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

Yeaju Diana Kim

2025-10-01

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

Walk-through Exercise

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
df_a<- tibble (
  ID = c(1,2,3,4,5),
  Name = c("Alice","Bob","Charlie","David","Eve"),
  Age = c(25,30,35,40,45)
)

df_b<- tibble (
  ID = c(2,4,6,8),
  City = c("New York","Los Angeles","Chicago","Houston"),
  Salary = c(50000,60000,55000,65000)
)
```

Play with joins (walkthrough)

```r
result <- df_a %>%
  left_join(df_b, by = "ID")
result
```

```
## # A tibble: 5 x 5
##      ID Name      Age City        Salary
##   <dbl> <chr>   <dbl> <chr>        <dbl>
## 1     1 Alice      25 <NA>            NA
## 2     2 Bob        30 New York     50000
## 3     3 Charlie    35 <NA>            NA
## 4     4 David      40 Los Angeles  60000
## 5     5 Eve        45 <NA>            NA
```

```r
result <- df_a %>%
  right_join(df_b, by = "ID")
result
```

```
## # A tibble: 4 x 5
##      ID Name    Age City        Salary
##   <dbl> <chr> <dbl> <chr>        <dbl>
## 1     2 Bob      30 New York     50000
## 2     4 David    40 Los Angeles  60000
## 3     6 <NA>     NA Chicago      55000
## 4     8 <NA>     NA Houston      65000
```

```r
result <- df_a %>%
  full_join(df_b, by = "ID")
result
```

```
## # A tibble: 7 x 5
##      ID Name      Age City        Salary
##   <dbl> <chr>   <dbl> <chr>        <dbl>
## 1     1 Alice      25 <NA>            NA
## 2     2 Bob        30 New York     50000
## 3     3 Charlie    35 <NA>            NA
## 4     4 David      40 Los Angeles  60000
## 5     5 Eve        45 <NA>            NA
## 6     6 <NA>       NA Chicago      55000
## 7     8 <NA>       NA Houston      65000
```

```r
result <- df_a %>%
  inner_join(df_b, by = "ID")
result
```

```
## # A tibble: 2 x 5
##      ID Name    Age City        Salary
##   <dbl> <chr> <dbl> <chr>        <dbl>
## 1     2 Bob      30 New York     50000
## 2     4 David    40 Los Angeles  60000
```

Carry out semi_join & anti-join with df_b as the left variable, and ID as the primary key.

i) But first, predict what you'd expect to see for either output below:

Answer: When semi joined, only the rows in the left dataset(df_a) that match their ID with the right dataset(df_b) will remain. When anti joined, only the rows in df_a that do not match with df_b will remain.

ii) Now carry out semi_join & anti_join

```
result <- df_a %>%
  semi_join(df_b, by = "ID")
result
```

```
## # A tibble: 2 x 3
##      ID Name    Age
##   <dbl> <chr> <dbl>
## 1     2 Bob      30
## 2     4 David    40
```

```
result <- df_a %>%
  anti_join(df_b, by = "ID")
result
```

```
## # A tibble: 3 x 3
##      ID Name      Age
##   <dbl> <chr>   <dbl>
## 1     1 Alice      25
## 2     3 Charlie    35
## 3     5 Eve        45
```

iii) Compare predictions against output.

Answer: It is as expected!

Lab Assignment

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) assume that 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/1buHo61YL8hw0q-Yzo8YMYUiu5QAvEQjS?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.

```
child <- read.csv('child_height.csv')
parent <- read.csv('parent_height.csv')
grandparent <- read.csv('grandparent_height.csv')
child_parent <- read.csv('galton_child_parent_heights.csv')

head(child)
```

```
##   family_id child_ht
## 1        F1     72.2
## 2        F2     73.2
## 3        F3     73.2
## 4        F4     73.2
## 5        F5     68.2
## 6        F6     69.2
```

```
head(parent)
```

```
##   family_id parent_ht
## 1        F1      74.5
## 2        F2      74.5
## 3        F3      74.5
## 4        F4      74.5
## 5        F5      73.5
## 6        F6      73.5
```

```
head(grandparent)
```

```
##   family_id gparent_ht
## 1        F1   73.18186
## 2        F2         NA
## 3        F3   74.31761
## 4        F4   73.87115
## 5        F5         NA
## 6        F6   71.86452
```

```
child_parent_df <- child %>%
  left_join(parent, by = "family_id")
head(child_parent_df)
```

```
##   family_id child_ht parent_ht
## 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
```

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

```
allgen3 <- child_parent_df %>%
  full_join(grandparent, by = "family_id")
head(allgen3)
```

```
##   family_id child_ht parent_ht gparent_ht
## 1        F1     72.2      74.5   73.18186
## 2        F2     73.2      74.5         NA
## 3        F3     73.2      74.5   74.31761
## 4        F4     73.2      74.5   73.87115
## 5        F5     68.2      73.5         NA
## 6        F6     69.2      73.5   71.86452
```

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.

```
allgen3_inner <- child_parent_df %>%
  inner_join(grandparent, by = "family_id")
allgen3_inner <- na.omit(allgen3_inner)
head(allgen3_inner)
```

```
##    family_id child_ht parent_ht gparent_ht
## 1         F1     72.2      74.5   73.18186
## 3         F3     73.2      74.5   74.31761
## 4         F4     73.2      74.5   73.87115
## 6         F6     69.2      73.5   71.86452
## 8         F8     70.2      73.5   71.47048
## 12       F12     72.2      73.5   72.95794
```

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

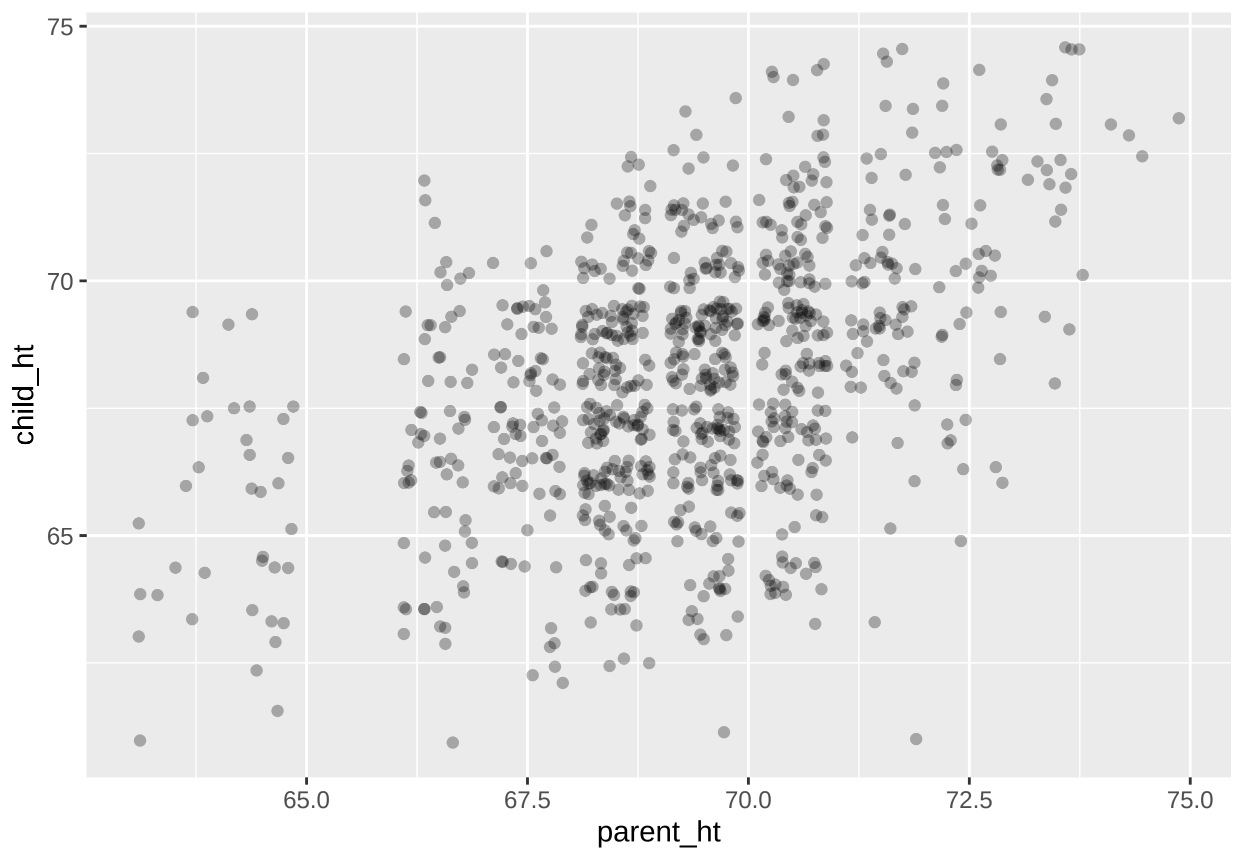
Visualization

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.

```
library(ggplot2)

ggplot(child_parent_df, aes(x = parent_ht, y = child_ht)) +
  geom_jitter(alpha = 0.3)
```

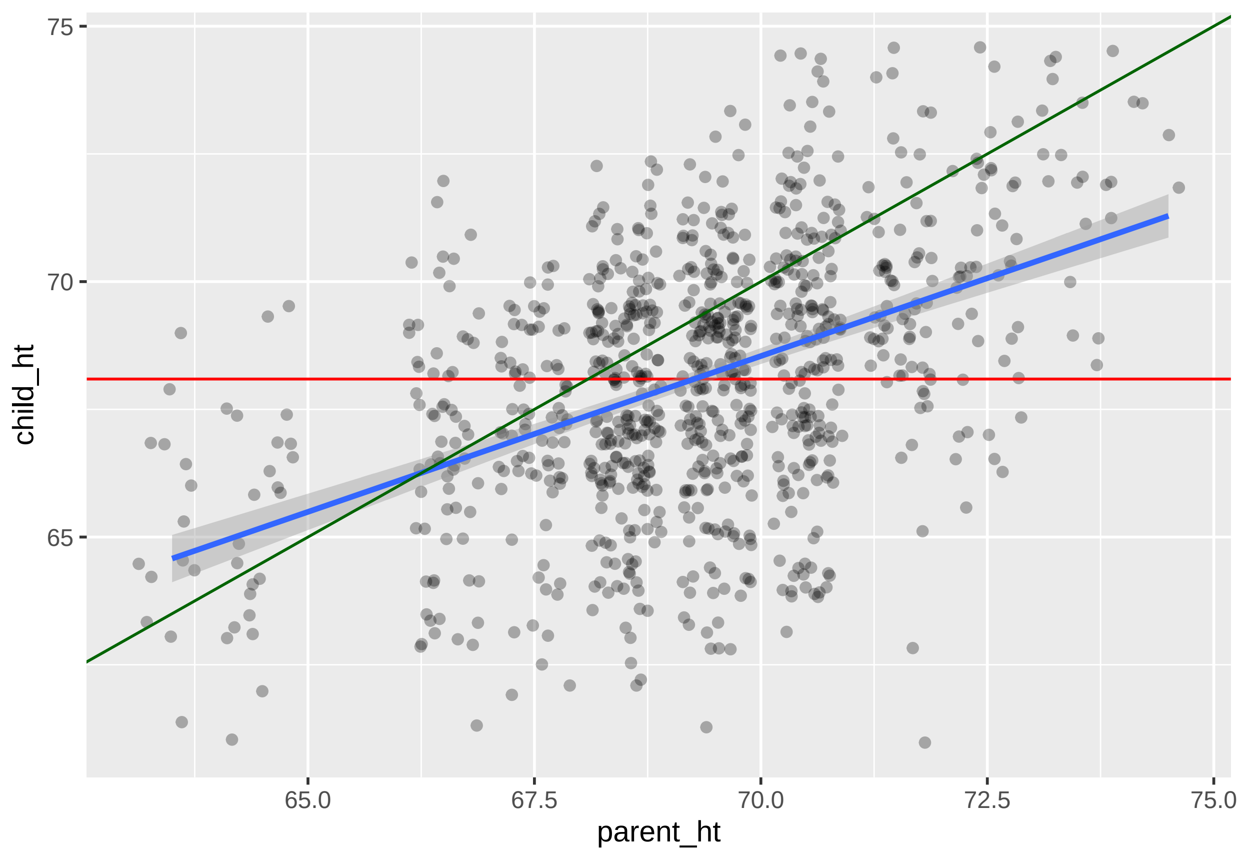


V2) Draw the following lines on the scatterplot: - a horizontal line at the mean of children's heights colored in red, using geom_hline() - a regression line using geom_smooth() by setting its' *method* to be equal to "lm" - diagonal line at the points where the parents' heights are the same as the children's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

```
child_mean = mean(child$child_ht)

ggplot(child_parent_df, aes(x = parent_ht, y = child_ht)) +
  geom_jitter(alpha = 0.3) +
  geom_hline(yintercept = child_mean, color = "red") +
  geom_smooth(method = 'lm') +
  geom_abline(color = "darkgreen")
```

```
## `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

```
empty <- lm(child_ht~NULL, data = child)
print(empty)
```

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

```
summary(empty)
```

```
##
## Call:
## lm(formula = child_ht ~ NULL, data = child)
```

```
##
## 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

```
parent2child <- lm(child_ht~parent_ht, data = child_parent_df)
print(parent2child)
```

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

```
summary(parent2child)
```

```
##
## Call:
## lm(formula = child_ht ~ parent_ht, data = child_parent_df)
##
## 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
library(broom)

tidy(empty)
```

```
## # 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
tidy(parent2child)
```

```
## # 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
sample_parent <- tibble(
  parent_ht = c(40, 64, 75)
)

predict(parent2child, sample_parent)
```

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

```r
predict(empty, sample_parent)
```

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

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

```r
x_predict <- tibble(
  parent_ht = seq(50, 100, by = 2) #parents' heights for which you are predicting childrens' heights
)
### Actually I'm not sure about the line above?? Should I put 'parent_ht' for the variable here

y_predict_null<- predict(empty, x_predict)

y_predict_one_explanatory_variable<- predict(parent2child, 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.

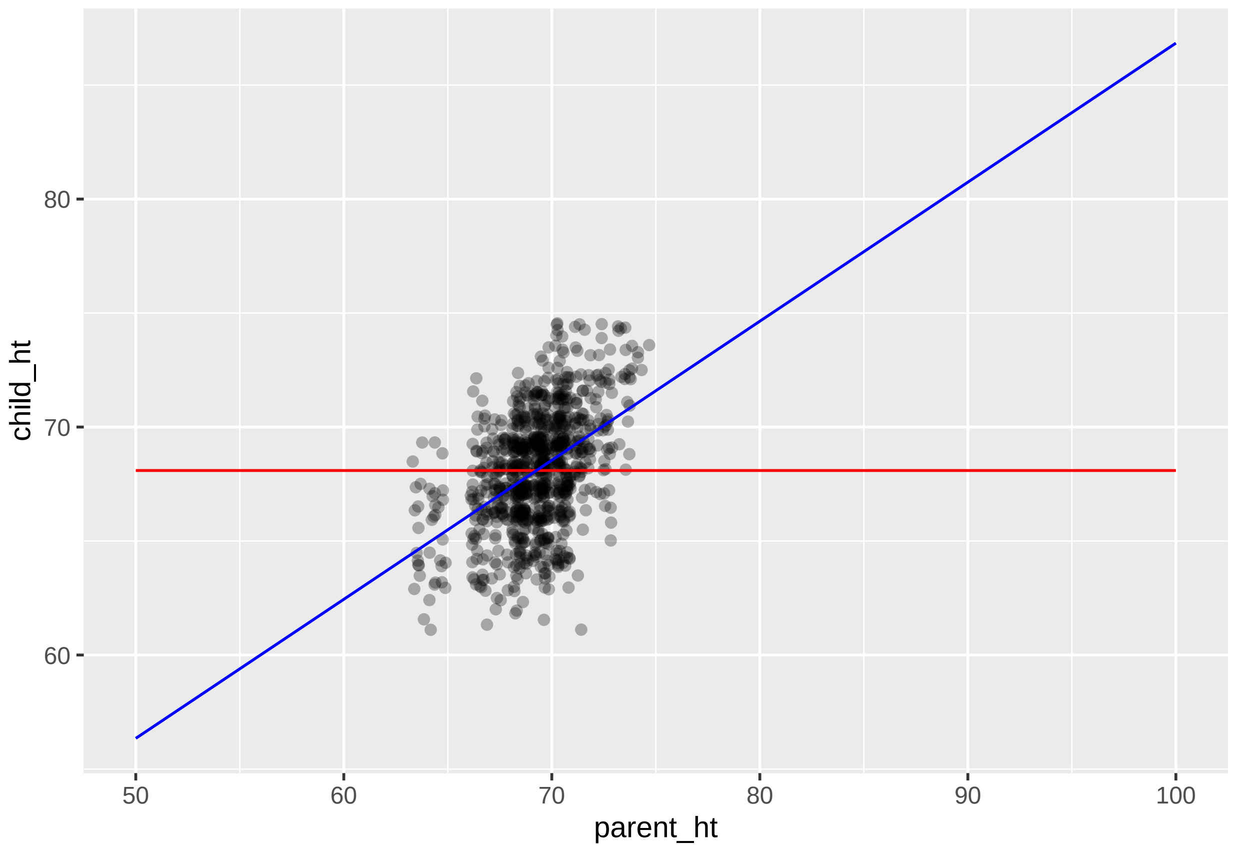
```
#two dataframes you'd use for the two geom_lines
df_predictions_null<- x_predict %>%
                bind_cols(y_predict_null)
```

```
## 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(child_parent_df, aes(x = parent_ht, y = child_ht)) +
  geom_jitter(alpha = 0.3) +
  geom_line(data = df_predictions_null, aes(x = parent_ht, y = y_predict_null), color = "red") +
  geom_line(data = df_predictions_explanatory, aes(x = parent_ht, y = y_predict_one_explanatory_variabl
```



R8) Why is this way of plotting predictions generally more useful than the method we had used so far?

Answer: The prediction line was only based on observed data points before but the new line can generalize to non existing points.

R9) predict() function is not as special as it seems. Create your own predict functions that uses the coefficients you extracted in (R3) 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.

```
my_predict_null <- function(b0, x){
  return(rep(b0, length(x)))
}

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

my_predict_null(68.093, c(40, 64, 75))
```

```
## [1] 68.093 68.093 68.093
```

```
my_predict_one_explanatory_variable(b0 = 25.849,
                                    b1 = 0.61,
                                    x = c(40, 64, 75))
```

```
## [1] 50.249 64.889 71.599
```

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

```
glance(empty)$r.squared
```

```
## [1] 0
```

```
glance(parent2child)$r.squared
```

```
## [1] 0.2104361
```

R10) 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.

```
augment(empty, data = child)
```

```
## # A tibble: 928 x 8
##    family_id child_ht .fitted .resid    .hat .sigma    .cooksd .std.resid
##    <chr>        <dbl>   <dbl>  <dbl>   <dbl>  <dbl>      <dbl>      <dbl>
##  1 F1            72.2    68.1  4.11  0.00108   2.54 0.00282        1.62
##  2 F2            73.2    68.1  5.11  0.00108   2.54 0.00436        2.01
##  3 F3            73.2    68.1  5.11  0.00108   2.54 0.00436        2.01
##  4 F4            73.2    68.1  5.11  0.00108   2.54 0.00436        2.01
##  5 F5            68.2    68.1  0.107 0.00108   2.54 0.00000190     0.0420
##  6 F6            69.2    68.1  1.11  0.00108   2.54 0.000205       0.436
##  7 F7            69.2    68.1  1.11  0.00108   2.54 0.000205       0.436
##  8 F8            70.2    68.1  2.11  0.00108   2.54 0.000741       0.829
##  9 F9            71.2    68.1  3.11  0.00108   2.54 0.00161        1.22
## 10 F10           71.2    68.1  3.11  0.00108   2.54 0.00161        1.22
## # i 918 more rows
```

```r
augment(parent2child, data = child_parent_df)
```

```
## # 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
```

R11) 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?

Answer: I get the predictor variable when '.fitted' and '.resid' are added. I guess .fitted refers to the model's predicted value and .resid is the difference between the prediction and the actual outcome (=residual).