

SMI606: Week 8

Multiple Linear Regression

Dr. Calum Webb

Sheffield Methods Institute, the University of Sheffield.

c.j.webb@sheffield.ac.uk



The
University
Of
Sheffield.



Sign In

[Link](#)

Learning Objectives

What will I learn?

How does this week fit into my course?

By the end of this week you will:

- Be able to incorporate multiple independent variables into a linear regression model to answer more interesting and complex research questions using **R**.
- Understand how multiple linear regression works by minimising squared residuals in multidimensional space.
- Be able to use the **mutate()** function to create recoded and transformed versions of variables, which can help create models with sensible reference points.
- Be able to check for linearity, heteroscedasticity, normality of residuals, outliers/leverage points, and multicollinearity in multiple regression models.

Learning Objectives

What will I learn?

How does this week fit into my course?

- By the end of this week, you will have gained the skills in quantitative research necessary to complete one of the possible options for assessment two for this module, and for a quantitative dissertation (if you wish to pursue one).
- This week will give you a solid foundation for extensions of the multiple linear regression model, including logistic regression (week 9), Poisson/negative binomial regression (advanced quant), and multilevel regression (advanced quant).

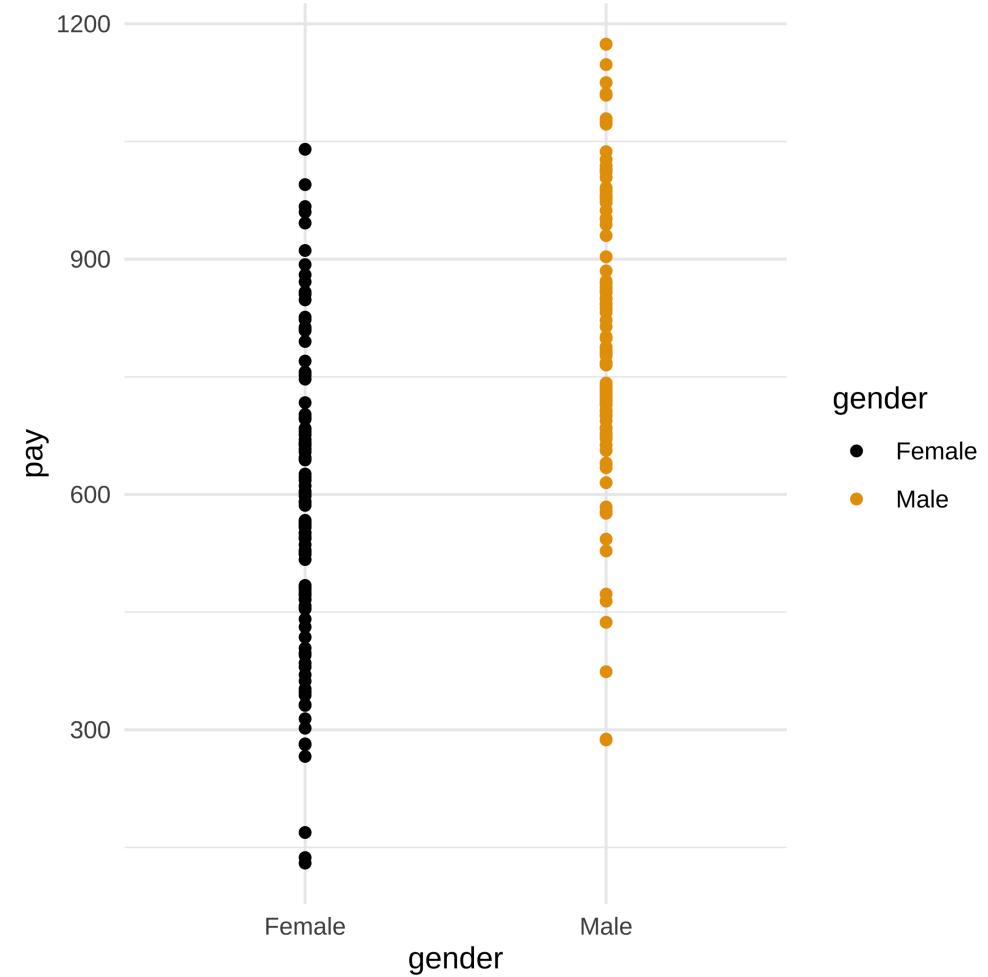
Are men paid more than women?

According to the Labour Force Survey 2018, among men and women earning less than £1,500 per week, the average (mean) gross weekly pay for men was £659, whereas the average weekly gross pay for women was £423. (-£236 difference)

Exploring the Gender Pay Gap with Multiple Regression

```
pay_data %>%
  group_by(gender) %>%
  summarise(mean_pay = mean(pay, na.rm = TRUE),
            median_pay = median(pay, na.rm = TRUE))

## # A tibble: 2 × 3
##   gender  mean_pay median_pay
##   <chr>     <dbl>      <dbl>
## 1 Female    582.       564
## 2 Male      804.       782
```



Exploring the Gender Pay Gap with Multiple Regression

```

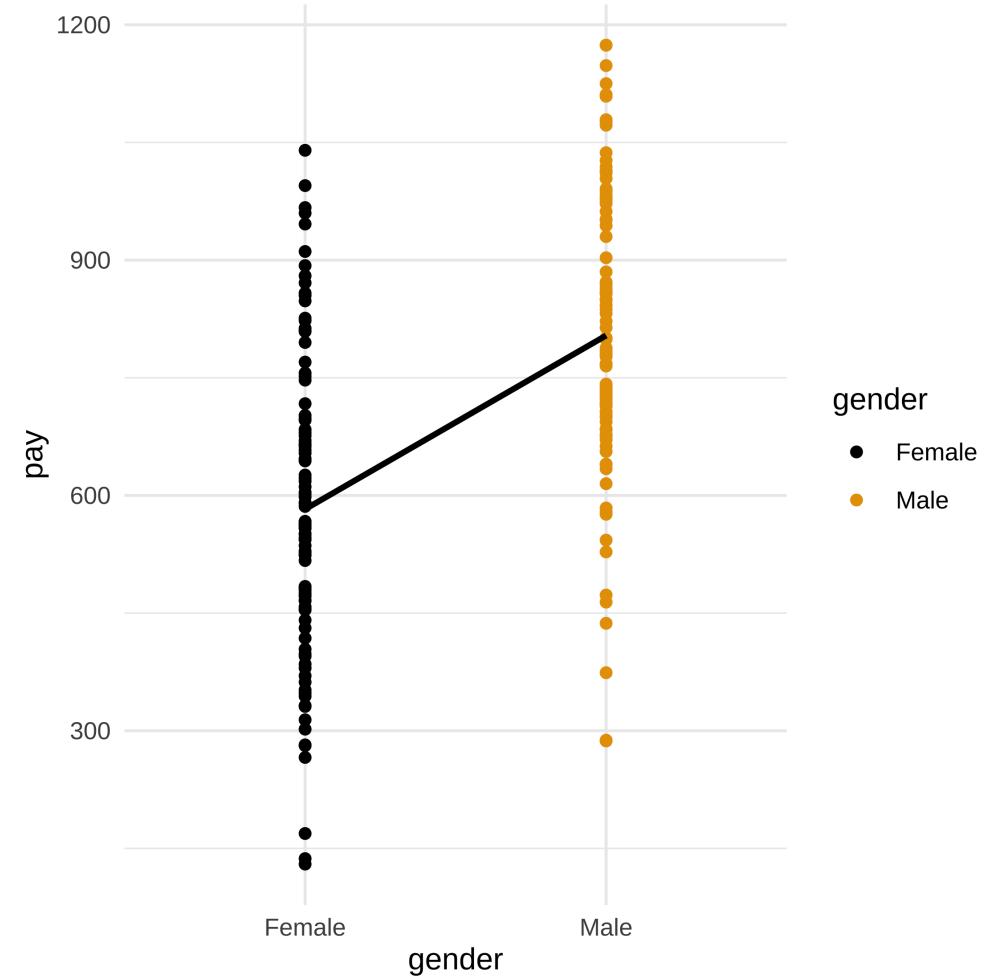
pay_data %>%
  group_by(gender) %>%
  summarise(mean_pay = mean(pay, na.rm = TRUE),
            median_pay = median(pay, na.rm = TRUE))

## # A tibble: 2 × 3
##   gender mean_pay median_pay
##   <chr>    <dbl>      <dbl>
## 1 Female     582.       564
## 2 Male       804.       782

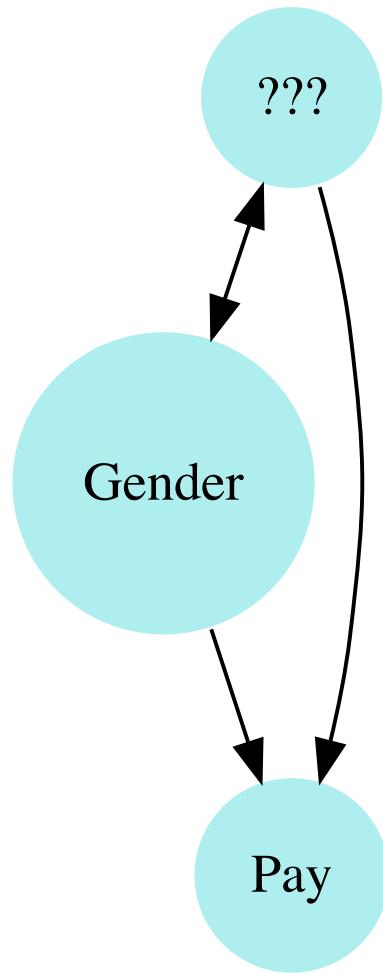
model_1 <- lm(data = pay_data, formula = pay ~ gender)
summary(model_1)

##
## Call:
## lm(formula = pay ~ gender, data = pay_data)
##
## Residuals:
##    Min     1Q     Median     3Q     Max 
## -516.88 -116.45   -19.66   128.23   457.55 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 582.45    18.75  31.067 < 2e-16 ***
## genderMale  221.42    27.06   8.183 3.32e-14 ***
## ---        
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 191.2 on 198 degrees of freedom
## Multiple R-squared:  0.2527, Adjusted R-squared:  0.2489 
## F-statistic: 66.95 on 1 and 198 DF,  p-value: 3.323e-14

```

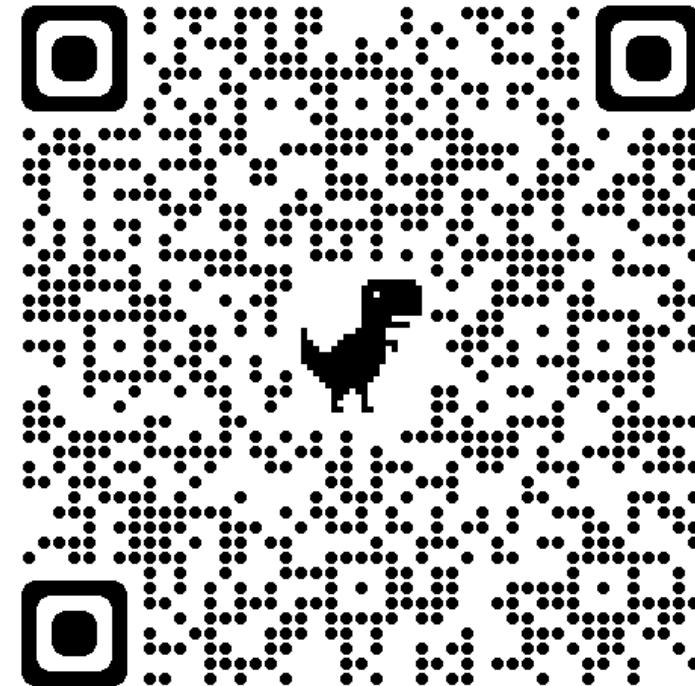


Exploring the Gender Pay Gap with Multiple Regression



[Link to Jamboard](#)

What other variables could be confounding the relationship between gender and pay?



Exploring the Gender Pay Gap with Multiple Regression

Bivariate Linear Regression

```
bivar_model <- lm(data = pay_data,  
                   formula = pay ~ gender)  
  
summary(bivar_model)
```

Multiple Linear Regression

```
mult_model <- lm(data = pay_data,  
                   formula = pay ~ gender + hrs_worked)  
  
summary(mult_model)
```

Exploring the Gender Pay Gap with Multiple Regression

Bivariate Linear Regression

```
bivar_model <- lm(data = pay_data,
                    formula = pay ~ gender)

summary(bivar_model)

##
## Call:
## lm(formula = pay ~ gender, data = pay_data)
##
## Residuals:
##    Min      1Q   Median     3Q    Max 
## -516.88 -116.45  -19.66  128.23  457.55 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 582.45     18.75  31.067 < 2e-16 ***
## genderMale  221.42     27.06   8.183 3.32e-14 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 191.2 on 198 degrees of freedom
## Multiple R-squared:  0.2527, Adjusted R-squared:  0.2489 
## F-statistic: 66.95 on 1 and 198 DF,  p-value: 3.323e-14
```

Multiple Linear Regression

```
mult_model <- lm(data = pay_data,
                    formula = pay ~ gender + hrs_worked)

summary(mult_model)

##
## Call:
## lm(formula = pay ~ gender + hrs_worked, data = pay_data)
##
## Residuals:
##    Min      1Q   Median     3Q    Max 
## -420.11 -100.48   10.53  101.20  444.89 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 122.087     47.113   2.591   0.0103 *  
## genderMale  69.524      26.352   2.638   0.0090 ** 
## hrs_worked 15.287      1.481   10.319  <2e-16 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 154.4 on 197 degrees of freedom
## Multiple R-squared:  0.5149, Adjusted R-squared:  0.51 
## F-statistic: 104.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

Exploring the Gender Pay Gap with Multiple Regression

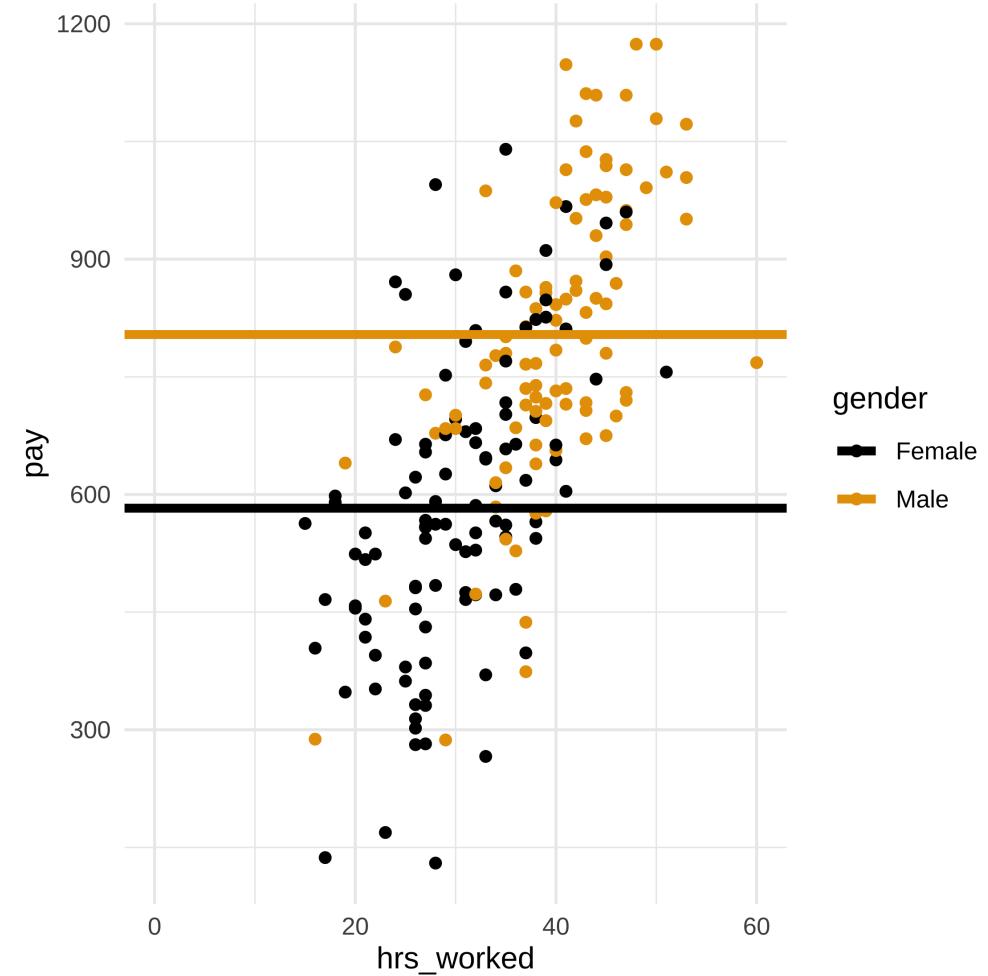
Bivariate Linear Regression

Bivariate simple regression model with just gender as a predictor.

```
bivar_model <- lm(data = pay_data,
                     formula = pay ~ gender)

summary(bivar_model)

##
## Call:
## lm(formula = pay ~ gender, data = pay_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -516.88  -116.45   -19.66   128.23   457.55
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 582.45     18.75  31.067 < 2e-16 ***
## genderMale  221.42     27.06   8.183 3.32e-14 ***
## ---
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 191.2 on 198 degrees of freedom
## Multiple R-squared:  0.2527, Adjusted R-squared:  0.2489 
## F-statistic: 66.95 on 1 and 198 DF,  p-value: 3.323e-14
```



Exploring the Gender Pay Gap with Multiple Regression

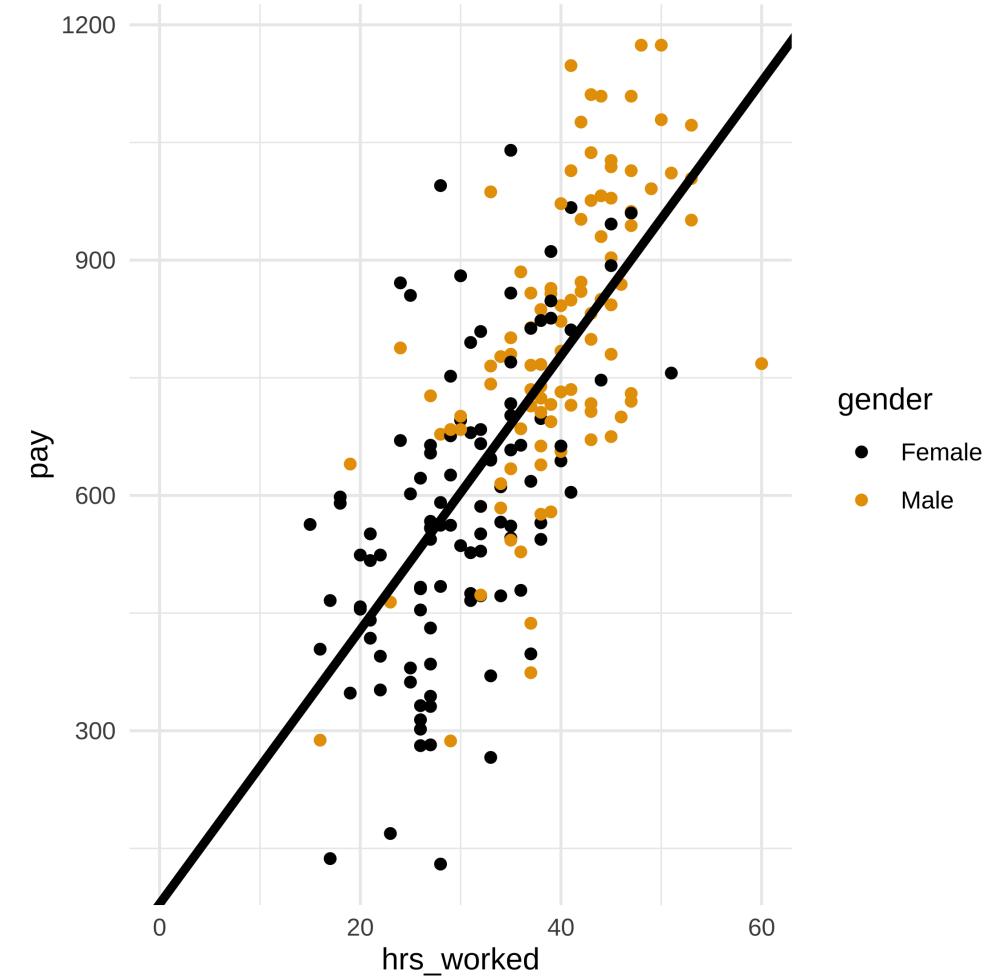
Bivariate Linear Regression

Bivariate simple regression model with just hours worked as a predictor.

```
bivar_model <- lm(data = pay_data,
                    formula = pay ~ hrs_worked)

summary(bivar_model)

##
## Call:
## lm(formula = pay ~ hrs_worked, data = pay_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -438.45 -107.38    6.46   95.57  426.55 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 79.297    44.894   1.766   0.0789 .  
## hrs_worked  17.470    1.247  14.009  <2e-16 *** 
## ---        
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 156.7 on 198 degrees of freedom
## Multiple R-squared:  0.4978, Adjusted R-squared:  0.4952 
## F-statistic: 196.2 on 1 and 198 DF,  p-value: < 2.2e-16
```



Exploring the Gender Pay Gap with Multiple Regression

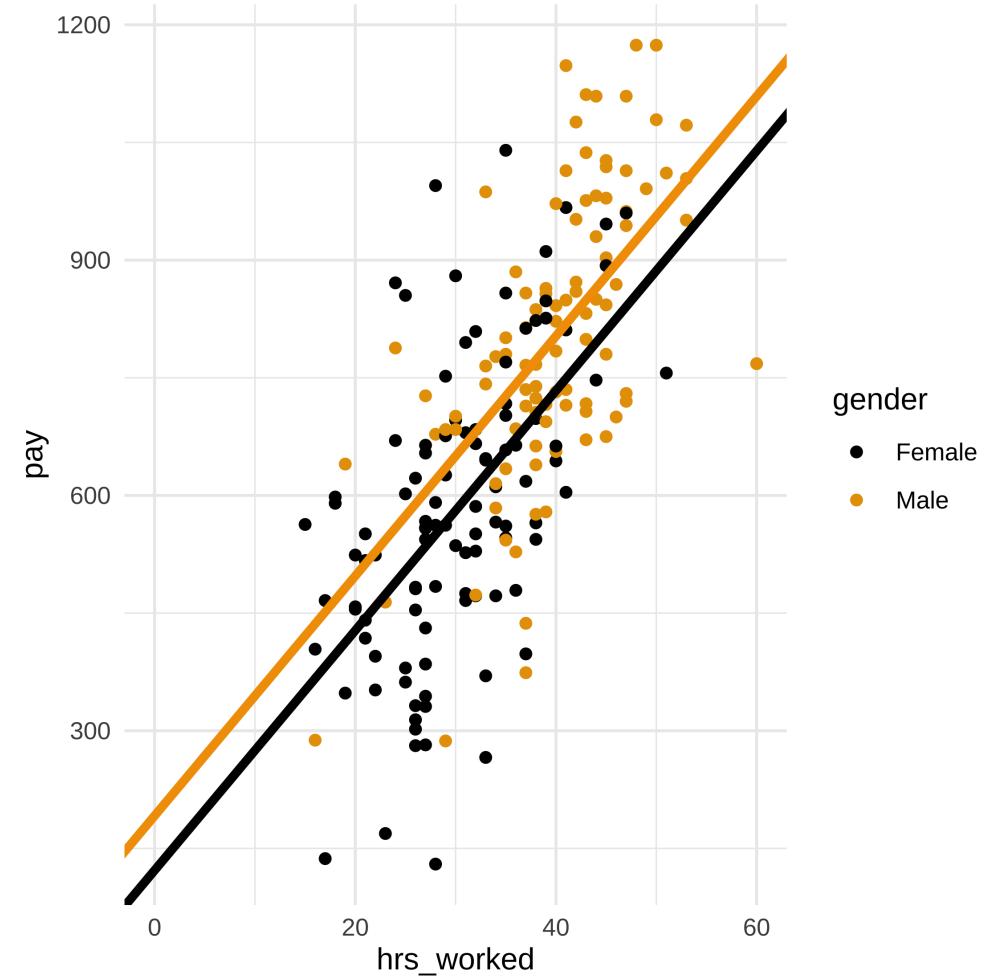
Multiple Linear Regression

Multiple linear regression model with both hours worked and gender as predictors.

```
mult_model <- lm(data = pay_data,
                   formula = pay ~ hrs_worked + gender)

summary(mult_model)

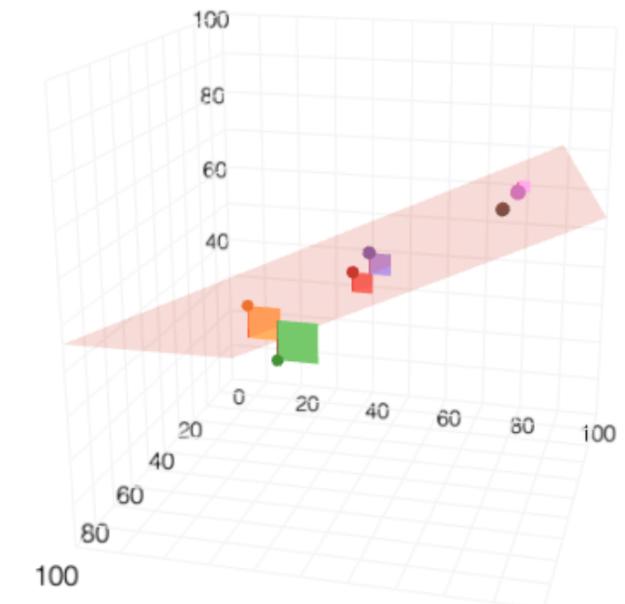
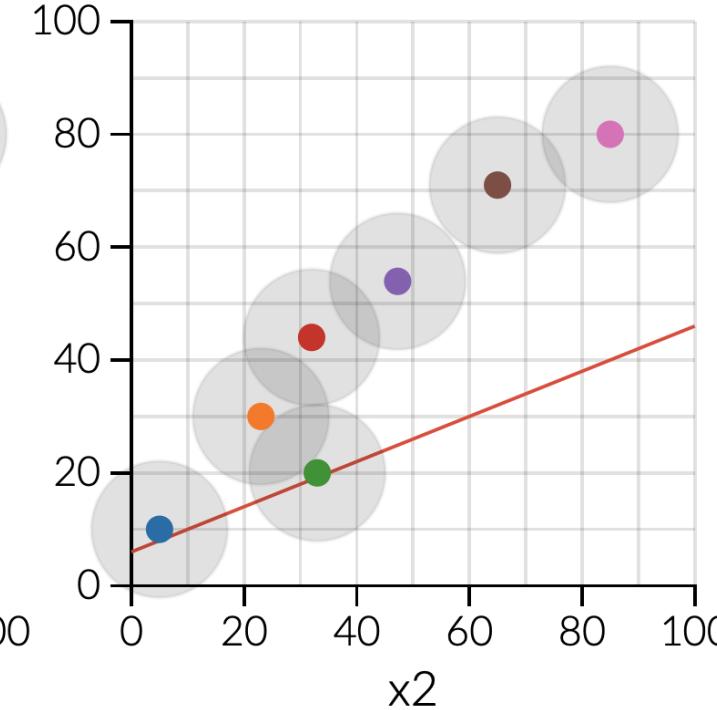
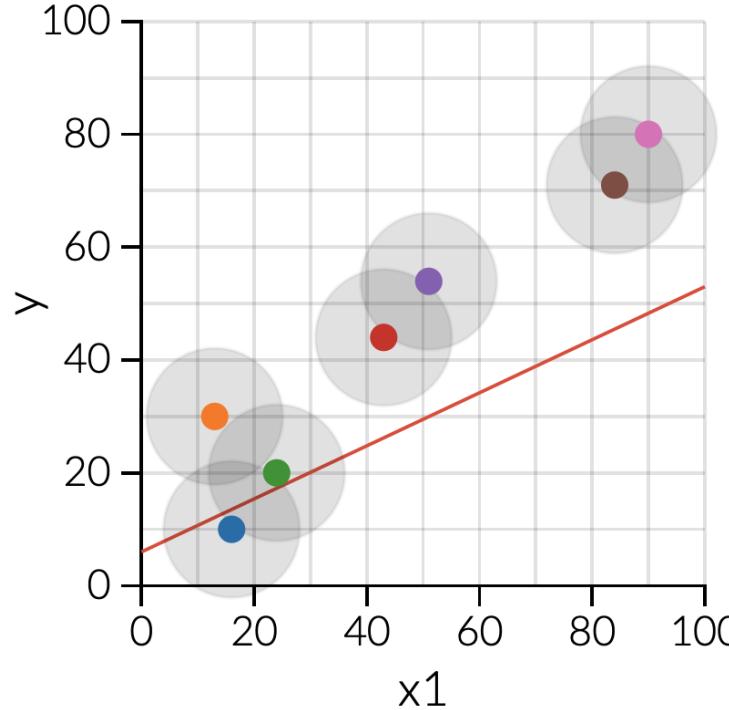
##
## Call:
## lm(formula = pay ~ hrs_worked + gender, data = pay_data)
##
## Residuals:
##   Min     1Q     Median      3Q     Max 
## -420.11 -100.48   10.53   101.20  444.89 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 122.087    47.113   2.591   0.0103 *  
## hrs_worked   15.287    1.481   10.319  <2e-16 *** 
## genderMale   69.524    26.352   2.638   0.0090 ** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 154.4 on 197 degrees of freedom
## Multiple R-squared:  0.5149, Adjusted R-squared:  0.51 
## F-statistic: 104.6 on 2 and 197 DF,  p-value: < 2.2e-16
```



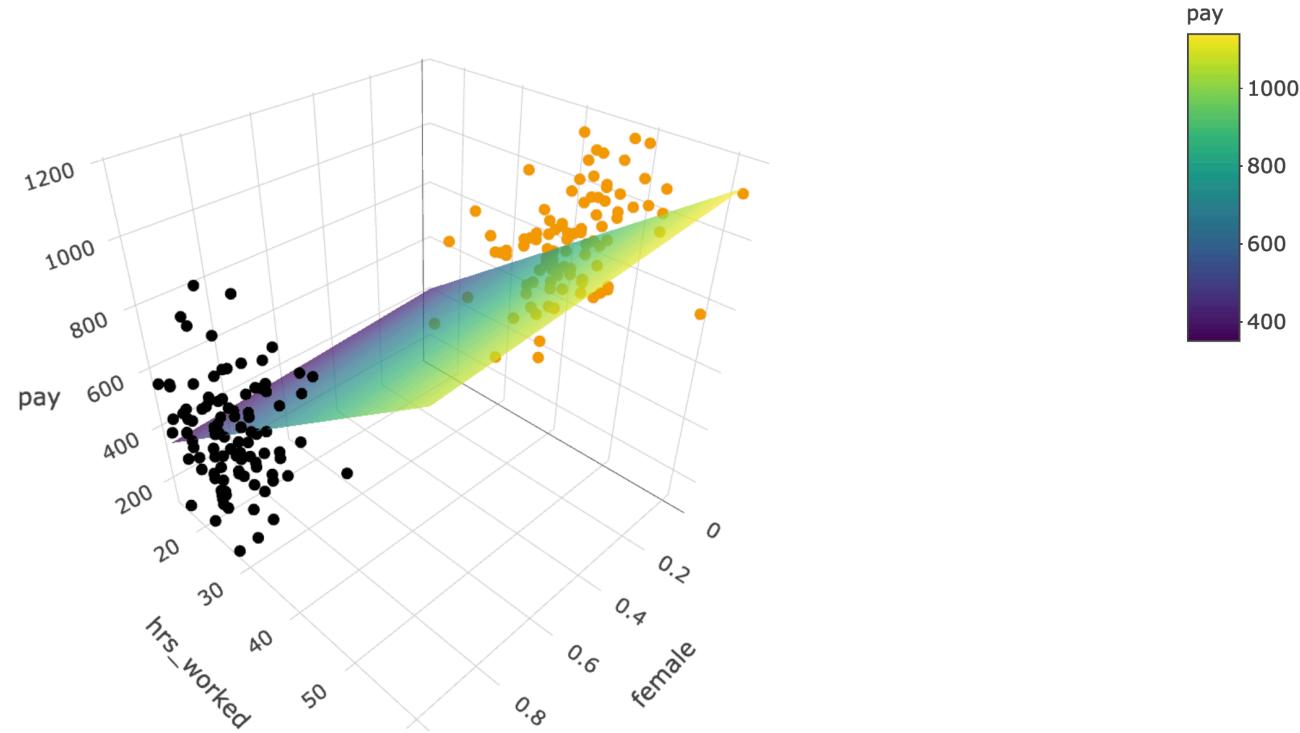
But how does it work?

Multiple Linear Regression is continuing to minimize the residuals between the predictions and the real values, but now it is considering more than two dimensional space.

Setosa Explained Visually Example



3D Visualisation



Four dimensions? Five dimensions?

In theory, we can include almost **as many independent variables as we want** in our multiple linear regression models, providing we have a large enough sample size (Degrees of Freedom = N-k-1).

This helps us answer more complex, more interesting research questions with greater nuance.

```
model_2 <- lm(data = pay_data,
                formula = pay ~ gender + hrs_worked + public_pr_ch
summary(model_2)
```

```
## 
## Call:
## lm(formula = pay ~ gender + hrs_worked + public_pr_chr, data = pay_dat
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -428.22 -101.66   15.29   97.04  436.78 
## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             137.319   49.370   2.781  0.00594 ***
## genderMale              67.644   26.411   2.561  0.01118 *  
## hrs_worked               15.032    1.502  10.010 < 2e-16 ***
## public_pr_chrPublic   -30.246   29.344  -1.031  0.30394  
## ---                     
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 154.4 on 196 degrees of freedom
## Multiple R-squared:  0.5175, Adjusted R-squared:  0.5101 
## F-statistic: 70.08 on 3 and 196 DF,  p-value: < 2.2e-16
```

Recoding and Transforming Variables with `mutate()`

Recode = Reordering, grouping, or ungrouping categories

Transform = Applying a mathematical function to convert a continuous value

Dummy variables (binary)

We often need to do some tidying of our data to make sure the predictors we include in our models are appropriate (e.g. either continuous or binary categorical, or do not break any assumptions).

```
pay_data
```

```
## # A tibble: 200 x 6
##   gender hrs_worked    pay public_pr_chr qualz    happy
##   <chr>     <dbl> <dbl> <chr>        <chr>    <chr>
## 1 Female      27    331 Private       No qualz Happy
## 2 Male        51   1011 Private       No qualz Happy
## 3 Female      26    483 Private       No qualz Happy
## 4 Female      37    398 Private      GCSE      Happy
## 5 Male        28    678 Private      University Happy
## 6 Male        37    714 Private      GCSE      Very happy
## 7 Female      24    871 Private      GCSE      Very happy
## 8 Female      27    567 Public       University Neutral
## 9 Female      40    644 Private      GCSE      Happy
## 10 Male       39    857 Private       No qualz Neutral
## # ... with 190 more rows
```

Dummy variables (binary)

When we use a binary variable in a regression model, we **always have to have a reference group** (the group that is zero).

```
model_2 <- lm(data = pay_data,
                 formula = pay ~ gender + hrs_worked + public_pr_ch
summary(model_2)
```

```
## 
## Call:
## lm(formula = pay ~ gender + hrs_worked + public_pr_chr, data = pay_data)
## 
## Residuals:
##      Min    1Q   Median    3Q   Max 
## -428.22 -101.66   15.29   97.04  436.78 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 137.319    49.370   2.781  0.00594 **  
## genderMale   67.644    26.411   2.561  0.01118 *   
## hrs_worked   15.032     1.502  10.010 < 2e-16 ***  
## public_pr_chrPublic -30.246   29.344  -1.031  0.30394    
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 154.4 on 196 degrees of freedom
## Multiple R-squared:  0.5175, Adjusted R-squared:  0.5101 
## F-statistic: 70.08 on 3 and 196 DF,  p-value: < 2.2e-16
```

By default, when we give **R** a character variable in a linear regression, the reference group chosen is always the first value alphabetically.

Dummy variables (binary)

When we use a binary variable in a regression model, we **always have to have a reference group** (the group that is zero).

```
model_2 <- lm(data = pay_data,
                 formula = pay ~ gender + hrs_worked + public_pr_ch
summary(model_2)

##
## Call:
## lm(formula = pay ~ gender + hrs_worked + public_pr_chr, data = pay_data)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -428.22 -101.66   15.29   97.04  436.78 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 137.319   49.370   2.781  0.00594 **  
## genderMale   67.644   26.411   2.561  0.01118 *   
## hrs_worked   15.032   1.502  10.010 < 2e-16 ***  
## public_pr_chrPublic -30.246  29.344  -1.031  0.30394  
## ---    
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 154.4 on 196 degrees of freedom
## Multiple R-squared:  0.5175, Adjusted R-squared:  0.5101 
## F-statistic: 70.08 on 3 and 196 DF,  p-value: < 2.2e-16
```

By default, when we give **R** a character variable in a linear regression, the reference group chosen is always the first value alphabetically.

However, we may not always want this — for example, **what if we wanted our reference group to be men rather than women?**

It's useful for us to be able to manually code our binary/dummy variables, or at least manually set our reference group. This is especially true if the reference group **R** chose was **very small** (e.g. setting Arab as the reference group in an analysis of ethnic inequality, rather than White British in the UK).

Dummy variables (binary)

dplyr::case_when() chr/fct/num

dplyr's `case_when()` function can be used to create a new variable based on conditional logic.

```
pay_data <- pay_data %>%
  # Mutate creates new variables
  mutate(
    female = case_when(
      # Start by setting all missing to stay missing
      is.na(gender) ~ NA_real_,
      # Set the group of interest to be 1
      gender == "Female" ~ 1,
      # Set everything else to be the reference category
      TRUE ~ 0
    )
  )
pay_data
```



```
## # A tibble: 200 x 7
##   gender hrs_worked   pay public_pr_chr qualis    happy  female
##   <chr>     <dbl> <dbl> <chr>       <chr>    <chr>    <dbl>
## 1 Female      27   331 Private     No qualis Happy     1
## 2 Male        51  1011 Private     No qualis Happy     0
## 3 Female      26   483 Private     No qualis Happy     1
## 4 Female      37   398 Private     GCSE      Happy     1
## 5 Male        28   678 Private     University Happy     0
## 6 Male        37   714 Private     GCSE      Very happy 0
## 7 Female      24   871 Private     GCSE      Very happy 1
## 8 Female      27   567 Public      University Neutral   1
## 9 Female      40   644 Private     GCSE      Happy     1
## 10 Male       39   857 Private     No qualis Neutral   0
## # ... with 190 more rows
```

Dummy variables (binary)

dplyr::case_when() chr/fct/num

Using the new `female` variable now sets the reference group to all men in the sample.

```
model_before <- lm(data = pay_data,
                     formula = pay ~ gender + hrs_worked)

summary(model_before)

## 
## Call:
## lm(formula = pay ~ gender + hrs_worked, data = pay_data)
## 
## Residuals:
##   Min     1Q     Median      3Q     Max 
## -420.11 -100.48    10.53   101.20   444.89 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 122.087    47.113   2.591   0.0103 *  
## genderMale   69.524    26.352   2.638   0.0090 ** 
## hrs_worked   15.287    1.481   10.319  <2e-16 *** 
## --- 
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 154.4 on 197 degrees of freedom
## Multiple R-squared:  0.5149,   Adjusted R-squared:  0.51 
## F-statistic: 104.6 on 2 and 197 DF,   p-value: < 2.2e-16
```

```
model_after <- lm(data = pay_data,
                     formula = pay ~ female + hrs_worked)

summary(model_after)

## 
## Call:
## lm(formula = pay ~ female + hrs_worked, data = pay_data)
## 
## Residuals:
##   Min     1Q     Median      3Q     Max 
## -420.11 -100.48    10.53   101.20   444.89 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 191.610    61.391   3.121   0.00207 ** 
## female      -69.524    26.352  -2.638   0.00900 ** 
## hrs_worked   15.287    1.481   10.319  < 2e-16 *** 
## --- 
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 154.4 on 197 degrees of freedom
## Multiple R-squared:  0.5149,   Adjusted R-squared:  0.51 
## F-statistic: 104.6 on 2 and 197 DF,   p-value: < 2.2e-16
```

Factor reference levels for categorical/ordinal variables

`relevel()` `fct`

Alternatively, we can turn our variable into a factor (if it's not already) and use `relevel` to set a reference category. This is very useful when you have lots of categories and can't be bothered to manually recode all of the non-reference categories into groups with `case_when`.

```
pay_data <- pay_data %>%
  # Mutate creates new variables
  mutate(
    gender_fct = relevel(factor(gender), ref = "Male")
  )
pay_data

## # A tibble: 200 x 8
##   gender hrs_worked  pay public_pr_chr qual… happy   female gender_fct
##   <chr>      <dbl> <dbl> <chr>       <chr>  <chr>   <dbl> <fct>
## 1 Female      27   331 Private     No qual… Happy   1 Female
## 2 Male        51   1011 Private    No qual… Happy   0 Male
## 3 Female      26   483 Private    No qual… Happy   1 Female
## 4 Female      37   398 Private    GCSE    Happy   1 Female
## 5 Male        28   678 Private    University Happy   0 Male
## 6 Male        37   714 Private    GCSE    Very happy 0 Male
## 7 Female      24   871 Private    GCSE    Very happy 1 Female
## 8 Female      27   567 Public     University Neutral 1 Female
## 9 Female      40   644 Private    GCSE    Happy   1 Female
## 10 Male       39   857 Private    No qual… Neutral 0 Male
## # ... with 190 more rows
```

Factor reference levels for categorical/ordinal variables

`relevel()` fct

```
model_chr <- lm(data = pay_data,
                  formula = pay ~ gender + hrs_worked)

summary(model_chr)

## 
## Call:
## lm(formula = pay ~ gender + hrs_worked, data = pay_data)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -420.11  -100.48   10.53  101.20  444.89 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 122.087    47.113   2.591   0.0103 *  
## genderMale   69.524    26.352   2.638   0.0090 ** 
## hrs_worked   15.287    1.481   10.319  <2e-16 *** 
## --- 
## Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 0.1 ' ' 1 
## 
## Residual standard error: 154.4 on 197 degrees of freedom
## Multiple R-squared:  0.5149, Adjusted R-squared:  0.51 
## F-statistic: 104.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

```
model_fct <- lm(data = pay_data,
                  formula = pay ~ gender_fct + hrs_worked)

summary(model_fct)

## 
## Call:
## lm(formula = pay ~ gender_fct + hrs_worked, data = pay_data)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -420.11  -100.48   10.53  101.20  444.89 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 191.610    61.391   3.121   0.00207 ** 
## gender_fctFemale -69.524    26.352  -2.638   0.00900 ** 
## hrs_worked   15.287    1.481   10.319  < 2e-16 *** 
## --- 
## Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 0.1 ' ' 1 
## 
## Residual standard error: 154.4 on 197 degrees of freedom
## Multiple R-squared:  0.5149, Adjusted R-squared:  0.51 
## F-statistic: 104.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

Ideally reference groups in models should be...

- The most common/largest category, or the most appropriate baseline categorisation
 - e.g. in a study of different attitudes towards climate change by political party affiliation, either Conservative or Labour voters would be a good reference category, whereas the Scottish Greens may not.
- A sensible middle point in an ordinal variable
 - e.g. On a scale of "Very unlikely" < "Unlikely" < "Neither likely nor unlikely" < "Likely" < "Very likely", a sensible reference point would probably be "Neither likely nor unlikely".

Grouping values

Quite often we want to group our responses into a new variable with a smaller number of factors.

This is especially the case when we have some values with a very small number of responses, or we need to create a binary version of a multiple-item variable (which we'll see next week is very useful for logistic regression!)

For example, the **happy** (how happy are you with your current pay) variable here.

```
pay_data
```

```
## # A tibble: 200 x 6
##   gender hrs_worked    pay public_pr_chr qualis     happy
##   <chr>      <dbl> <dbl> <chr>        <chr>      <chr>
## 1 Female       27    331 Private       No qualis  Happy
## 2 Male         51   1011 Private      No qualis  Happy
## 3 Female       26    483 Private      No qualis  Happy
## 4 Female       37    398 Private      GCSE       Happy
## 5 Male         28    678 Private      University Happy
## 6 Male         37    714 Private      GCSE       Very happy
## 7 Female       24    871 Private      GCSE       Very happy
## 8 Female       27    567 Public       University Neutral
## 9 Female       40    644 Private      GCSE       Happy
## 10 Male        39   857 Private      No qualis  Neutral
## # ... with 190 more rows
```

Grouping values with `case_when()`

We can use the same conditional logic to create groupings.

First, I check what all the possible values of the variable are with a `tabyl`.

```
pay_data %>% janitor::tabyl(happy)
```

```
##      happy n percent
##      Happy 60  0.300
## Neutral 62  0.310
## Unhappy 33  0.165
## very happy 28  0.140
## very unhappy 17  0.085
```

Grouping values with `case_when()`

We can use the same conditional logic to create groupings.

Then, I can use `case_when` to group these into something different.

```
pay_data <- pay_data %>%
  mutate(
    happy_bn = case_when(is.na(happy) ~ NA_character_,
                          happy == "Very happy" ~ "Happy",
                          happy == "Happy" ~ "Happy",
                          TRUE ~ "Not happy")
  )
pay_data

## # A tibble: 200 x 7
##   gender hrs_worked   pay public_pr_chr qual_s    happy  happy_bn
##   <chr>     <dbl> <dbl> <chr>       <chr>    <chr>    <chr>
## 1 Female      27   331 Private     No qual_s Happy    Happy
## 2 Male        51  1011 Private     No qual_s Happy    Happy
## 3 Female      26   483 Private     No qual_s Happy    Happy
## 4 Female      37   398 Private     GCSE     Happy    Happy
## 5 Male        28   678 Private     University Happy    Happy
## 6 Male        37   714 Private     GCSE     Very happy Happy
## 7 Female      24   871 Private     GCSE     Very happy Happy
## 8 Female      27   567 Public      University Neutral Not happy
## 9 Female      40   644 Private     GCSE     Happy    Happy
## 10 Male       39   857 Private     No qual_s Neutral Not happy
## # ... with 190 more rows
```

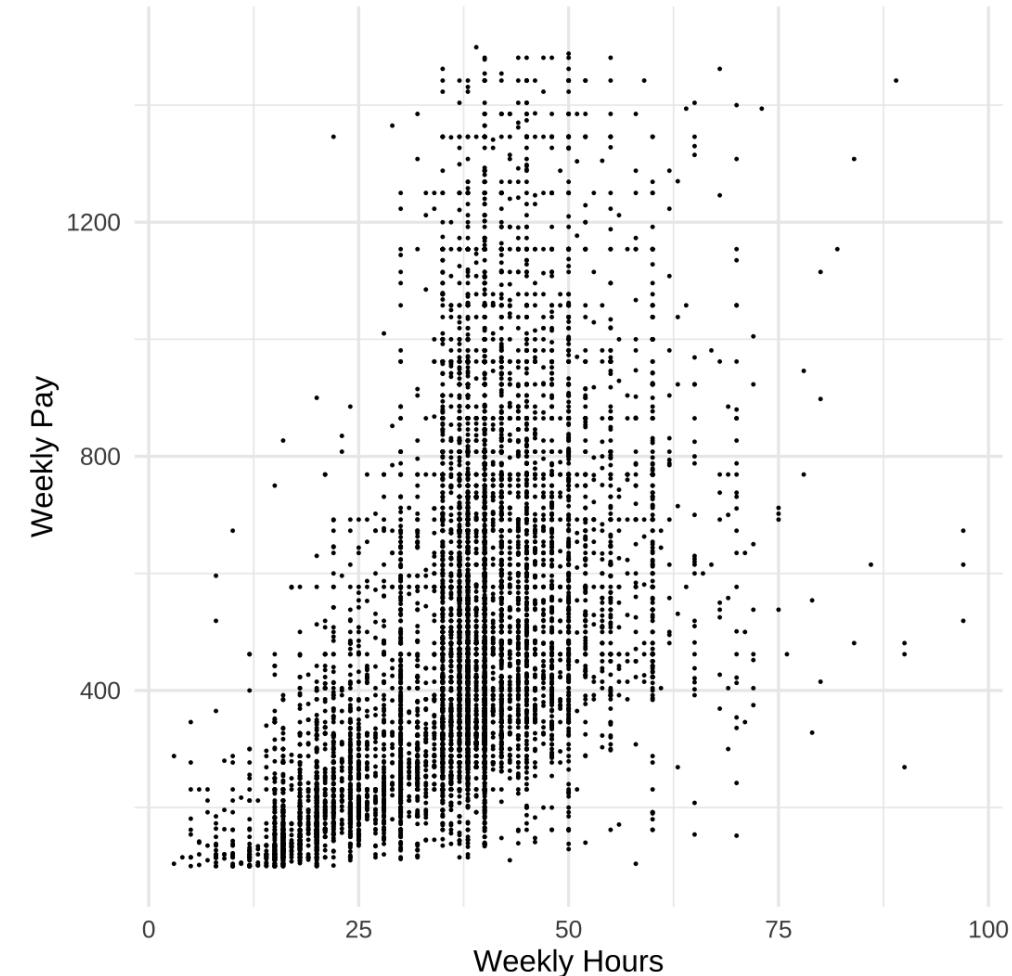
Note that a quirk with `case_when()` is that you have to specify what kind of missing the missing value is. `NA_real_` for numeric variables and `NA_character_` for character ones.

Log and Square Root Transformation

Natural log and square root transformations can be very useful when our variables are positively-skewed (right tailed). However, they do change the interpretation of our coefficients slightly: [see a short guide here](#). For example, if both X and Y are logged the interpretation becomes for a 1% increase in X what % change is there in Y.

```
lfs_real <- read_csv("lfs_sample.csv")

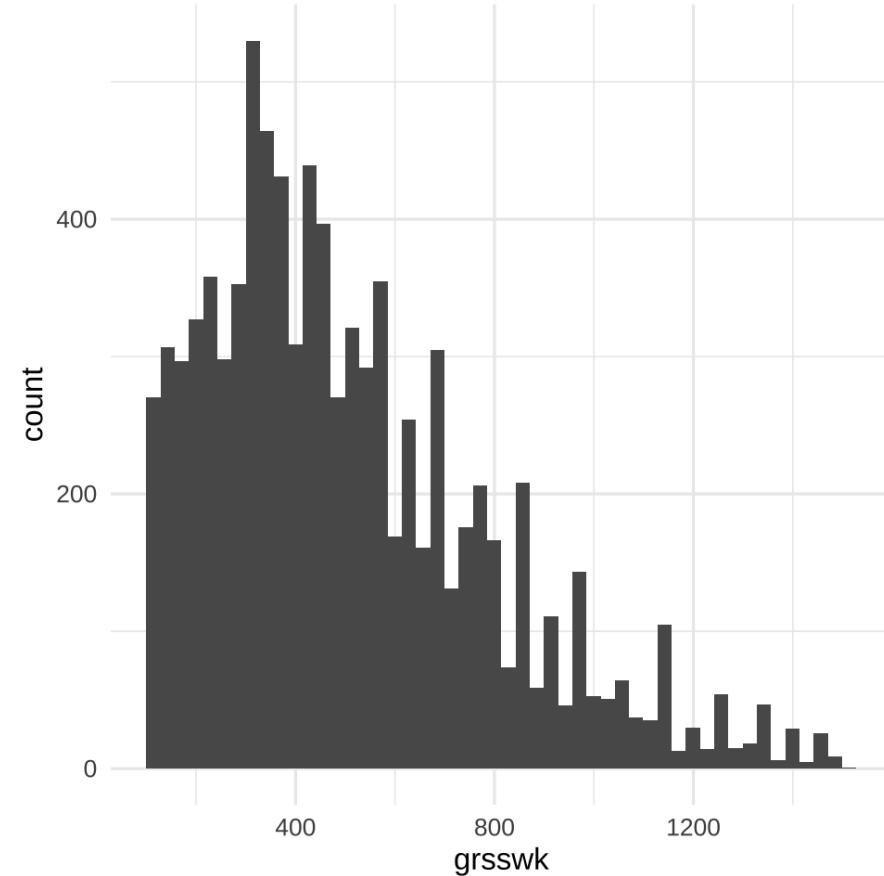
lfs_real %>%
  ggplot() +
  geom_point(aes(x = ttushr, y = grsswk), size = 0.1) +
  ylab("Weekly Pay") +
  xlab("Weekly Hours") +
  theme_minimal()
```



Log and Square Root Transformation

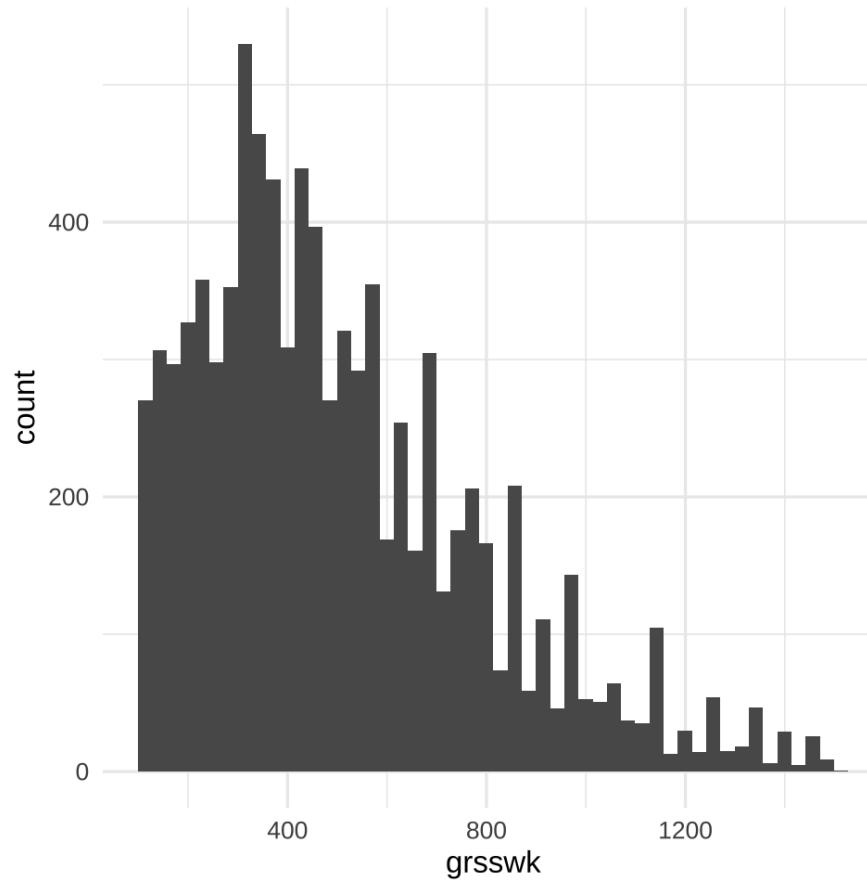
Natural log and square root transformations can be very useful when our variables are positively-skewed (right tailed). However, they do change the interpretation of our coefficients slightly: [see a short guide here](#). For example, if both X and Y are logged the interpretation becomes for a 1% increase in X what % change is there in Y.

```
lfs_real %>%
  ggplot() +
  geom_histogram(aes(x = grsswk),
                 bins = 50) +
  theme_minimal()
```

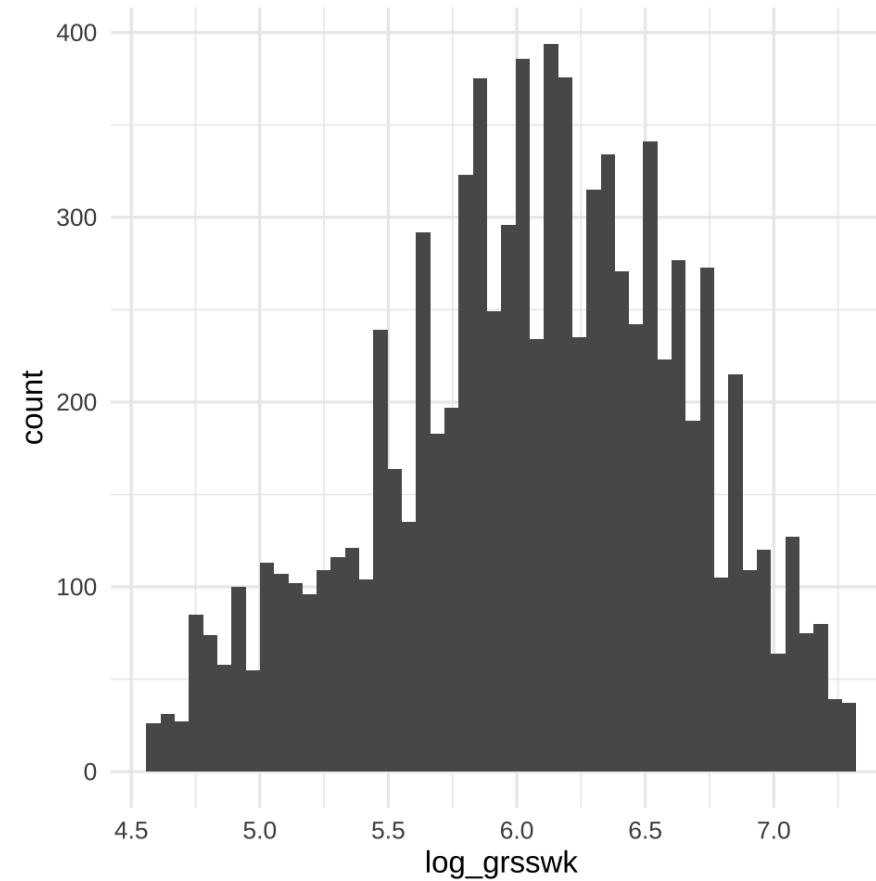


Log and Square Root Transformation

Untransformed

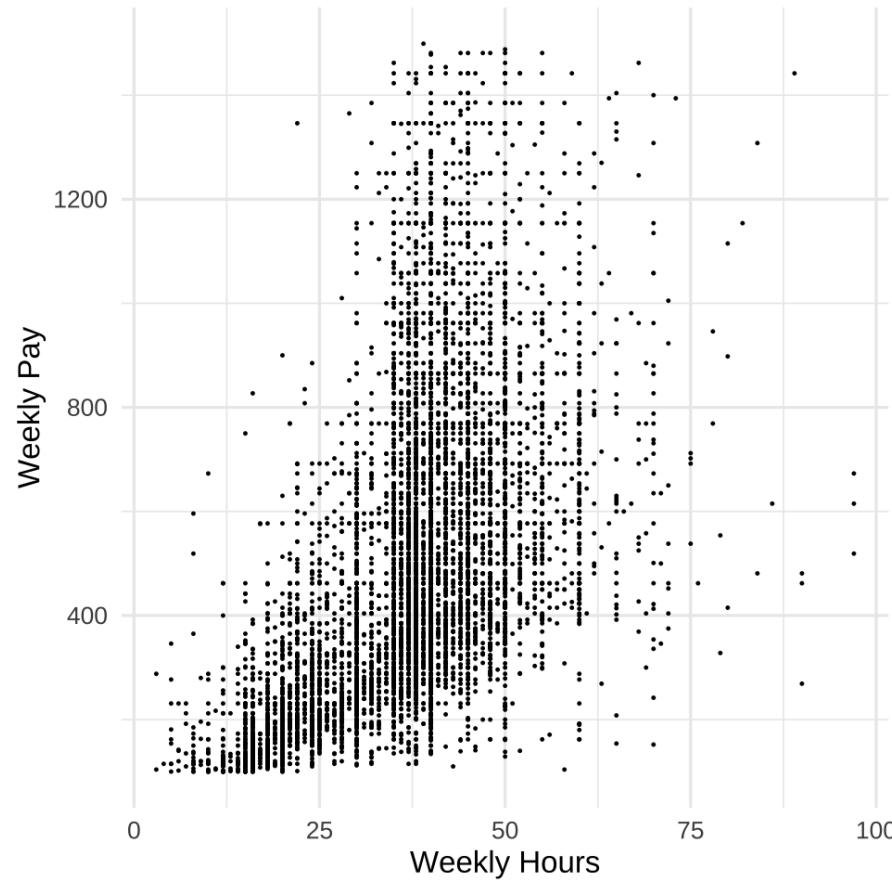


Transformed using Log

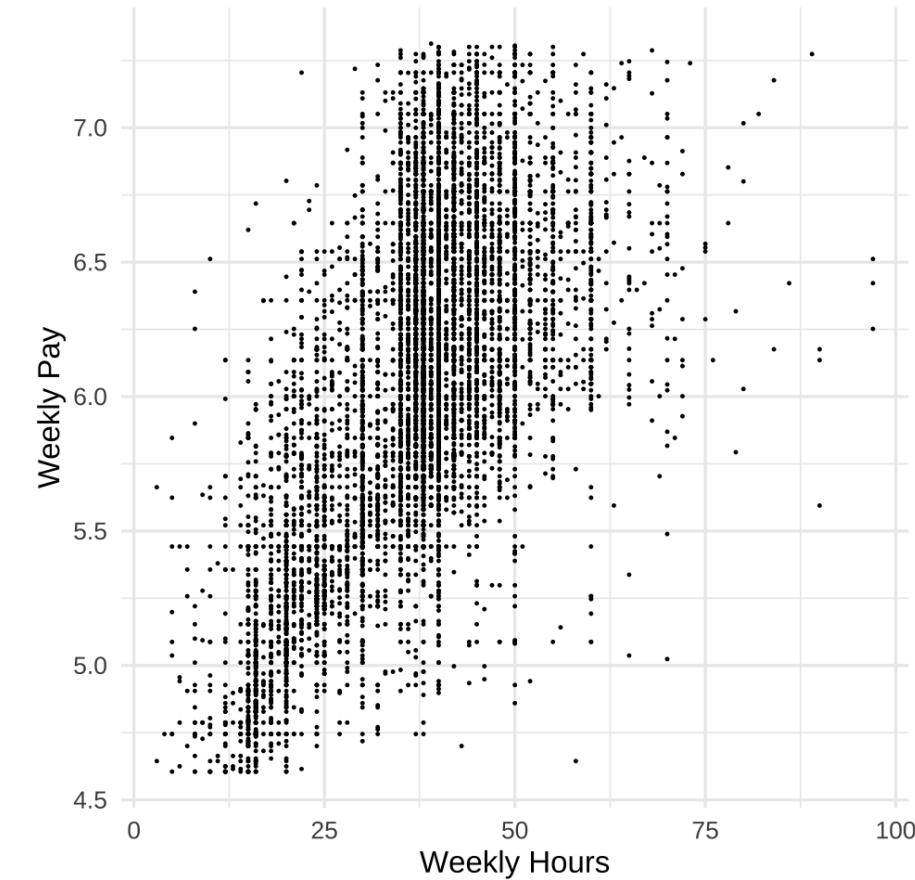


Log and Square Root Transformation

Untransformed

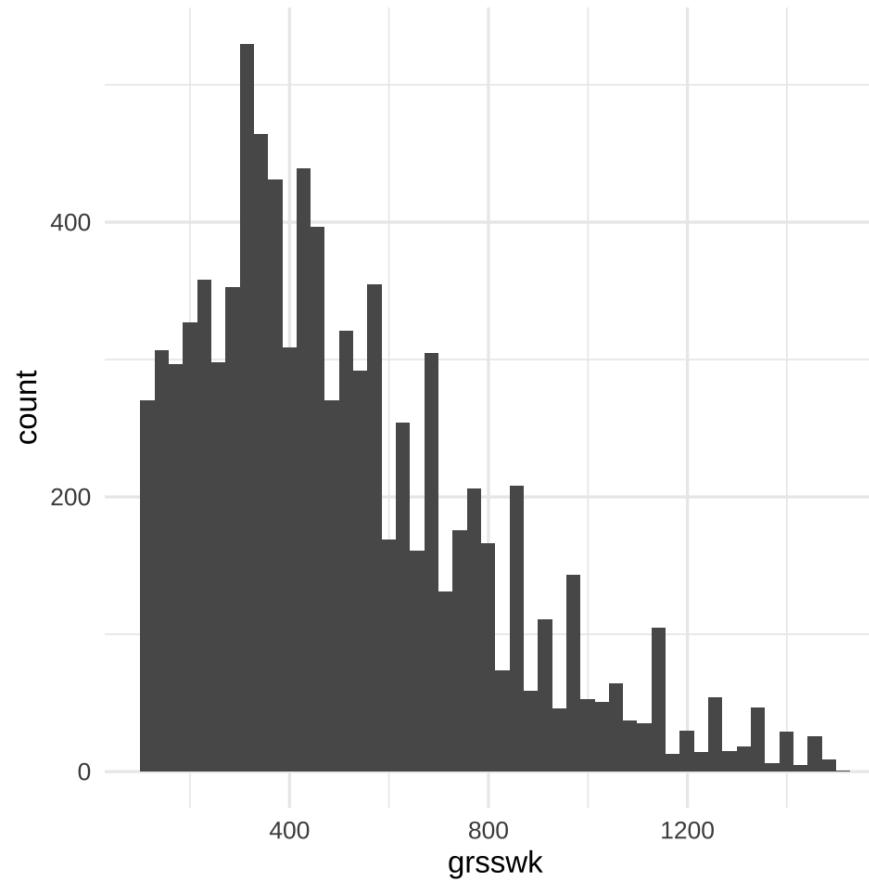


Transformed using Log

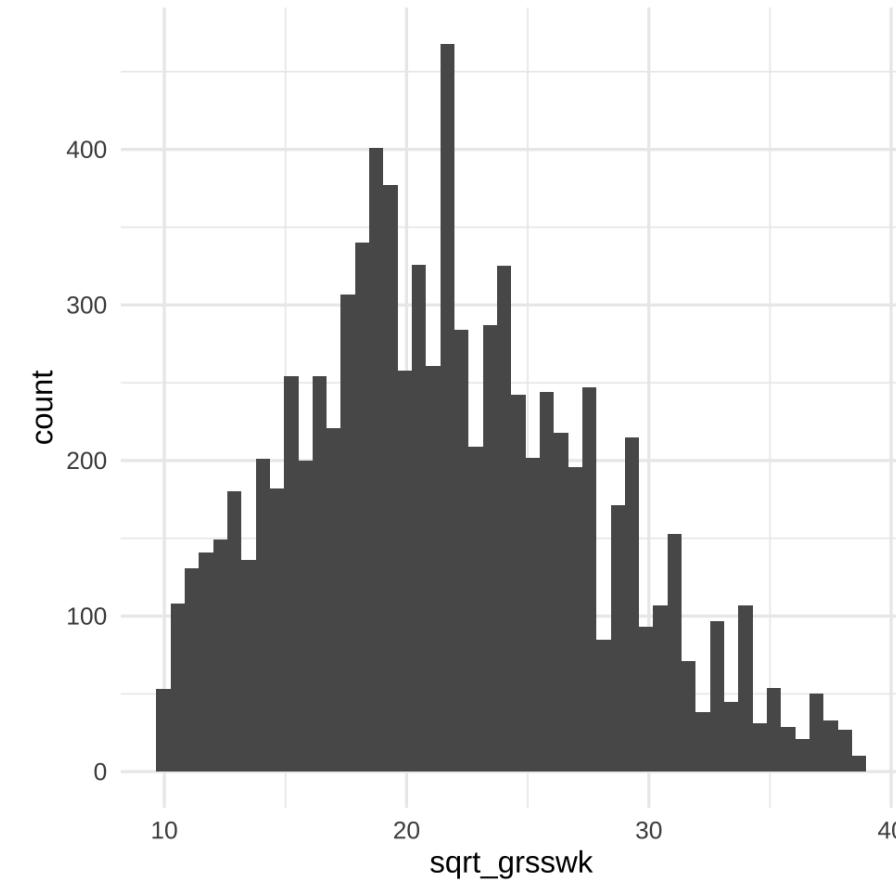


Log and Square Root Transformation

Untransformed

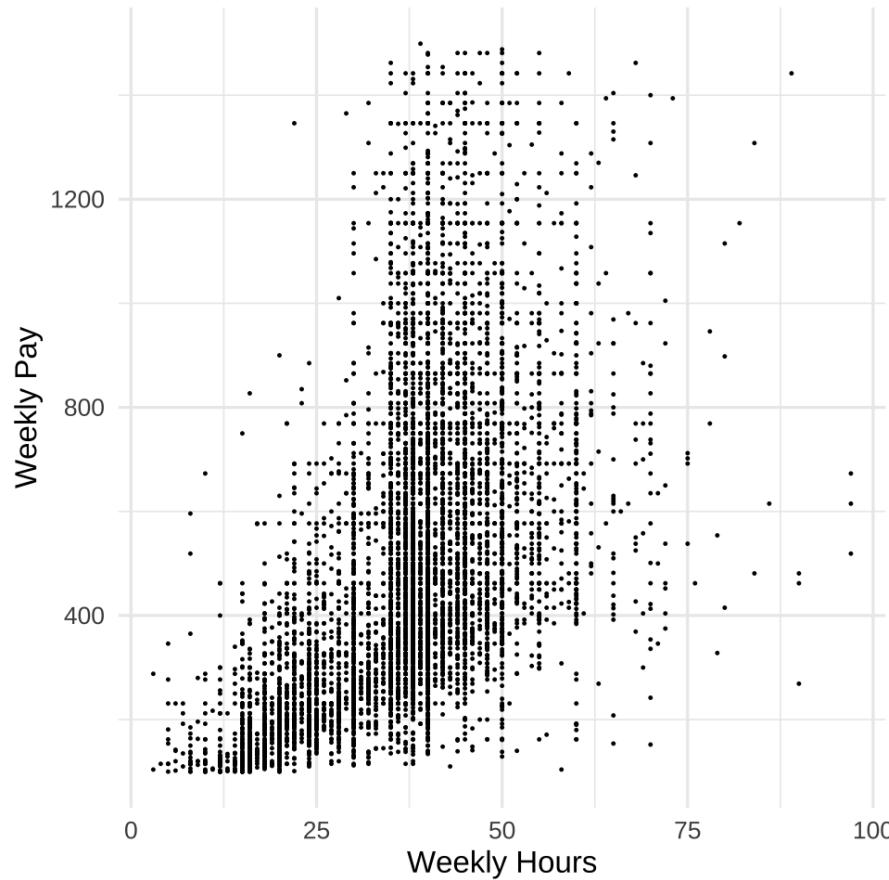


Transformed using `sqrt()`

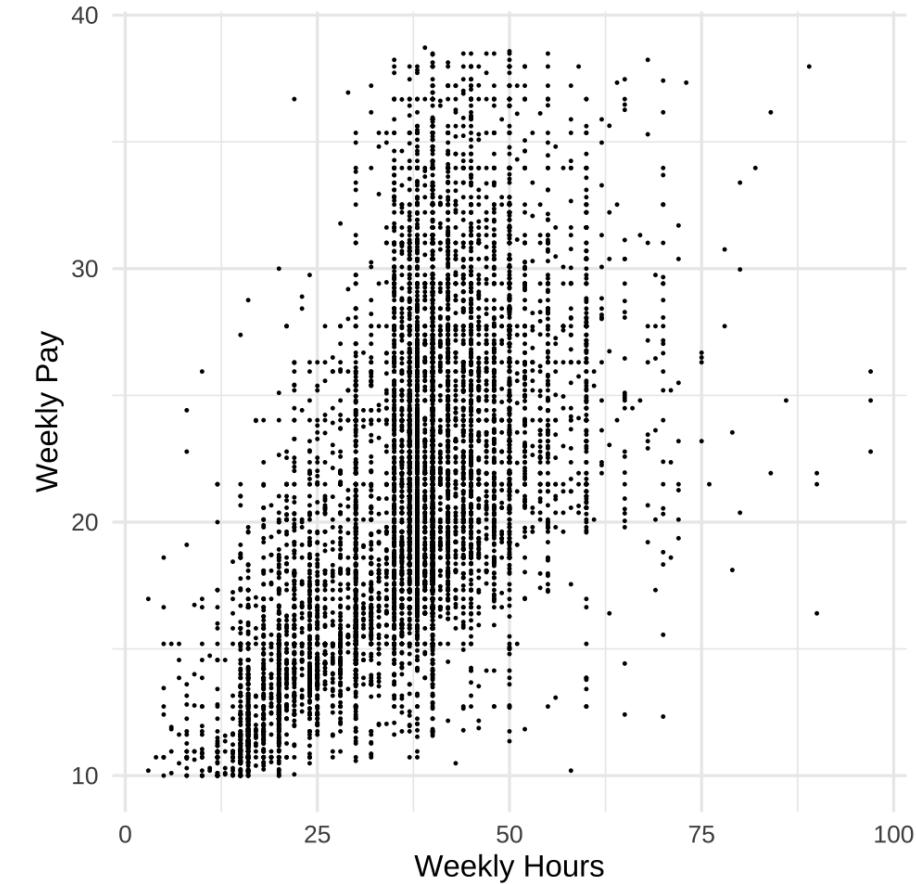


Log and Square Root Transformation

Untransformed



Transformed using `sqrt()`



Log and Square Root Transformation

```
lfs_real %>%
  mutate(
    log_grsswk = log(grsswk),
    sqrt_grsswk = sqrt(grsswk)
  )
```

Important notes on log and square root transformation

- Only numbers above 0 have a defined logarithmic value. If you try and take the log of 0 you get negative infinity, if you try and take the log of a negative number you get a **NaN** result.
- (In R), the square root of a negative number is undefined, and will give a **NaN** result. This means that any negative values transformed using **sqrt()** will become missing.
- If you need to log or take the square root of a variable that has negative or zero values, you can add a **constant** value to make all of the numbers positive (e.g. Pay + 500). Just remember to remove the constant if back-transforming.

```
lfs_real %>%
  mutate(
    log_grsswk = log(grsswk + 500),
    sqrt_grsswk = sqrt(grsswk + 500)
  )
```

Data manipulation in R (and other programming languages for that matter) is always a bit fiddly and often takes some googling!

It's always worth checking the results look like what you would expect by eyeballing the data or using a table/visualisation.



Assumptions Updated!

Last week we learned about four assumptions of linear regression.

- [Linearity](#)
- [Homoscedasticity](#)
- [Outliers and Leverage Points](#)
- [Normality of residuals](#)

However, it's no longer very easy to create a visualisation in 4, 5, or 6 dimensions to check these like we could do with our scatterplot! We now need to check multivariate linearity, multivariate homoscedasticity, etc.

Luckily, there are specialised kinds of plots to check these assumptions that are easy to use in R.

Assumptions Updated!

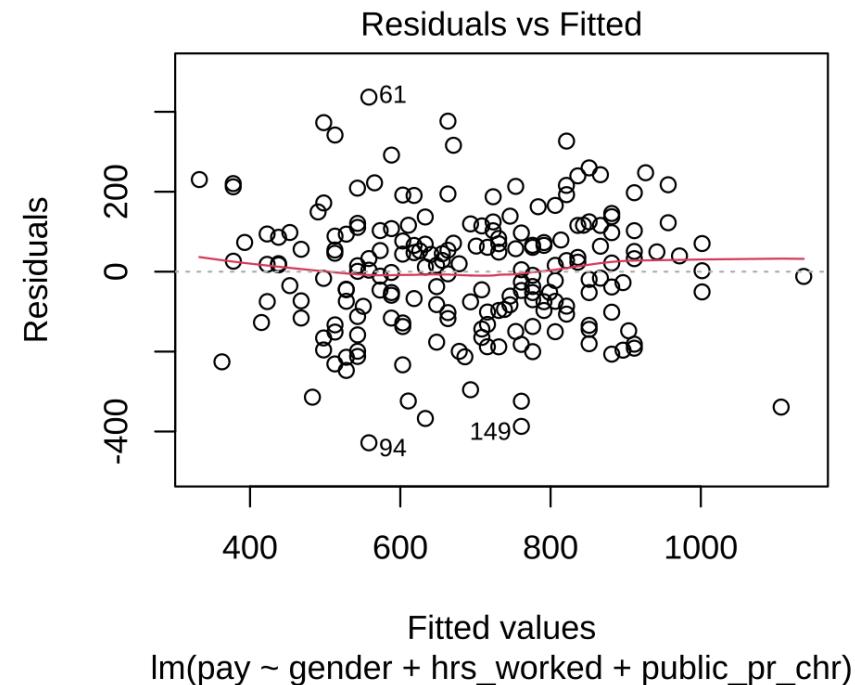
- Linearity

We can check linearity using a **residuals versus fitted values plot**. You can produce this plot in R by using `plot(model_2, which = 1)`.

If the red line roughly follows the dotted line at 0, the linear fit is close. If it diverges, a non-linear fit may be better.

- Homoscedasticity
- Outliers and Leverage Points
- Normality of residuals

```
plot(model_2, which = 1)
```



Assumptions Updated!

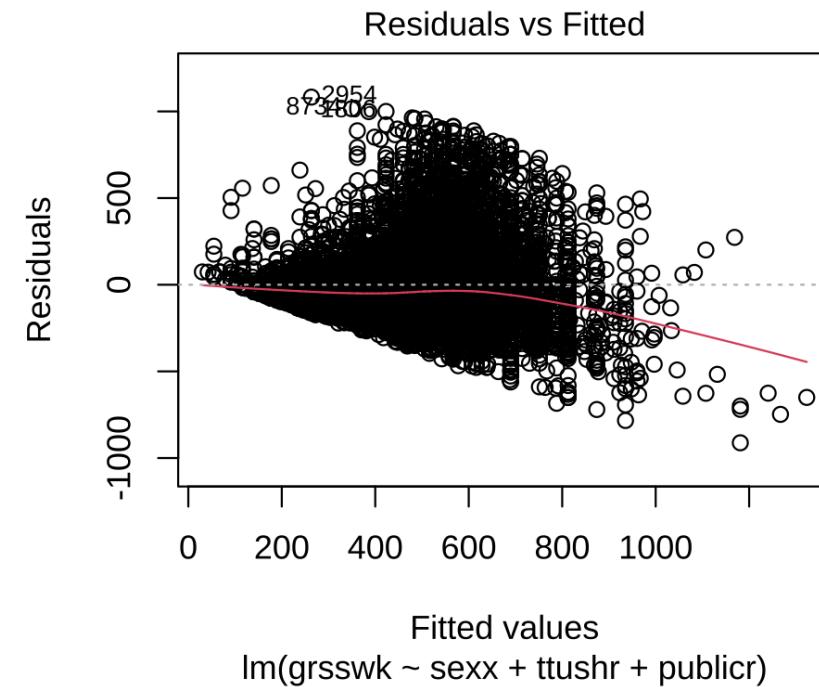
- Linearity

We can check linearity using a **residuals versus fitted values plot**. You can produce this plot in R by using `plot(model_2, which = 1)`.

If the red line roughly follows the dotted line at 0, the linear fit is close. If it diverges, a non-linear fit may be better.

- Homoscedasticity
- Outliers and Leverage Points
- Normality of residuals

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



Assumptions Updated!

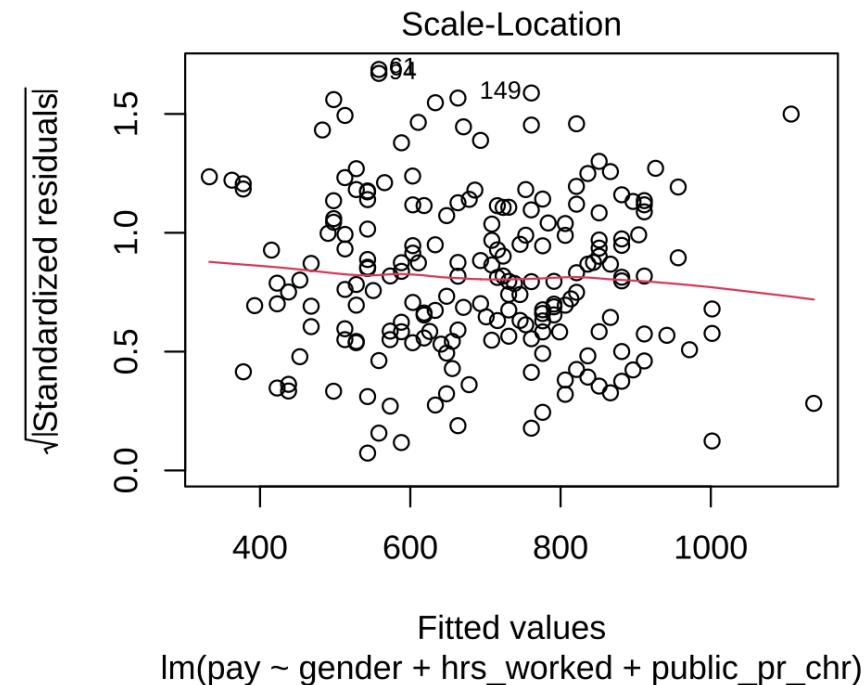
- Linearity
- Homoscedasticity

We can check for heteroscedasticity using a [spread/scale-location plot](#). You can produce this plot with `plot(model_2, which = 3)`.

The red line should be close to be horizontal, and the spread around the line should be roughly uniform (the same as we would interpret a bivariate scatterplot).

- Outliers and Leverage Points
- Normality of residuals

```
plot(model_2, which = 3)
```



Assumptions Updated!

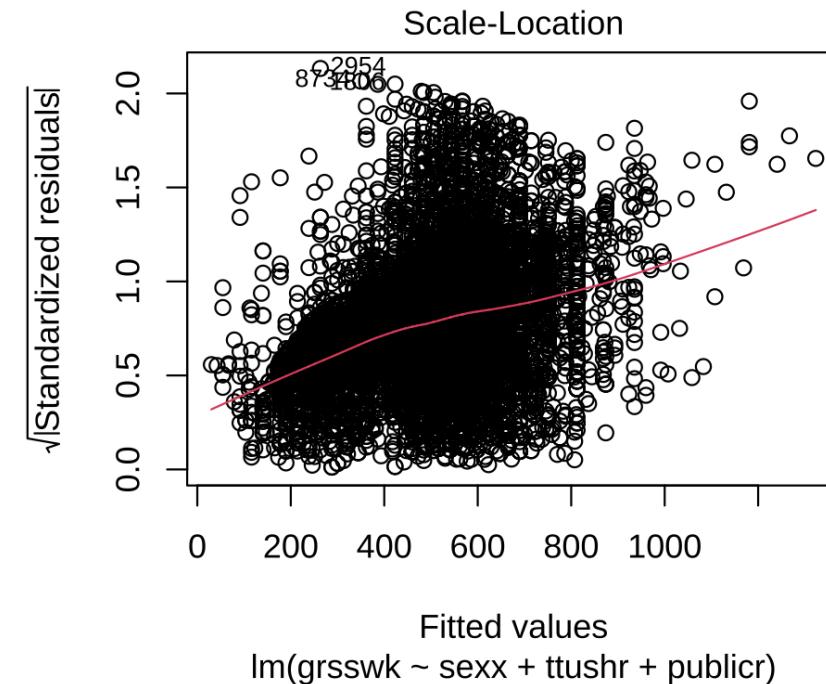
- Linearity
- Homoscedasticity

We can check for heteroscedasticity using a [spread/scale-location plot](#). You can produce this plot with `plot(model_2, which = 3)`.

The red line should be close to be horizontal, and the spread around the line should be roughly uniform (the same as we would interpret a bivariate scatterplot).

- Outliers and Leverage Points
- Normality of residuals

```
plot(model_real, which = 3)
```



Assumptions Updated!

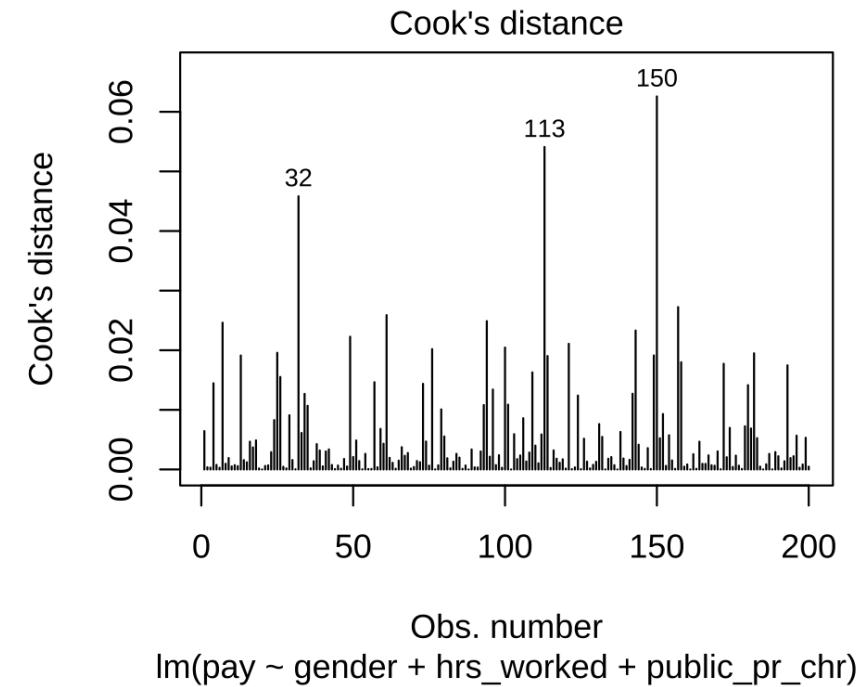
- Linearity
- Homoscedasticity
- Outliers and Leverage Points

There are a few different ways to check for multivariate outliers/leverage points, but probably the easiest one to interpret is the **residuals versus leverage plot** and the **Cook's distance** plot.

The Cook's distance plot can be generated using `plot(model_2, which = 4)`, and the residuals versus leverage plot can be generated using `plot(model_2, which = 5)`. The numbers on the points/lines refer to the **observation row number**.

- Normality of residuals

```
plot(model_2, which = 4)
```



Assumptions Updated!

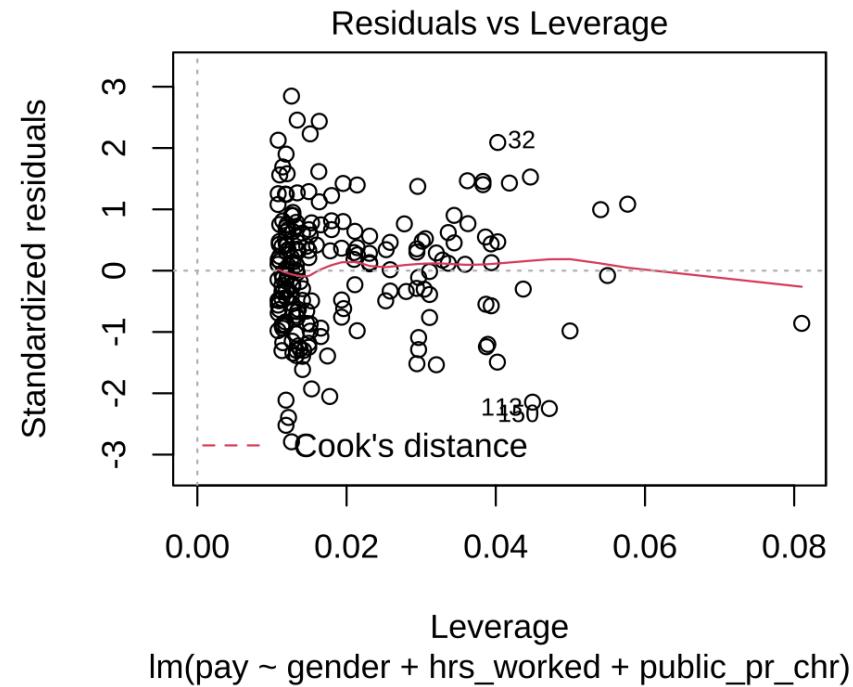
- Linearity
- Homoscedasticity
- Outliers and Leverage Points

There are a few different ways to check for multivariate outliers/leverage points, but probably the easiest one to interpret is the **residuals versus leverage plot** and the **Cook's distance** plot.

The Cook's distance plot can be generated using `plot(model_2, which = 4)`, and the residuals versus leverage plot can be generated using `plot(model_2, which = 5)`. The numbers on the points/lines refer to the **observation row number**.

- Normality of residuals

```
plot(model_2, which = 5)
```



Assumptions Updated!

- Linearity
- Homoscedasticity
- Outliers and Leverage Points

There are a few different ways to check for multivariate outliers/leverage points, but probably the easiest one to interpret is the **residuals versus leverage plot** and the **Cook's distance** plot.

The Cook's distance plot can be generated using `plot(model_2, which = 4)`, and the residuals versus leverage plot can be generated using `plot(model_2, which = 5)`. The numbers on the points/lines refer to the **observation row number**.

- Normality of residuals

Outliers can be removed by row position using the `slice()` function.

```
summary(model_2)

##
## Call:
## lm(formula = pay ~ gender + hrs_worked + public_pr_chr, data = pay_dat
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -428.22 -101.66   15.29   97.04  436.78 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 137.319   49.370   2.781  0.00594 **  
## genderMale   67.644   26.411   2.561  0.01118 *   
## hrs_worked   15.032   1.502   10.010 < 2e-16 ***  
## public_pr_chrPublic -30.246   29.344  -1.031  0.30394  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 154.4 on 196 degrees of freedom
## Multiple R-squared:  0.5175,    Adjusted R-squared:  0.5101 
## F-statistic: 70.08 on 3 and 196 DF,  p-value: < 2.2e-16
```

Assumptions Updated!

- Linearity
- Homoscedasticity
- Outliers and Leverage Points

There are a few different ways to check for multivariate outliers/leverage points, but probably the easiest one to interpret is the **residuals versus leverage plot** and the **Cook's distance** plot.

The Cook's distance plot can be generated using `plot(model_2, which = 4)`, and the residuals versus leverage plot can be generated using `plot(model_2, which = 5)`. The numbers on the points/lines refer to the **observation row number**.

- Normality of residuals

Outliers can be removed by row position using the `slice()` function.

```
pay_data_no_outliers <- pay_data %>%
  slice(-32, -113, -150)

model_3 <- lm(data = pay_data_no_outliers,
  formula = pay ~ gender + hrs_worked + public_pr_chr)

summary(model_3)
```

```
## 
## Call:
## lm(formula = pay ~ gender + hrs_worked + public_pr_chr, data = pay_dat
## 
## Residuals:
##   Min     1Q   Median     3Q    Max 
## -426.94 -101.52   12.19   95.16  438.06 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 121.182    48.848   2.481   0.014 *  
## genderMale   66.205    26.109   2.536   0.012 *  
## hrs_worked   15.563    1.494  10.417 <2e-16 *** 
## public_pr_chrPublic -29.631    29.277  -1.012   0.313  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 150.1 on 193 degrees of freedom 
## Multiple R-squared:  0.5387,   Adjusted R-squared:  0.5316 
## F-statistic: 75.13 on 3 and 193 DF,   p-value: < 2.2e-16
```

Assumptions Updated!

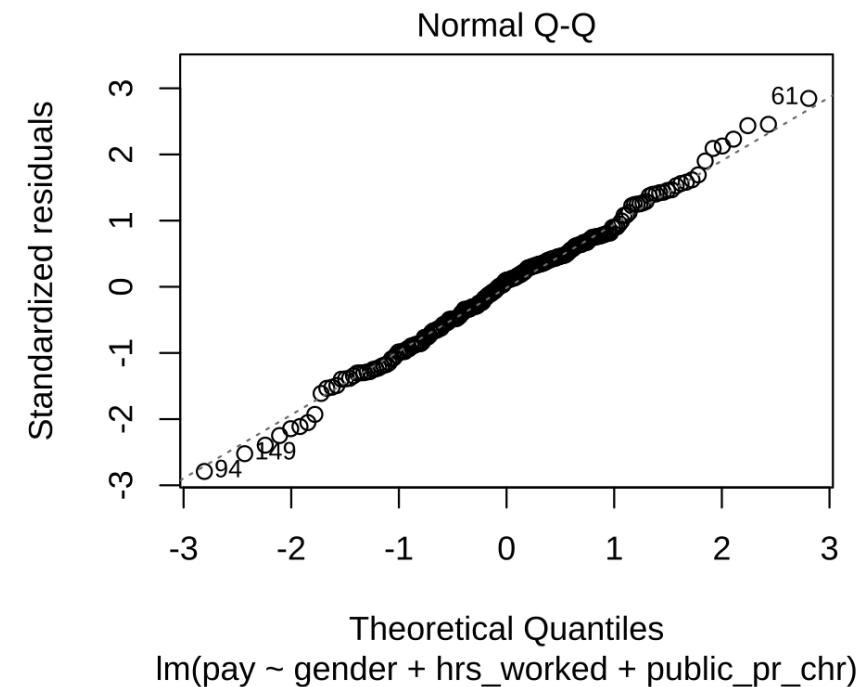
- Linearity
- Homoscedasticity
- Outliers and Leverage Points
- Normality of residuals

Finally, multivariate normality of residuals can be checked using a **Normal Q-Q plot**. This can be generated in R using the `plot(model_2, which = 2)` function.

If residuals are normally distributed, they should approximately follow the diagonal line.

Approximately normally distributed.

```
plot(model_2, which = 2)
```



Assumptions Updated!

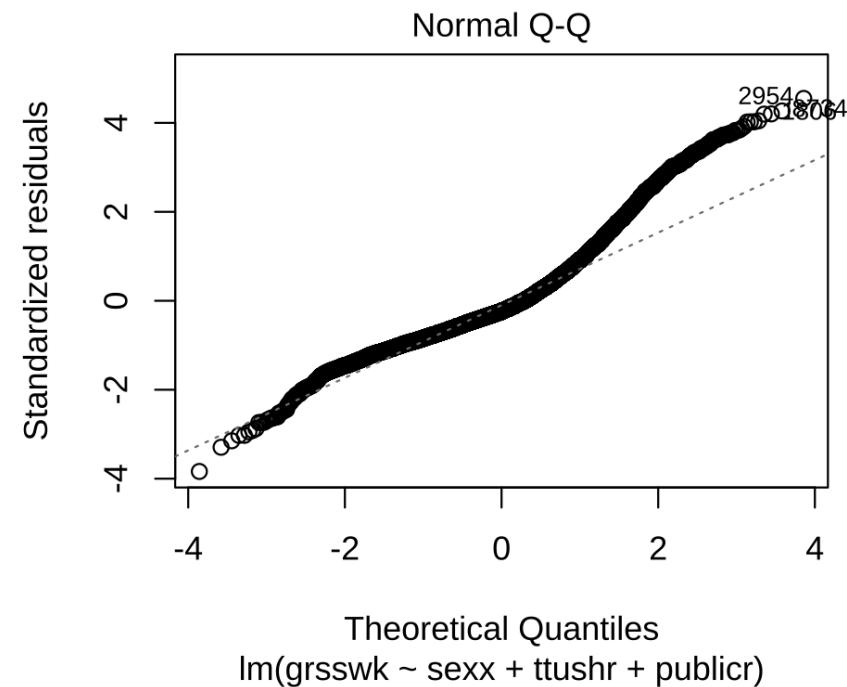
- Linearity
- Homoscedasticity
- Outliers and Leverage Points
- Normality of residuals

Finally, multivariate normality of residuals can be checked using a **Normal Q-Q plot**. This can be generated in R using the `plot(model_2, which = 2)` function.

If residuals are normally distributed, they should approximately follow the diagonal line.

Not normally distributed.

```
plot(model_real, which = 2)
```



Reminder - Cheat sheet on violated assumptions.

- **Linearity**

This means that your estimates will be over- and under-estimated at different parts of the line. If your dependent variable is right-skewed (long tail of observations in the + direction), you can transform it to its log value using `log()`. Alternatively, you can fit a curvilinear model — bit more advanced but not too difficult!

- **Homoscedasticity**

Heteroscedasticity means that the standard errors will be biased (in different ways depending on the shape) — this can be resolved using robust standard errors and/or a weighted least squares estimator.

- **Outliers and leverage points**

Outliers and leverage points can generally have an undue influence on the slope in a regression. Remove any error-based outliers (e.g. someone enters their age as 400 instead of 40). Compare results with 'true' outliers included and excluded to see how they differ, present both.

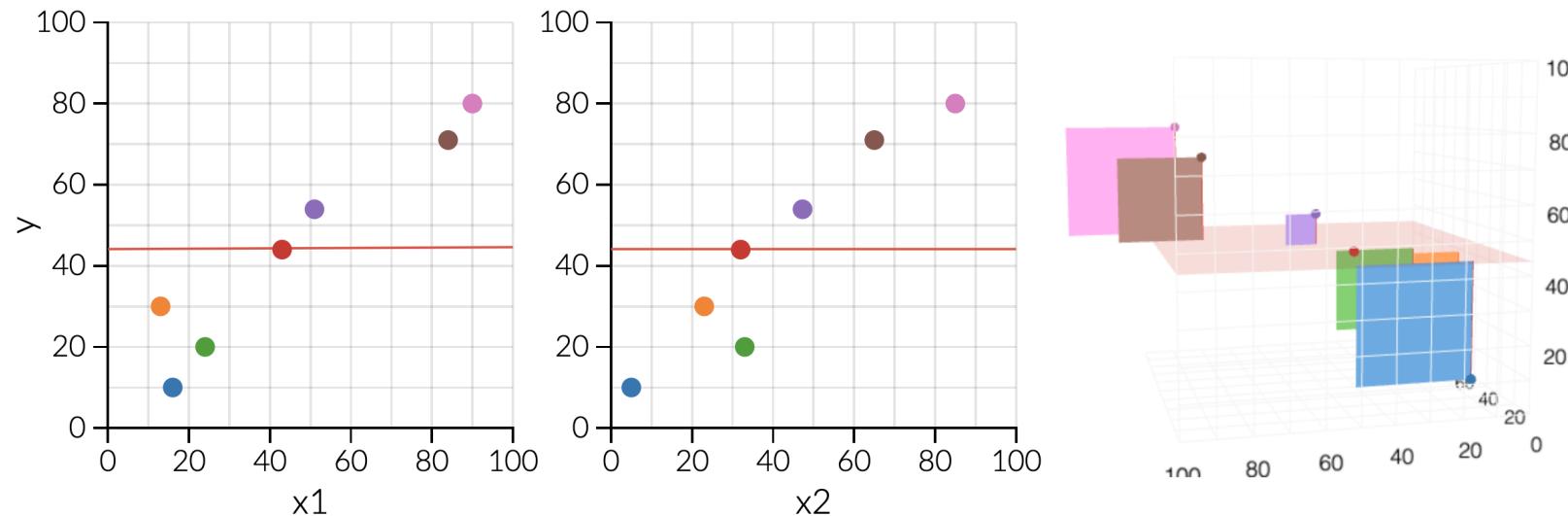
- **Normality of residuals**

Non-normality of residuals can mean that standard errors are smaller than they should be, which can effect decisions made in hypothesis testing in marginal cases — but this is not generally a big problem outside of very small studies ($N < 10$ per variable) (Schmidt & Finan, 2018). Make this clear if either is true.

One more assumption: Multicollinearity

Independent variables are supposed to be **independent**. If two independent variables are strongly correlated (around $r > 0.8$), the estimates for the effect of each will not be accurate.

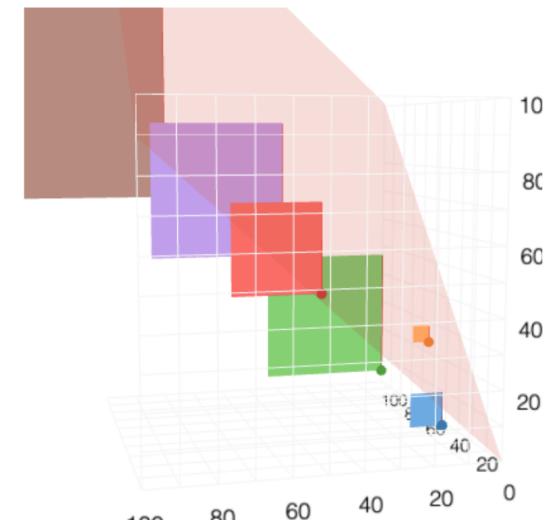
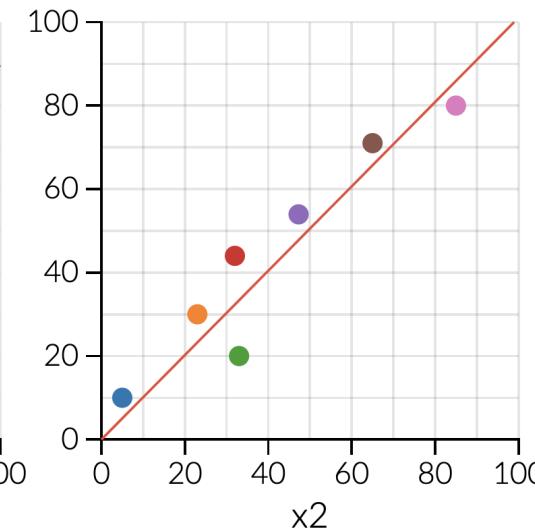
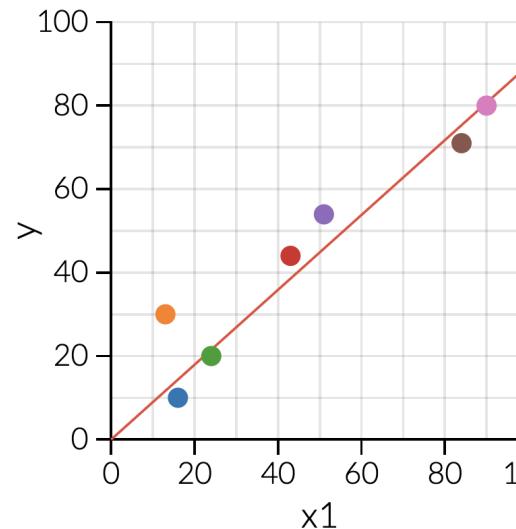
$$44.13 + 0.00 * \text{hand size} + 0.00 * \text{hand size} = \text{height}$$



One more assumption: Multicollinearity

Independent variables are supposed to be **independent**. If two independent variables are strongly correlated (around $r > 0.8$), the estimates for the effect of each will not be accurate.

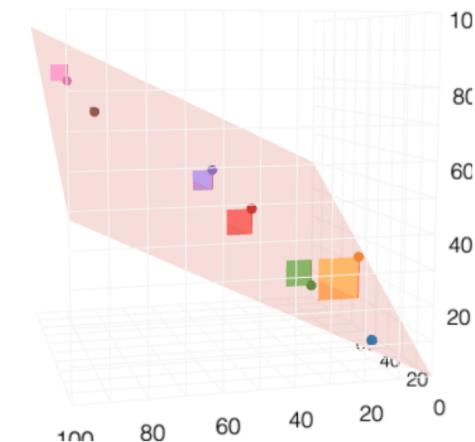
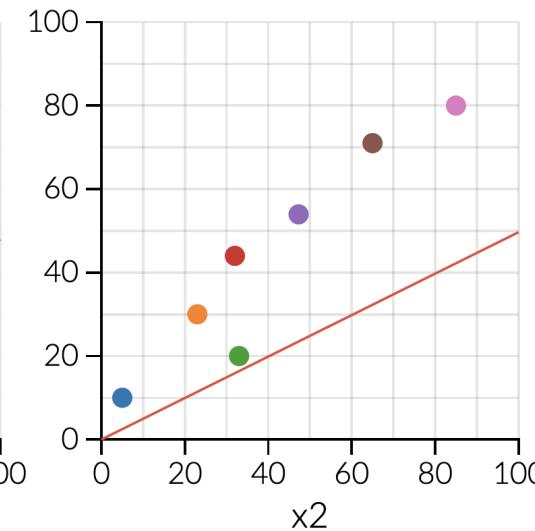
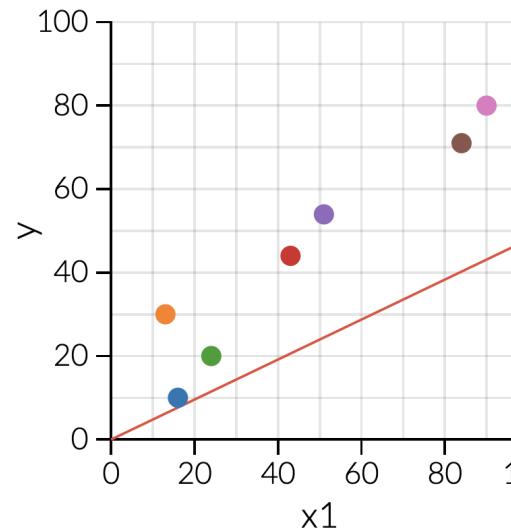
$$0.00 + 0.90 * \text{hand size} + 1.01 * \text{hand size} = \text{height}$$



One more assumption: Multicollinearity

Independent variables are supposed to be **independent**. If two independent variables are strongly correlated (around $r > 0.8$), the estimates for the effect of each will not be accurate.

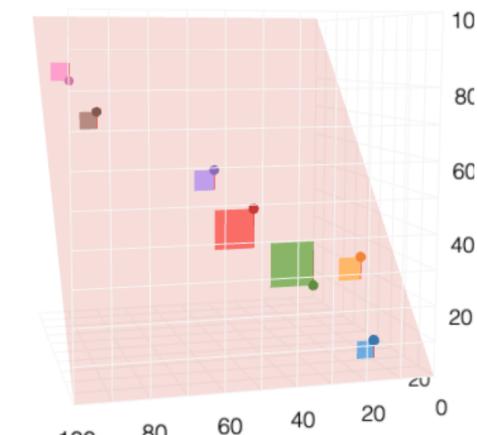
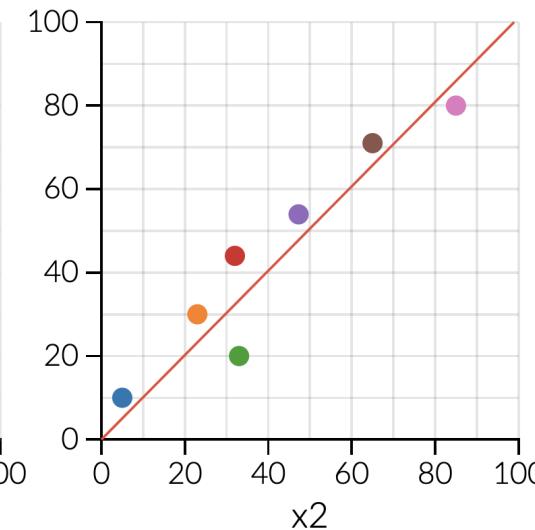
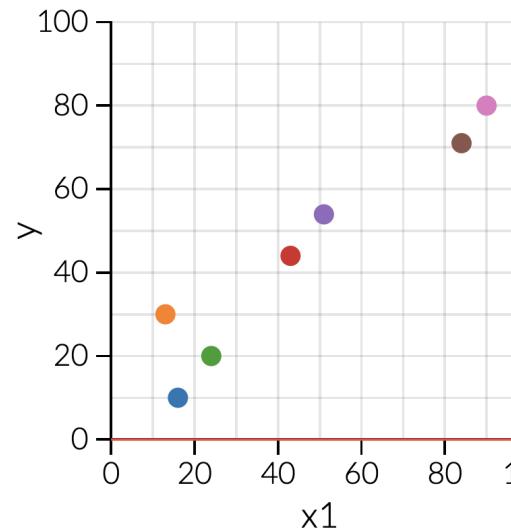
$$0.00 + 0.48 * \text{hand size} + 0.50 * \text{hand size} = \text{height}$$



One more assumption: Multicollinearity

Independent variables are supposed to be **independent**. If two independent variables are strongly correlated (around $r > 0.8$), the estimates for the effect of each will not be accurate.

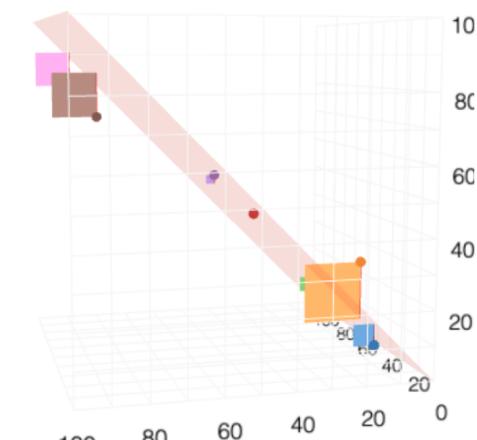
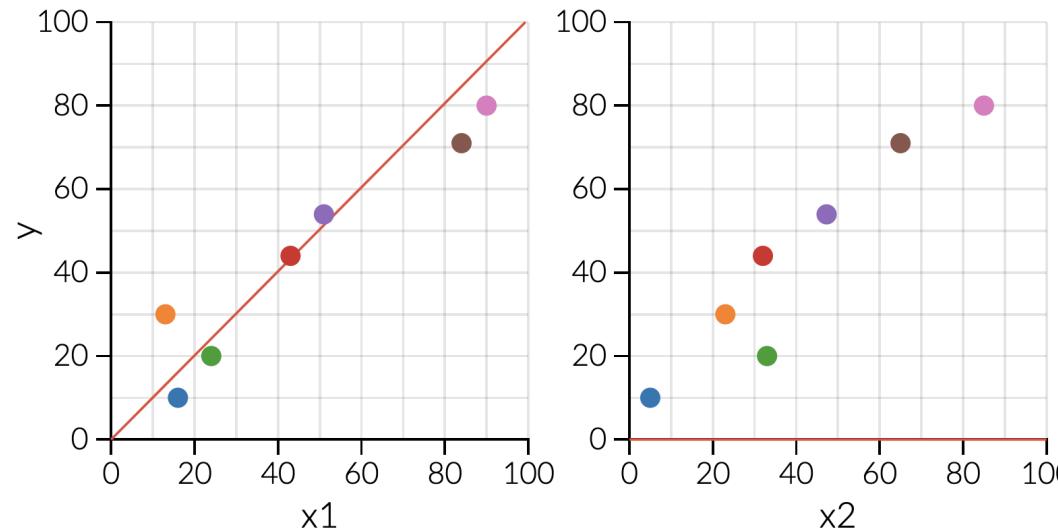
$$0.00 + 0.00 * \text{hand size} + 1.01 * \text{hand size} = \text{height}$$



One more assumption: Multicollinearity

Independent variables are supposed to be **independent**. If two independent variables are strongly correlated (around $r > 0.8$), the estimates for the effect of each will not be accurate.

$$0.00 + 1.01 * \text{hand size} + 0.00 * \text{hand size} = \text{height}$$



Multicollinearity

- Check the correlation between your continuous independent variables before creating your model. Also check the crosstabs between independent dummy variables and Cramer's V.
 - If two variables have $R/V > 0.8$ you may wish to exclude one and consider why they are so highly correlated (e.g. hourly pay and weekly pay are likely highly correlated, but are not really conceptually distinct).
 - If two variables are conceptually distinct, estimate models with both, and each variable separately and describe how the estimates differ.

Multicollinearity

- Use the **Variance Inflation Factor** estimate to check for multicollinearity in your model.
 - Install and load the **car** package with **library(car)**, or call it with **car::**
 - Use the **vif()** function on your model.

```
car::vif(model_2)
```

```
##      gender    hrs_worked public_pr_chr
## 1.460524  1.494024     1.066182
```

- A VIF of 1 indicates no correlation between independent variables.
- A VIF between 1 and 5 indicates some correlation between independent variables, but not much to be of any concern.
- A VIF higher than 5 indicates high correlation between this independent variable and at least one other independent variable. The inclusion of both/all of these variables may bias model estimates.

Regression is a powerful and intuitive tool, and it becomes easy to misuse – especially when assumptions are not checked. It's also very difficult to people to check you haven't violated any assumptions if they don't have your data.

Always make sure you check the assumptions and briefly describe any possible issues there may be in your model, and what future research could do to adjust for these.

Summary

- Multiple linear regression allows us to answer more interesting research questions, taking account of potential **confounders** or competing explanations.
- The number of independent variables that can be added has a very high ceiling, depending on our sample size, making **complex analytical designs feasible**.
- Sometimes, we need to use some data **recoding/transformation techniques** to make our models "tidy" and straightforward to interpret, or to correct for violated assumptions.
- Checking for violated assumptions (including a new assumption of **independent of predictors/multicollinearity**), requires the use of some additional visualisations and statistics. However, these are reasonably easy to produce in **R** using the **plot()** function and **car** package.

R Exercises

This week, we are going to explore the gender pay gap using a sample of [real labour force survey](#) data, incorporating a number of variables.

- Download the [week-8-r-exercises.zip](#) file from Blackboard and open the .Rproj folder and .Rmd file.