PSY 503: Lab 04 - Joins, Broom, Regression Intro

Bella Fascendini

2024-10-02

PSY 503: Lab 04 - Joins, Broom, Regression Intro

For this lab assignment, let's imagine that Galton collected and saved height measurements of families in the following manner:

a) he denoted each of the 928 families surveyed with a family id
b) children's heights in families were saved in child-height.csv
c) parents' heights were saved in parent_height.csv
d) for a side project, he also collected grandparents' heights in families in to grandparent_height.csv [NOTE: this never happened]

These files can be found in the following Google drive directory: https://drive.google.com/drive/folders/1JW1KQ9DYwFC3nb3iktWBF0hNG4WfHNjP?usp=sharing

Joins

Your tasks are to:

J1) Using a join command, combine the dataframes for **children** and **parents** based on their family identifier. Display this dataframe. Save this dataframe for future use.

```r
children <- read_csv("child_height.csv")
```

```
## Rows: 928 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): family_id
## dbl (1): child_ht
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
parents <- read_csv("parent_height.csv")
```

```
## Rows: 928 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): family_id
## dbl (1): parent_ht
```

```
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
# Join children and parents by family id
children_parents <- left_join(children, parents, by = "family_id")
print(children_parents)
```

```
## # A tibble: 928 x 3
##    family_id child_ht parent_ht
##    <chr>        <dbl>     <dbl>
##  1 F1            72.2      74.5
##  2 F2            73.2      74.5
##  3 F3            73.2      74.5
##  4 F4            73.2      74.5
##  5 F5            68.2      73.5
##  6 F6            69.2      73.5
##  7 F7            69.2      73.5
##  8 F8            70.2      73.5
##  9 F9            71.2      73.5
## 10 F10           71.2      73.5
## # i 918 more rows
```

```
write_csv(children_parents, "children_parents.csv")
```

J2) Next, use the appropriate join command to merge the dataframe you derived from (1) and grandparent_height.csv file to produce a single dataframe that saves within one dataframe all the information across the original 3 .csv files. Display this dataframe

```
grandparents <- read_csv("grandparent_height.csv")
```

```
## Rows: 928 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): family_id
## dbl (1): gparent_ht
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
all_data <- left_join(children_parents, grandparents, by = "family_id")
print(all_data)
```

```
## # A tibble: 928 x 4
##    family_id child_ht parent_ht gparent_ht
##    <chr>        <dbl>     <dbl>      <dbl>
##  1 F1            72.2      74.5       73.2
##  2 F2            73.2      74.5       NA
##  3 F3            73.2      74.5       74.3
##  4 F4            73.2      74.5       73.9
##  5 F5            68.2      73.5       NA
##  6 F6            69.2      73.5       71.9
##  7 F7            69.2      73.5       NA
##  8 F8            70.2      73.5       71.5
##  9 F9            71.2      73.5       NA
## 10 F10           71.2      73.5       NA
## # i 918 more rows
```

J3) While you are at it, use the appropriate join to find only those families for which all measurements across children, parents, and grandparents were available.Display this dataframe.

```
complete_data <- inner_join(inner_join(children, parents, by = "family_id"), grandparents, by = "family_
print(complete_data)
```

```
## # A tibble: 928 x 4
##    family_id child_ht parent_ht gparent_ht
##    <chr>        <dbl>     <dbl>      <dbl>
##  1 F1            72.2      74.5       73.2
##  2 F2            73.2      74.5       NA
##  3 F3            73.2      74.5       74.3
##  4 F4            73.2      74.5       73.9
##  5 F5            68.2      73.5       NA
##  6 F6            69.2      73.5       71.9
##  7 F7            69.2      73.5       NA
##  8 F8            70.2      73.5       71.5
##  9 F9            71.2      73.5       NA
## 10 F10           71.2      73.5       NA
## # i 918 more rows
```

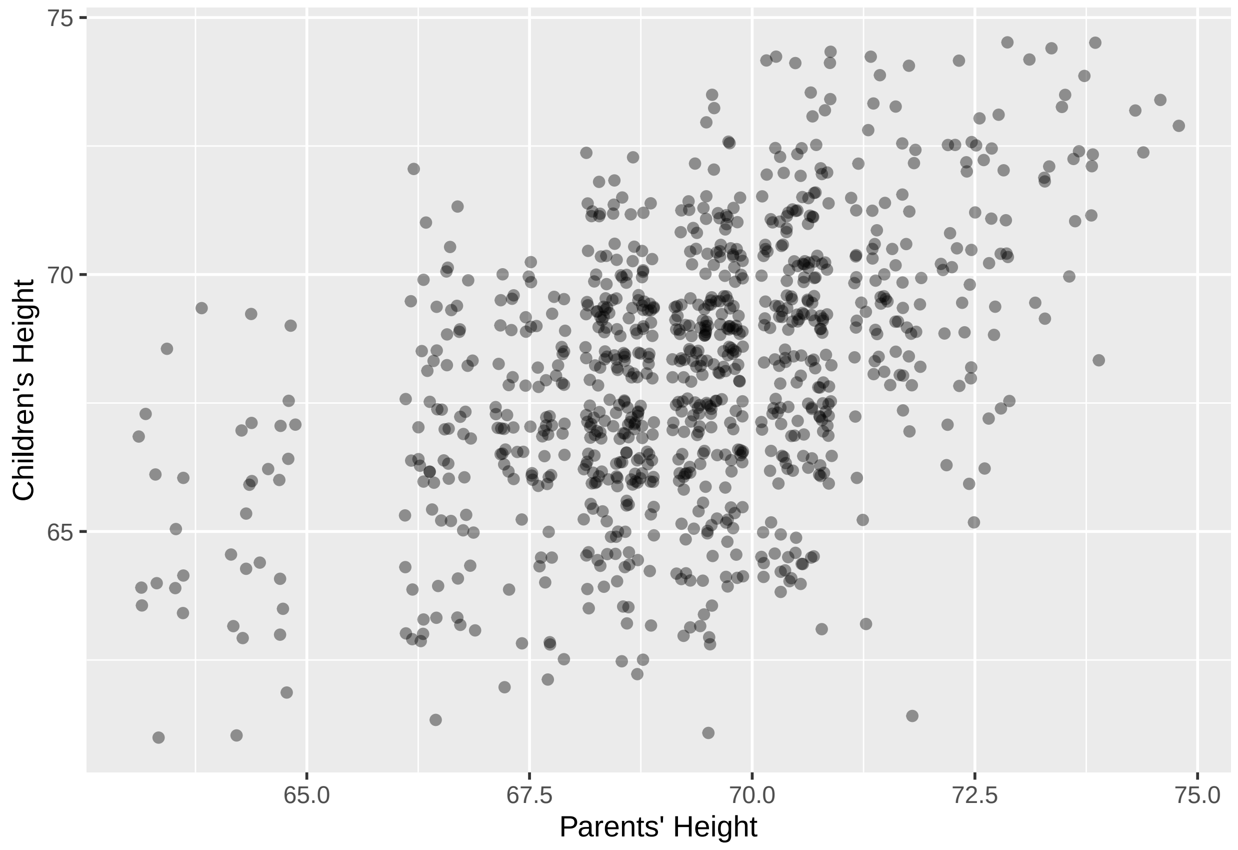
Having used different joins, lets go back to the dataframe produced in (J1), and use it for visualization and for building regression models.

Visualization

Scatterplots are the best way to visualize relationships that you are hoping to model.

V1) Produce a scatterplot with children's height on the y-axis and parents' height on the x-axis. Use geom_jitter, and set transparency with the alpha parameter so that any overlapping data points are easy to spot.

```
ggplot(children_parents,
       aes(x = parent_ht,
           y = child_ht)) +
  geom_jitter(alpha = 0.4) +
  labs(x = "Parents' Height",
       y = "Children's Height")
```

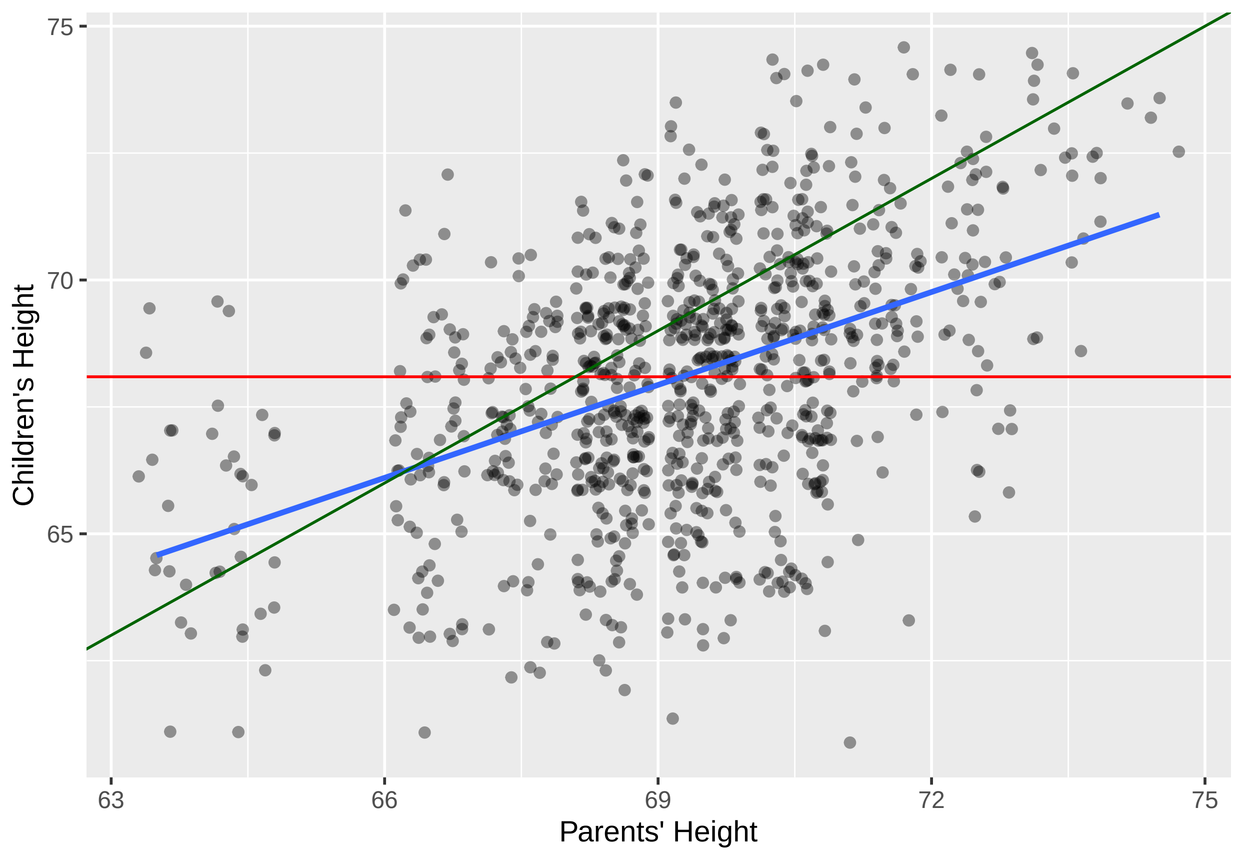


V2) Draw the following lines on the scatterplot: - a horizontal line at the mean of children's heights colored in red, using geom_hline() - a regression line using geom_smooth() by setting its' *method* to be equal to "lm" - diagonal line at the points where the parents' heights are the same as the children's heights. Use geom_abline() and set the color to "dark green"

```
mean_child_ht <- mean(children_parents$child_ht)

ggplot(children_parents, aes(x = parent_ht,
                             y = child_ht)) +
  geom_jitter(alpha = 0.4) +
  geom_hline(yintercept = mean_child_ht,
             color = "red") +
  geom_smooth(method = "lm",
              se = FALSE) +
  geom_abline(slope = 1,
              intercept = 0,
              color = "darkgreen") +
  labs(x = "Parents' Height",
       y = "Children's Height")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Regression

While visual results are great, we want to use the full power of lm() to find estimates of model parameters that minimize error, to make predictions, and to calculate various diagnostics of model fit.

R1) Build a null model (intercept-only) using lm() for predicting children's height. Show the result of model fitting by printing the model. Show more detailed results by using the summary() command

```
null_model <- lm(child_ht ~ 1, data = children_parents)
print(null_model)
```

```
##
## Call:
## lm(formula = child_ht ~ 1, data = children_parents)
##
## Coefficients:
## (Intercept)
##       68.09
```

```
summary(null_model)
```

```
##
## Call:
## lm(formula = child_ht ~ 1, data = children_parents)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.8933 -1.8933  0.1067  2.1067  6.1067
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 68.09332    0.08346   815.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.543 on 927 degrees of freedom
```

R2) Build a model using parents' height as the explanatory variable. Show the result of model fitting by printing the model. Show more detailed results by using the summary() command

```r
model_parent <- lm(child_ht ~ parent_ht, data = children_parents)
print(model_parent)
```

```
##
## Call:
## lm(formula = child_ht ~ parent_ht, data = children_parents)
##
## Coefficients:
## (Intercept)    parent_ht
##     25.8486       0.6099
```

```r
summary(model_parent)
```

```
##
## Call:
## lm(formula = child_ht ~ parent_ht, data = children_parents)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2577 -1.4280  0.1323  1.5720  5.7918
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 25.84856    2.69009   9.609   <2e-16 ***
## parent_ht    0.60992    0.03882  15.710   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.26 on 926 degrees of freedom
## Multiple R-squared:  0.2104, Adjusted R-squared:  0.2096
## F-statistic: 246.8 on 1 and 926 DF,  p-value: < 2.2e-16
```

R3) While the print() and summary() commands are useful for displaying results, it's hard to automatically extract results of model-fitting using them.

Use the tidy() command in broom to save model fitting results in a new dataframe. Using this dataframe, display the coefficients of the model fit (i.e. parameter estimates of intercept and slope) of the two models.

```r
null_model_tidy <- tidy(null_model)
model_parent_tidy <- tidy(model_parent)

print(null_model_tidy)
```

```
## # A tibble: 1 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)     68.1    0.0835      816.       0
```

```r
print(model_parent_tidy)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   25.8      2.69        9.61 6.67e-21
## 2 parent_ht      0.610    0.0388     15.7  1.76e-49
```

R4) Use the predict() function to generate predictions for both models for three different parent heights. 40 inches, 64 inches, 75 inches. Do this with a single call of predict by passing to it the appropriate data structure with these 3 values.

```r
parents_ht_selected <- tibble(parent_ht = c(40, 64, 75))
predict(null_model, parents_ht_selected)
```

```
##        1        2        3
## 68.09332 68.09332 68.09332
```

```r
predict(model_parent, parents_ht_selected)
```

```
##        1        2        3
## 50.24531 64.88336 71.59246
```

R5) Actually, lets generate predictions for a larger set of values.

```r
x_predict <- tibble(parent_ht = seq(50, 100, by = 2))
  x = seq(50, 100, by = 2) #parents' heights for which you are predicting childrens' heights

y_predict_null<- predict(null_model, x_predict)

y_predict_one_explanatory_variable<- predict(model_parent, x_predict)
```

R6) Let's plot these predictions on top of our previous scatterplot, by adding two geom_line() commands which uses this new data. Note that these geom_lines will be using data frames different from the original dataframe that was used for scatterplot.

```r
#two dataframes you'd use for the two geom_lines
df_predictions_null<- x_predict %>%
                bind_cols(y_predict_null)
```

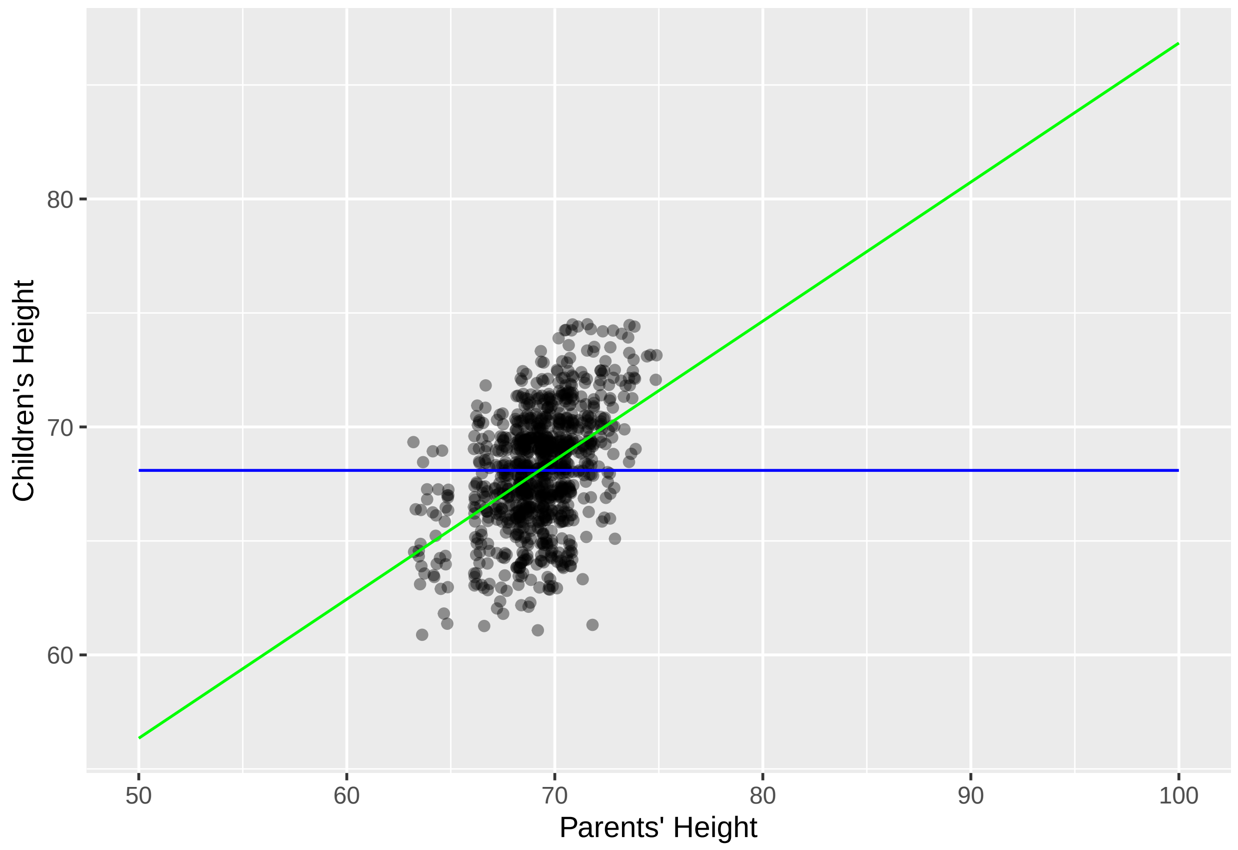
```
## New names:
## * `` -> `...2`
```

```
df_predictions_explanatory<- x_predict %>%
                bind_cols(y_predict_one_explanatory_variable)
```

```
## New names:
## * `` -> `...2`
```

```
#Reproduce your earlier scatterplot from V1 below, and add two geom_line layers

ggplot(children_parents, aes(x = parent_ht,
                             y = child_ht)) +
  geom_jitter(alpha = 0.4) +
  geom_line(data = df_predictions_null, aes(x = parent_ht,
                                            y = y_predict_null), color = "blue") +
  geom_line(data = df_predictions_explanatory, aes(x = parent_ht,
                                                   y = y_predict_one_explanatory_variable),
            color = "green") +
  labs(x = "Parents' Height",
       y = "Children's Height")
```



Why is this way of plotting predictions generally more useful than the method we had used so far?

This plot is more useful because it shows the predictions of the model on top of the actual data, which allows us to see how well the model fits the data.

R7) predict() function is not as special as it seems. Create your own predict functions that uses the coefficients you extracted in (3c) and plugs it into the formulation of (i) the null model and (ii) linear model with 1 explanatory variable.

Find predictions when x is 40 inches, 64 inches, and 75 inches.

```r
predict_null<- function(b0, x){
  return(b0)
}

predict_one_explanatory_variable<- function(b0, b1, x){
  return(b0 + b1 *x)
}

# Coefficients
b0_null <- null_model_tidy$estimate[1]
b0 <- model_parent_tidy$estimate[1]
b1 <- model_parent_tidy$estimate[2]

# Predictions for specific values
predict_null(b0_null, c(40, 64, 75))
```

```
## [1] 68.09332
```

```r
predict_one_explanatory_variable(b0, b1, c(40, 64, 75))
```

```
## [1] 50.24531 64.88336 71.59246
```

R6) The glance() function in broom shows you many "goodness of fit" measures. Compare the r-squared values you obtain for the two models.

```r
glance(null_model)
```

```
## # A tibble: 1 x 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             0  2.54        NA      NA    NA -2182. 4368. 4378.
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```r
glance(model_parent)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.210         0.210  2.26      247. 1.76e-49     1 -2073. 4151. 4166.
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

R7) The augment() function in broom allows you to add columns having to do with model predictions to the dataset. Use augment to create expanded tables for both models.

```r
augmented_null <- augment(null_model, data = children_parents)
augmented_explanatory <- augment(model_parent, data = children_parents)

print(augmented_null)
```

```
## # A tibble: 928 x 9
##    family_id child_ht parent_ht .fitted .resid    .hat .sigma .cooksd .std.resid
##    <chr>        <dbl>     <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>      <dbl>
##  1 F1            72.2      74.5    68.1  4.11  0.00108   2.54 2.82e-3     1.62
##  2 F2            73.2      74.5    68.1  5.11  0.00108   2.54 4.36e-3     2.01
##  3 F3            73.2      74.5    68.1  5.11  0.00108   2.54 4.36e-3     2.01
##  4 F4            73.2      74.5    68.1  5.11  0.00108   2.54 4.36e-3     2.01
##  5 F5            68.2      73.5    68.1  0.107 0.00108   2.54 1.90e-6     0.0420
##  6 F6            69.2      73.5    68.1  1.11  0.00108   2.54 2.05e-4     0.436
##  7 F7            69.2      73.5    68.1  1.11  0.00108   2.54 2.05e-4     0.436
##  8 F8            70.2      73.5    68.1  2.11  0.00108   2.54 7.41e-4     0.829
##  9 F9            71.2      73.5    68.1  3.11  0.00108   2.54 1.61e-3     1.22
## 10 F10           71.2      73.5    68.1  3.11  0.00108   2.54 1.61e-3     1.22
## # i 918 more rows
```

```r
print(augmented_explanatory)
```

```
## # A tibble: 928 x 9
##    family_id child_ht parent_ht .fitted .resid    .hat .sigma .cooksd .std.resid
##    <chr>        <dbl>     <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>      <dbl>
##  1 F1            72.2      74.5    71.3  0.912 0.00917   2.26 7.61e-4      0.406
##  2 F2            73.2      74.5    71.3  1.91  0.00917   2.26 3.34e-3      0.850
##  3 F3            73.2      74.5    71.3  1.91  0.00917   2.26 3.34e-3      0.850
##  4 F4            73.2      74.5    71.3  1.91  0.00917   2.26 3.34e-3      0.850
##  5 F5            68.2      73.5    70.7 -2.48  0.00637   2.26 3.88e-3     -1.10
##  6 F6            69.2      73.5    70.7 -1.48  0.00637   2.26 1.38e-3     -0.656
##  7 F7            69.2      73.5    70.7 -1.48  0.00637   2.26 1.38e-3     -0.656
##  8 F8            70.2      73.5    70.7 -0.478 0.00637   2.26 1.44e-4     -0.212
##  9 F9            71.2      73.5    70.7  0.522 0.00637   2.26 1.72e-4      0.232
## 10 F10           71.2      73.5    70.7  0.522 0.00637   2.26 1.72e-4      0.232
## # i 918 more rows
```

Identify the two variables in this resulting dataset that when added give you the predictor variable (i.e. child's height). What do you think they are referring to?

If I understood the question correctly - I think the two varaibles are .resid and .fitted. .resid is the residuals and .fitted is the predicted values. When these two values are added together, it should sum up to the original observed child's height for each observation.