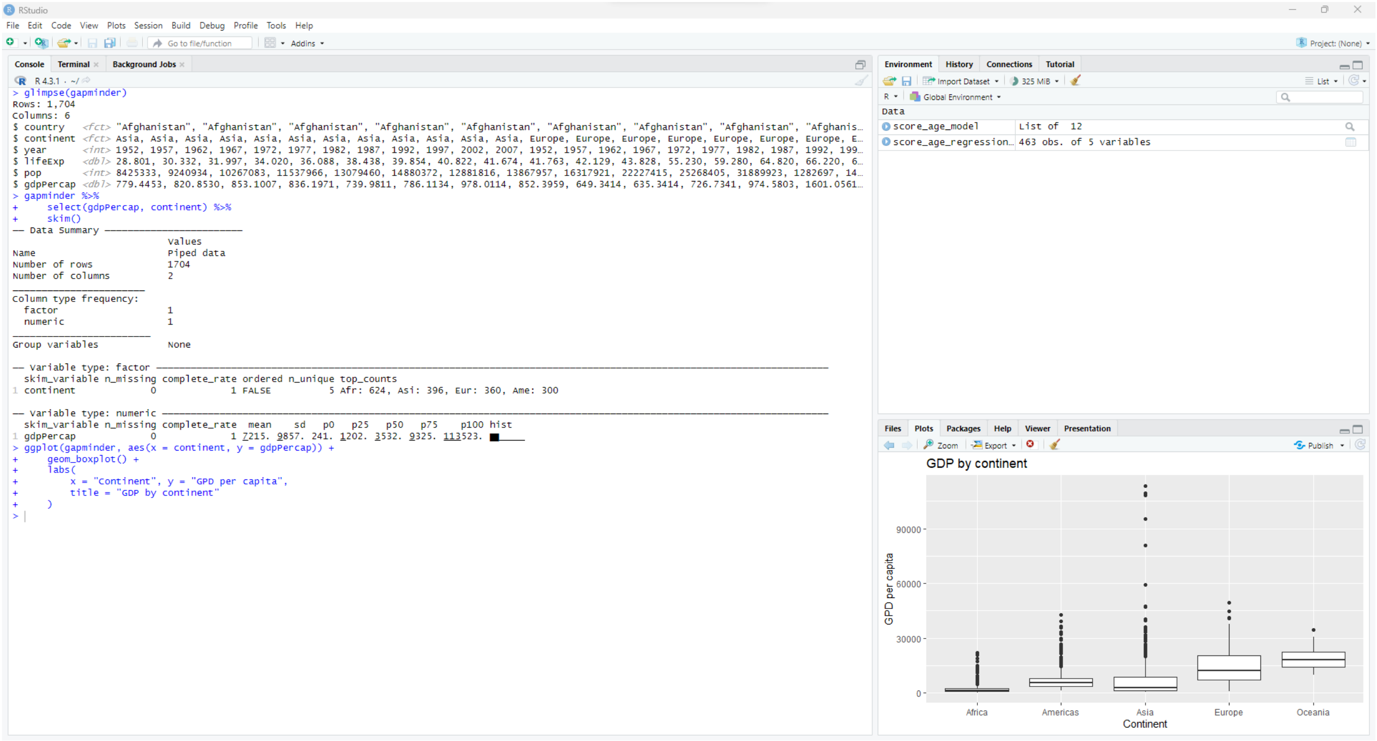
```
> gapminder %>%
+     select(gdpPercap, continent) %>%
+     skim()
── Data Summary ────────────────────────
                           Values
Name                       Piped data
Number of rows             1704
Number of columns          2
_______________________
Column type frequency:
  factor                   1
  numeric                  1
________________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique top_counts
1 continent             0             1 FALSE          5 Afr: 624, Asi: 396, Eur: 360, Ame: 300

── Variable type: numeric ─────────────────────────────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd   p0   p25   p50   p75    p100 hist
1 gdpPercap             0             1 7215. 9857. 241. 1202. 3532. 9325. 113523. ▇▁▁▁▁
> |
```

ggplot(gapminder, aes(x = continent, y = gdpPercap)) +
 geom_boxplot() +
 labs(
  x = "Continent", y = "GPD per capita",
  title = "GDP by continent"
)



```
> ggplot(gapminder, aes(x = continent, y = gdpPercap)) +
+     geom_boxplot() +
+     labs(
+         x = "Continent", y = "GPD per capita",
+         title = "GDP by continent"
+     )
> |
```



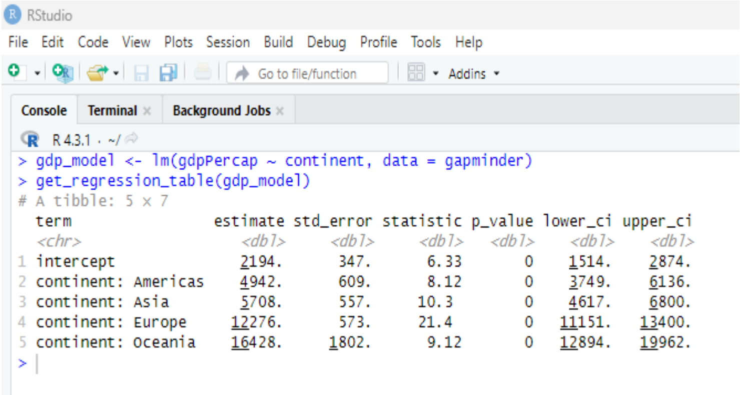


```
> glimpse(gapminder)
Rows: 1,704
Columns: 6
$ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis…
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Europe, Europe, Europe, Europe, Europe, Europe, Europe, E…
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002, 2007, 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 199…
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.822, 41.674, 41.763, 42.129, 43.828, 55.230, 59.280, 64.820, 66.220, 6…
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12881816, 13867957, 16317921, 22227415, 25268405, 31889923, 1282697, 14…
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, 978.0114, 852.3959, 649.3414, 635.3414, 726.7341, 974.5803, 1601.0561…
> gapminder %>%
+     select(gdpPercap, continent) %>%
+     skim()
── Data Summary ────────────────────────
                           Values
Name                       Piped data
Number of rows             1704
Number of columns          2
_______________________
Column type frequency:
  factor                   1
  numeric                  1
________________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique top_counts
1 continent             0             1 FALSE          5 Afr: 624, Asi: 396, Eur: 360, Ame: 300

── Variable type: numeric ─────────────────────────────────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd   p0  p25  p50  p75    p100 hist
1 gdpPercap             0             1 7215. 9857. 241. 1202. 3532. 9325. 113523. ▇▁▁▁▁
> ggplot(gapminder, aes(x = continent, y = gdpPercap)) +
+     geom_boxplot() +
+     labs(
+         x = "Continent", y = "GPD per capita",
+         title = "GDP by continent"
+     )
> |
```

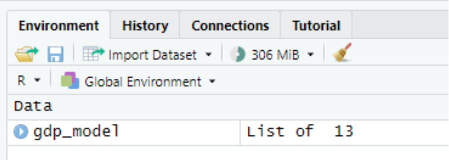
Data	
score_age_model	List of 12
score_age_regression…	463 obs. of 5 variables

(LC5.5) Fit a new linear regression using lm(gdpPercap ~ continent, data = gapminder2007) where gdpPercap is the new outcome variable y . Get information about the "best-fitting" line from the regression table by applying the get_regression_table() function. How do the regression results match up with the results from your previous exploratory data analysis?

gdp_model <- lm(gdpPercap ~ continent, data = gapminder)
get_regression_table(gdp_model)



```
> gdp_model <- lm(gdpPercap ~ continent, data = gapminder)
> get_regression_table(gdp_model)
# A tibble: 5 × 7
  term                estimate std_error statistic p_value lower_ci upper_ci
  <chr>                  <dbl>     <dbl>     <dbl>   <dbl>    <dbl>    <dbl>
1 intercept              2194.      347.      6.33       0    1514.    2874.
2 continent: Americas    4942.      609.      8.12       0    3749.    6136.
3 continent: Asia        5708.      557.     10.3        0    4617.    6800.
4 continent: Europe     12276.      573.     21.4        0   11151.   13400.
5 continent: Oceania    16428.     1802.      9.12       0   12894.   19962.
> |
```



```
Data
gdp_model          List of  13
```

According to our earlier exploratory data study, continent appears to be a statistically significant predictor of an area's GDP. We may build an equation to predict gdpPercap using the continent as statistically significant predictors by fitting a new linear regression using lm (gdpPercap continent, data = gapminder) where gdpPercap is the new outcome variable y. As a consequence, the regression results agree with the findings of our earlier exploratory data study.

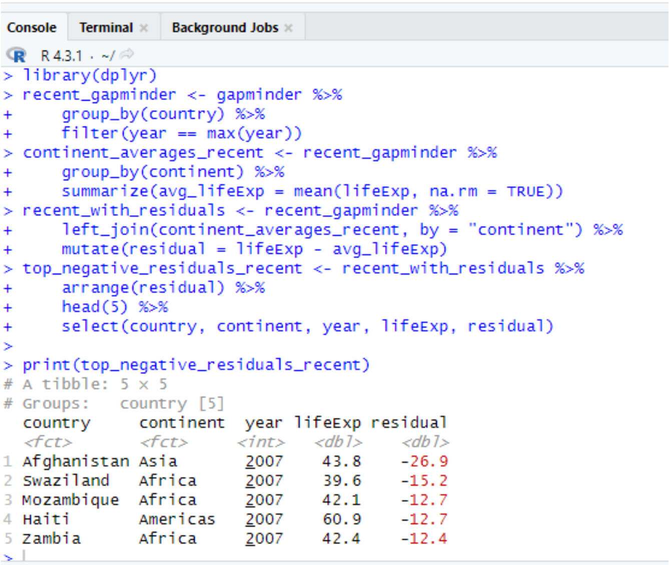
(LC5.6) Using either the sorting functionality of RStudio's spreadsheet viewer or using the data wrangling tools you learned in Chapter 3, identify the five countries with the five smallest (most negative) residuals? What do these negative residuals say about their life expectancy relative to their continents' life expectancy?

We can find the five nations with the smallest (most negative) residuals using the sorting capability of RStudio's spreadsheet viewer: Afghanistan, Swaziland, Mozambique, Haiti, and Zambia.

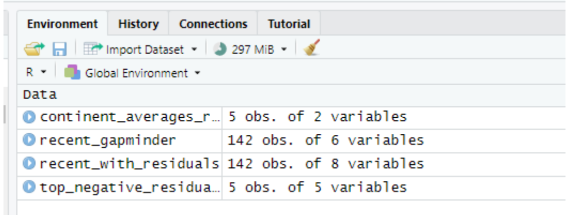
These negative residuals show that these data points deviate the most from their group averages.

This signifies that the average life expectancy in these five countries is the lowest when compared to the average life expectancy in their respective continents.

```
library(dplyr)

recent_gapminder <- gapminder %>%
group_by(country) %>%
filter(year == max(year))

continent_averages_recent <- recent_gapminder %>%
group_by(continent) %>%
summarize(avg_lifeExp = mean(lifeExp, na.rm = TRUE))

recent_with_residuals <- recent_gapminder %>%
left_join(continent_averages_recent, by = "continent") %>%
mutate(residual = lifeExp - avg_lifeExp)

top_negative_residuals_recent <- recent_with_residuals %>%
arrange(residual) %>%
head(5) %>%
select(country, continent, year, lifeExp, residual)

print(top_negative_residuals_recent)
```



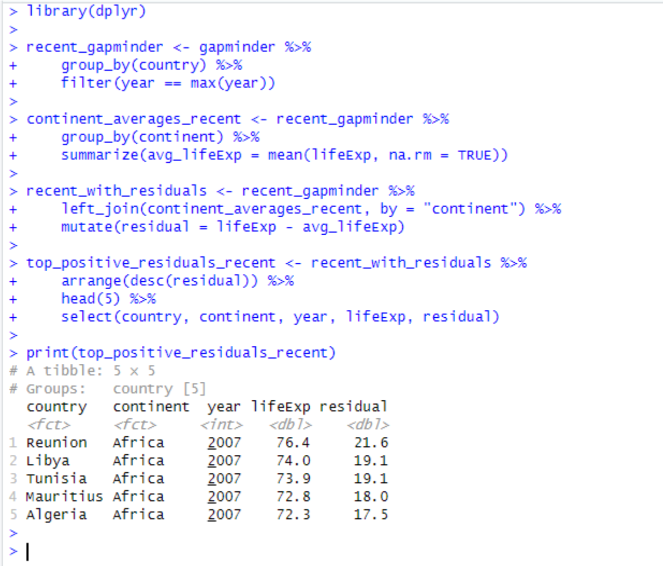
```
> library(dplyr)
> recent_gapminder <- gapminder %>%
+     group_by(country) %>%
+     filter(year == max(year))
> continent_averages_recent <- recent_gapminder %>%
+     group_by(continent) %>%
+     summarize(avg_lifeExp = mean(lifeExp, na.rm = TRUE))
> recent_with_residuals <- recent_gapminder %>%
+     left_join(continent_averages_recent, by = "continent") %>%
+     mutate(residual = lifeExp - avg_lifeExp)
> top_negative_residuals_recent <- recent_with_residuals %>%
+     arrange(residual) %>%
+     head(5) %>%
+     select(country, continent, year, lifeExp, residual)
>
> print(top_negative_residuals_recent)
# A tibble: 5 x 5
# Groups:   country [5]
  country     continent  year lifeExp residual
  <fct>       <fct>     <int>   <dbl>    <dbl>
1 Afghanistan Asia       2007    43.8    -26.9
2 Swaziland   Africa     2007    39.6    -15.2
3 Mozambique  Africa     2007    42.1    -12.7
4 Haiti       Americas   2007    60.9    -12.7
5 Zambia      Africa     2007    42.4    -12.4
> 
```



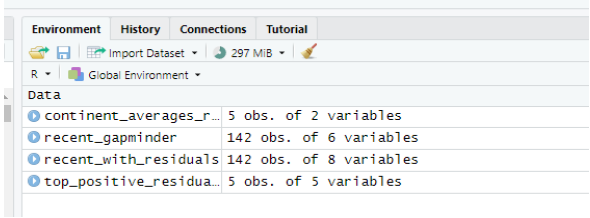
```
Data
continent_averages_r…  5 obs. of 2 variables
recent_gapminder       142 obs. of 6 variables
recent_with_residuals  142 obs. of 8 variables
top_negative_residua…  5 obs. of 5 variables
```

(LC5.7) Repeat this process, but identify the five countries with the five largest (most positive) residuals. What do these positive residuals say about their life expectancy relative to their continents' life expectancy?

```
library(dplyr)

recent_gapminder <- gapminder %>%
        group_by(country) %>%
        filter(year == max(year))
continent_averages_recent <- recent_gapminder %>%
        group_by(continent) %>%
        summarize(avg_lifeExp = mean(lifeExp, na.rm = TRUE))
recent_with_residuals <- recent_gapminder %>%
        left_join(continent_averages_recent, by = "continent") %>%
        mutate(residual = lifeExp - avg_lifeExp)
top_positive_residuals_recent <- recent_with_residuals %>%
        arrange(desc(residual)) %>%
        head(5) %>%
        select(country, continent, year, lifeExp, residual)

print(top_positive_residuals_recent)
```



```
> library(dplyr)
>
> recent_gapminder <- gapminder %>%
+     group_by(country) %>%
+     filter(year == max(year))
>
> continent_averages_recent <- recent_gapminder %>%
+     group_by(continent) %>%
+     summarize(avg_lifeExp = mean(lifeExp, na.rm = TRUE))
>
> recent_with_residuals <- recent_gapminder %>%
+     left_join(continent_averages_recent, by = "continent") %>%
+     mutate(residual = lifeExp - avg_lifeExp)
>
> top_positive_residuals_recent <- recent_with_residuals %>%
+     arrange(desc(residual)) %>%
+     head(5) %>%
+     select(country, continent, year, lifeExp, residual)
>
> print(top_positive_residuals_recent)
# A tibble: 5 x 5
# Groups:   country [5]
  country   continent  year lifeExp residual
  <fct>     <fct>     <int>   <dbl>    <dbl>
1 Reunion   Africa     2007    76.4     21.6
2 Libya     Africa     2007    74.0     19.1
3 Tunisia   Africa     2007    73.9     19.1
4 Mauritius Africa     2007    72.8     18.0
5 Algeria   Africa     2007    72.3     17.5
>
> |
```



```
Data
continent_averages_r…  5 obs. of 2 variables
recent_gapminder       142 obs. of 6 variables
recent_with_residuals  142 obs. of 8 variables
top_positive_residua…  5 obs. of 5 variables
```

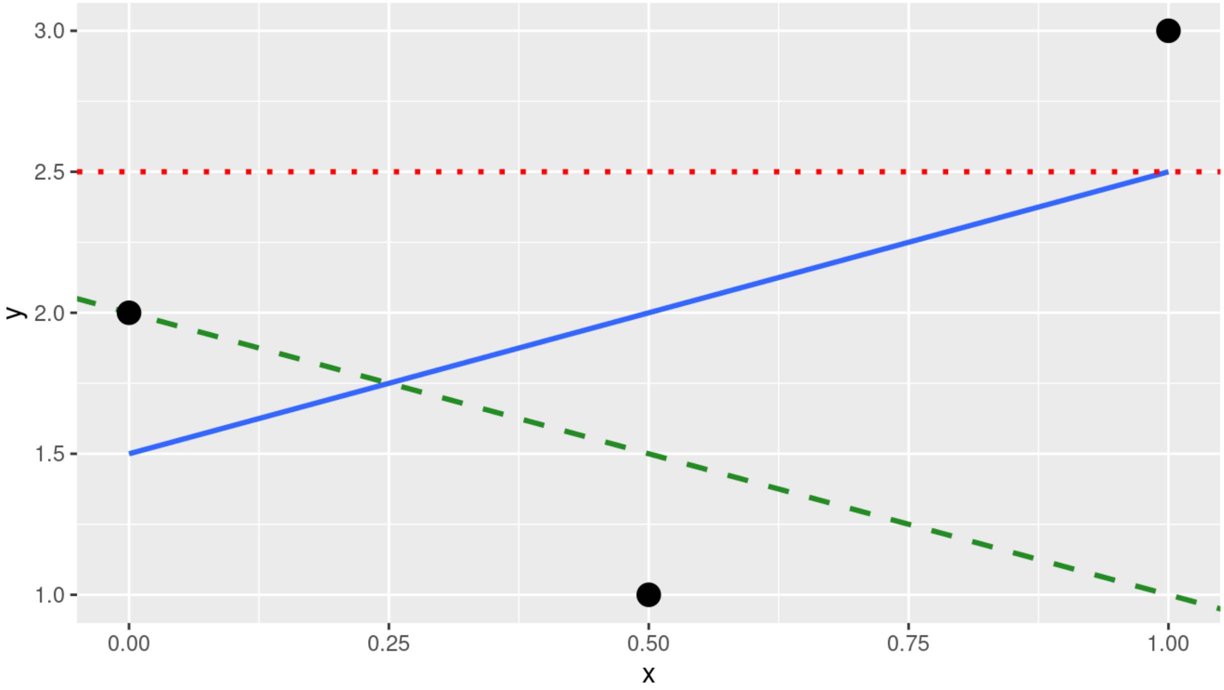
We can find the five nations with the highest (most positive) residuals using the sorting capabilities of RStudio's spreadsheet viewer: Reunion, Libya, Tunisia, Mauritius, and Algeria.

These positive residuals show that the data points are above the regression line by the greatest distance.

This signifies that the average life expectancy in these five countries is the greatest when compared to the average life expectancy in their respective continents.

(LC5.8) Note in the following plot there are 3 points marked with dots along with:

- The "best" fitting solid regression line in blue
- An arbitrarily chosen dotted red line
- Another arbitrarily chosen dashed green line



Compute the sum of squared residuals by hand for each line and show that of these three lines, the regression line in blue has the smallest value.

```
library(ggplot2)

x <- c(0, 0.5, 1)

y <- c(2, 1, 3)

data <- data.frame(x, y)

blue_line_end_y <- 1.5 + 1 * max(x)

if (blue_line_end_y > 2.5) {

 blue_line_end_x <- (2.5 - 1.5) / 1

} else {

 blue_line_end_x <- max(x)

}

ggplot(data, aes(x = x, y = y)) +

geom_point(size = 3) +

geom_segment(aes(x = 0, y = 1.5, xend = blue_line_end_x, yend = 1.5 + 1 * blue_line_end_x), color = "blue", linewidth = 1) +

geom_hline(yintercept = 2.5, color = "red", linetype = "dotted", linewidth = 1) +

geom_abline(intercept = 2.0, slope = -1.0, color = "green", linetype = "dashed", linewidth = 1) +

theme_minimal() +

labs(x = "X", y = "Y")

y_hat_blue <- c(1.5, 2.0, 2.5)

SSR_blue <- sum((y - y_hat_blue)^2)

y_hat_red <- rep(2.5, 3)

SSR_red <- sum((y - y_hat_red)^2)

y_hat_green <- c(2.0, 1.5, 1.0)
```

SSR_green <- sum((y - y_hat_green)^2)

print(paste("SSR for the blue line:", SSR_blue))

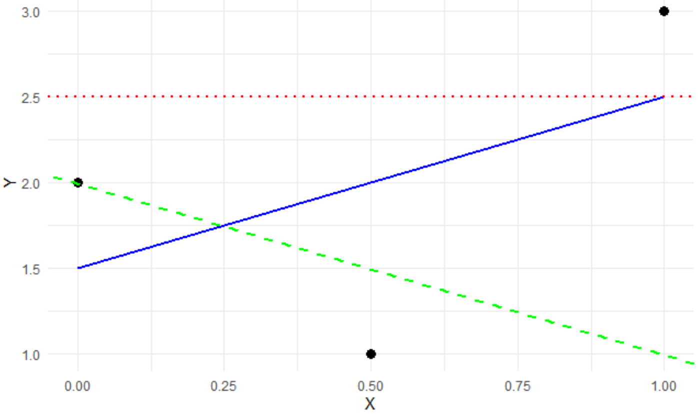
print(paste("SSR for the red dotted line:", SSR_red))

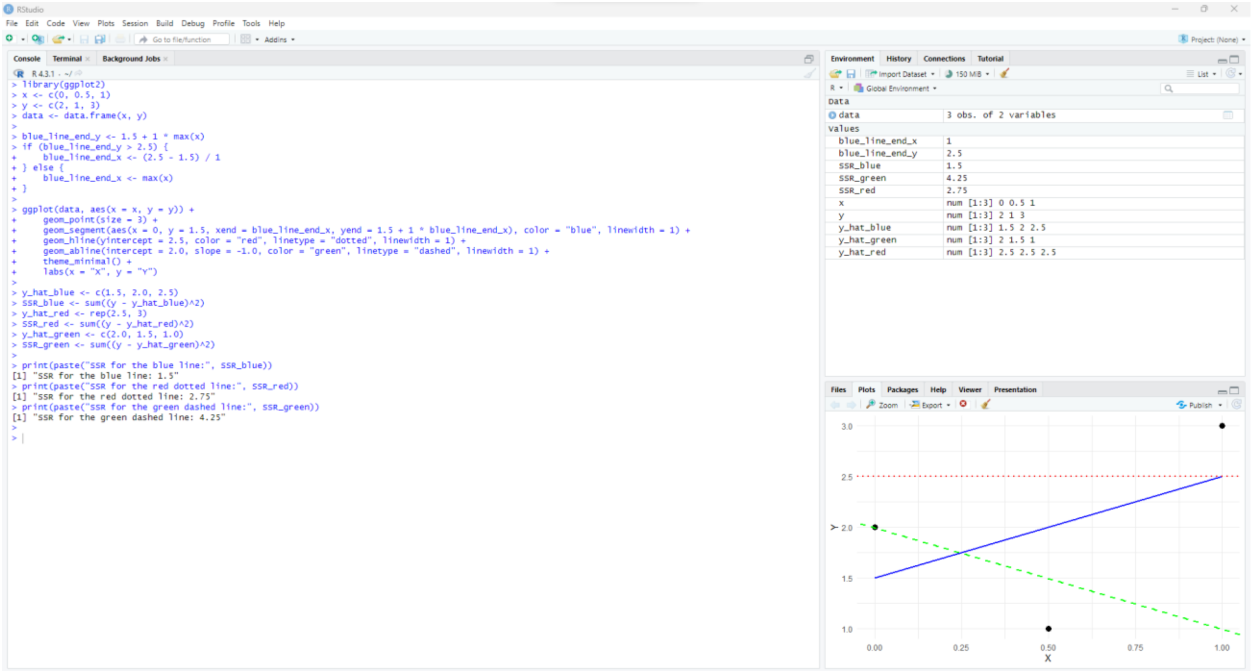
print(paste("SSR for the green dashed line:", SSR_green))

The “best” fitting solid regression line in blue: 1.5

An arbitrarily chosen dotted red line: 2.75

Another arbitrarily chosen dashed green line: 4.25





```
> library(ggplot2)
> x <- c(0, 0.5, 1)
> y <- c(2, 1, 3)
> data <- data.frame(x, y)
>
> blue_line_end_y <- 1.5 + 1 * max(x)
> if (blue_line_end_y > 2.5) {
+     blue_line_end_x <- (2.5 - 1.5) / 1
+ } else {
+     blue_line_end_x <- max(x)
+ }
>
> ggplot(data, aes(x = x, y = y)) +
+     geom_point(size = 3) +
+     geom_segment(aes(x = 0, y = 1.5, xend = blue_line_end_x, yend = 1.5 + 1 * blue_line_end_x), color = "blue", linewidth = 1) +
+     geom_hline(yintercept = 2.5, color = "red", linetype = "dotted", linewidth = 1) +
+     geom_abline(intercept = 2.0, slope = -1.0, color = "green", linetype = "dashed", linewidth = 1) +
+     theme_minimal() +
+     labs(x = "X", y = "Y")
>
> y_hat_blue <- c(1.5, 2.0, 2.5)
> SSR_blue <- sum((y - y_hat_blue)^2)
> y_hat_red <- rep(2.5, 3)
> SSR_red <- sum((y - y_hat_red)^2)
> y_hat_green <- c(2.0, 1.5, 1.0)
> SSR_green <- sum((y - y_hat_green)^2)
>
> print(paste("SSR for the blue line:", SSR_blue))
[1] "SSR for the blue line: 1.5"
> print(paste("SSR for the red dotted line:", SSR_red))
[1] "SSR for the red dotted line: 2.75"
> print(paste("SSR for the green dashed line:", SSR_green))
[1] "SSR for the green dashed line: 4.25"
>
> |
```

Data	
data	3 obs. of 2 variables
Values	
blue_line_end_x	1
blue_line_end_y	2.5
SSR_blue	1.5
SSR_green	4.25
SSR_red	2.75
x	num [1:3] 0 0.5 1
y	num [1:3] 2 1 3
y_hat_blue	num [1:3] 1.5 2 2.5
y_hat_green	num [1:3] 2 1.5 1
y_hat_red	num [1:3] 2.5 2.5 2.5