Data Analysis with R for Social Scientists

Jakob Tures & Jasper Tjaden

2023-08-29

Contents

Intro

This course offers an accessible and easy introduction to one of the fastest growing statistical packages used in social science and data science more generally.

Please download the data used in the course here. To find more about me, have a look at my website. Also, feel free to watch me as I walk you through each lesson here.

Overview over the Course :

- **Week 1: Introduction to Seminar**
- **Week 2: Exploratory Data Analysis-I**
- **Week 3: Exploratory Data Analysis-II**
- **Week 4: DAGs**
- **Week 5: Linear Regression Theory I: Simple Linear Regression**
- **Week 6: Linear Regression Theory II: Multiple Linear Regression**
- **Week 7: Linear Regression Theory III: Diagnostics**
- **Week 8: Linear Regression - Application**
- **Week 9: Logistic Regression - Exercises**
- **Week 10: Mediation**
- **Week 11: Prediction - Theory**
- **Week 12: Prediction - Application**
- **Week 13: Other Estimators**
- **Week 14: Course Paper - Discussion - Outlook**

Chapter 1

Introduction to Seminar

1.1 Introduction

Welcome to this course! In this course, you will learn how to analyse data using multiple regression in R. The course is aimed at undergraduate students who have completed the course "Intro to R for Social Scientists.

1.2 Why should I take this course?

- What is regression?
- What do you use it for?

1.3 Objectives

-
-

1.4 What is not covered

-
-

1.5 Prerequisites

- Basic knowledge of how to use R (data cleaning, management etc.) (include hyperlink to "Intro to R course")
- Descriptive statistics
- Data visualization using ggplot()

1.6 Structure

1. Intro to seminar

BLOCK I - Pre-processing/ EDA

2. Exploratory data analysis (EDA) - I (ggplot; gtsummary; x)
3. EDA - II -> in class exercise

–> add visualization/ reporting here.

–> knitting

BLOCK II - Modelling/ Linear regression

4. DAGS
5. Linear Regression - theory I: Simple Linear Regression
6. Linear Regression - theory II: Multiple Linear Regression
7. Linear Regression - theory III: Diagnostics

Block III - Application

8. Linear Regression – application
9. Linear Regression - exercise

Block IV - Advanced uses

10. Mediation (?)
11. Prediction - Theory
12. Prediction - Application
13. Other estimators: Logistic/ Poisson/ Multilevel/ FE
14. Course paper/Discussion/ Outlook

Chapter 2

Exloratory Data Analysis - I

2.1 Objectives

- Remember how to load and explore datasets in R
- Conduct basic descriptive data analysis
- Undestand and visualize distribution of and relationship between variables

2.2 R functions covered this week

- `load()`
- `read_excel()`
- `str()`
- `glimpse()`
- `table()`
- `summary()`
- `mutate()`
- `case_when()`
- `ggplot()`
- `corr ()`

2.3 Why is EDA so important?

- Every regression analysis is based on proper EDA; EDA often used for "hypothesis generation" (i.e. finding things that could be interesting to study).

- `EDA` helps understand the data and issues in the data. The better we understand the data, the better we can "fine-tune" our regression model later.
- `EDA` helps prepping data for regression (i.e. "cleaning"; "pre-processing"). Small errors in the data can lead to completely wrong conclusions or even prevent the model from working altogether.

2.4 Importing data into R

The first step of any data analysis is getting data into R. To get started, we first need to follow some preparatory steps

1. In this course, we will use data on NBA players (Basketball). Usually, you need to first download the data and documentation and save them in a folder on your own computer. We have already done this, and provide the data for you here.
2. Install R and Rstudio. If you don't already have R installed, here is a link to how it is done (hyperlink)
3. In the folder which you will use for this class, create a new R project. You will see that all files appear in the bottom right window in R studio.

Now, Let's get started.

2.5 Import data into R

First, we need to install some packages.

```
library("tidyverse")
library("readxl")
```

Now, let's import the data.You can see in the folder that we have 2 csv files. We can use `read_delim()` or `read_csv` function. Note that you need other function for Stata datasets (`read_data` from the haven package). To load Rdata files, you use the `load()` function.

Make sure you name the correct sub-directory in case you saved the data a sub-folder of your project folder (which I have done).

```
# import data
nba_salaries <- read_csv("../datasets/nba/salaries_1985to2018.csv", show_col_types = FA
nba_players <- read_csv("../datasets/nba/players.csv", show_col_types = FALSE)
```

Great, the two dataframes should appear in your environment in the upper right side in R studio.

Let's take a quick look at these trend dataframe using str() function The str() function shows the number of rows (observations) and columns (variables). It also provides information on the name of each column, its type and an example of some of the values in each column.

```
str(nba_players)
```

```
## spc_tbl_ [4,685 x 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ index      : num [1:4685] 0 1 2 3 4 5 6 7 8 9 ...
##  $ _id        : chr [1:4685] "abdelal01" "abdulza01" "abdulka01" "abdulma02" ...
##  $ birthDate  : chr [1:4685] "June 24, 1968" "April 7, 1946" "April 16, 1947" "March 9, 1969"
##  $ birthPlace : chr [1:4685] "Cairo, Egypt" "Brooklyn, New York" "New York, New York" "Gulfpor
##  $ career_AST : num [1:4685] 0.3 1.2 3.6 3.5 1.1 2.5 1.2 1 0.7 0.5 ...
##  $ career_FG% : chr [1:4685] "50.2" "42.8" "55.9" "44.2" ...
##  $ career_FG3%: chr [1:4685] "0.0" NA "5.6" "35.4" ...
##  $ career_FT% : chr [1:4685] "70.1" "72.8" "72.1" "90.5" ...
##  $ career_G   : num [1:4685] 256 505 1560 586 236 830 319 1 56 174 ...
##  $ career_PER : chr [1:4685] "13.0" "15.1" "24.6" "15.4" ...
##  $ career_PTS : num [1:4685] 5.7 9 24.6 14.6 7.8 18.1 5.6 0 9.5 5.3 ...
##  $ career_TRB : chr [1:4685] "3.3" "8.0" "11.2" "1.9" ...
##  $ career_WS  : num [1:4685] 4.8 17.5 273.4 25.2 3.5 ...
##  $ career_eFG%: chr [1:4685] "50.2" NA "55.9" "47.2" ...
##  $ college    : chr [1:4685] "Duke University" "Iowa State University" "University of Californ
##  $ draft_pick : chr [1:4685] "25th overall" "5th overall" "1st overall" "3rd overall" ...
##  $ draft_round: chr [1:4685] "1st round" "1st round" "1st round" "1st round" ...
##  $ draft_team : chr [1:4685] "Portland Trail Blazers" "Cincinnati Royals" "Milwaukee Bucks" "D
##  $ draft_year : chr [1:4685] "1990" "1968" "1969" "1990" ...
##  $ height     : chr [1:4685] "6-10" "6-9" "7-2" "6-1" ...
##  $ highSchool : chr [1:4685] "Bloomfield in Bloomfield, New Jersey" "John Jay in Brooklyn, New
##  $ name       : chr [1:4685] "Alaa Abdelnaby" "Zaid Abdul-Aziz" "Kareem Abdul-Jabbar" "Mahmoud
##  $ position   : chr [1:4685] "Power Forward" "Power Forward and Center" "Center" "Point Guard"
##  $ shoots     : chr [1:4685] "Right" "Right" "Right" "Right" ...
##  $ weight     : chr [1:4685] "240lb" "235lb" "225lb" "162lb" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   index = col_double(),
##   ..   `_id` = col_character(),
##   ..   birthDate = col_character(),
##   ..   birthPlace = col_character(),
##   ..   career_AST = col_double(),
##   ..   `career_FG%` = col_character(),
##   ..   `career_FG3%` = col_character(),
##   ..   `career_FT%` = col_character(),
```

```
##   ..   career_G = col_double(),
##   ..   career_PER = col_character(),
##   ..   career_PTS = col_double(),
##   ..   career_TRB = col_character(),
##   ..   career_WS = col_double(),
##   ..   `career_eFG%` = col_character(),
##   ..   college = col_character(),
##   ..   draft_pick = col_character(),
##   ..   draft_round = col_character(),
##   ..   draft_team = col_character(),
##   ..   draft_year = col_character(),
##   ..   height = col_character(),
##   ..   highSchool = col_character(),
##   ..   name = col_character(),
##   ..   position = col_character(),
##   ..   shoots = col_character(),
##   ..   weight = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(nba_salaries)
```

```
## spc_tbl_ [14,163 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ index       : num [1:14163] 0 1 2 3 4 5 6 7 8 9 ...
##  $ league      : chr [1:14163] "NBA" "NBA" "NBA" "NBA" ...
##  $ player_id   : chr [1:14163] "abdelal01" "abdelal01" "abdelal01" "abdelal01" ...
##  $ salary      : num [1:14163] 395000 494000 500000 805000 650000 1530000 2030000 2
##  $ season      : chr [1:14163] "1990-91" "1991-92" "1992-93" "1993-94" ...
##  $ season_end  : num [1:14163] 1991 1992 1993 1994 1995 ...
##  $ season_start: num [1:14163] 1990 1991 1992 1993 1994 ...
##  $ team        : chr [1:14163] "Portland Trail Blazers" "Portland Trail Blazers" "B
##  - attr(*, "spec")=
##   .. cols(
##   ..   index = col_double(),
##   ..   league = col_character(),
##   ..   player_id = col_character(),
##   ..   salary = col_double(),
##   ..   season = col_character(),
##   ..   season_end = col_double(),
##   ..   season_start = col_double(),
##   ..   team = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

We see that most variables/ columns, already are in the type which we want. R automatically picked up on the format. Text variables are “chr” for “character”;

Numeric variables are "num". It is very important that you understand the types of columns/ variables that R recognized. If you need a refresher, go here [hyperlink].

2.6 Merge datasets

We see that "nba_salaries" contains the salaries of players for various seasons. "players" contains many career statistics about players, for example, how many points they scored on average per game, across their whole career.

Now we want to link both datasets. We can do this using the `merge()` function. An alternative would be `join()`.

```
data_nba <- merge(nba_players, nba_salaries, by.x = c("_id"), by.y=c("player_id"))
rm(nba_players, nba_salaries)
```

2.7 Clean dataset

We can use the select() function to kick out columns and the filter() function to kick out rows. First, we need to look at the codebook and the questionnaire to understand the what each variable refers to (see .txt file "data_description").

When using the select function, we will also rename the variables to make them more intuitive.

First, let's filter out Let's filter all years between 2012-2022.

Afterwards, we use the save() function to store the data as a .Rdata file and the write_excel() function to store the reduced dataset. This way we can simply load that one next time and save time.

```
data_nba <- data_nba %>%
        select(everything(), -league, -highSchool) %>%
        filter(season_start>=1998)
save(data_nba, file ="../datasets/nba/data_nba.RData")
```

2.8 change variables

Let's do some data cleaning. First, we want to calculate each players' age at the beginning of each season. Currently, we only have the date of birth and the year for each season.

Second, we want recode the "position" variable. Some players played multiple position, so that info is messy. We want to create varies dummy variables for each position.

```
# let's calculate age for every season
class(data_nba$birthDate) # nice, it is already a date variable
```

```
## [1] "character"
```

```
library(lubridate)
data_nba <- data_nba %>%
  mutate(year_of_birth = year(mdy(birthDate)),
         age = season_start - year_of_birth)


# let's clean the position

table(data_nba$position)
```

```
##
##                                                                 Center
##                                                                   1204
##                                              Center and Power Forward
##                                                                    686
##                          Center and Power Forward and Small Forward
##                                                                      3
##                          Center and Small Forward and Power Forward
##                                                                     30
##                                                            Point Guard
##                                                                   1162
## Point Guard and Power Forward and Small Forward and Shooting Guard
##                                                                      5
##                                         Point Guard and Shooting Guard
##                                                                    573
##                     Point Guard and Shooting Guard and Small Forward
##                                                                     13
##                                       Point Guard and Small Forward
##                                                                      3
##                     Point Guard and Small Forward and Shooting Guard
##                                                                     21
##                                                       Power Forward
##                                                                    667
##                                              Power Forward and Center
##                                                                    884
```

```
##                         Power Forward and Center and Small Forward
##                                                                  51
##                                  Power Forward and Shooting Guard
##                                                                  14
##                Power Forward and Shooting Guard and Small Forward
##                                                                  31
##                                   Power Forward and Small Forward
##                                                                 418
##                         Power Forward and Small Forward and Center
##                                                                   3
##                Power Forward and Small Forward and Shooting Guard
##                                                                  11
##                                                       Shooting Guard
##                                                                 735
##                                    Shooting Guard and Point Guard
##                                                                 581
##                  Shooting Guard and Point Guard and Small Forward
##                                                                  17
##                Shooting Guard and Power Forward and Small Forward
##                                                                  25
##                                  Shooting Guard and Small Forward
##                                                                 592
##                  Shooting Guard and Small Forward and Point Guard
##                                                                  73
##                Shooting Guard and Small Forward and Power Forward
##                                                                  25
##                                                     Small Forward
##                                                                 620
##                                          Small Forward and Center
##                                                                   4
##                         Small Forward and Center and Power Forward
##                                                                  72
##                  Small Forward and Point Guard and Shooting Guard
##                                                                  19
##                                   Small Forward and Power Forward
##                                                                 425
##                         Small Forward and Power Forward and Center
##                                                                  49
##                Small Forward and Power Forward and Shooting Guard
##                                                                  33
##                                  Small Forward and Shooting Guard
##                                                                 588
##                  Small Forward and Shooting Guard and Point Guard
##                                                                  22
##                Small Forward and Shooting Guard and Power Forward
##                                                                  69
```

```
data_nba <- data_nba %>%
  mutate(
    position_center =
      case_when(position = str_detect(position,"Center") ~ 1,
                TRUE ~ 0),
    position_sf =
      case_when(position = str_detect(position,"Small Forward") ~ 1,
                TRUE ~ 0),
    position_pf =
      case_when(position = str_detect(position,"Power Forward") ~ 1,
                TRUE ~ 0),
    position_sg =
      case_when(position = str_detect(position,"Shooting Guard") ~ 1,
                TRUE ~ 0),
    position_pg =
      case_when(position = str_detect(position,"Point Guard") ~ 1,
                TRUE ~ 0))



data_nba <- data_nba %>%
  select("_id", name, age, weight, height, birthPlace, everything(), -position, -birthl
save(data_nba, file ="../datasets/nba/data_nba.RData")
```

Now, we want to create a variable that gives us the number of seasons (i.e.years) that each player played. Since the dataset is organized in seasons, each row is one season. Counting the rows per player gives us the years they played.

```
data_nba <- data_nba %>%
  group_by("_id") %>%
  mutate(seasons_played = n()) %>%
  ungroup()
```

Almost done. When we browse the dataset, we recognize that the height and weight variables are stored as characters. Let's convert them to numeric, so we can use them in operations.

```
str(data_nba$weight)
```

```
##  chr [1:9728] "162lb" "223lb" "223lb" "223lb" "223lb" "223lb" "223lb" ...
```

```
str(data_nba$height)
```

```
##  chr [1:9728] "6-1" "6-6" "6-6" "6-6" "6-6" "6-6" "6-6" "6-6" "6-6" "6-6" ...
```

```
data_nba <- data_nba %>%
  mutate(weight = str_replace(weight, "lb", ""),
         weight = as.numeric(weight),
         height = str_replace(height, "-", "."),
         height = as.numeric(height))

str(data_nba)
```

```
## tibble [9,728 x 36] (S3: tbl_df/tbl/data.frame)
##  $ _id            : chr [1:9728] "abdulma02" "abdulta01" "abdulta01" "abdulta01" ...
##  $ name           : chr [1:9728] "Mahmoud Abdul-Rauf" "Tariq Abdul-Wahad" "Tariq Abdul-Wahad"
##  $ age            : num [1:9728] 31 24 25 26 27 28 29 30 31 32 ...
##  $ weight         : num [1:9728] 162 223 223 223 223 223 223 223 223 223 ...
##  $ height         : num [1:9728] 6.1 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 ...
##  $ birthPlace     : chr [1:9728] "Gulfport, Mississippi" "Maisons Alfort, France" "Maisons Alf
##  $ index.x        : num [1:9728] 3 4 4 4 4 4 4 4 4 4 ...
##  $ career_AST     : num [1:9728] 3.5 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ career_FG%     : chr [1:9728] "44.2" "41.7" "41.7" "41.7" ...
##  $ career_FG3%    : chr [1:9728] "35.4" "23.7" "23.7" "23.7" ...
##  $ career_FT%     : chr [1:9728] "90.5" "70.3" "70.3" "70.3" ...
##  $ career_G       : num [1:9728] 586 236 236 236 236 236 236 236 236 236 ...
##  $ career_PER     : chr [1:9728] "15.4" "11.4" "11.4" "11.4" ...
##  $ career_PTS     : num [1:9728] 14.6 7.8 7.8 7.8 7.8 7.8 7.8 7.8 7.8 7.8 ...
##  $ career_TRB     : chr [1:9728] "1.9" "3.3" "3.3" "3.3" ...
##  $ career_WS      : num [1:9728] 25.2 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 ...
##  $ career_eFG%    : chr [1:9728] "47.2" "42.2" "42.2" "42.2" ...
##  $ college        : chr [1:9728] "Louisiana State University" "University of Michigan, San Jos
##  $ draft_pick     : chr [1:9728] "3rd overall" "11th overall" "11th overall" "11th overall" ..
##  $ draft_round    : chr [1:9728] "1st round" "1st round" "1st round" "1st round" ...
##  $ draft_team     : chr [1:9728] "Denver Nuggets" "Sacramento Kings" "Sacramento Kings" "Sacra
##  $ draft_year     : chr [1:9728] "1990" "1997" "1997" "1997" ...
##  $ shoots         : chr [1:9728] "Right" "Right" "Right" "Right" ...
##  $ index.y        : num [1:9728] 17 19 20 21 22 23 24 25 26 27 ...
##  $ salary         : num [1:9728] 798500 1411000 1594920 4500000 5062500 ...
##  $ season         : chr [1:9728] "2000-01" "1998-99" "1999-00" "2000-01" ...
##  $ season_end     : num [1:9728] 2001 1999 2000 2001 2002 ...
##  $ season_start   : num [1:9728] 2000 1998 1999 2000 2001 ...
##  $ team           : chr [1:9728] "Vancouver Grizzlies" "Sacramento Kings" "Denver Nuggets" "De
##  $ position_center: num [1:9728] 0 0 0 0 0 0 0 0 0 0 ...
##  $ position_sf    : num [1:9728] 0 0 0 0 0 0 0 0 0 0 ...
##  $ position_pf    : num [1:9728] 0 0 0 0 0 0 0 0 0 0 ...
##  $ position_sg    : num [1:9728] 0 1 1 1 1 1 1 1 1 1 ...
##  $ position_pg    : num [1:9728] 1 0 0 0 0 0 0 0 0 0 ...
##  $ "_id"          : chr [1:9728] "_id" "_id" "_id" "_id" ...
##  $ seasons_played : int [1:9728] 9728 9728 9728 9728 9728 9728 9728 9728 9728 9728 ...
```

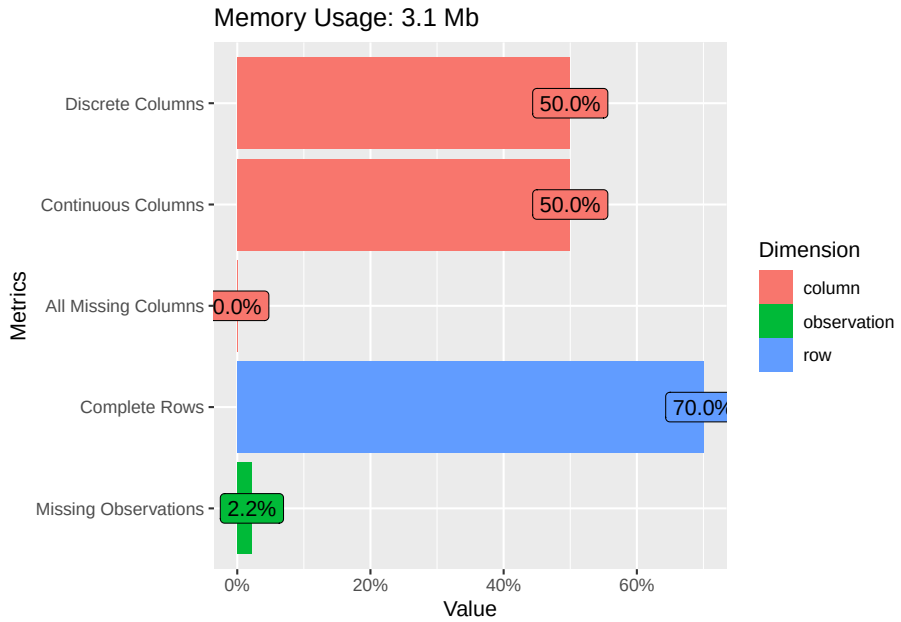
2.9 Explore the whole dataset

Now, let's explore some packages which help us to explore the dataframe as a whole. Let's start with the DataExplorer package. It is nice to get an overview of variables and the "missingness" of data.

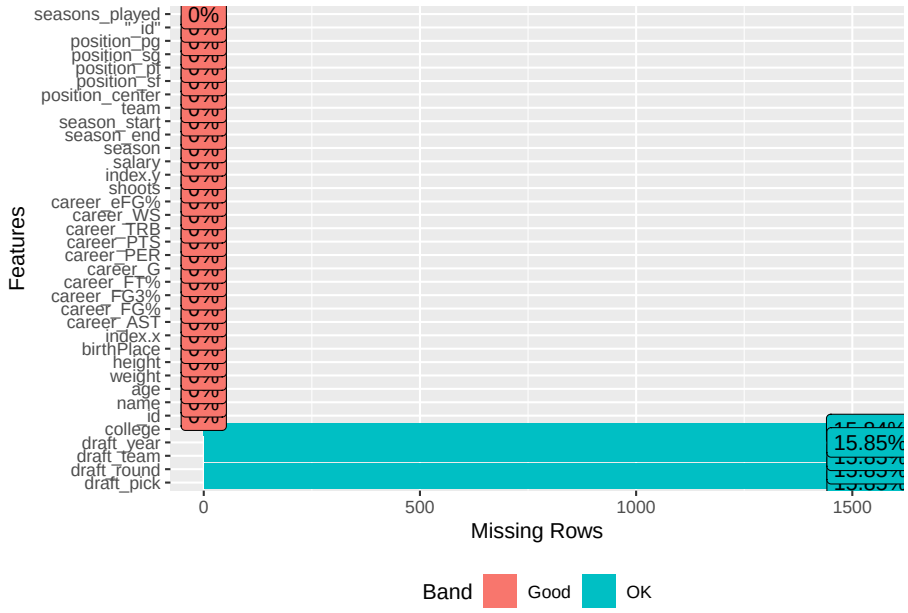
```
library(DataExplorer)

# overview of types of variables and missingness
introduce(data_nba)
```

```
## # A tibble: 1 x 9
##    rows columns discrete_columns continuous_columns all_missing_columns
##   <int>   <int>            <int>              <int>               <int>
## 1  9728      36               18                 18                   0
## # i 4 more variables: total_missing_values <int>, complete_rows <int>,
## #   total_observations <int>, memory_usage <dbl>
```

```
# plots the info from above
plot_intro(data_nba)
```

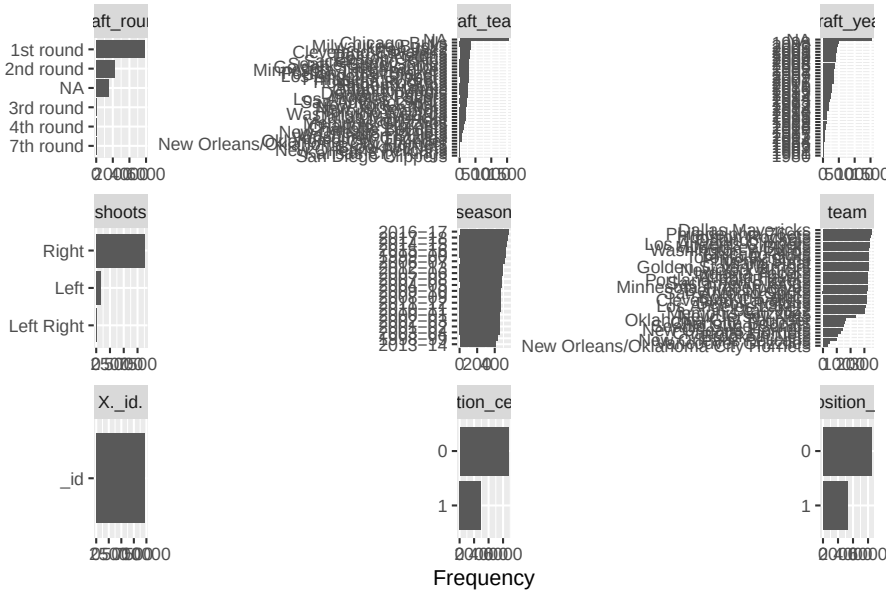


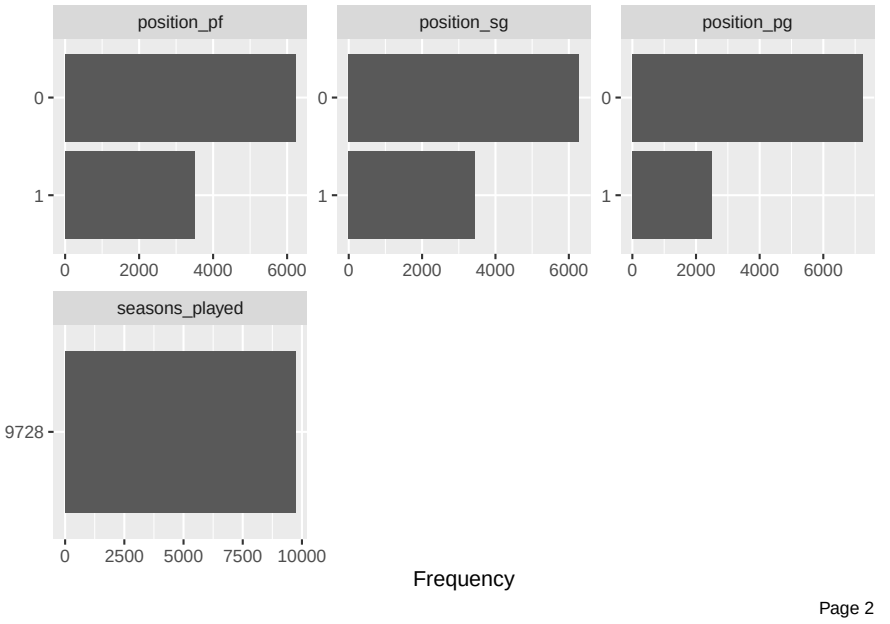

```
#plots percentages missing across variables
plot_missing(data_nba)
```




```
# plots frequencies across variables
plot_bar(data_nba)
```

```
## 11 columns ignored with more than 50 categories.
## X_id: 1794 categories
## name: 1790 categories
## birthPlace: 849 categories
## career_FG.: 333 categories
## career_FG3.: 297 categories
## career_FT.: 413 categories
## career_PER: 274 categories
## career_TRB: 113 categories
## career_eFG.: 310 categories
## college: 380 categories
## draft_pick: 67 categories
```





Now, let's try gtSummary for summary tables. A summary table is always a useful start once you have identified the type of variables you are interested in.

Let's assume we are interested in age, seasons played, career points, salary, position, and right or left handed (i.e. "shoots")

```
library(gtsummary)
```

```
## #Uighur
```

```
data_nba %>%
  select(age, seasons_played, shoots, career_PTS, salary, contains("position")) %>%
         tbl_summary(
           statistic = all_continuous() ~ c("{mean} ({min}, {max})"))
```

```
## Table printed with `knitr::kable()`, not {gt}. Learn why at
## https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
## To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	**N = 9,728**
age	27.0 (18.0, 42.0)
seasons_played	
9728	9,728 (100%)
shoots	
Left	856 (8.8%)
Left Right	7 (<0.1%)
Right	8,865 (91%)
career_PTS	8.9 (0.0, 30.1)
salary	4,072,633 (2,706, 34,682,550)
position_center	2,986 (31%)
position_sf	3,222 (33%)
position_pf	3,501 (36%)
position_sg	3,447 (35%)
position_pg	2,489 (26%)

Now, let's look at a correlation matrix between all numeric variables in the dataset.

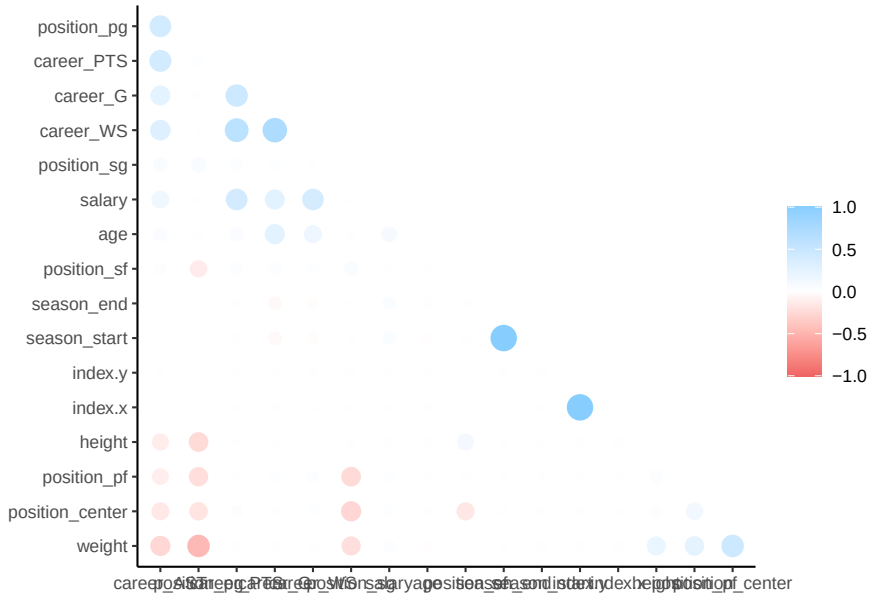
```
library("corrr")
library("corrplot")

data_nba_numeric <- data_nba %>%
  select(where(is.numeric)) %>%
  na.omit()

# Find constant columns
constant_columns <- sapply(data_nba_numeric, function(x) length(unique(x)) == 1)

# Remove constant columns
data_nba_numeric <- data_nba_numeric[, !constant_columns]
```

```
# as number
corrmatrix <- data_nba_numeric %>%
  correlate() %>%    # Create correlation data frame (cor_df)
  rearrange() %>%  # rearrange by correlations
  shave()

fashion(corrmatrix)
```

```
##              term career_AST position_pg career_PTS career_G career_WS
## 1       career_AST
## 2      position_pg        .61
## 3       career_PTS        .61         .08
## 4         career_G        .47         .04        .65
## 5        career_WS        .52        -.01        .76       .83
## 6      position_sg        .19         .22        .12       .09        .01
## 7           salary        .35        -.04        .60       .48        .59
## 8              age        .18         .03        .18       .49        .36
## 9      position_sf       -.08        -.33        .12       .13        .07
## 10      season_end        .01         .01        .04      -.17       -.09
## 11    season_start        .01         .01        .04      -.17       -.09
## 12         index.y        .01         .00       -.01      -.04       -.03
## 13         index.x        .01         .00       -.01      -.04       -.03
## 14          height       -.31        -.46       -.01      -.02       -.00
## 15     position_pf       -.29        -.44        .01       .11        .12
## 16 position_center       -.36        -.39       -.10       .04        .08
## 17          weight       -.49        -.66       -.06      -.03        .05
##    position_sg salary  age position_sf season_end season_start index.y index.x
## 1
## 2
## 3
## 4
## 5
## 6
## 7         -.03
## 8          .03    .25
## 9          .18    .03  .03
## 10         .02    .16 -.07       -.04
## 11         .02    .16 -.07       -.04       1.00
## 12        -.05   -.02 -.02        .01       -.03        -.03
## 13        -.06   -.02 -.02        .01       -.03        -.03    1.00
## 14        -.03    .03 -.04        .30        .02         .02     .05     .05
## 15        -.46    .09  .03        .04       -.00        -.00     .01     .01
## 16        -.49    .10  .04       -.37       -.06        -.06     .01     .02
## 17        -.43    .12 -.07       -.01        .04         .04     .04     .04
##    height position_pf position_center weight
```

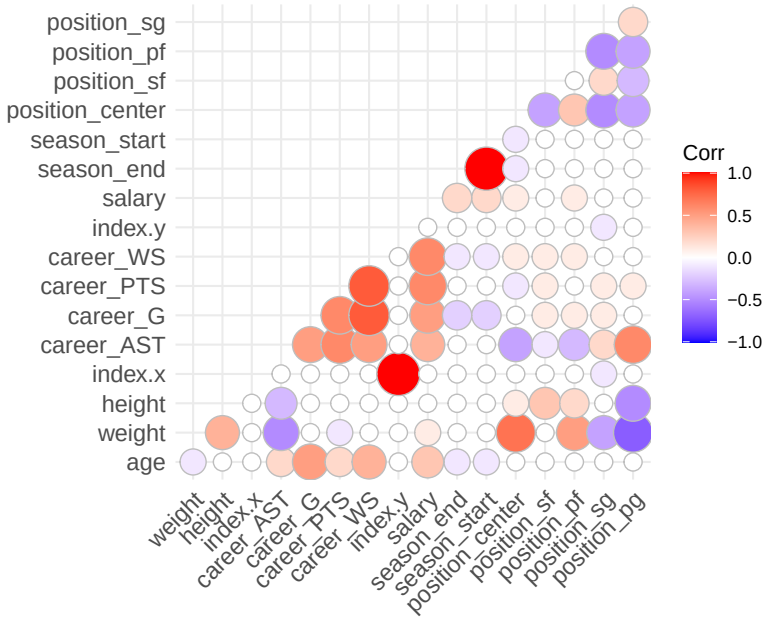
```
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15    .15
## 16    .12         .33
## 17    .41         .45            .66
```

```
# as plot
rplot(corrmatrix)
```




```
# or...ggplot approach
library("ggcorrplot")
```

```
corrmatrix <- round(cor(data_nba_numeric), 1)

ggcorrplot(corrmatrix,
           method = "circle",
           type="lower")
```



This is interesting for a first look. For example, it seems that the weight is strongly correlated with whatever position you play. Centers are heavy, point guards are light weights. We also see that most performance metrics ("career_…") are correlated with each other and also with salary. Good players seem to be good in many things, and good players seem to be paid more.

2.10 explore individual variables

Now, that we have a feeling for the whole dataset, we want to explore individual variables. To keep it focused, we want to further explore the question of whether players that score more on average are also paid more.

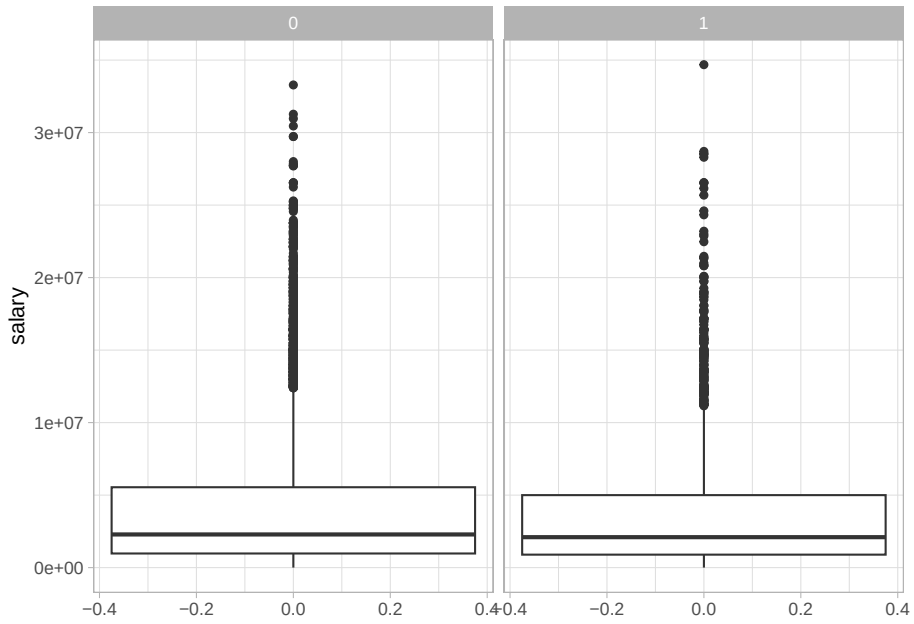
Maybe roles are clearly divided on the team. Maybe really good passers are highly paid because they give great passes to people who then score. Or maybe teams don't care about passers and just pay more to people who score more.

So, now, let's look at salary and average number of points scored by game. Also, we want to know whether point guards (short people who pass a lot) are paid

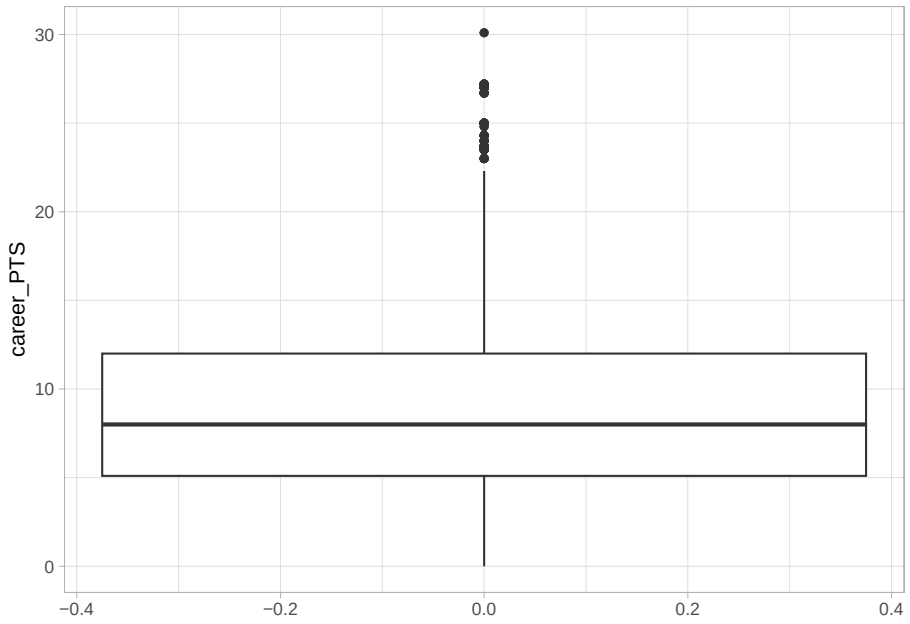
less then other positions (who score more).

First, let's look how the two continuous variables are distributed:

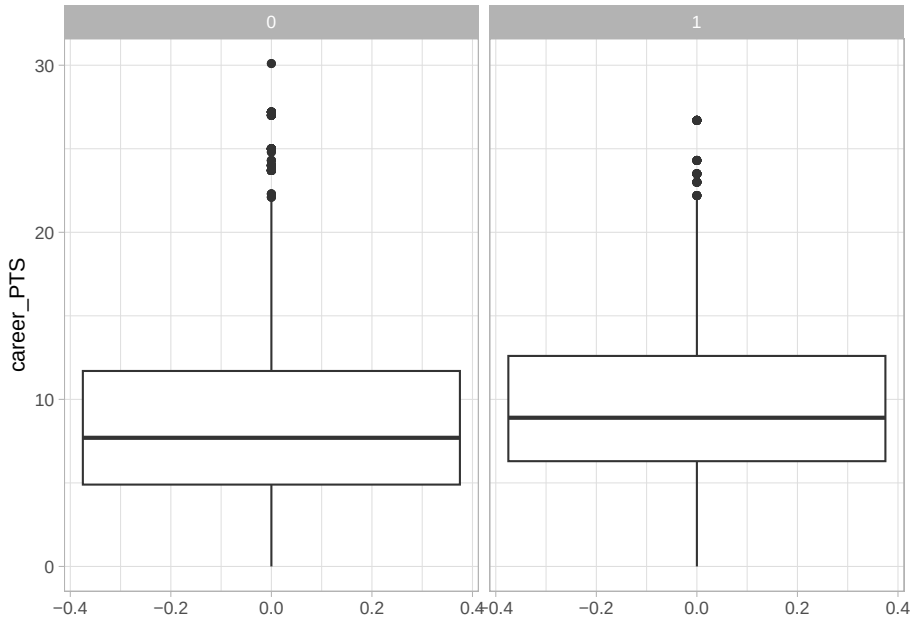
```
data_nba %>% ggplot() +
geom_boxplot(aes(y=salary)) +
  facet_wrap(~position_pg) +
  theme_light()
```




```
data_nba %>% ggplot() +
geom_boxplot(aes(y=career_PTS)) +
  theme_light()
```




```
data_nba %>% ggplot() +
geom_boxplot(aes(y=career_PTS)) +
  facet_wrap(~position_pg) +
  theme_light()
```



Let's look at the relationship between salary and position as well as the relationship between position and points.

```
str(data_nba$position_pg)
```

```
##  num [1:9728] 1 0 0 0 0 0 0 0 0 0 ...
```

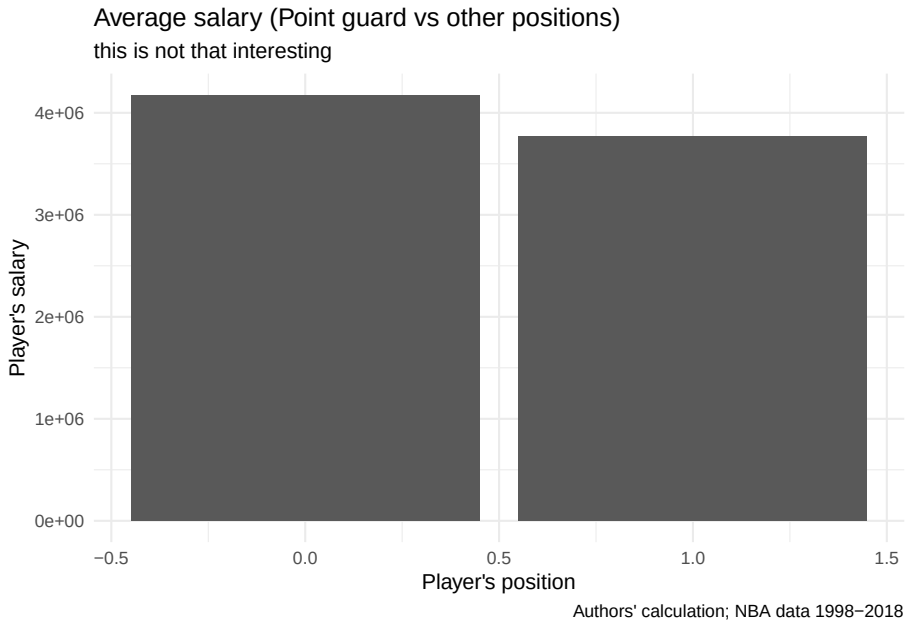
```
str(data_nba$salary)
```

```
##  num [1:9728] 798500 1411000 1594920 4500000 5062500 ...
```

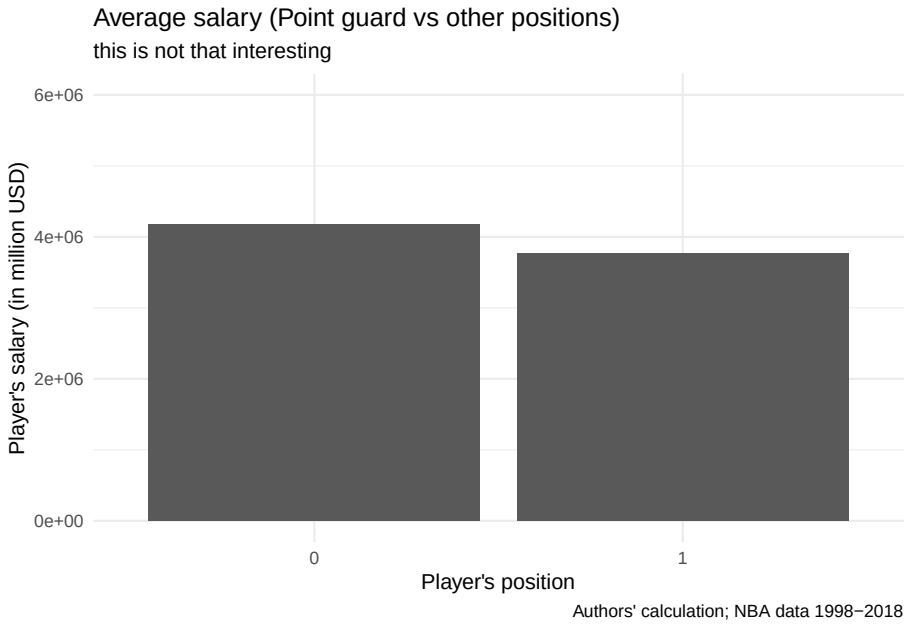
```
str(data_nba$career_pts)
```

```
##  NULL
```

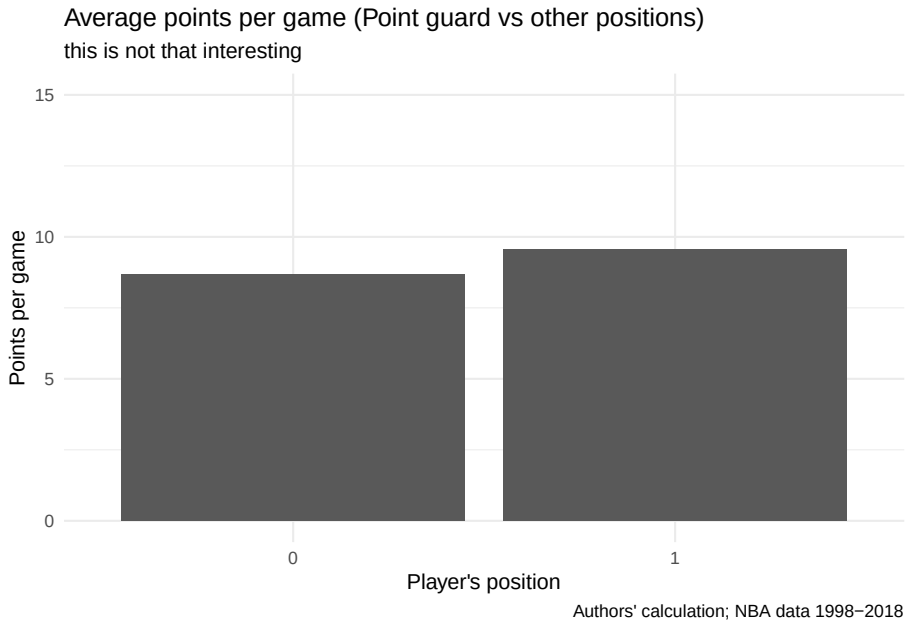
```
data_nba %>%
  ggplot() +
  geom_bar(aes(x = position_pg,
               y = salary),
           position = "dodge",
           stat = "summary",
           fun = "mean") +
  #xlim(0, 10000000) +
  labs(title = "Average salary (Point guard vs other positions)",
       subtitle = "this is not that interesting",
       caption = "Authors' calculation; NBA data 1998-2018") +
  xlab("Player's position") +
  ylab("Player's salary") +
  theme_minimal()
```




```
## alternatively:

data_nba %>%
  group_by(position_pg) %>%
  summarize(mean_salary = mean(salary, na.rm=T)) %>%
  ggplot() +
  geom_bar(aes(x = as.factor(position_pg),
               y = mean_salary),
           stat = "identity") +
  ylim(0, 6000000) +
  labs(title = "Average salary (Point guard vs other positions)",
       subtitle = "this is not that interesting",
       caption = "Authors' calculation; NBA data 1998-2018") +
  xlab("Player's position") +
  ylab("Player's salary (in million USD)") +
  theme_minimal()
```




```
# Now the relationship between position and points:

data_nba %>%
  group_by(position_pg) %>%
  summarize(mean_points = mean(career_PTS, na.rm=T)) %>%
  ggplot() +
  geom_bar(aes(x = as.factor(position_pg),
               y = mean_points),
           stat = "identity") +
  ylim(0,15) +
  labs(title = "Average points per game (Point guard vs other positions)",
       subtitle = "this is not that interesting",
       caption = "Authors' calculation; NBA data 1998-2018") +
  xlab("Player's position") +
  ylab("Points per game") +
  theme_minimal()
```

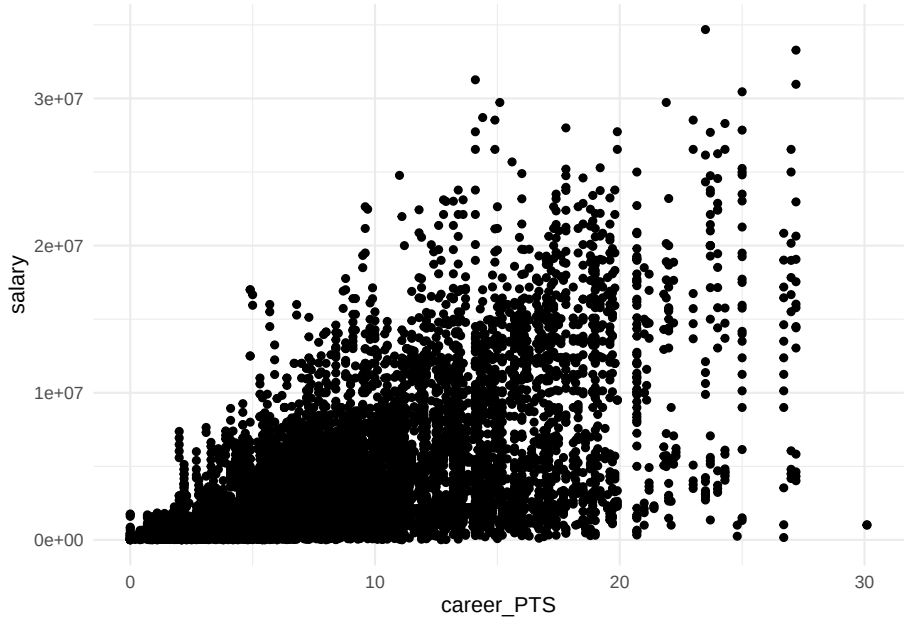


So, it seems that point gards are paid a little less even though they make a few more points on average. Interesting puzzle to explore.

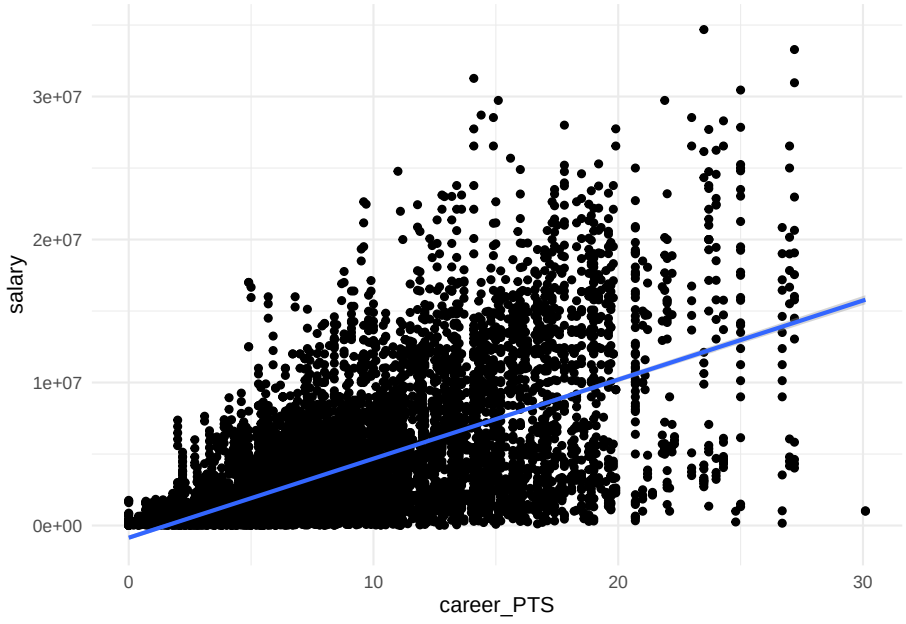
```
# find example for using histogramm()
# find example for first just creating cross tabs as perecntages tbl_cross(); summariz
```

Now, let's explore the relationship which is at the heart of our analysis from now on: salary and average points.

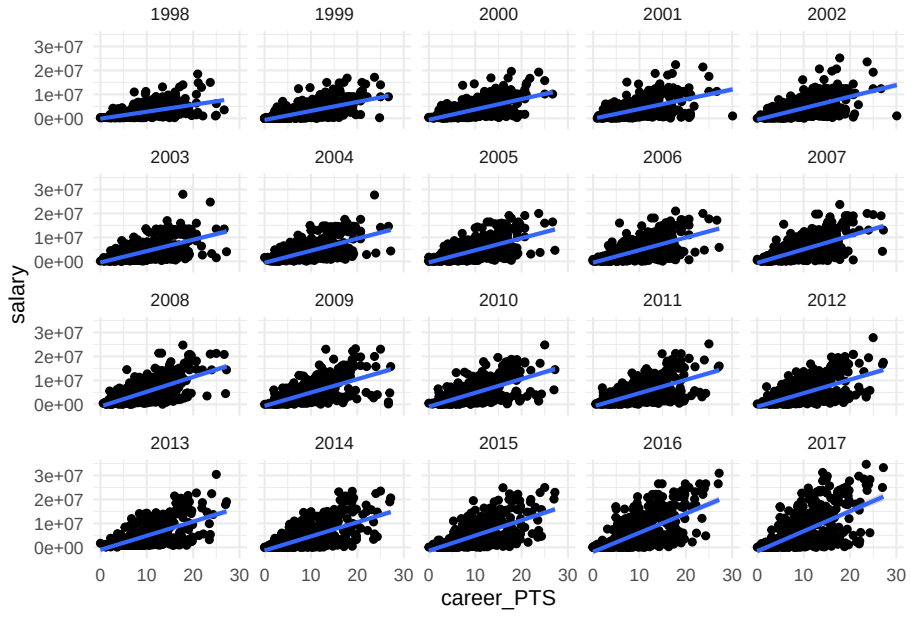
```
# simple scatterplot
data_nba %>% ggplot() +
geom_point(aes(y=salary, x=career_PTS)) +
  theme_minimal()
```



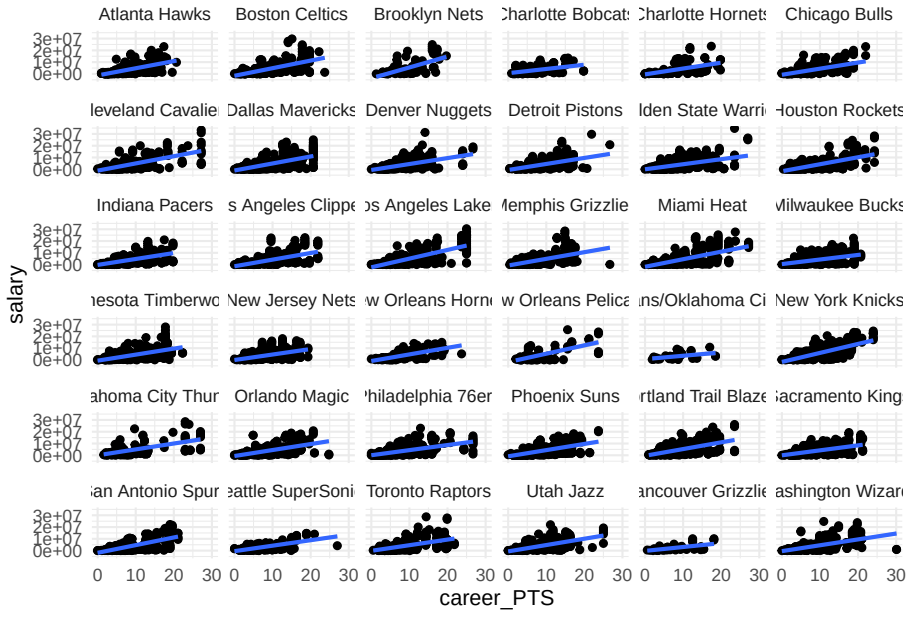

```
# Now, let's add a line
data_nba %>%
  ggplot(aes(y=salary, x=career_PTS)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()
```




```
# let's look the relatiship separate for every year
data_nba %>%
  ggplot(aes(y=salary, x=career_PTS)) +
    geom_point() +
     stat_smooth(method = "lm") +
        facet_wrap(~season_start) +
       theme_minimal()
```

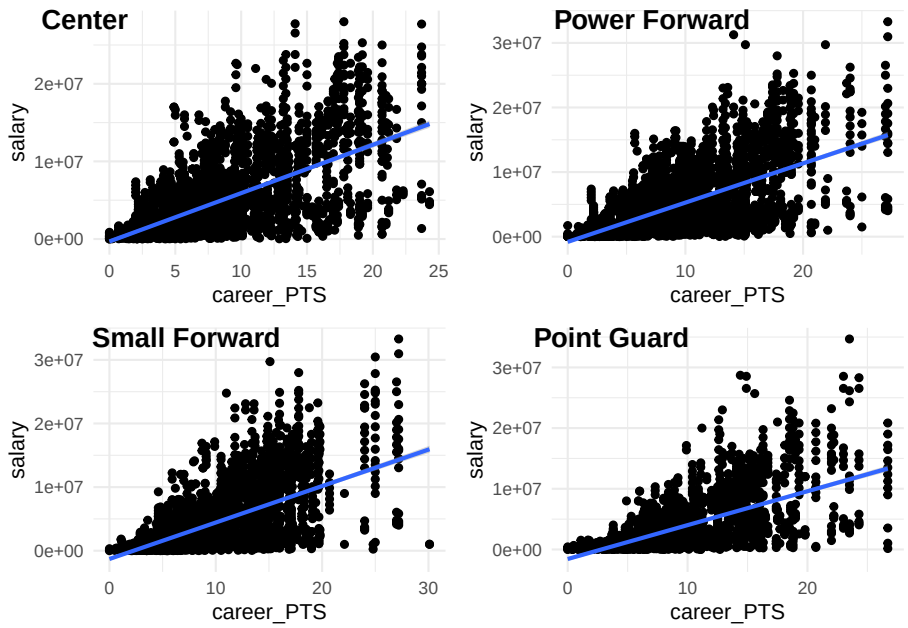



```
# let's look the relatiship for separate teams
data_nba %>%
  ggplot(aes(y=salary, x=career_PTS)) +
    geom_point() +
     stat_smooth(method = "lm") +
        facet_wrap(~team) +
       theme_minimal()
```

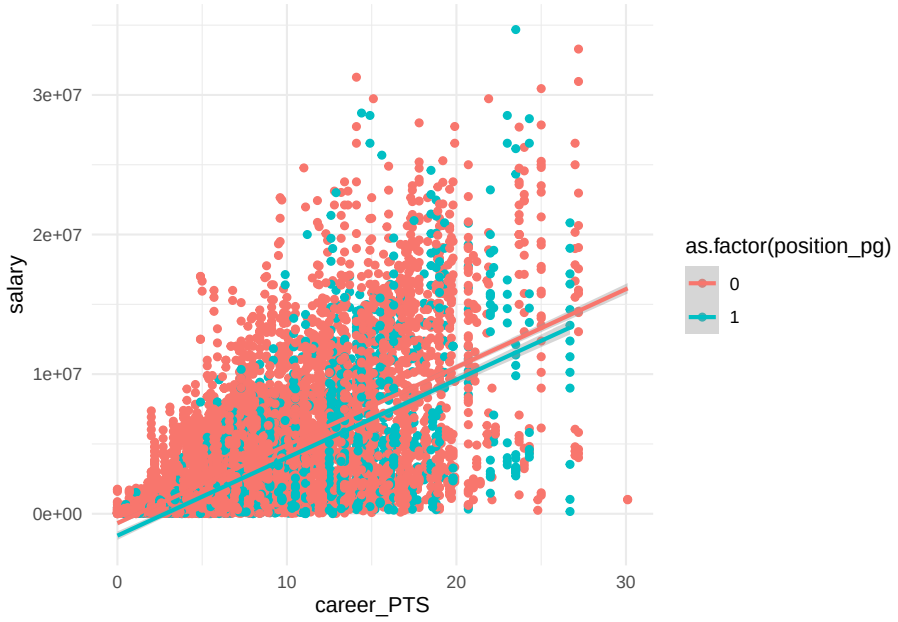



```
# let's look at it by position
scatter_pg <- data_nba %>% filter(position_pg ==1) %>%
  ggplot(aes(y=salary, x=career_PTS)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()

scatter_center <- data_nba %>% filter(position_center ==1) %>%
  ggplot(aes(y=salary, x=career_PTS)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()

scatter_pf <- data_nba %>% filter(position_pf ==1) %>%
  ggplot(aes(y=salary, x=career_PTS)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()

scatter_sf <- data_nba %>% filter(position_sf ==1) %>%
  ggplot(aes(y=salary, x=career_PTS)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()
```

```
library("ggpubr")
ggarrange(scatter_center, scatter_pf,
          scatter_sf, scatter_pg,
          ncol = 2, nrow = 2,
          labels = c("Center",
                     "Power Forward",
                     "Small Forward",
                     "Point Guard"))
```




```
# yet a better way to compare this could be:

data_nba %>%
  ggplot(aes(y=salary, x=career_PTS, color=as.factor(position_pg))) +
    geom_point() +
     stat_smooth(method = "lm",
                 aes(group=as.factor(position_pg))) +
       theme_minimal()
```




```
data_nba %>%
  ggplot(aes(y=salary, x=career_PTS, color=as.factor(position_center))) +
    geom_point() +
     stat_smooth(method = "lm",
                 aes(group=as.factor(position_center))) +
       theme_minimal()
```

Let's reflect a moment what we can learn from all this.

First, there seems to be a somewhat linear relationship between how many points a player scores and how much they are paid. This relationship seems pretty robust across years and teams. It also holds true for point guards just as much as for non-point guards. However, the average salary for point guards is lower in comparison. We also learn that the link between salary and points is stronger for centers. They seems to be paid more, the more they score.

We have set the stage now for linear regression (next week). Linear regression is all about further exploring the relationship between two variables, one outcome (often called "y") and one independent variable (often called "x"). Independent variables have many names. They are sometimes called "covariates"; "predictors"; "exposure", depending on the context.

There are a number of cool things that regression can do for us that simple EDA cannot:

1) It can model the relationship between two variables while considering simultaneously the potential influence of other factors. Image we are interested in the effect of points per game on salary regardless of position, season or team. With regression we can estimate how much more a player would ear every season if he scored 10 more points a game (regardless of the position he plays).

2) It can assess what explains the effect of one variable on another (i.e. mediation). Maybe we find that point guards earn less and we want to know

why. Is it because they score less? Is it because they play less time on average?

3) It can be used to predict salaries for players for whom we don't know the salary or even for hypothetical players. We could also look at the performance trend of players and predict whether they earn more next season or not.

Chapter 3

Exploratory Data Analysis - II

This week we will try to apply our last weeks knowledge into analysis.

3.1 Markdown Introduction

R Markdown is a powerful tool that allows you to create dynamic documents, presentations, and reports using R code. It combines the core syntax of markdown (an easy-to-write plain text format) with embedded R code chunks that are run when the document is rendered[1].

R Markdown documents are fully reproducible, meaning that anyone can re-run the code and generate the same results. This makes it easy to share your work with others and ensure that your results are accurate and reliable.

One of the great things about R Markdown is its flexibility. You can use it to create a wide variety of output formats, including HTML, PDF, and Microsoft Word documents. You can even create interactive documents with Shiny components[1].

To get started with R Markdown, you'll need to install the `rmarkdown` package from CRAN. This can be done by running the command `install.packages("rmarkdown")` in the R console[2]. Once you have the package installed, you can create a new R Markdown document in RStudio by going to `File > New File > R Markdown`.

An R Markdown document is made up of text written in markdown syntax and chunks of R code. When you render the document, the R code is executed and its output (such as plots or tables) is inserted into the final document[1].

When you render this document, the text and code will be combined to create an HTML file that includes both the markdown text and the output of the R code chunks.

You can easily add images to an R Markdown document using the standard markdown syntax for images. The basic syntax for adding an image is `![Alt text](image_url)`, where `Alt text` is the text that will be displayed if the image cannot be loaded, and `image_url` is the URL of the image you want to include.

I hope this helps you understand how to add images to an R Markdown document! Let me know if you have any further questions

R Markdown is a powerful tool that offers many advantages for data analysis and reporting. Some of the key benefits of using R Markdown include:

1. **Reproducibility**: R Markdown documents are fully reproducible, meaning that anyone can re-run the code and generate the same results. This makes it easy to share your work with others and ensure that your results are accurate and reliable.

2. **Flexibility**: R Markdown is incredibly flexible and can be used to create a wide variety of output formats, including HTML, PDF, and Microsoft Word documents. You can even create interactive documents with Shiny components.

3. **Ease of use**: R Markdown is easy to use, even for people with little or no programming experience. The core syntax of markdown is simple and intuitive, and the ability to embed R code directly into the document makes it easy to include dynamic content.

4. **Integration with R**: R Markdown is tightly integrated with R, making it easy to access and use the full power of the R language for data analysis and visualization.

5. **Collaboration**: R Markdown makes it easy to collaborate with others on data analysis projects. You can share your code and results with others, and they can easily reproduce your work and build on it.

Overall, R Markdown is a powerful tool that offers many advantages for data analysis and reporting. It's a great way to create dynamic, reproducible documents that are easy to share and collaborate on

I hope this introduction helps you understand what R Markdown is and how it can be used. If you want to learn more, there are many great resources available online, including the R Markdown website

3.2 Applying EDA(WVS/own data)

3.2.1 Exercise - 1

Boston Housing Dataset

Housing data contains 506 census tracts of Boston from the 1970 census. The dataframe BostonHousing contains the original data by Harrison and Rubinfeld (1979), the dataframe BostonHousing2 the corrected version with additional spatial information.

The dataset has two prototasks: `nox`, in which the nitrous oxide level is to be predicted; and `price`, in which the median value of a home is to be predicted.

Other Details: *Origin* -The origin of the boston housing data is Natural. *Usage* -This dataset may be used for Assessment. *Number of Cases* -The dataset contains a total of 506 cases. *Order* -The order of the cases is mysterious. *Variables* -There are 14 attributes in each case of the dataset. They are: 1. CRIM - per capita crime rate by town 2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft. 3. INDUS - proportion of non-retail business acres per town. 4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise) 5. NOX - nitric oxides concentration (parts per 10 million) 6. RM - average number of rooms per dwelling 7. AGE - proportion of owner-occupied units built prior to 1940 8. DIS - weighted distances to five Boston employment centres 9. RAD - index of accessibility to radial highways 10. TAX - full-value property-tax rate per $10,000 11. PTRATIO - pupil-teacher ratio by town 12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town 13. LSTAT - % lower status of the population 14. MEDV - Median value of owner-occupied homes in $1000's

You can include this data by installing mlbench library:

```
#install.packages("mlbench") ## installing the library
library(mlbench) #adding the library
library(openxlsx)
library(dplyr)
data(BostonHousing2)
housing <- BostonHousing2
write.xlsx(housing, "../datasets/boston.xlsx")
```

1. **Read the dataset**: Read the Boston Housing Dataset from the Excel file.

```
library(readxl)
BostonHousing <- read_excel("../datasets/boston.xlsx")
```

2. **Inspect the dataset**: Use the proper functions to inspect the structure and contents of the dataset. How many Categorical variables are there? How many numerical variables are there? Is there any null values?

```
str(BostonHousing)
```

```
## tibble [506 x 19] (S3: tbl_df/tbl/data.frame)
##  $ town   : chr [1:506] "Nahant" "Swampscott" "Swampscott" "Marblehead" ...
##  $ tract  : num [1:506] 2011 2021 2022 2031 2032 ...
##  $ lon    : num [1:506] -71 -71 -70.9 -70.9 -70.9 ...
##  $ lat    : num [1:506] 42.3 42.3 42.3 42.3 42.3 ...
##  $ medv   : num [1:506] 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
##  $ cmedv  : num [1:506] 24 21.6 34.7 33.4 36.2 28.7 22.9 22.1 16.5 18.9 ...
##  $ crim   : num [1:506] 0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num [1:506] 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num [1:506] 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : chr [1:506] "0" "0" "0" "0" ...
##  $ nox    : num [1:506] 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524
##  $ rm     : num [1:506] 6.58 6.42 7.18 7 7.15 ...
##  $ age    : num [1:506] 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num [1:506] 4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : num [1:506] 1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num [1:506] 296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num [1:506] 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ b      : num [1:506] 397 397 393 395 397 ...
##  $ lstat  : num [1:506] 4.98 9.14 4.03 2.94 5.33 ...
```

```
glimpse(BostonHousing)
```

```
## Rows: 506
## Columns: 19
## $ town    <chr> "Nahant", "Swampscott", "Swampscott", "Marblehead", "Marblehea~
## $ tract   <dbl> 2011, 2021, 2022, 2031, 2032, 2033, 2041, 2042, 2043, 2044, 20~
## $ lon     <dbl> -70.9550, -70.9500, -70.9360, -70.9280, -70.9220, -70.9165, -7~
## $ lat     <dbl> 42.2550, 42.2875, 42.2830, 42.2930, 42.2980, 42.3040, 42.2970,~
## $ medv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
## $ cmedv   <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 22.1, 16.5, 18.9, 15~
## $ crim    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829,~
## $ zn      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ indus   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## $ chas    <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0~
## $ nox     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524,~
## $ rm      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631,~
## $ age     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
```

```
## $ dis     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ rad     <dbl> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4,~
## $ tax     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 31~
## $ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ b       <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ lstat   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
```

```r
# Check for missing values in all columns
colSums(is.na(BostonHousing))
```

```
##    town   tract     lon     lat    medv   cmedv    crim      zn   indus    chas
##       0       0       0       0       0       0       0       0       0       0
##     nox      rm     age     dis     rad     tax ptratio       b   lstat
##       0       0       0       0       0       0       0       0       0
```

3. **Summarize categorical variables**: Create frequency table for categorical variables CHAS (Charles River dummy variable).

```r
table(BostonHousing$chas)
```

```
##
##   0   1
## 471  35
```

4. **Summarize numerical variables**: Generate summary statistics for numerical variables CRIM (per capita crime rate by town) and RM (average number of rooms per dwelling).

```r
summary(BostonHousing$crim)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##   0.00632   0.08204   0.25651   3.61352   3.67708 88.97620
```

```r
summary(BostonHousing$rm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.561   5.886   6.208   6.285   6.623   8.780
```

5. **Create new variables**: Create new variable indicating whether a house is located near the Charles River or not.

```
library(dplyr)
BostonHousing <- BostonHousing %>%
  mutate(near_river = case_when(chas == 1 ~ "Yes",
                                chas != 1 ~ "No"))
```

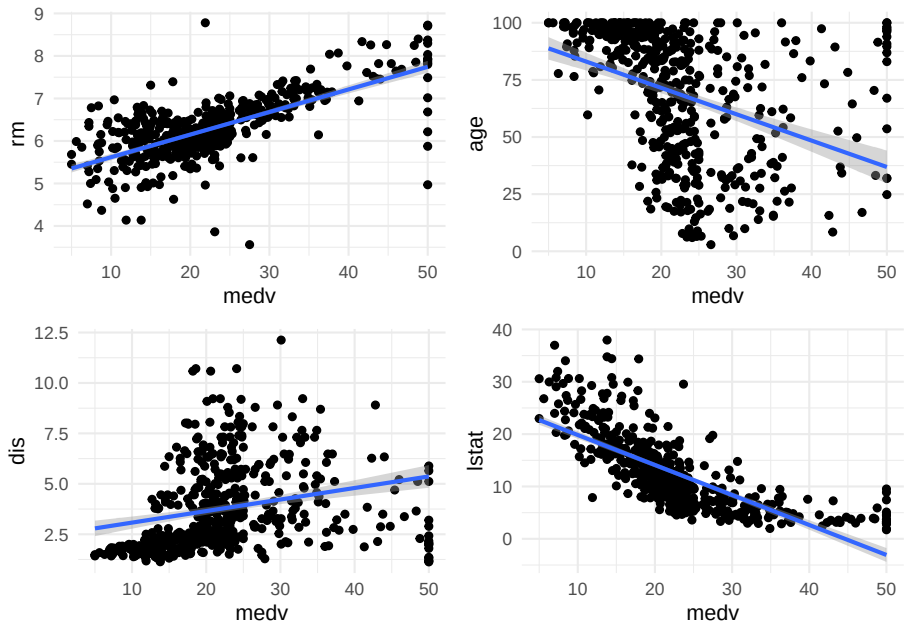
6. **Recoding variables**: Create a new variable that groups houses by their proximity to employment centers.

```
BostonHousing <- BostonHousing %>%
  mutate(distance_group = case_when(
    dis < 2 ~ "Near",
    dis >= 2 & dis < 5 ~ "Medium",
    dis >= 5 ~ "Far"
  ))
```

7. **Visualize relationships**: Create scatter plots to explore the relationship between housing prices and other numeric variables(`rm`, `age`, `dis`, `lstat`). Interpret the plots. Also show the distribution of ptratio by distance_group.

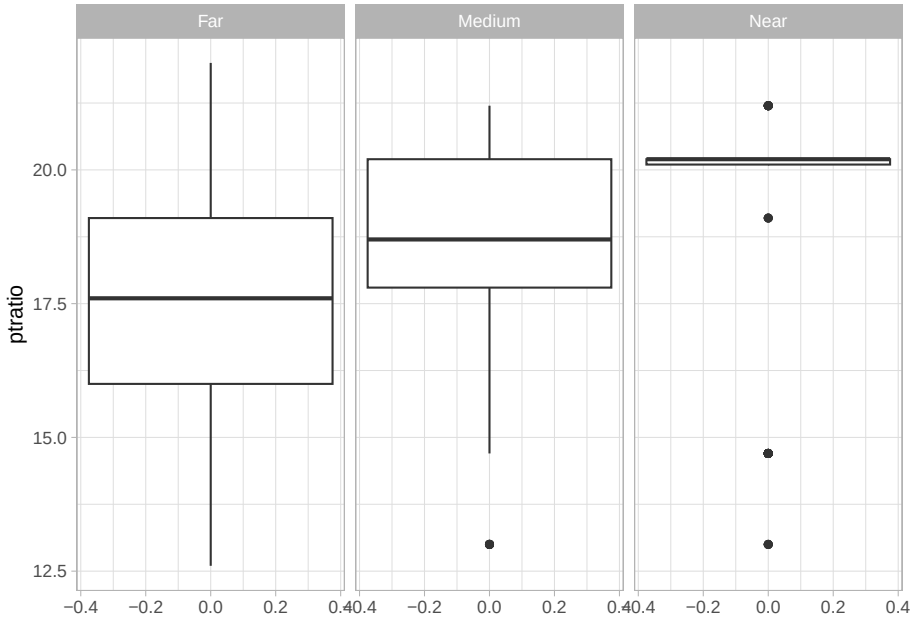
```
# Create a list of numeric variables to plot against MEDV
vars <- c("rm", "age", "dis", "lstat")

# Create a scatter plot for each variable
#rm
rm_sp <- BostonHousing %>%
  ggplot(aes(y=rm, x=medv)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()
#age
age_sp <- BostonHousing %>%
  ggplot(aes(y=age, x=medv)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()
#dis
dis_sp <- BostonHousing %>%
  ggplot(aes(y=dis, x=medv)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()
#lstat
lstat_sp <- BostonHousing %>%
```

```
  ggplot(aes(y=lstat, x=medv)) +
    geom_point() +
     stat_smooth(method = "lm") +
       theme_minimal()
#adding all plots in one
library("ggpubr")
ggarrange(rm_sp, age_sp,
          dis_sp, lstat_sp,
          ncol = 2, nrow = 2)
```



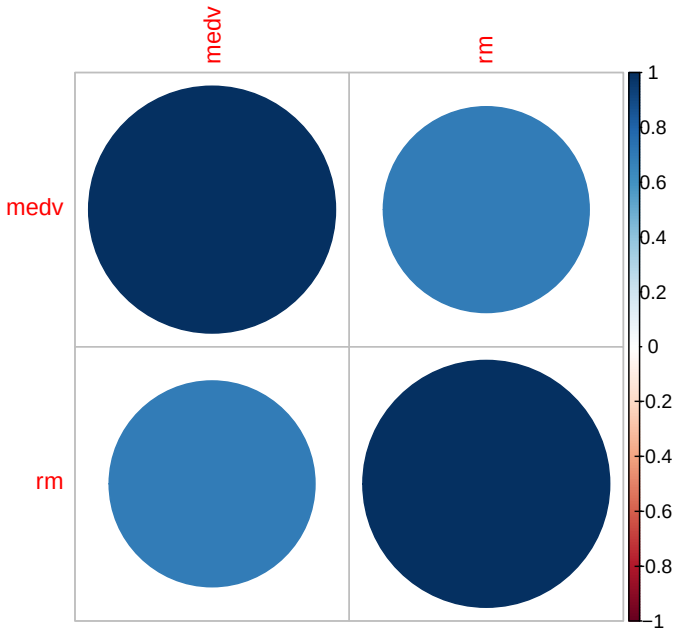
boxplot

```
# Create a boxplot of PTRATIO grouped by distance_group
BostonHousing %>% ggplot() +
geom_boxplot(aes(y=ptratio)) +
  facet_wrap(~distance_group) +
  theme_light()
```

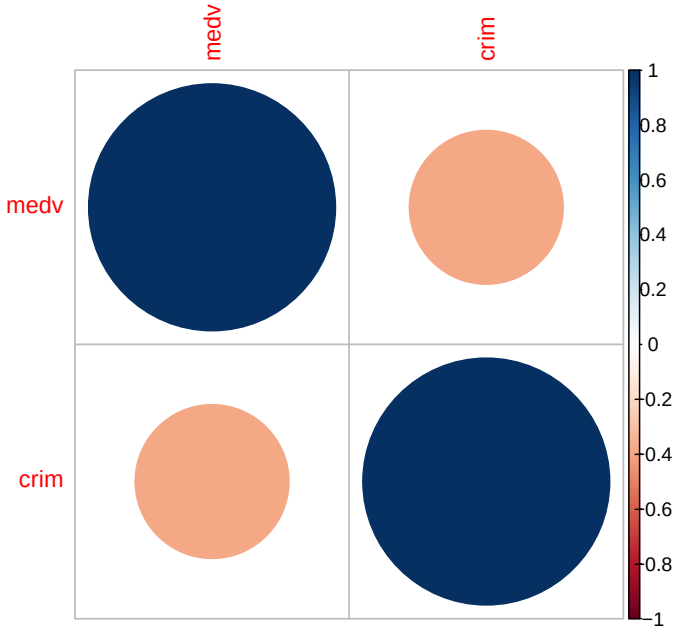


8. **Correlation analysis**: Calculate correlation coefficients between housing prices and average number of rooms per dwelling, also between housing prices and crime rate. Interpret the result.

```
library(corrplot)
corr_matrix0 <- cor(BostonHousing[, c("medv", "rm")])
corr_matrix1 <- cor(BostonHousing[, c("medv", "crim")])
corrplot(corr_matrix0)
```




```
corrplot(corr_matrix1)
```



Chapter 4

DAGs

4.1 Objectives

- Getting to know DAGs
- Understand its implications on achieving unbiased estimates for effects of interest
- Building a DAG for the NBA data to model the effect of scored points on salary (in session 8)

4.2 Modelling

This session kicks off the block on linear regression as a statistic modelling technique, but before we jump into the deep end and learn what linear regression is and how we can apply it, we first have to understand what modelling is and why we might do it. We also need a way to find out what we actually have to include in the model and what we might not want or even can not include to achieve robust results. This is what this session is all about.

4.2.1 What is modelling?

Before we approach this question, we should briefly think about what steps a typical data analysis project comprises. We usually start with an interesting problem and derive a research question from it. Based on this question we would go into the literature and read up on theories and already published research papers that are relevant to our question. We construct a theoretical framework for our particular problem and formulate some hypotheses, collect or identify appropriate data and conduct an exploratory data analysis. At this point we

should already have a firm understanding about what we actually want to find out and how our data is structured. The next step would be modelling.

So what is modelling? In general we have one *dependent variable*, typically denoted as y. This variable has some varying values. Our goal in modelling is to estimate how these values are generated. Generating here refers to the *data generating process* that we assume responsible for y having the values it has. One or multiple *independent variables*, x_1 x_2 ... x_k, influence how the values of y are generated; thus y *depends* on the values of our independent variable(s).

When we model, we do not know the data generating process, but based on theoretical considerations, careful thinking and exploratory data analysis, we can make assumptions on how we think the process operates. DAGs are a tool that can assist us in this step. We can use it to formally clarify our assumptions on the data generating process and to formulate a model based on its implications.

4.2.2 Estimating effects vs. prediction

There are two main reasons for modelling in the social sciences.

Our goal can be *prediction*, as in predicting our dependent variable y with the highest possible accuracy. This maybe is the less classical approach to modelling, but one that has come to the forefront in recent years, especially in the context of *machine learning*.

Take ChatGPT for instance. The underlying GPT model, a large language model (LLM), is used to predict what the next word in a sequence of words should be. Based on the context of the question and the prior words in the answer, which we can understand as independent variables for our example, it calculates what word has the highest probability of being the correct next one. It is all about prediction.

An example closer to home are annotations for text data. Imagine you have a lot of text, hundred thousands of social media posts, and you want to explore the sentiment expressed in those. Do they lean to the positive or to the negative? You can now go and take a "small" sample, let us say a few thousands and annotate them by hand. A lot of work, but based on those manually annotated posts you can train a machine learning model that learns from those posts and then, if everything goes well, is able to automatically annotate the remaining hundred thousands of posts for you. Again, this is all about prediction; here predicting the sentiment of a post, the dependent variable, based on the words it contains, the independent variables.

When prediction is our goal, we most often are not primarily interested in understanding what independent variables influence the dependent variables in which direction and with which magnitude. We are interested in the most accurate prediction for the dependent variable possible. These approaches are therefore also called *y-centered*.

When our interest is focused on one or multiple independent variables, our approach is *x-centered*. This is the more classical usage of modelling in statistics, at least for the social sciences. Here our goal is to estimate an effect of interest as accurately as possible. y is still our dependent variable but our focus lies on understanding which x variables influence y, in which direction this influence goes and what the magnitude is.

Let us say we are interested in why people cast their vote for a certain party. We may have some hypotheses that proposes that voters who find certain issues important have a higher probability of casting their ballot for this party. Our interest would not be predicting the vote accurately but explaining why someone votes the way they do. We can build a model from our assumptions, maybe there are other important factors that correlate with the issues and the vote, and test our hypotheses based on the results. Does holding certain issues important really increase the chances of voting for this party or is there no effect?

Over the last sessions we build an interest in the relationship between points scored and the salary received for NBA players. We could approach this as a prediction problem, i.e. trying to predict the salary as accurately as possible based on a model that incorporates the scored points as well as other factors that we assume of having an effect on the salary. We will return to this in session 11. We could also approach this as estimation problem, i.e. trying to estimate the effect of scored points on the salary. We will most probably have to include other variables that affect the relationship of score on salary as well, but the model used will not necessarily be the same. This approach is what we will tackle in this and the next 5 sessions.

Having settled on estimating the effect of score on salary, how can we find out which variables we have to include in the model? The first step, and we can never replace this with ever so fancy a statistical technique, is thinking about the problem. We should also have a theoretical understanding of our problem, know the current research on the topic and do some exploratory data analysis. Based on this we will already have developed some assumptions concerning our proposed underlying data generating process. Should we now throw everything into our model that we deem interesting or relevant for the relationship between score and salary? No, we should not. What we should do is use a tool that helps us formalise our assumptions and figure out which variables are actually relevant for measuring our effect of interest; and which variables we can not include in our model as they would potentially lead to incorrect estimations. This is where DAGs come in.

4.3 DAGs

4.3.1 Directed acyclical graphs

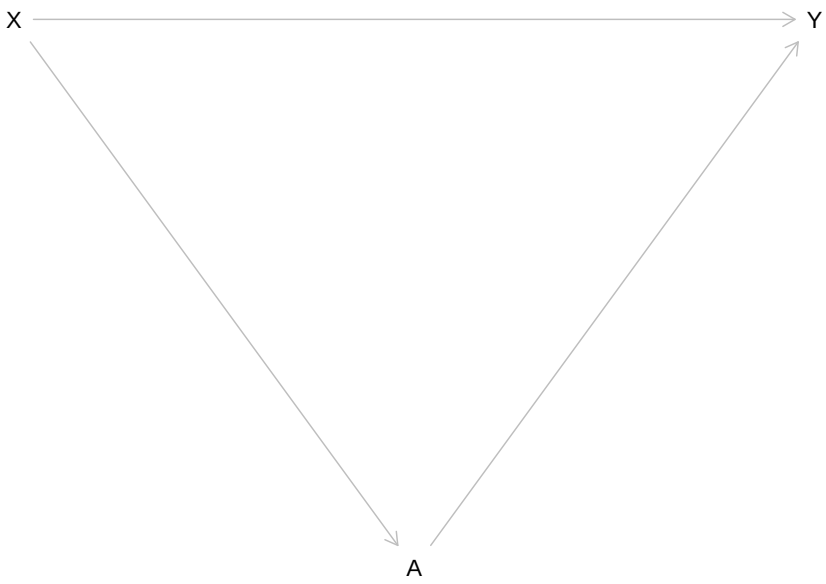
DAG is short for *directed acyclical graph*. DAGs are *graphs* that display the assumed relationship between variables as arrows, or missing arrows, between them. An arrow represents our assumption that one variable has an effect on the other, a missing arrow represents our assumption that one variable has no effect on the other. These arrows are *directed*. This represents our assumptions about the direction of the effect. We do not only assume that two variables are somehow associated, but we explicitly state which one influences the other.

A simple DAG could look like this:

What are the assumptions about the data generating process we have encoded here? We have one dependent variable X that we assume to have a direct effect on the independent variable Y. We know that our assumption was that X has an effect on Y and not the other way around, because the arrow is *directed* from X to Y.

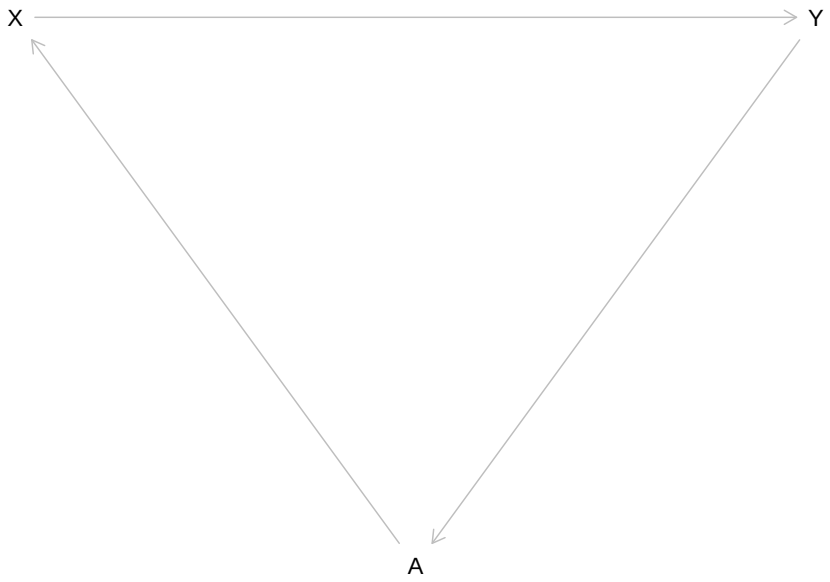
We call the sequence of one or many arrows that do not pass a single variable more than once a *path*. When trying to estimate an effect of interest we are foremost interested in the paths going from our independent variable of interest to the dependent variable. In the first example we only have the one path $X \rightarrow Y$, but in most "real" DAGs we will have multiple paths between X and Y.

Consider this DAG, were we introduce a second variable A:



Here there are two paths from X to Y, $X \rightarrow Y$ and $X \rightarrow A \rightarrow Y$. Our assumption here was that X directly influences Y, but that X also directly influences A which in turn directly influences Y. The latter is an indirect effect from X on Y through A.

DAGs are also *acyclical*, meaning that there can not be any cyclical relationships between variables. A cyclical relationship would be present if we start from one variable and follow a path that leads us back to this variable.



In this example there is a path $X \to Y \to A \to X$ that leads us from X back to X. This is not allowed in a DAG.

Now that we know the basics, what do we actually do with a DAG? A DAG is a way to graphically formalise our assumptions about the data generating process. But it is about more than drawing nice formalisations, it is also about figuring out which variables we have to include and which we are not allowed to include to get an unbiased estimate of our effect of interest. We do this by *blocking* all paths from X to Y that do not represent the relationship we want to estimate and at the same time opening up all paths that do. For this to make sense, we need to know the three patterns of relationships between a set of three variables and how we can open or close paths with them.

4.3.2 Patterns of relationships

4.3.2.1 Chains/Pipe

Three variables can be connected in a *chain* or *pipe* like this:

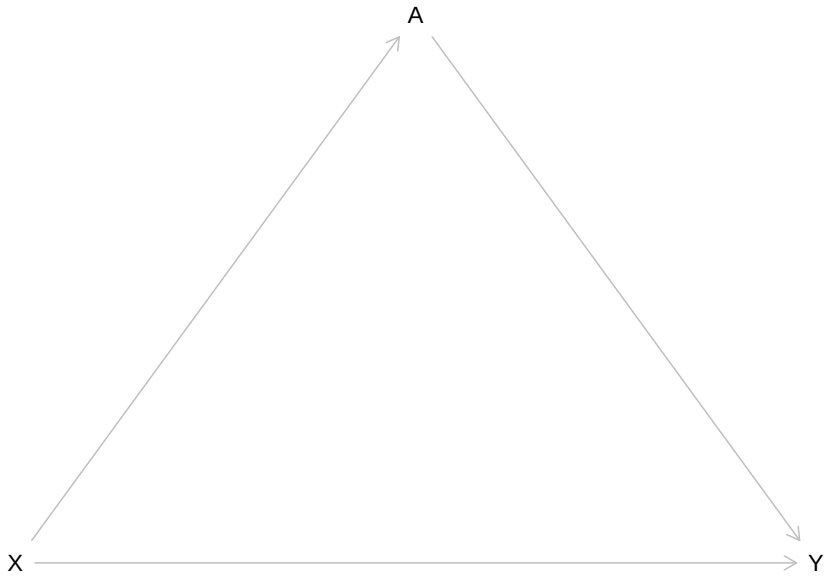
X has an effect on A which in turn affects the value of Y. This implies that X and Y are statistically correlated. When the value of X changes, the value of Y also changes, "transmitted" by the indirect effect X has on Y through A. Remember we are still interested in measuring the effect of X on Y, so we also want to measure this indirect effect.

The DAG tells us that there is a relationship between all three variables. We therefore could be tempted to include A in our model as well. But what would happen is that by including A we would *block* the path between X and Y. We would not be able to measure the association we are actually interested in. Including such a variable and thereby blocking a path of interest is called *overcontrol bias*.

In some cases overcontrolling can make the effect of interest unmeasurable. We would then conclude from our analysis that X has no effect on Y and that our hypotheses was wrong, while there actually could be an effect that we made "disappear" by blocking its path. Drawing a DAG based on our assumptions helps us to prevent this pitfall.

4.3.2.1.1 Mediation A special case of pipes is *mediation*. We will only take about this briefly here and return to the topic in session 10.

Consider this DAG:



We see a direct effect through the path $X \to Y$ as well as indirect effect through $X \to A \to Y$. Should we include A in our model? This depends on what we want to measure.

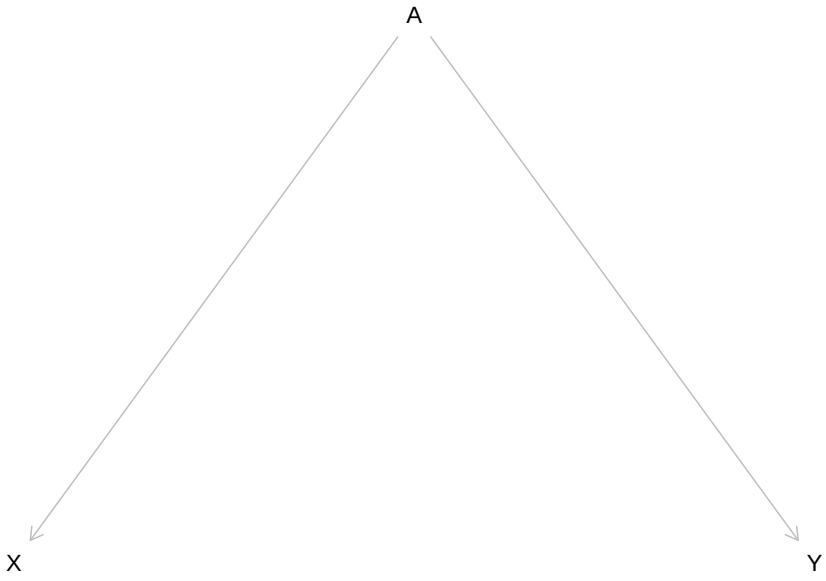
If we are interested in the *total effect* of X on Y, we should not. Both effects, the direct and the indirect paths, are of interest here, so we keep both paths open by not controlling for A.

Our interest could also lie exclusively in the *direct effect*. Here we would want to measure the effect of X stripped by all indirect effects. In this case we want the path $X \to Y$ to keep open, but we would close the path $X \to A \to Y$ by controlling for A.

We could also only be interested in the *indirect effect*, the effect of X on Y that goes through A. We can not directly model this, but we can compute the indirect effect as the difference between total and direct effect.

4.3.2.2 Confounders

The second pattern we may see is the *fork* or *confounder*.

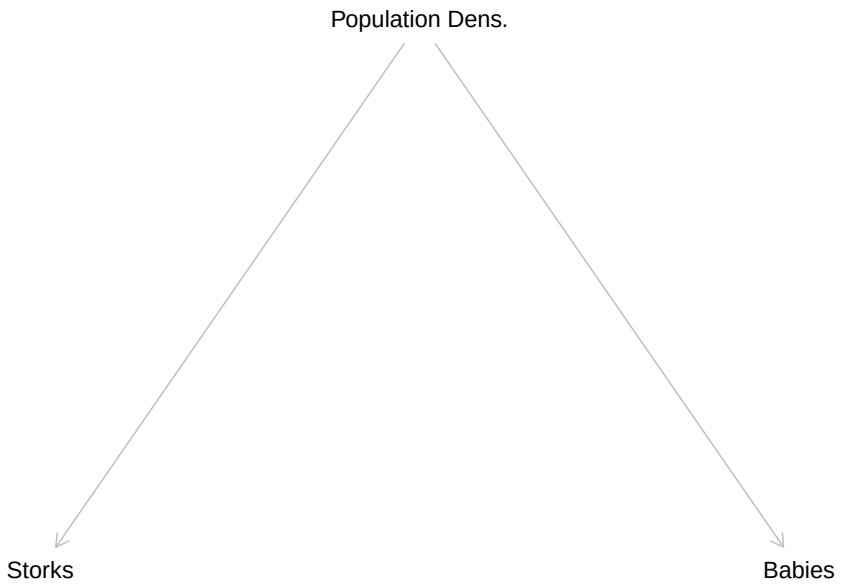


We see that there is no implied direct relationship between X and Y but that both variables are influenced by A. If we would measure the relationship between X and Y we *should* see no statistical correlation. The problem is, that we *would* see a correlation despite this. Why is that? The values of X and Y both depend on the value of A. If we for example assume that both effects are positive the value of X would rise with the value of A and the value of Y would rise with the value of A also. The opposite would be true if both effects were negative. Lower values for A would lead to lower values for X and Y. But even the effects would be opposite, they would never cancel each other out perfectly. X and Y vary together. If we just include X and Y in our model this would show up as an effect from X on Y.

In DAG terms, there is an open path between X and Y although we did not draw a direct path: $X \leftarrow A \rightarrow Y$. This may seem counterintuitive at first, as the arrows go in opposite directions, but for paths in a DAG to exist, the direction of the arrows does not matter. Every connection between variables is a path. So there is an open path, in this case a so called *backdoor path*, that leads to a statistical association between X and Y, but we can close it by controlling for A. If we do this, we would see no remaining association between X and Y and thus get an unbiased estimate for the effect of interest, i.e. no effect.

If this still seems unintuitive consider the following example, which you can also find in many statistical textbooks. Do storks bring babies? We could tackle this analytically by taking a measure of babies born in a region as our dependent variable and the number of storksightings in the same region as our dependent variable. Let statistics come to the rescue and help us discard the notion of

the baby bringing stork once and for all. But alas, our model will tell us that there is a positive effect from storksightings on the number of newborns. Should we conclude that everything we learned from our parents and teachers was one big lie to hide away the magical truth about reproduction? Before we do that, let us return to rationality. Maybe we have missed something important. It turns out that there is a confounder we did not include in our model. More rural areas have higher birthrates and also have a higher rate of storksightings while both variables have lower values in more urban regions. Our dependent and independent variables both vary by the value of the confounder and thus it seems that there is a correlation where there actually is none. If we now include a measure for the type of region, let us say population density, this spurious association disappears and we can return to normalitty. Storks do not bring babies after all.



4.3.2.3 Colliders

The last pattern we have to consider are *colliders*. A collider is a variable on a path that has two arrows pointing into it:

Here A is influenced by the value of X as well as Y. Again, there should be no effect of X on Y and in this case there is none if just include X and Y in our model. If we also include A we introduce an association between X and Y even if there should be none. This implies that we should not control for colliders because we would open up a path that creates a spurious association between two variables that are not related.

4.3.3 Adjustment set

Now we have all the building blocks for identifying which variables we have to include in our model and which we are not allowed to include. If we do not follow these rules we may statistically find relationships where there are none or miss relationships that actually exist. We then would draw the wrong conclusions for our hypotheses and research question. We would produce bad science.

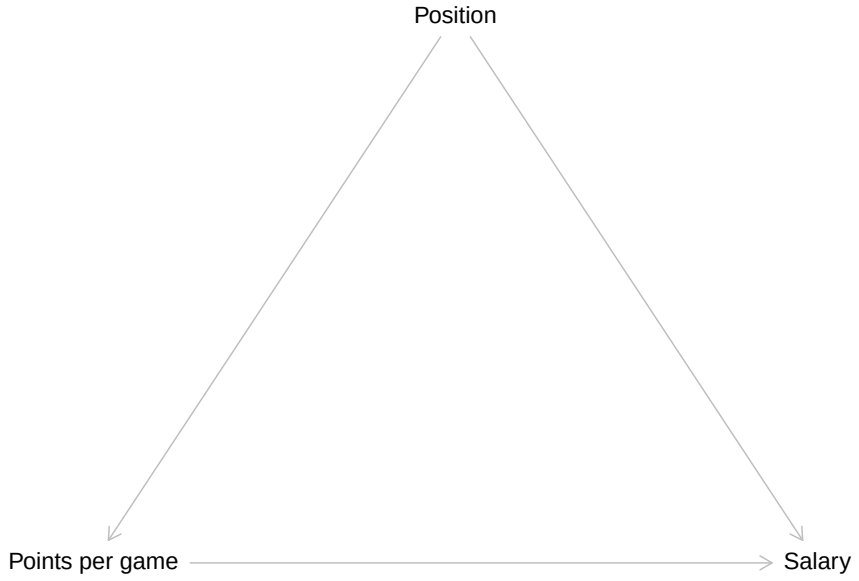
If we draw out our DAG and use its implications to identify the correct *adjustment set* Z of control variables, we do not fall into this trap. We only control what we have to, and nothing that we should not. We thus create the best model to get an unbiased estimate for our effect of interest; but there is always a caveat and this is a big one. The model is only correct if our DAG is also correct and we can never know for certain if it is. We could make wrong assumptions, forget important relationships, and make all matters of mistakes while building our DAG. While DAGs are a great tool for identifying the adjustment set, the technique alone can never replace careful thinking.

4.4 NBA DAG

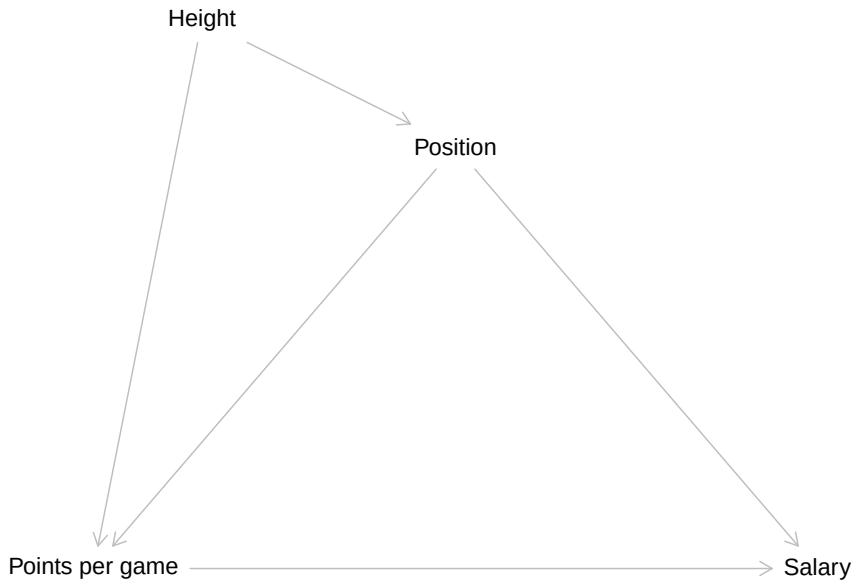
We will now pick up where we left off in session 2 and return to the NBA data. Equipped with our new tool we can now draw a DAG with the goal of building a model to estimate the effect of points scored on the salary a player receives. The assumption is, that the higher the point average, the higher the salary. This makes intuitive sense as a high scoring player is more valuable to the team and thus may receive a higher monetary compensation.

Let us start building a DAG with the information we already have.

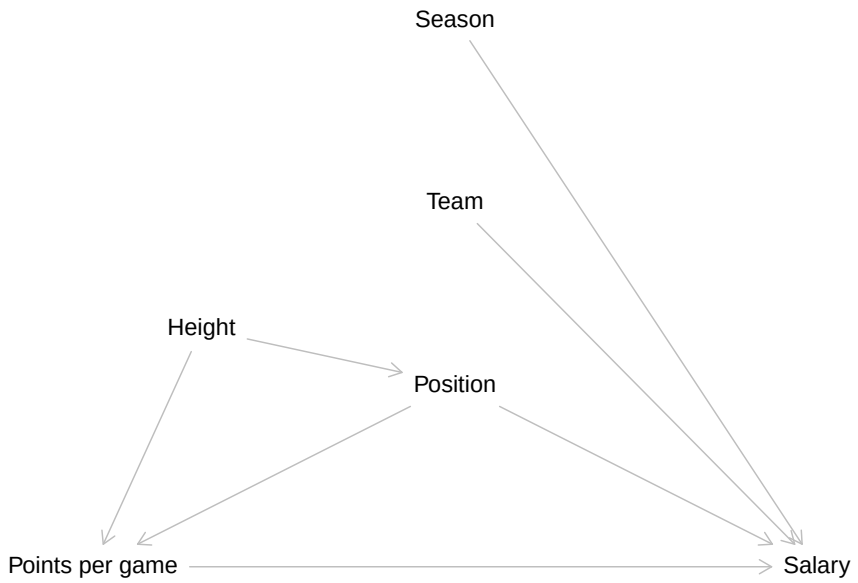
Now we have to think about other factors that could influence the relationship between points and salary. One variable we already identified as having an effect on both was the position a player occupies. The position influences how many points per game a player can score and we also already saw that centers make more money compared to point guards. Right now we have no reason to believe that other positions do not also have an effect on the received salary. Following this reasoning, position is a confounder for points and salary.



It is also reasonable that body height influences which position a player can occupy. Centers have to be big while smaller players tend to play on other positions. At the same time, height is an advantage if you want to score in a basketball game. Thus height is a confounder for points and position.



Another factor that will have an effect on the salary is the team a player plays for. More successful teams will be able to pay higher salaries. The season an observation was recorded in should also influence the paid salary as we can expect a general inflation of salaries over time.



This is our final DAG. Does it correctly depict the underlying data generating process? We do not know for sure, but the DAG correctly reflects our assumptions at this point. Those may be wrong. Maybe the data generating process actually works slightly or even completely different; but to the best of our current knowledge, this is how the scored points affect the salary a player receives.

We can now inspect what implications the DAG has for our model. To do this, let us list the paths from our independent to the dependent variable.

$$A : Points \rightarrow Salary$$

$$B : Points \leftarrow Position \rightarrow Salary$$

$$C : Points \leftarrow Height \rightarrow Position \rightarrow Salary$$

Path A is a direct path from our independent variable to our dependent variable. We will have to include scored points and salary into our model, but that is a given.

Paths B & C are both backdoor paths - these are easily spotable by an arrow pointing into the independent variable - that we have to address somehow. Let

us consider B first. Position is a confounder for points and salary. Above we already learned how to deal with this, we control for it. This removes the spurious part of the association between points and salary introduced by the confounder. Path C also includes a confounder, namely the body height of a player. Do we also control for this variable to close path C? No, we do not. If we further examine path C we will see that it also includes position as a pipe. When we control for a variable in a pipe, the path gets closed. As we have to include position to close path B, path C is also already closed.

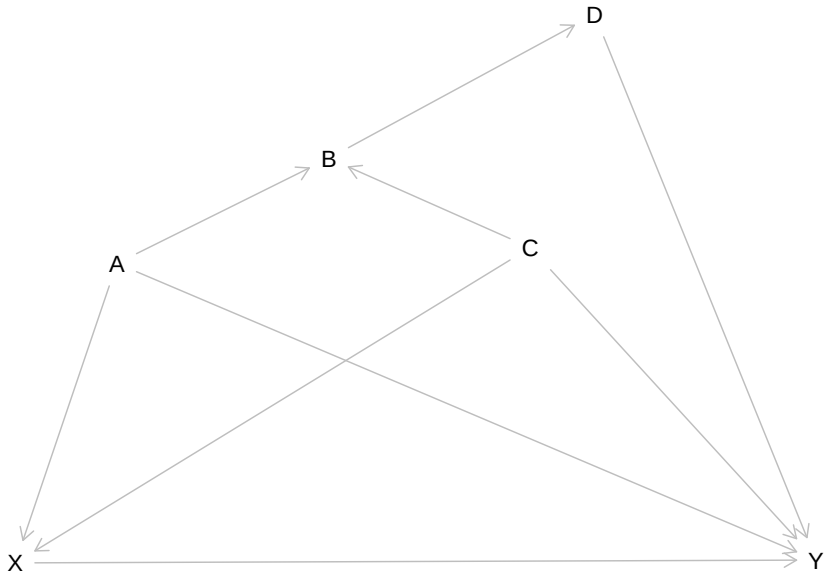
The two remaining variables, team and season, have direct effects on the salary but do not lie on a path from points to salary. This implies, that we do **not** have to control for them if our goal is estimating the effect of points on salary.

Above we briefly talked about the two possible goals of modelling. Here and over the next sessions our goal is estimating an effect of interest without bias. We used the DAG to identify an adjustment set of variables we have to control for to reach this goal. If our goal was predicting the salary as accurately as possible we would made other conclusions. The team a player is employed by and the season where an observation was measured are both relevant predictors for the received salary. For prediction we would have included both variables because this should increase the accuracy of the prediction. We will return to this in more detail in sessions 11 and 12.

4.5 Resources

4.5.1 dagitty.net

While the underlying rules of DAGs are relatively straightforward, identifying the adjustment set can get harder the more complex a DAG gets. Consider this for instance:



What variables do we have to include in our adjustment set to get unbiased measure for the effect from X on Y? We would suggest you take out pen and paper and try to figure it out. You can approach this as above. List all paths from X to Y that do not include a variable twice and try to find the adjustment set that closes all paths besides the one(s) needed to measure the effect from X on Y. Spoiler: In this case the only path we want to keep open is the direct effect $X \rightarrow Y$.

While this is doable, you may be unsure if you found the correct set. Luckily there is a convenient way to find out. https://dagitty.net/ provides a browser tool where you can draw DAGs - as well as export them for your papers - and check for the correct adjustment set. After launching the tool you should click on "Model" and then "New model". Now you can start drawing. New variables are added by clicking on the canvas and naming them, arrows are added by first clicking on the variable where the arrow should start and then on the variable where it should end. For everything to work you should also declare the independent variable of interest the "Exposure" and the dependent variable the "Outcome". Both can be set on the left under "Variable" When the DAG is drawn correctly, the top right panel "Causal effect identification" should show which variables need to be part of the adjustment set to estimate the effect of interest.

Note that "Causal effect identification" is set to display the adjustment set for the total effect. This also is what we are interested in here. If we are interested in mediation, we can also set the panel to display the set for the direct effect.

4.5.2 More on DAGs

If you are interested in diving deeper into DAGs, we can recommend this resources, which were also used for writing this session.

Richard McElreath provides a great introduction into the topic with many clear examples. While the book chapter on DAGs may require some knowledge of advanced statistical topics, the corresponding lecture on YouTube is more approachable:

McElreath, Richard (2020). Statistical Rethinking: A Bayesian Course with Examples in R and STAN. Second Edition. Boca Raton & Oxon: CRC Press. Chapter 5: The Many Variables & The Spurious Waffles, 123-160.

Corresponding YouTube lecture: https://www.youtube.com/watch?v=mBEA7PKDmiY

Felix Elwert wrote a concise paper on DAGs with a perspective that is more focused on causality than we have presented here:

Elwert, Felix (2013). Graphical Causal Models. In S. L. Morgan (Hrsg.), Handbook of Causal Analysis for Social Research, 245–273. Dordrecht [u.a.]: Springer.

If you really want to get into it, the work of Judea Pearl was central for establishing DAGs. An approachable starting point would be the "Book of Why":

Pearl, Judea & Dana Mackenzie (2018). The book of why : the new science of cause and effect. New York: Basic Books.

Chapter 5

Linear Regression Theory I: Simple Linear Regression

The next three sessions will be an introduction for linear regressions. We will look at the theoretical underpinnings, the interpretation of results and the underlying assumptions of these models. For this introduction we will keep the NBA data aside and use some simulated data that plays nice with us. We will return to the NBA data in session 8 applying everything we learned to assess the effect of scored points on salary.

5.1 Objectives

- Understand simple linear regression
- Understand the regression formula
- Interpret the results

5.2 What is Linear Regression

As we eluded to last session, there are two main approaches to using statistical modelling in the social sciences. The more classical approach is to use modelling for estimating the effect that one or several independent variables have on one dependent variable. Maybe we are interested in knowing if a higher income has an effect on life satisfaction and if yes, what the direction and magnitude of this effect is. Does more money actually make you happier?

The other and more recent approach is to use modelling for making predictions with high accuracy. Based on the relationships between many independent

variables and one dependent variable. We try to predict the latter for actual or hypothetical cases based on their values for the independent variables. This approach lies at the heart of *machine learning* and drives many of the technologies we use on a daily basis from E-Mail spam filters to ChatGPT. Returning to the example above, we are not interested in measuring the effect of money on life satisfaction, but in predicting the value for life satisfaction based on money and a host of other variables as accurately as possible.

Linear regression is one of the many available modelling techniques and it can serve both approaches. Over the next sessions we will focus on using linear regression for estimating an effect of interest but we will return to prediction in session 11 & 12.

How do we know if we should choose linear regression for a specific task? This is not easy to answer as there are many alternatives and even variations of linear regression which may be better suited for a specific empirical problem. As this is an introduction to modelling and time is of the essence we opted to solely focus on linear regression. This technique is suited for many problems and is comparably easy to understand and use. Also, after learning the ins and outs of linear regression, we are in a good position to build upon that knowledge and learn all of those more complex and specific models that we will encounter in textbooks and scientific papers.

With the pool of options trimmed down to one, the central question remains unanswered. Should I use linear regression for my task? As we have no alternatives to chose from, we can change the question to: Can I use linear regression for my task? The answer no mostly depends on what type of dependent variable we want to use. If it is metric, we can use linear regression. In our cases, the simulated data and our NBA data, our dependent variables both are metric. If we had other types of dependent variables, e.g. binary or categorical, we would have to use different models. We will give you some pointers for these at a later point.

5.3 Examplary research question & data

For this introduction, et us imagine that we are interested in a research question that asks what makes a good grade in a seminar paper. In particular we are interested in the effect that the hours a student invests in working on it has on the grade. Based on some theoretical considerations, and maybe some idealistic views, we derive our main hypotheses that putting in more hours will result in a better grade.

Now we also - hypothetically - held a small survey and asked 200 imaginary students some questions on how they approached writing a seminar paper. In particular we asked them how much time they spent working on the paper, if they have attended (almost) all seminar sessions, how closely they worked with

their lecturers in preparing the paper and what the mean grade for previous papers was. As these imaginary students have already turned in their papers, we also know the grades they achieved.

Please note, that this is data on **imaginary** students, meaning we have simulated the data making some assumptions on how to achieve a good (or bad) grade in a paper. The assumptions we made do not necessarily reflect the way *you* write a good paper, while still being based in our experience on what it takes to achieve a good grade. But remember, no real students were harmed in making up this data.

Let us have a first look on the data: XXX ALIGN WITH EDA XXX

```
## Warning: package 'skimr' was built under R version 4.2.3
```

```
##
## Attaching package: 'skimr'
```

```
## The following object is masked from 'package:corrr':
##
##     focus
```

```
## Warning: package 'knitr' was built under R version 4.2.3
```

skim_type	skim_variable	n_missing	complete_rate	factor.ordered	factor.n_unique	factor
factor	contact	0	1	FALSE	3	No :
logical	attendance	0	1	NA	NA	NA
numeric	grade	0	1	NA	NA	NA
numeric	hours	0	1	NA	NA	NA
numeric	previous_grades	0	1	NA	NA	NA
numeric	previous_grades_centered	0	1	NA	NA	NA
numeric	hours_centered	0	1	NA	NA	NA

Right now, the observations are ordered by the grade of the seminar paper which run from 1.0 to 5.0 in increments of 0.1. While this is somewhat unrealistic - the German grading system actually only uses the increments .0, .3 and .7 - simulating the data in this way will make the demonstrations on linear regression easier and more straightforward. The variable `previous_grades` is set up in the same way and represents the mean of the grades the student received up to this point. `hours` represents the time a student spent on writing the paper, ranging from $23 - 57$ hours, with a mean of about 40. Besides these metric variables, the data set also contains two categorical measures. `attendance` is a *binary* or *dummy variable*, meaning it can only have the values 1 or 0 or `TRUE` and `FALSE` in this case, as it is saved as a logical variable. `TRUE` represents that a student attended almost all seminar sessions before writing the paper - which about 77 did -, `FALSE` states that they did not. `contact` is a factor variable with three

categories and shows the answers to the imaginary question on how much contact the student had to the lecturer before starting the writing process. Besides `No contact` the students could have had `E-Mail` contact to state their research question and get some short written feedback or meet the lecturer `In Person` to achieve a deeper discussion of the question and laid out plan for writing the paper. The two additional variables are versions of `previous_grades` and `hours` that are centered on their respective means. They will come into play at a later point in this session.

Let's have a look at some observations.

```
## # A tibble: 10 x 7
##    grade hours previous_grades attendance contact   previous_grades_centered
##    <dbl> <int>           <dbl> <lgl>      <fct>                       <dbl>
##  1     1    50             1.4 TRUE       E-Mail                     -1.54
##  2     1    46             1   TRUE       E-Mail                     -1.94
##  3     1    42             1   TRUE       In Person                  -1.94
##  4     1    49             1   FALSE      In Person                  -1.94
##  5     1    42             1.2 TRUE       In Person                  -1.74
##  6     1    46             1.8 TRUE       In Person                  -1.14
##  7     1    44             1.4 FALSE      In Person                  -1.54
##  8     1    45             2   TRUE       In Person                  -0.935
##  9     1    48             1   TRUE       In Person                  -1.94
## 10     1    45             2   TRUE       In Person                  -0.935
## # i 1 more variable: hours_centered <dbl>
```

From this first 10 rows, we can see that the students with the best grades spent more than 40 hours on writing, have already achieved good grades in their papers up to this point and at least had some contact to the lecturers. Most also regularly attended the seminar but two did not and still achieved a 1.0 in their grade.

So what makes a bad grade?

```
## # A tibble: 10 x 7
##    grade hours previous_grades attendance contact    previous_grades_centered
##    <dbl> <int>           <dbl> <lgl>      <fct>                         <dbl>
##  1   4.8    37             4.2 TRUE       No contact                    1.27
##  2   4.8    38             4.3 TRUE       E-Mail                        1.36
##  3   4.8    35             4.4 TRUE       E-Mail                        1.47
##  4   4.9    40             4.2 TRUE       E-Mail                        1.27
##  5   5      35             3.9 FALSE      No contact                    0.965
##  6   5      41             4.9 TRUE       No contact                    1.97
##  7   5      24             4.7 TRUE       E-Mail                        1.76
##  8   5      33             5   TRUE       E-Mail                        2.06
##  9   5      29             4.1 FALSE      E-Mail                        1.16
```

```
## 10   5      50             4.6 FALSE      E-Mail                      1.66
## # i 1 more variable: hours_centered <dbl>
```

Here the picture seems less clear. While most students did not put in as many hours, some did and still failed to pass. Half of the students that received a 5.0 regularly attended and most at least had E-Mail contact before writing their paper. What seems to be more consistent though is that the mean of the previous grades is rather low.

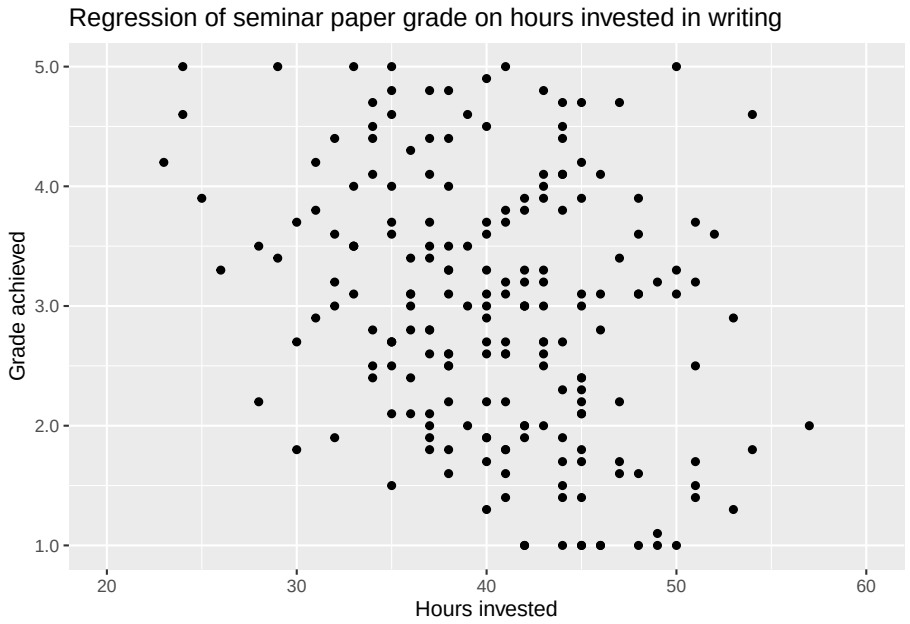
So what do we know now? Does a good or bad track record in grades predict all future grades? This seems not only unrealistic but is also a kind of sad take home message. To get a better understanding on which of the potential influential variables has an effect on the final grade and what the magnitude and direction of these effects is, we now turn to linear regression.

5.4 Simple Linear Regression

In a *simple linear regression*, the model is used to describe the relationship between *one* dependent and *one* independent or explanatory variable. The question this model can answer for us is, by how much does the dependent variable increase or decrease, when the explanatory variable increases by 1?

Returning to our exemplary research question on what makes a good grade in a seminar paper, an intuitive hypotheses would be that the grade gets better the more hours a student invests in writing the paper. In this case we assume a linear relationship between the independent variable `hours` and the dependent variable `grade`. As German grades are better the lower their value, we thus would assume a negative effect from `hours` on `grade`.

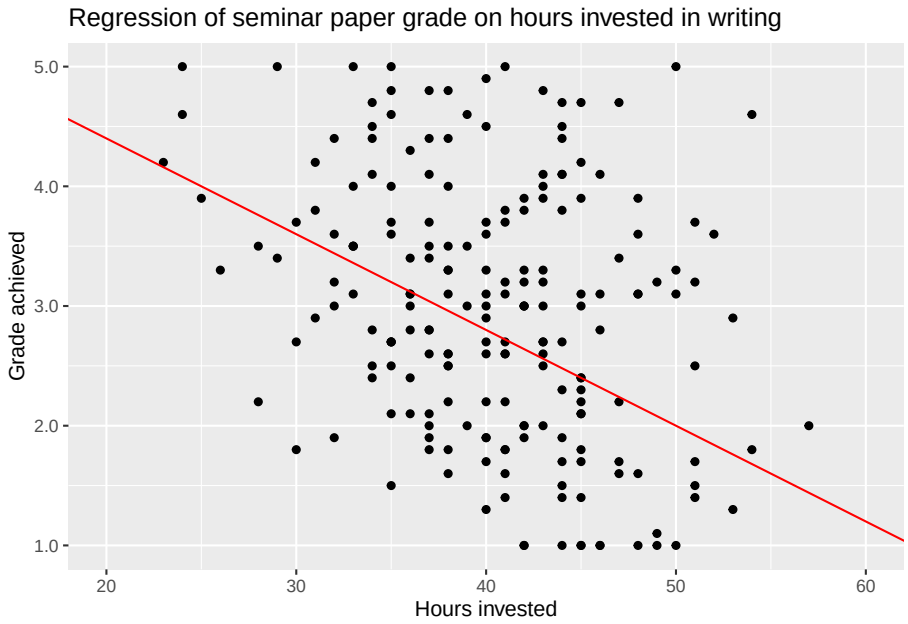
Before turning to the formalities and practical application of a simple linear regression model, let us first have a look on this relationship by plotting the variables against each other.



When we are talking about dependent and independent variables, there is the convention to plot former on the x-axis and the latter on the y-axis. So the *y-variable* is to be explained and the *x-variable* is used to explain it. This convention will also be used in all formulas in this seminar.

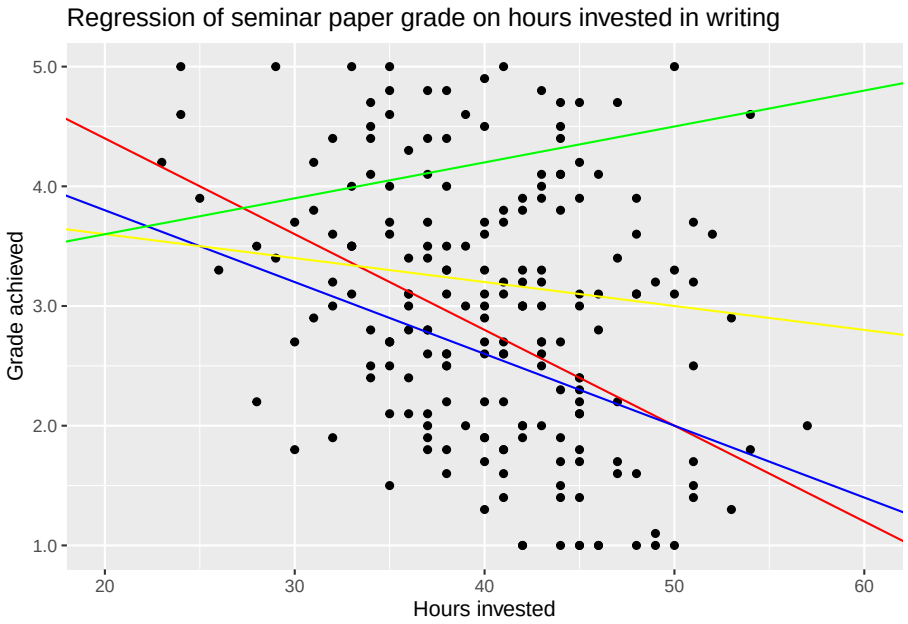
Looking at the plot we first see a cloud of dots, representing all combinations of `hours` and `grade` in all our 200 observations. It may be hard to pick out any pattern, but looking closely we can observe that overall the dots seem to follow a downward slope from the upper left - indicating few hours worked and a worse grade - towards the lower right - indicating more invested hours and a better grade. This would be the relationship stated in our hypotheses. The more hours a student works on a seminar paper the better the final grade will be.

We can try to describe this pattern by adding a line from the upper left to the lower right.



This describes the realtionship between the two variables as linear. Each hour invested decreases the grade by a certain amount, for this proposed line by exactly 0.08 points. Remember that decreasing the value of the grade actually means getting a better grade.

But is this the only possible line or even the *correct* one? Most certainly not, as the values used to draw the line were only a wild guess. We could imagine several other lines that also look more or less reasonable - as well as some that look unreasonable - and add them to the plot.



While we have some intuition that the green line completely misses the mark, we can't really decide between the others just by looking at the plot. The data points are way to dispersed to see the relationship clearly.

The goal of using a simple linear regression model is to identify the *one* line that describes the relationship the best. Here *best* means, with as little error as possible.

5.4.1 Regression Formula

To understand how these lines in the above plot were conceived and how to find the line with the best *fit*, i.e. the lowest error, we have to understand the formula for linear regression. While formulas may always be kind of daunting, we are in luck as this particular one is actually quite easy to understand, especially when paired with a graphical representation.

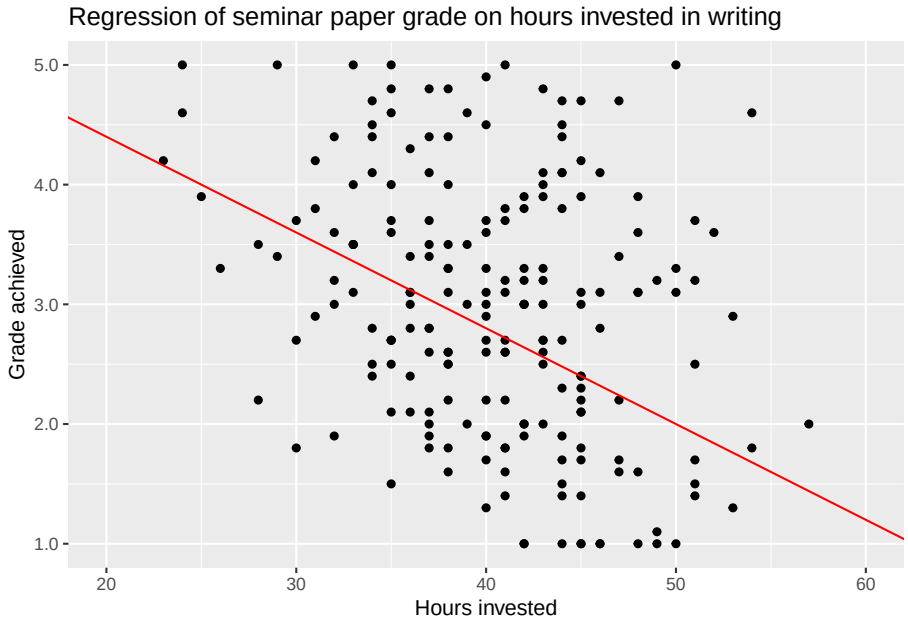
$$y = \beta_0 + \beta_1 * x_1 + \epsilon$$

Let us first look at the parts we already know. y is the dependent variable, in our case the grade achieved. So one thing is for sure, the whole right part of the equation has to be used to calculate the value of y from the data, i.e. from the dependent variable x. Here we have three terms. Let us skip the first one for now and focus on the second one $\beta_1 * x_1$.

x_1 is the dependent variable, in our case `hours`. β_1 is the *regression coefficient* for x_1. This value gives us the *slope* of the regression line. Based on this, we can start rewriting the general formula and tailor it to our specific use case.

$$y_{grade} = \beta_0 + \beta_{hours} * x_{hours} + \epsilon$$

Let us return to the first wild guess we made above.

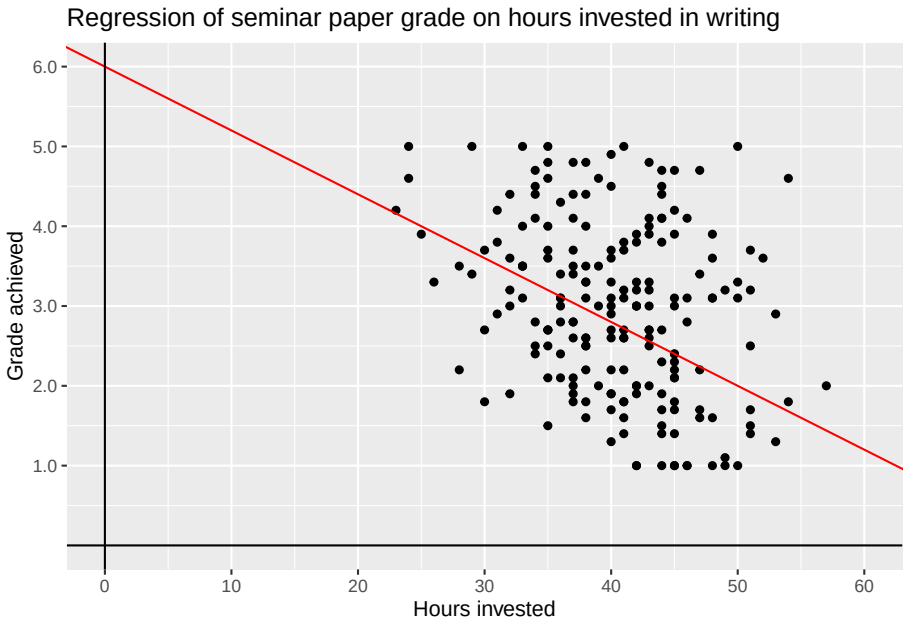


Here we guessed that an increase in invested time of one hour decreases the value of `grade` by 0.08. This is the slope of the red line and thus also the coefficient in the regression formula that is used in computing said line. So, $\beta_{hours} = -0.08$. We can insert this value into our formula.

$$y_{grade} = \beta_0 - 0.08 * x_{hours} + \epsilon$$

In this way the value of x_{hours} is multiplied by -0.08. Let us assume a student worked 40 hours on their paper. $-0.08 * 40$ being -3.2, we assume that working 40 hours on a paper *on average* - more on that later - leads to a 3.2 lower grade value. But 3.2 lower than what?

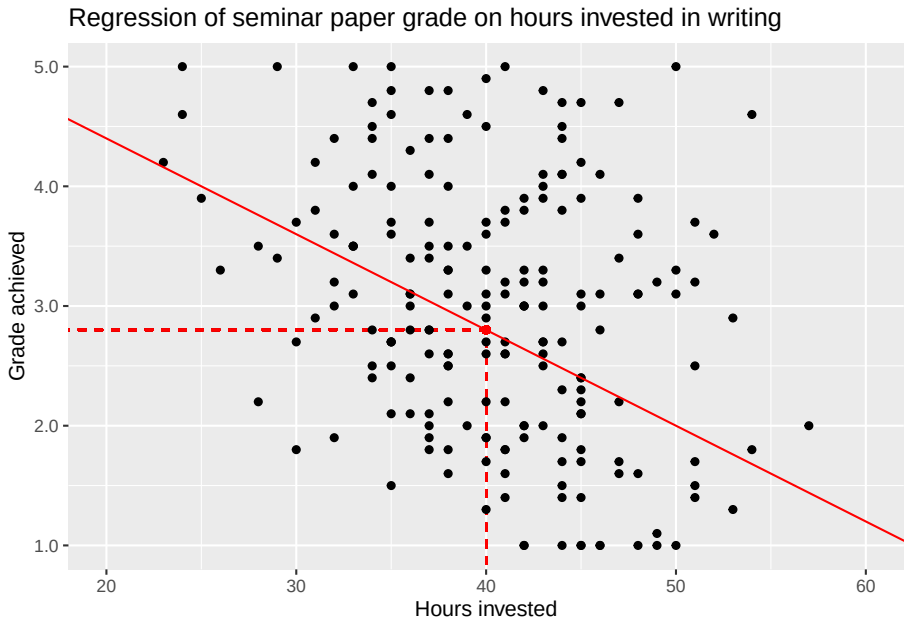
Looking at the formula again, we see that we subtract this value from β_0. This is the *intercept*, the value at which the line intersects with the y-axis. Let us zoom out on our plot to see what happens.



We can now see the point where the red line intersects with the y-axis. This is the intercept of this line, i.e. $\beta_0 = 6$.

$$y_{grade} = 6 - 0.08 * x_{hours} + \epsilon$$

If we now again assume a time investment of 40 hours, we can compute $6 - 0.08 * 40 = 2.8$. So our red regression line - which is still only a wild guess - assumes, that working 40 hours on a seminar paper will result in a grade of 2.8, on average. We can mark these values in our plot:



The red dot is the intersection of the values `hours = 40` and `grade = 2.8`. As this is the value for y our regression line assumes a student with a time investment of 40 hours achieves, the red dot also lies exactly on the red line.

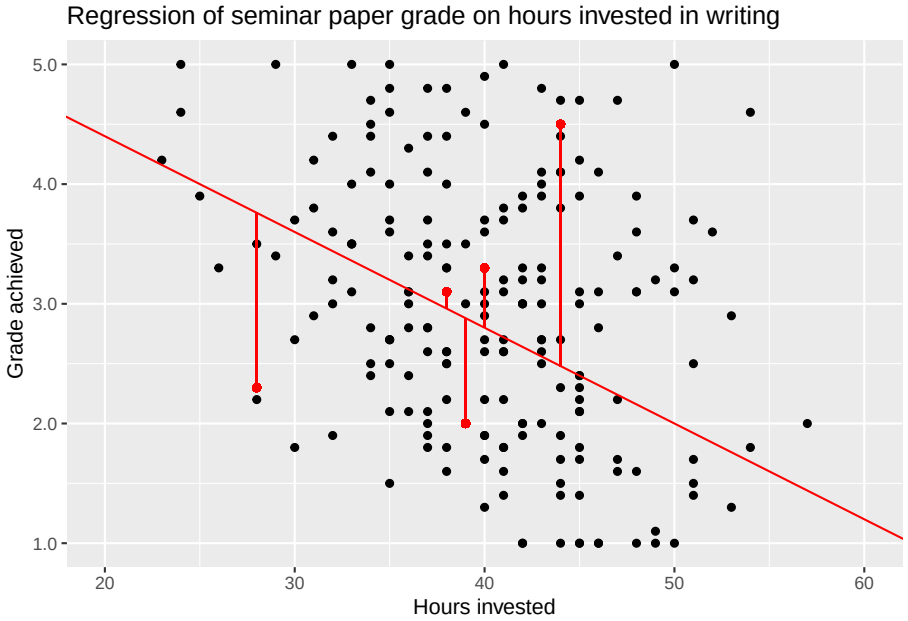
But if we look at the plot once again, we can see that most actual observations for students that invested 40 hours do not actually lie on the regression line but are scattered above and below the line. Some of these students achieve much worse or much better grades than 2.8 investing the same amount of time in their work. This leads us to the last part of the formula, ϵ.

This is the *error term*. Having data that is dispersed like this - and any real world data will always be - our linear line will never be able to pass exactly through every data point. Some points may lie exactly on the line, but many or most will not.

We can visualize this. To keep the plot readable, we only do this for some random observations but in reality the distance of every data point from the regression line is taken into account.

```
## # A tibble: 5 x 7
##   grade hours previous_grades attendance contact    previous_grades_centered
##   <dbl> <int>           <dbl> <lgl>      <fct>                         <dbl>
## 1   2.3    45             2.6 TRUE       No contact                   -0.335
## 2   1.9    44             2.8 TRUE       In Person                    -0.135
## 3   3.1    38             3.7 TRUE       In Person                     0.765
## 4   3.3    40             3.4 TRUE       E-Mail                        0.465
## 5   4.6    54             4.5 TRUE       No contact                    1.56
```

```
## # i 1 more variable: hours_centered <dbl>
```



The distance of these or rather all points from the line, the *residuals*, are represented in the error term ϵ. It is a measure for how wrong our line is in describing the data in its entirety. So why is it wrong? We can not say for sure, but there are two common main reasons.

For one, there may be other variables that also influence the relationship between invested hours and achieved grade, something that we will return to in the next session, when we expand the idea of linear regression to multiple independent variables.

But there is also random variation present in every bit of real world data. While our data is simulated we also added random variation on purpose. Because this is what real world data is, it is messy and it is noisy.

Not every seminar paper that had the same time investment, e.g. 40 hours, will have the same quality in results. There may be other influential variables, e.g. the student's general skill level or if they sought assistance by their lecturer in preparing the paper, influencing the final grade. But even if the quality of the paper after working 40 hours would be the same for each student, measurement error, i.e. noise, will be introduced because not every lecturer will grade exactly the same or maybe because papers were submitted at different time points and grading standards may have changed. If we can not measure these variables we have to accept these unobservable sources of noise and hope, where *hope* actually means thorough theoretical and methodical thinking, that we can still measure

our effect of interest. This also means, that measuring and modelling **always** includes uncertainty. We never know for certain if and to what extent our results are influenced by unobservable variables and random variation. Still, there are ways to assess this uncertainty, which we will regularly return to during the course. This should not stop quantitative social scientists from making strong or even bold arguments based in thorough theoretical thinking and responsible data analysis, but we always have to acknowledge the uncertainty included in every step and make it a part of our interpretations and conclusions.

The error term ϵ is the final piece of the puzzle in actually computing a linear regression model. Without jumping into the mathematics of it all, the technique that is used to estimate the coefficients β_0 and β_1 is called *OLS* - Ordinary Least Squares. What it basically does, is to take the squares of all residuals, i.e. the distances of the data points from the regression line, sum them up and minimise this value. All this substantially means is, that OLS searches for the regression line with the lowest amount of error, i.e. the lowest overall distance from the actual data points.

OLS gives us estimates for the regression coefficients in this formula:

$$\hat{y} = b_0 + b_1 * x_1$$

We can see two differences to the formula we started with. First, we write $\hat{y}$ - pronounced as "y hat" - instead of y. At the same time, we exclude the error term ϵ. This means that we are no longer computing the actual value of y, as in the point on the regression line for a certain value of x_1 + the error, but the estimate $\hat{y}$, as in the point on the regression line that is predicted for a certain value of x_1. Second, we write b instead of β. This also alludes to the fact that we are now computing an estimate for the coefficients based on the data available and not the real but unknown value of β.

This implies that we now estimate the same grade for every student who invested the same amount of time, the $\hat{y}$ that lies exactly on our regression line at a certain value of x_1. For all students who invested 40 hours in writing, we would estimate exactly the same grade. As we have seen above, these students received different grades in reality, or more accurately our simulated reality. The value of $\hat{y}$ is still the best guess our model can make. That is what we mean when we say "on average". On average a student is estimated to receive the grade $\hat{y}$ after investing x_1 hours in writing the paper. We have to keep in mind, that this will not be true for many students; there is always an error involved in our estimates.

5.4.2 Regressing `grade` on `hours`

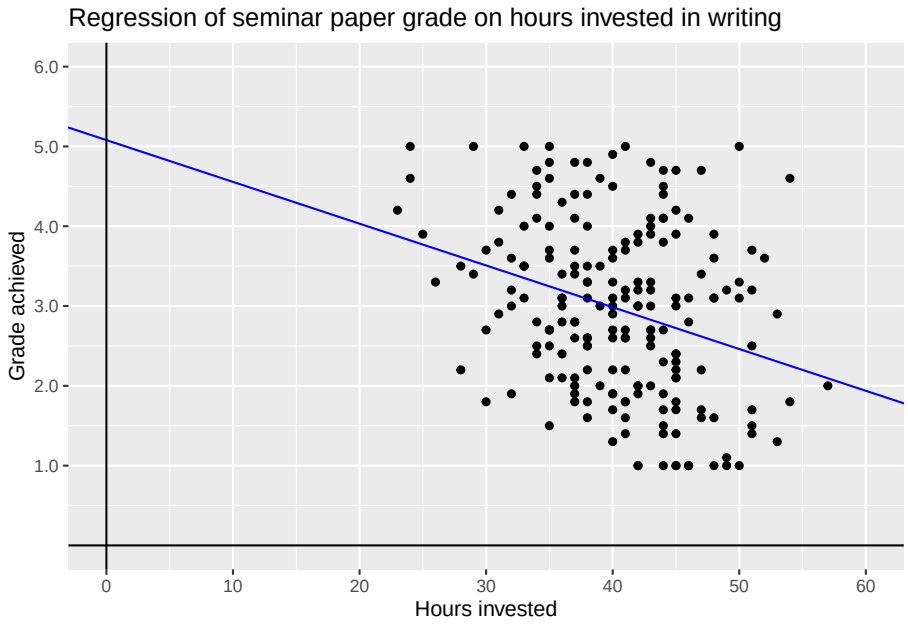
Now that we have a firmer understanding on what linear regression actually is and does, we can finally get to the fun part and use the technique for estimating the effect of `hours` on `grade` or in other words, regress `grade` on `hours`.

```
##
## Call:
## lm(formula = grade ~ hours, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88006 -0.83961 -0.08006  0.77006  2.53881
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.07912    0.47306  10.737  < 2e-16 ***
## hours       -0.05236    0.01159  -4.517 1.07e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.028 on 198 degrees of freedom
## Multiple R-squared:  0.09344,    Adjusted R-squared:  0.08886
## F-statistic: 20.41 on 1 and 198 DF,  p-value: 1.075e-05
```

This is the output from a simple linear regression for `grade` on `hours` R returns to us. How we can do this in practice and what the first two lines mean will be the topic of session 8. For now we will focus on the estimates in the coefficient block and introduce the additional elements of the output on by one over the next sessions.

The column `Estimate` gives us the values for β_0 and β_1 discussed above. The estimated coefficient for `hours` tells us that out intuition was right, the more hours a student invests in writing a paper, the better the grade will be. In this case every additional hour spent on working is estimated to decrease the value of the grade by -0.05236 points. In keeping with the example of a 40 hour workload this leads to a decrease of $-0.05236 * 40 = -2.0944$ points. Adding the intercept from the same column, the estimated grade after working 40 hours is $5.07912 - 0.05236 * 40 = 2.98472$. So on average a student from our simulated data set will pass after 40 hours of work but will not get a great grade. Remember, this is the expected average value. This does not mean that some students will not get better or worse grades, or even fail to pass with this amount of time investment.

Now that we know the coefficients for the regression line with the best fit, i.e. the lowest error, we can again visualise the result.



What grade can a student expect, on average, if they invest exactly 0 hours, i.e. do nothing and hand in a blank paper. We can look at the graph or, to achieve a more precise result, calculate it.

$$5.07912 - 0.05236 * 0 = 5.07912$$

For this theoretical example of $x_{hours} = 0$, the estimated value $\hat{y}$ or y_{grade} is the same as the intercept β_0. This is what the intercept represents in general, the estimated value $\hat{y}$ when the dependent variable equals 0.

Investing zero hours in a seminar paper is not only not advisable, it is also not a value we observed in our data. If the data would include observations with zero hours of time invested, the grade would be a firm 5.0 and the same would be true for low single digits, i.e. turning in a two-pager as a seminar paper. The takeaway is, that the model is highly dependent on the data that it is trained on. If the data would have included such cases we could expect a higher intercept and a steeper slope, i.e. stronger negative coefficient.

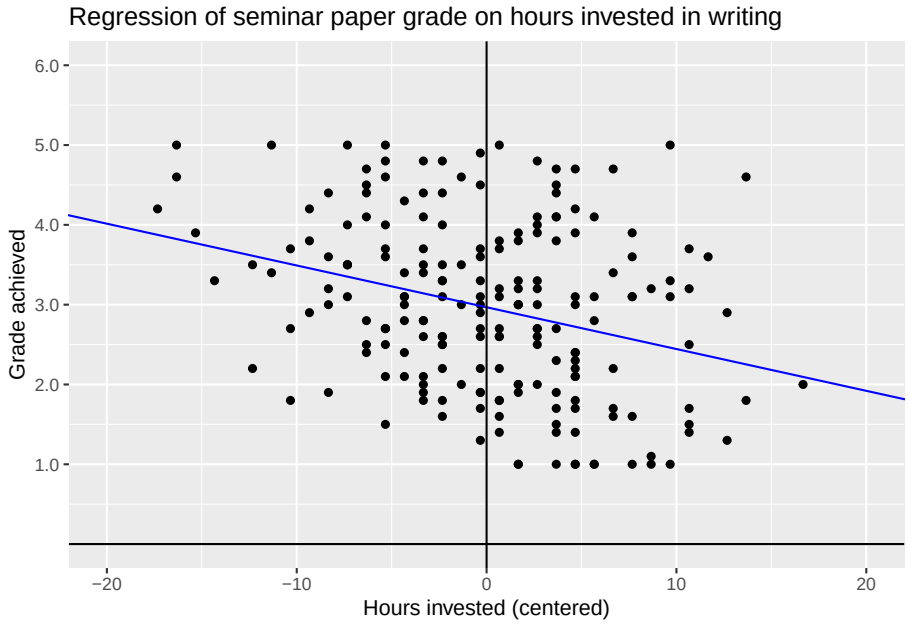
Luckily all our simulated students have put in at least some hours. But as we do not have data for zero to 22 hours, we can not really make reliable estimates in this range. Because of this, it does not really make sense to enter `hours` into the regression model as ranging from 0 to 57. One solution that is often used for metric variables is to center them on their mean. This can be achieved by simply subtracting the mean of x from each individual value:

$$x_i - \bar{x}$$

We can now rerun the regression.

```
##
## Call:
## lm(formula = grade ~ hours_centered, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88006 -0.83961 -0.08006  0.77006  2.53881
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.96750    0.07267  40.835  < 2e-16 ***
## hours_centered -0.05236    0.01159  -4.517 1.07e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.028 on 198 degrees of freedom
## Multiple R-squared:  0.09344,    Adjusted R-squared:  0.08886
## F-statistic: 20.41 on 1 and 198 DF,  p-value: 1.075e-05
```

Comparing the results to the first model shows us, that the coefficient for $b_{hours\_centered}$ is exactly the same as for b_{hours}. So the effect of working more hours has not changed. What has changed is the value of the intercept. This will make more sense if we again plot the regression line.



By centering the x-Variable on its mean we have changed its interpretation. A value of `hours = 0` now stands for investing as much time as the mean of `hours` in the whole data set, which in this case is 40.33 hours. Positive values indicate that a student worked x hours more, negatives indicate $-x$ hours less compared to the mean. In this way, we also moved the y-axis and thus changed the interpretation of the intercept. Its new value of 2.9675 now indicates the estimate for a student who invests the mean value of `hours` in their work, i.e. 40.33.

5.5 Moving on

Based on our simple linear regression model we achieved an estimate for our effect of interest. Working more hours results in receiving a better grade. But there could be other variables that influence this relationship. In the next session we learn how we can include these in our model, when we move from simple to multiple linear regression.

Chapter 6

Linear Regression Theory II: Multiple Linear Regression

Maybe explaining the grade a student receives solely based on the hours of invested time, does not paint the whole picture. As we have alluded to, there may be other variables that could affect the relationship between `hours` and `grade`. If we fail to include these in our model, we may not get an unbiased estimate for our effect of interest. Maybe the actual effect for `hours` is even stronger, maybe it is weaker, or maybe there is no effect at all. To assess this, we have to move from simple to multiple linear regression.

6.1 Objectives

- Expand the idea to multiple linear regression
- How can we interpret different types of independent variables?
- Understand measures of uncertainty

6.2 Multiple Linear Regression

A *simple linear regression* only allows for one independent variable. This is why we need *multiple linear regression* if we want to start introducing additional variables into the model. Luckily this is easy to understand as we already know the formula for a simple linear regression:

$$y = \beta_0 + \beta_1 * x_1 + \epsilon$$

To change a simple into a multiple linear regression, we just start adding the additional variables and their coefficients additively to the formula.

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_k * x_k + \epsilon$$

So to add a second variable and its coefficient we add the term $+\beta_2 * x_2$ and so on until we added all independent variables of interest k to the model. Everything else works exactly as described above for the simple model.

6.2.1 Adding additional metric variables

We already expected that the mean of the previous grades could be a strong predictor for future grades. We could understand these as a *proxy* variable for the general skill level of a student. The higher the skill level, the higher previous grades will have been.

How we can add additional variables in R code will again be a topic for the next session, but let us look at the results of a regression of `grade` on `hours_centered` and `previous_grades_centered`, the latter being centered on the mean previous grade of 2.935.

```
##
## Call:
## lm(formula = grade ~ hours_centered + previous_grades_centered,
##     data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44462 -0.30556  0.00622  0.32878  1.31002
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.967500   0.038316  77.449   <2e-16 ***
## hours_centered            -0.056543   0.006114  -9.248   <2e-16 ***
## previous_grades_centered   0.904079   0.039830  22.699   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5419 on 197 degrees of freedom
## Multiple R-squared:  0.7492, Adjusted R-squared:  0.7467
## F-statistic: 294.3 on 2 and 197 DF,  p-value: < 2.2e-16
```

As we added a new variable, we now see three coefficients. The intercept has not changed. It now indicates the estimated grade for a student who invests the mean amount of hours, 40.33, and whose previous grades are exactly 2.935, the mean of the variable.

The coefficient for `hours_centered` got mildly more negative, still telling us that the value of `grade` gets lower, the more hours are invested in writing the paper. This coefficient now gives us the effect while *controlling* for the effect of `previous_grades_centered`. This is what multiple linear regression does, giving us the coefficients for our variables of interest while keeping all other independent variables at specific values. As we have centered the variable for previous grades, the coefficient for `hours centered` gives us the effect when the previous grades were exactly at the mean of $2,935$.

In the same way, the coefficient for `previous_grades_centered` gives us the effect of previous grades when the invested hours are controlled for, in this case when the invested hours were exactly 40.33. The coefficient is rather high and positive. This indicates that a student with a previous grade value that is 1 above the mean, is estimated to receive a new grade that is 0.9 points above the intercept. This means, that the previous grade is a very strong predictor for the new grade.

While plotting in more than two dimensions gets really hard, we can still calculate $\hat{y}$ for certain values of both independent variables. We already know the predicted grade for a student with mean values on both independent variables, as this is the intercept. To make sure that we correct, we can calculate it again.

$$b_0 + b_{hours\_centered} * 0 + b_{previous\_grades\_centered} * 0 = 2.9675$$

For this case we can see, that the previous grade actually is a strong predictor, as the previous and new grades are substantially the same.

What if a student whose previous grades were 1 above the mean, so just below 4.0 but who decides to invest 10 hours more than the mean for the new paper?

$$2.9675 - 0.056543 * 10 + 0.904079 * 1 = 3.306149$$

So the good message is, while previous grades are a strong predictor, putting in more hours still leads to better grades.

What if a really good student decides to rely on their skill and to work less this time?

$$2.9675 - 0.056543 * -10 + 0.904079 * -2 = 1.724772$$

While 1.7 is still a very good grade, working 10 less hours than the mean of students leads to a substantially worse estimate compared to the about 1.0 received in previous grades.

6.2.2 Adding dummy variables

Another variable that could be of interest in explaining the received grade, is if a student attended most of the seminar sessions. `attendance` holds this information in the form of a dummy variable. Dummies can only have two states. "Yes" or "No", "1" or "0" or in this case "`TRUE`" or "`FALSE`".

Let us add the variable to our model.

```
##
## Call:
## lm(formula = grade ~ hours_centered + previous_grades_centered +
##     attendance, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.41059 -0.30910  0.01667  0.35607  1.29849
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.157411   0.078658  40.141  < 2e-16 ***
## hours_centered            -0.053942   0.006088  -8.860 4.85e-16 ***
## previous_grades_centered   0.911802   0.039282  23.212  < 2e-16 ***
## attendanceTRUE            -0.248250   0.090246  -2.751   0.0065 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5331 on 196 degrees of freedom
## Multiple R-squared:  0.7586, Adjusted R-squared:  0.7549
## F-statistic: 205.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

This gives us a new line in the R Output holding an estimate for `attendanceTRUE`. What is meant by this? In contrast to the metric variables we have uses in our model up to this point, a dummy variable - or binary variable - can only have two states. As we are using a logical variable here, it can only have the value `TRUE` - here indicating regular attendance - or `FALSE`. So what the output shows us, is the effect of attendance being `TRUE` compared to being `FALSE`. If a student did regularly attend the seminar, the estimated grade is -0.248250 lower compared to when they did not.

We can observe what happens in the formula:

$$\hat{y} = b_0 + b_{hours\_centered} * x_{hours\_centered} + b_{previous\_grades\_centeterd} * x_{previous\_grades\_centeterd} + b_{attendance} * x$$

If you calculate with `TRUE` and `FALSE` in R, the values 1 and 0 are used respectively. So $x_{attendance}$ can either have the value 1 for regular attendance or 0 for not so regular attendance.

If a student did regularly attend, the coefficient $b_{attendance} becomes a part of the estimate $\hat{y}$:

$$\hat{y} = b_0 + b_{hours\_centered} * x_{hours\_centered} + b_{previous\_grades\_centeterd} * x_{previous\_grades\_centeterd} + b_{attendance} * 1$$

If student did not regularly attended, this happens:

$$\hat{y} = b_0 + b_{hours\_centered} * x_{hours\_centered} + b_{previous\_grades\_centeterd} * x_{previous\_grades\_centeterd} + b_{attendance} * 0$$

$$\hat{y} = b_0 + b_{hours\_centered} * x_{hours\_centered} + b_{previous\_grades\_centeterd} * x_{previous\_grades\_centeterd}$$

The coefficient is no longer a part of the estimate. One can basically say, the coefficient gets switched on or off by the value of the dummy variable.

So while the estimate for a student with mean values for invested hours and previous grades who did not attend is equal to the intercept of 3.157411, for a similar student who attended we can calculate the estimate as:

$$3.157411 - 0.053942 * 0 + 0.911802 * 0 - 0.248250 * 1 = 3.157411 - 0.248250 = 2.909161$$

It seems attending class is an easy way to raise one's grades.

6.2.3 Adding categorical variables

We have one further variable in our simulated data set that could be of interest in explaining, what makes a good grade in a seminar paper.`contact` is a categorical variable, or a factor variable in R terms. It can take three different categories. `No contact` indicates that the student did not contact the lecturer to discuss a research question or the laid out plan for the paper. `E-Mail` means that there was some written contact and at least the basics for the paper were discussed before writing. Lastly, `In Person` stands for an in depth discussion with the lecturer, clearing up problems beforehand and thus potentially having a more stringent vision for the paper before writing the first word.

Let us add the variable to our model.

```
##
## Call:
```

```
## lm(formula = grade ~ hours_centered + previous_grades_centered +
##     attendance + contact, data = grades)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3835 -0.2525  0.0167  0.2678  0.9347
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.617949   0.068077  53.145  < 2e-16 ***
## hours_centered            -0.050830   0.004433 -11.466  < 2e-16 ***
## previous_grades_centered   0.874123   0.028657  30.503  < 2e-16 ***
## attendanceTRUE            -0.324653   0.065781  -4.935 1.72e-06 ***
## contactE-Mail             -0.413808   0.069817  -5.927 1.39e-08 ***
## contactIn Person          -0.853252   0.063964 -13.340  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3869 on 194 degrees of freedom
## Multiple R-squared:  0.8741, Adjusted R-squared:  0.8709
## F-statistic: 269.4 on 5 and 194 DF,  p-value: < 2.2e-16
```

Wait, we entered three categories into the model and got estimates for two of them. What happened? What R does is to create two dummy variables on the fly. The first discerns between having E-Mail contact and no contact at all. The second one between having contact in person and no contact at all. So for categorical variables in regression models we always compare being in one of the category to being in the *base category*. In this case the base category is `No contact`, but we could also change the base category. It depends on what we are interested in comparing to. For our example comparing the effects of having more in depth contact to having none makes sense.

Let us look at our formula again:

$$\hat{y} = b_0 + b_{hours\_centered} * x_{hours\_centered} + b_{previous\_grades\_centeterd} * x_{previous\_grades\_centeterd} + b_{attendance} * a$$

Now there are three possibilities. A student can have no contact at all. In this case both dummy variables equal 0. To make our formula easier to read, we have abbreviated the middle part for now:

$$\hat{y} = b_0 + ... + b_{E-Mail} * 0 + b_{InPerson} * 0$$

So in this case controlling all other independent variables at their default values, the mean for the metric variables and `FALSE` for `attendance`, the intercept gives

us the estimate for the grade as both dummy variables that were created for `contact` are "switched off".

The two other possibilities are that a student either had E-Mail contact or an in person discussion:

$$\hat{y} = b_0 + ... + b_{E-Mail} * 1 + b_{InPerson} * 0$$

$$\hat{y} = b_0 + ... + b_{E-Mail} * 0 + b_{InPerson} * 1$$

In both cases the relevant dummy variable is "switched on" while the other does not factor into the equation.

Looking at the estimates we can see that having contact to the lecturer before writing has strong negative effects, resulting in better grades. Having E-Mail contact reduces the value of `grade` by -0.413808 points, having an in person discussion by -0.853252.

So what grade can a student whose previous grades were at the mean of 2.935, but who decided to put in 20 hours more compared to their peers, regularly attend the seminar and have an in-depth personal discussion before writing their paper expect on average as their new grade?

$$3.617949 - 0.050830 * 20 + 0.874123 * 0 - 0.324653 * 1 - 0.413808 * 0 - 0.853252 * 1 = 1.423444$$

Putting in the hours, attending and working with your lecturer seems to pay off, at least in our simulated data set.

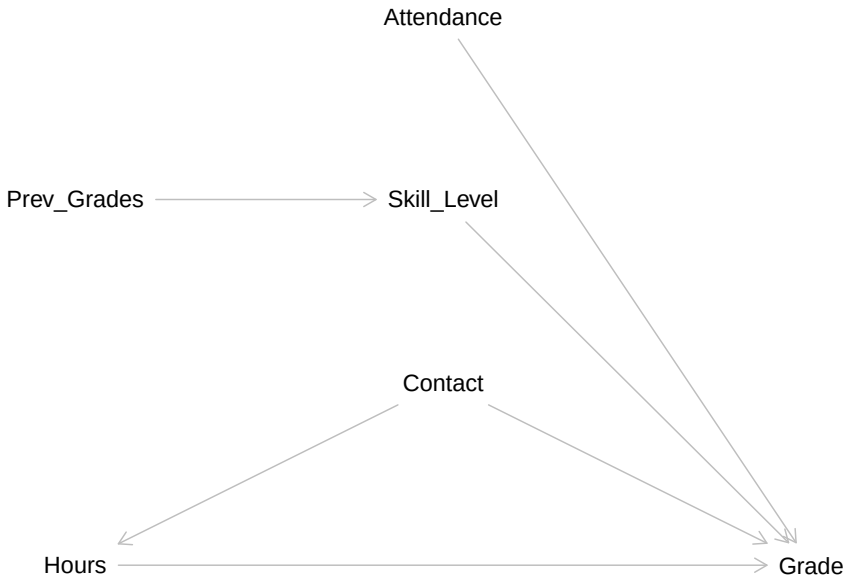
6.3 Returning to our research question

Our exemplary research question concerned itself with what makes a good grade in a seminar paper. In particular we were interested in the effect of the invested hours, as our main hypothesis was that more hours lead to better grades. What do we know now?

All analysis point towards a clear effect from `hours` on `grade`. This effect was consistently visible in all of our models. But did we correctly identify and estimate the effect of interest? Maybe. The problem is, we actually did not approach the analysis correctly. In a real analysis we should **absolutely** refrain from adding variables to our model that *could be* relevant until we are satisfied or even until all available variables are bunched into one huge model. It was fine to do this in this introduction to linear regression to learn how different types of variables can be used in a regression model. But in a real project, we have to

invest time to think about which variables to add because we assume that they have a relevant effect based on theoretical assumptions about the processes we are interested in.

So let us do this now and vow do make this our first step in all future endeavors. While we do not have a clear theoretical basis, we can make clear assumptions on the data generating process and draw these in a DAG.



Our central assumption, and the effect we want to identify and estimate, is the direct effect from `hours` on `grade` in the bottom line. The more hours a student invests, the better the grade should be.

The assumed effect of `contact` is more complex. For one we assume that a more in-depth contact with the lecturer will increase the grade directly. The research question will be more focused, the student will know what is important to a certain lecturer, common mistakes can be avoided if they are cleared up beforehand and so on. But we will also assume that `contact` will have an effect on `hours` in the sense that the hours invested can be used more efficiently if an in-depth discussion has taken place. Instead of wasting time running into problems that could have been avoided most of the invested time can actually go into constructive work. This makes `contact` a confounder for `grade` and `hours`.

A student's skill level will also have a direct effect on `grade`. As we do not have a direct measure of skill in our data, we use `previous_grades` as a proxy for skill level. `attendance` is also assumed to have a direct effect on `grade` as students who were present in the seminar will not only have learned the

seminar's contents, but will also have a better understanding of what is expected in their seminar papers.

Tapping into the knowledge from session 4, we can now make implications for our model from the DAG. Let us list all paths from `hours` to `grade`:

$$A : Hours \rightarrow Grade$$

$$B : Hours \leftarrow Contact \rightarrow Grade$$

Path A represents our effect of interest. On path B, `contact` is a confounder for `hours` and `grade`. To close this path, we have to control for `contact`. As neither skill level - or `previous_grades` - nor `attendance` lie on a path from our independent variable of interest to our dependent variable, we should not control for them in our model. That leaves us with `hours` and `contact` to be included in our linear regression, if our goal is to get an unbiased estimate for the effect of invested time on the final grade. So let us do this:

```
##
## Call:
## lm(formula = grade ~ hours_centered + contact, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85595 -0.74624 -0.02106  0.66648  2.50161
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.44352    0.10404  33.098  < 2e-16 ***
## hours_centered   -0.04967    0.01052  -4.723 4.43e-06 ***
## contactE-Mail    -0.46482    0.16785  -2.769  0.00616 **
## contactIn Person -1.02804    0.15240  -6.746 1.67e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9305 on 196 degrees of freedom
## Multiple R-squared:  0.2643, Adjusted R-squared:  0.253
## F-statistic: 23.47 on 3 and 196 DF,  p-value: 5.072e-13
```

This is our estimate. Each hour invested beyond the mean of 40.33 hours changes the grade by about -0.05 points. This supports our hypotheses and we can conclude, that investing more hours into writing a seminar paper actually is a worthwhile investment.

But remember: This is correct as long as our DAG is drawn correctly. This is always debatable. Maybe we should assume an effect from skill level on `hours`. The higher the skill level the more efficiently the available time can be used. For this example we know the DAG is correct, because we have simulated the data exactly in this way. For real world applications we never know if our DAG is correct. All we can - and have to - do is base it on thorough thought, theoretical work, exploratory data analysis and sound arguments.

This all is true *if* our goal is to estimate an effect of interest as precisely as possible. But as we have alluded to in the introduction to this session we could also use modelling with a different goal, i.e. predicting a grade as accurately as possible. For this task, the model which only includes `hours` and `contact` will not do the best job. From our DAG we know that `attendance` and `previous_grade` should have an effect on `grade`, as we have also seen in our models. For this task the full model including all these variables will produce better estimates. We will return to this in a later session, but for now we should remember that we have to know our task because the task dictates which is the best model to use.

6.4 Adressing the uncertainty

Looking at the coefficient block from the output, we see more than just our estimates. The `Std. Error` - *standard error* - is a measure for the uncertainty of our estimates. It basically tells us, how far away the actual values of the observations used to compute the model are from our estimate *on average*. The smaller the *standard error*, the more accurate our estimate. The standard error is presented in the units of the estimate and we can thus compare them. A large standard error for a large estimate is far less problematic compared to a large standard error for a small estimate.

The estimate and it's standard error are the basis for *hypothesis testing*. What we are testing is the *alternative hypotheses* H_a that there actually is an effect of our independent variable on the dependent variable against the *null hypothesis* H_0 that there is no effect. To reject the null hypothesis and be confident that we are observing an actual effect, versus an effect that is just based on random variation in our sample, the estimate has to be far away enough from 0 and be accurate enough, i.e. have a small standard error. This relationship is computed in the *t-statistic*, `t value` in our output. From this the *p-value* can be computed, `Pr(>|t|)` in the output. The *p-value* tells us the probability to observe an association between the independent and the dependent variable as large or larger than our estimate suggests, if the true association would actually be 0. If the p-value is small enough, we can reject H_0 and conclude that we observed an actual effect. There are certain agreed upon cutoffs in statistics while values that meet this cutoffs are considered *statistically significant*. The most common cutoff in social sciences is 0.05 indicated by one `*` in the output. Other common

cutoffs are indicated by more asterisks.

Interpreting p-values correctly and not falling into the common pitfalls is a topic on its own. We do not have the time to dive into this here, so for now we can agree that p-values below 0.05 indicate that we can reject H_0 and thus conclude that we have actually observed an effect. Still, our interpretation of regression results should not focus solely on p-values or lead us to disregard any effects that did not meet the cutoff. For example, we can have very small p-values for effects that are so small that they are substantially irrelevant. One way to address this is to inspect the actual magnitudes of the effects. On the other hand, we can have p-values larger than 0.05 for effects that are still relevant. Maybe the problem is not that there is no effect but that we were not able to measure the variable in question precisely enough or that we just did not have enough observations. We can not go any deeper than this here, but we should remember that the practice of declaring every effect with stars a win and disregarding everything without them may still be common but is not the way to go forward.

In our model, we can see that the effect of interest is statistically highly significant. We can thus conclude, that we have observed an actual effect from `hours` on `grade`. Our estimate is large enough and our standard error small enough to reach this conclusion.

6.5 Moving on

We have attained an estimate for our effect of interest which supports our hypotheses that investing more hours into writing a paper leads to better grades. So can we wrap a bow on the question and move on to finally figuring out what is going on in our NBA data? Almost, but not yet. We still do not know, if our model actually works as intended. Linear regression, as well as every modelling technique, has some underlying assumptions that we have to meet for the model to accurately estimate an effect. In the next session we will get to know this assumptions and how we can test for them.

Chapter 7

Linear Regression Theory III: Diagnostics

In this session we will get to know the central underlying assumptions for linear regression models. To find out if our model actually works as intended and thus gives us a reliable estimate for the effect of `hours` on `grade`, we have to check if we have met this assumptions, and if we did not, we have to correct our model accordingly. Before we do this, we should briefly consider another part of the regression output, the model fit.

7.1 Objectives

- Learn about model fit and its limits
- Understand the statistical assumptions underlying linear regression
- Test for violated assumptions and learn how to correct for those

7.2 Model fit

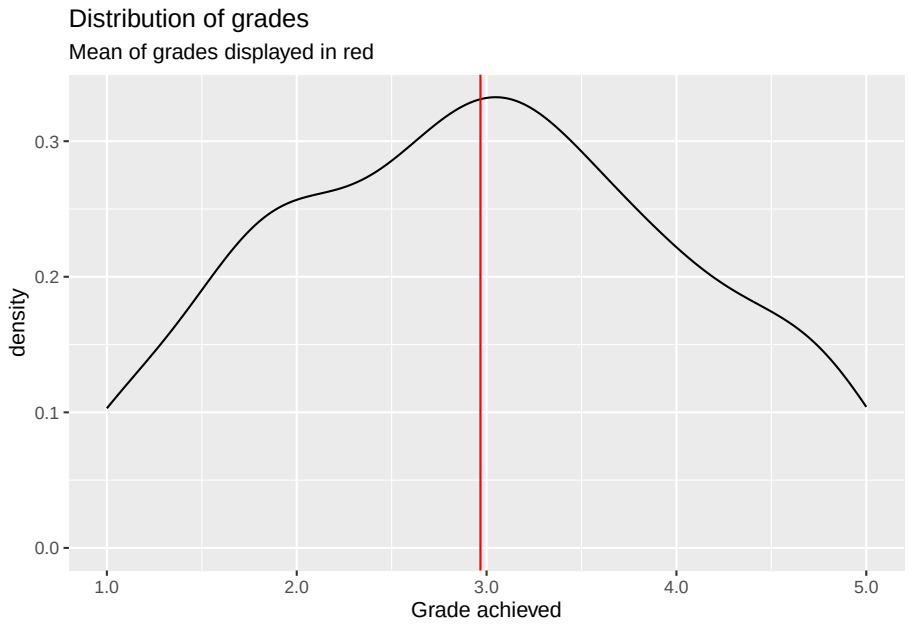
Let us again inspect the output from the simplest model we computed, regressing the grade solely on the invested hours:

```
##
## Call:
## lm(formula = grade ~ hours_centered, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -1.88006 -0.83961 -0.08006  0.77006  2.53881
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.96750    0.07267  40.835  < 2e-16 ***
## hours_centered -0.05236    0.01159  -4.517 1.07e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.028 on 198 degrees of freedom
## Multiple R-squared:  0.09344,    Adjusted R-squared:  0.08886
## F-statistic: 20.41 on 1 and 198 DF,  p-value: 1.075e-05
```

Up to now, we exclusively talked about the coefficient block. We will return to the "Call" next session and to the "Residuals" later in this session. For now let us focus on the bottom block in the output.

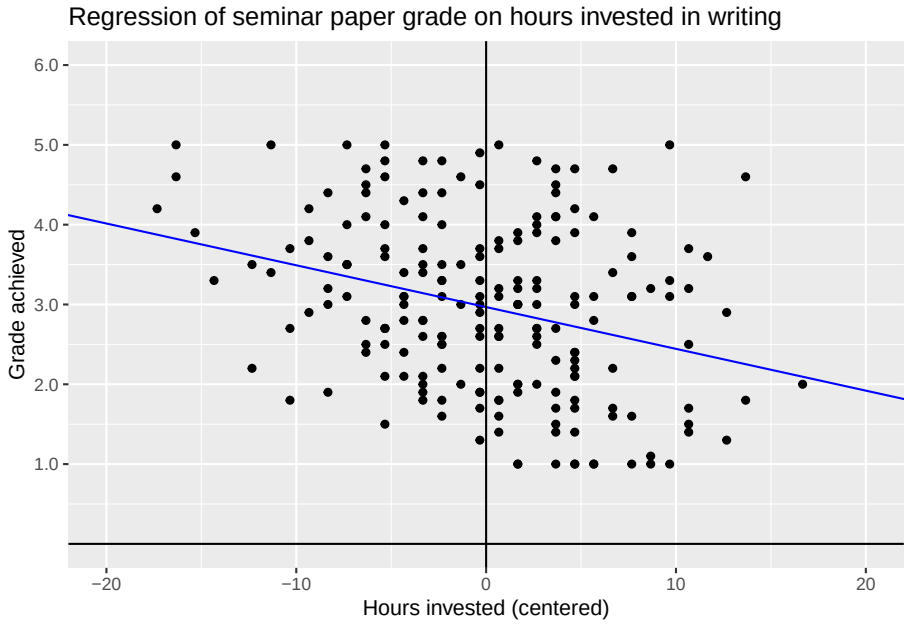
R^2 or *R-squared* is a measure for the amount of variance in the data that is "explained" by the model. Real world data will always have variance. Not every value will neatly fall onto the mean value of a variable. Rather the data is dispersed around it. The same is true for our dependent variable:



This is a density plot and shows the distribution of a metric variable as a smoothed line. We do not see every individual actual value but the general shape of the data. The red line represent the mean of `grade`, which is about 2.97. Most actual values are not exactly at the mean but are rather dispersed

around it, ranging between 1.0 and 5.0. This is the variance in our outcome variable.

Let us now plot `grade` against `hours_centered` and add the regression line from our model above:



Without our regression line, all we would have is a cloud of points without much order to it. What linear regression does, is trying to bring order into this by fitting a line that best explains the variance of the dependent variable, `grade` in our case by its relationship to one or multiple dependent variables, here `hours_centered`. But this linear line can never explain the variance completely. For this it had to pass through every data point. Our line does not. Actually most data points do not lie on the regression line but at some distance to it. You will remember that OLS computes *the* regression line for which the squared distances are smallest. This is the line that explains most of the variance of y by its relationship to x, but not all variance is explained. An unexplained part remains. These are the residuals, the distance that points fall from the regression line. R^2 tells us the relative amount of how much we reduced the initial variance by fitting the line and thus explaining a part of said variance.

A R^2 of 0 would mean that no variance is explained, a value of 1 that all variance is explained. Two highly unlikely outcomes. We will almost always explain something and never explain everything.

In our model R^2 equals 0.09344. This means we explained about 9.3% of the variance of `grade` by its relationship with `hours_centered`. That's nice, but this also means that over 90% are still unexplained. We will not explain all of

the variance, i.e. $R^2 = 1$, but in general a higher R^2 is desirable.

So what can we do? We can try to add additional variables to the model that help to explain the variance in the outcome variable. Last session we concluded that the best model to measure the effect of invested hours on the achieved grade would also have to include `contact`:

```
##
## Call:
## lm(formula = grade ~ hours_centered + contact, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85595 -0.74624 -0.02106  0.66648  2.50161
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.44352    0.10404  33.098  < 2e-16 ***
## hours_centered   -0.04967    0.01052  -4.723 4.43e-06 ***
## contactE-Mail    -0.46482    0.16785  -2.769  0.00616 **
## contactIn Person -1.02804    0.15240  -6.746 1.67e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9305 on 196 degrees of freedom
## Multiple R-squared:  0.2643, Adjusted R-squared:  0.253
## F-statistic: 23.47 on 3 and 196 DF,  p-value: 5.072e-13
```

We can use R^2 to compare the *model fit* of multiple models. Here the larger model achieved a considerably higher value of $R^2 = 0.2643$. The model fit improved as we can now explain a higher ratio of the variance in `grade`.

After R^2 we see another value, *Adjusted R-squared*. This becomes relevant if we add additional variables to our model. R^2 almost always increases, and never decreases, when adding additional variables to the model, especially if we have few observations. Because of this R^2 can get less reliable when we have many variables and few observations. *Adjusted R-squared* corrects for this by including both factors in the calculation. When we have many observations the differences are negligible. This is true for our case. We have relatively many observations and few variables in our model, so the values of both measures are rather close. But in cases where this relationship is not as favorable, adjusted R-squared should be used in place of the regular R^2.

The block in our output also gives us the *Residual standard error*. As we have seen above, most actual data point do not lie on the regression line but some distance away from it. These are the residuals. Thus their standard error basically tells us how much we miss the spot on average. As it is given in units

of the dependent variable, we can say that the estimates for `grade` based on our second model are on average 0.93 off. A considerable amount, as this is almost one whole grading step. This is still an improvement from the 1.028 in the first model but nevertheless a substantial error.

The last line in the output gives us two connected measures. The *F-statistic* is the test statistic for R^2 and is used the compute the corresponding *p-value*. In this case we are testing if the R^2 our model returned based on our sample is possible, when the actual population value of R^2 is 0. In other words, could we have achieved this R^2 by chance if the independent variables in our model actually do not explain part of the variance in the population? Both of our models have very small p-values, so it is highly unlikely that we have just explained some variance by chance. This gives further credibility to our model specification.

We can conclude that the second model was an improvement over the first. But can we do more? Sure! We can add additional explanatory variables:

```
##
## Call:
## lm(formula = grade ~ hours_centered + previous_grades_centered +
##     attendance + contact, data = grades)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3835 -0.2525  0.0167  0.2678  0.9347
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.617949   0.068077  53.145  < 2e-16 ***
## hours_centered            -0.050830   0.004433 -11.466  < 2e-16 ***
## previous_grades_centered   0.874123   0.028657  30.503  < 2e-16 ***
## attendanceTRUE            -0.324653   0.065781  -4.935 1.72e-06 ***
## contactE-Mail             -0.413808   0.069817  -5.927 1.39e-08 ***
## contactIn Person          -0.853252   0.063964 -13.340  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3869 on 194 degrees of freedom
## Multiple R-squared:  0.8741, Adjusted R-squared:  0.8709
## F-statistic: 269.4 on 5 and 194 DF,  p-value: < 2.2e-16
```

The p-value is even lower, and the F-statistic even higher, compared to our second model, but this was never an issue. What is more interesting is that we have substantially increased R^2 and decreased the residual standard error. As we have concluded last week, this larger model is better at predicting the actual

values of `grade`. Thus the explained variance has to increase and the average error in estimating y has to decrease. But is this the better model? The values on the model fit would suggest so. And this also is true, if our aim is predicting `grade` to the best of our abilities. But if our aim is still measuring the effect of `hours` on `grade` we know from our DAG that we do not have to or even should not control for the additional variables to get an unbiased estimator for the effect of interest.

What can we take away from this? While the model fit measures are an important tool for comparing multiple possible models and better values are desirable in general, it should not be our goal to just max out all measures and declare this model the "winner". It is never that easy in statistics. One thing we can never replace is thorough theoretical work before even computing our first model. Based on our DAG, if it is correct, we know that we do not have to control for previous grades and attendance. Including them may give us a larger R^2, but is still not the correct way to build our model.

Based on this our best model is still the second one:

```
summary(m2)
```

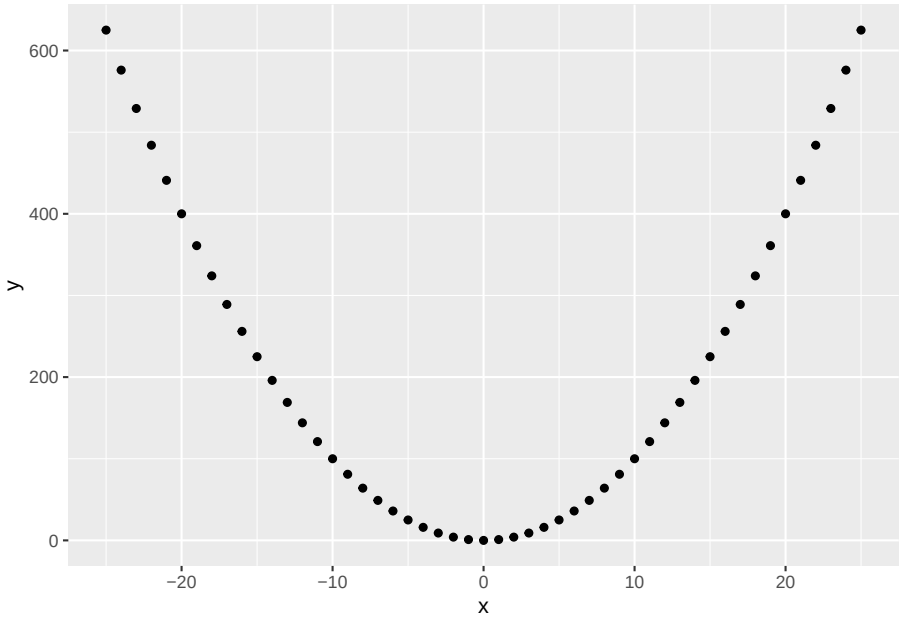
```
##
## Call:
## lm(formula = grade ~ hours_centered + contact, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85595 -0.74624 -0.02106  0.66648  2.50161
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.44352    0.10404  33.098  < 2e-16 ***
## hours_centered   -0.04967    0.01052  -4.723 4.43e-06 ***
## contactE-Mail    -0.46482    0.16785  -2.769  0.00616 **
## contactIn Person -1.02804    0.15240  -6.746 1.67e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9305 on 196 degrees of freedom
## Multiple R-squared:  0.2643, Adjusted R-squared:  0.253
## F-statistic: 23.47 on 3 and 196 DF,  p-value: 5.072e-13
```

7.3 Regression diagnostics

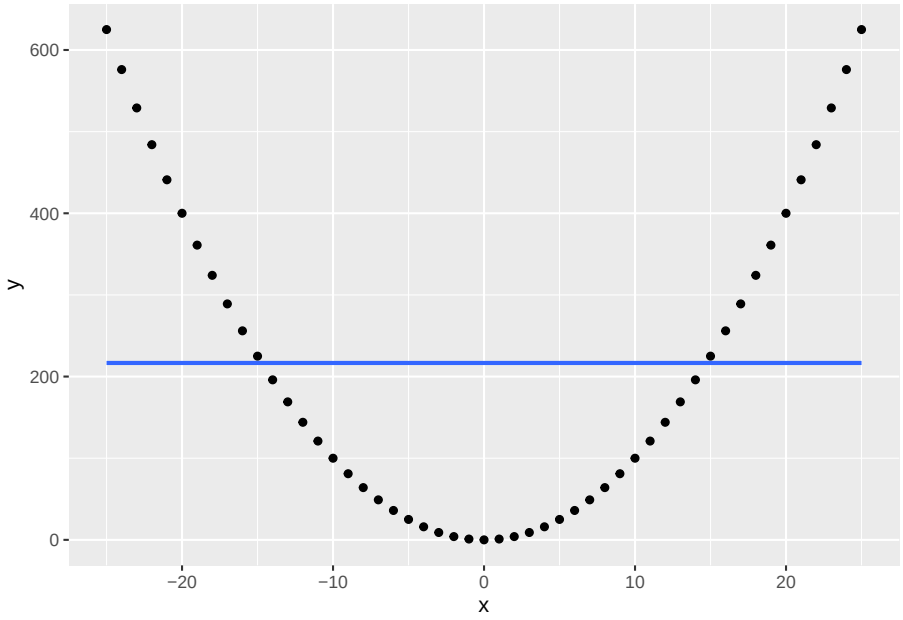
As linear regression is a statistical technique, there are certain statistical assumptions we have to meet. If we violate those, the best laid plans may falter and our results may be not as robust as we hoped. Let us go through these assumptions and the tests to check for them one by one.

7.3.1 Linearity

The name already gives it away, a *linear* regression is used to estimate **linear** relationships between variables. For this to work, the relationships actually have to be linear. But a relationship between two variables can have other functional forms. Consider this for example:

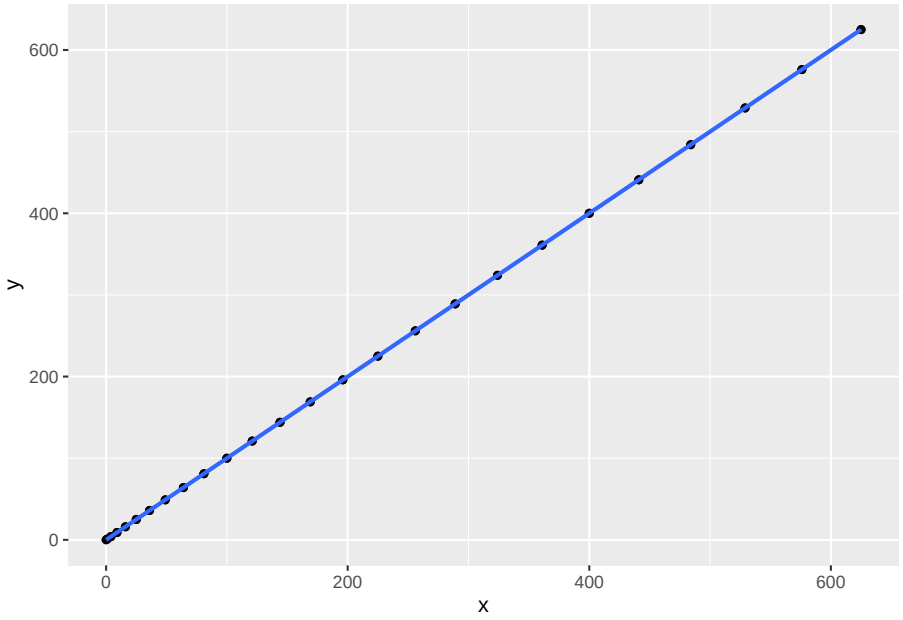


The relationship is clearly not linear. But we can still fit a regression and get a result:



The regression line shows us that `x` and `y` are completely uncorrelated. This is clearly not true, but as our linear regression assumes linearity, it tries to model the relationship in linear terms.

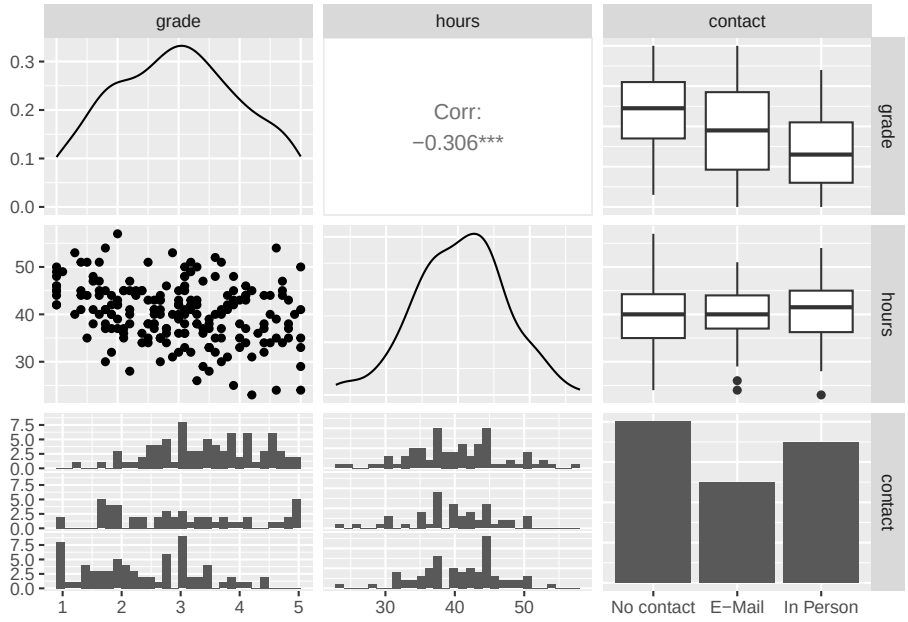
What we can do in such cases is to transform the variable in question in a non-linear way. Here the quadratic relationship is easy to spot, so if we transform x to x^2, this happens:



The non-linear relationship between `x` and `y` has been transformed into a linear one.

For real world data, the non-linearity most often is not as straightforward to spot as in this example. A first step to approach this, is inspecting a scatterplot matrix. This is usually done before starting to model to identify relationships between the variables used.

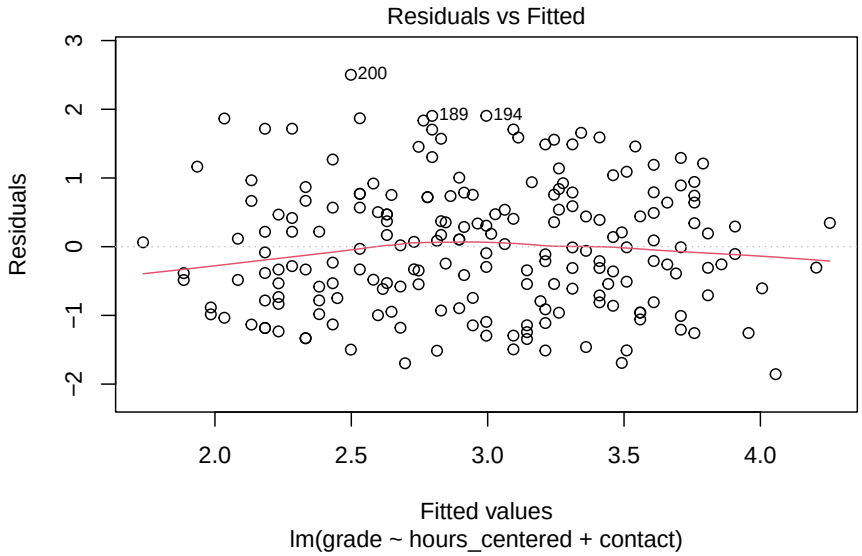
```
## Warning: package 'GGally' was built under R version 4.2.3
```



The diagonal displays the distribution of all variables included. Here metric variables are displayed as density plots and categorical variables as bar plots. Below and above the diagonal the relationship between two variables is shown. The scatterplot on the left of the second row is the one between `hours` and `grade` we have already seen several times. There is no indication of non-linearity here. What we have not inspected yet, is the relationship between `contact` and the two other variables.

The bottom row contains histograms of the two metric variables by the category of contact, the right column boxplots for the same combination. Without going into too much detail on both types of plots, both show us how the distribution for both metric variables changes by category. The more personal the contact with the lecturer, the lower the distribution of final grades is. This makes sense, as we have already seen this correlation in the results of our model. Between `hours` and `contact` there seems to be no correlation. The amount of hours a student invests in writing the paper, does not lower the hours invested in a systematic way.

But this does not clear the model of suspicions of non-linearity just yet. Even when all pairwise relationships are linear, controlling for multiple variables at the same time can introduce non-linearity for this specific combination. One way to approach this is to inspect the *Residuals vs. Fitted* plot. As the name suggests, this plots the fitted values, i.e. the estimates for our dependent variables based on the model, against the residuals of the dependent variable. When the relationship is linear, we should see a more or less straight line along the x-axis, where $y = 0$.



For many use cases, the line is straight enough, indicating no clear and strong patterns of non-linearity. Still the residuals seem to be slightly off for very good and very bad estimated grades.

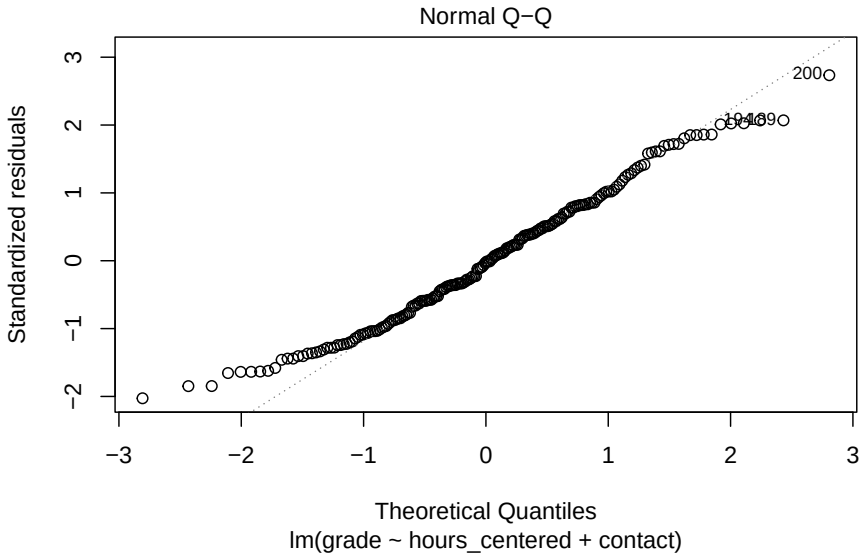
Besides violating the assumption of linearity, patterns in the residuals vs. fitted plot can also indicate that there is some important explanatory variable missing from the model. We will return to this after we have applied the further tests and discussed the remaining assumptions.

7.3.2 Normally distributed residuals

Another assumptions in linear regression is that the residuals are normally distributed. This is especially relevant for sample with a small n as the test statistics tend to get unreliable in these cases if the residuals are not normally distributed. For larger samples, as in our case, this is not as problematic. Still, systematic deviations from normality can indicate that the model is not *parsimonious*. This means that either not all relevant variables are included or that variables are included that are not necessary for the model.

We get a first idea of the distribution from the model summary. This shows us the median as well as the 25- & 75-percentiles and the minimum and maximum values. While strong and clear violations against the normality assumption could already be visible here, these measures are not enough to actually test for normality. A more informative and accurate approach is using a *Q-Q plot*. This plots the standardizes residuals, the residuals divided by their standard

error, against a theoretical normal distribution. If the residuals are perfectly normally distributed, each data point lies on the diagonal, if they are not they move away from the line. Small deviations, especially in the tails, should not be over emphasized. What we are looking for are clear and systematic deviations.



In the case of our model, most data points lie on the diagonal, while there are some small deviations in the tails. As our n is large enough, this should not be problematic. It may indicate that the model is not parsimonious but there is no clear cause for concern here.

7.3.3 Homoscedasticity

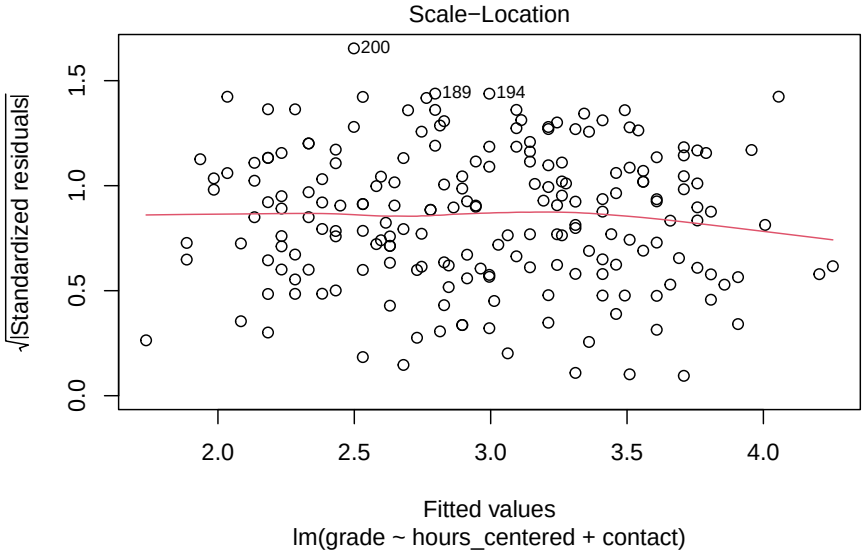
The *homoscedasticity* assumption states that the residuals are expected to have a constant variance over the whole range of the dependent variable. Let us assume that the variance of our residuals would be lower for very good grades and higher for very bad ones. This would indicate that we can make more accurate estimates for better grades than for worse ones as a small variance would indicate smaller residuals and thus a smaller error. For the assumption to hold we must be able to make about the same quality, be it high or low, for all values of `grade`.

The problem is that the computation of the standard errors, test statistics and p-values depends on this assumption. If the assumption is violated, if we have *heteroscedasticity*, these measures are not reliable anymore.

The problem often occurs when the dependent variable is not symmetric. In

the scatterplot matrix above, we already saw that `grade` is fairly symmetrically distributed, so we would not expect problems here. If our dependent variable was unsymmetrical, transforming it to be more symmetrical, e.g. by using the logarithm or a square root, could help.

To check for problems with heteroscedasticity, we can use the *Scale-Location* plot. This plots the fitted values against the square root of the standardized residuals. For homoscedasticity to hold, we should see our data points as a horizontal band with more or less constant width running from the left to the right. The same goes for the plotted line.



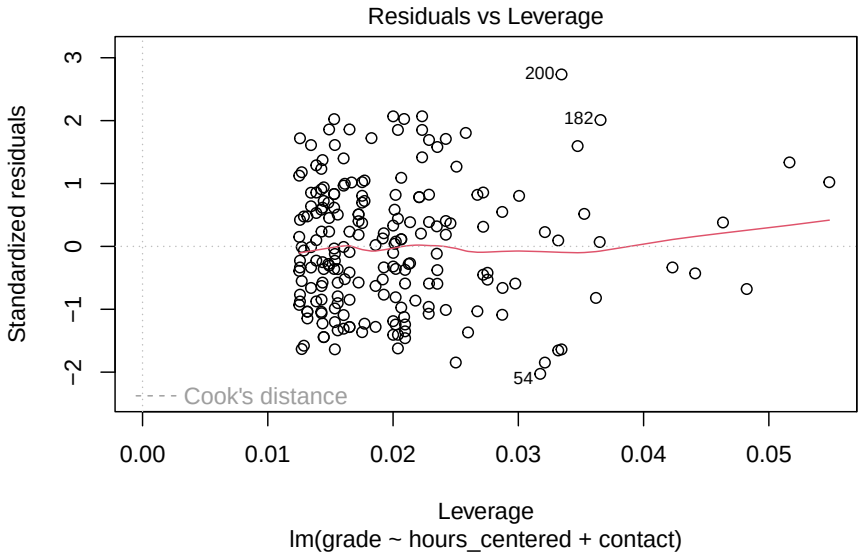
In our case the homoscedasticity assumption holds. Slight variations are not problematic and overall the variance is constant.

7.3.4 No overly influential data points

Observations can get highly influential if they have unusual values. Sometimes these are extremely low or high values on some variable. But even "normal" values on two or more variables can get unusual in their combination. Imagine a student with 60 invested hours. A high value but not overly extreme. Now the same student had in person contact with their lecturer but still received a 5.0. This could potentially be an influential data point as this combination is unusual in terms of what the model expects. In case of our model this observation would most probably not be overly influential. But imagine the same observation with 300 invested hours. Such extreme cases can influence the fit by figuratively

"puling" the regression line in their direction.

We can divide influential data points into unusual values on the dependent variable, *outliers*, and unusual values on independent variables, *high leverage points*. The latter have high leverage because they "pull" on the regression lines and thus change the slope. As a rule of thumb, we can consider values with standardized residuals over 3 or under -3 outliers. Concerning the dependent variables we can compute the *leverage statistic*. Here values that exceed $2 * (p + 1)/n$, where p is the number of predictors, are considered as having high leverage. We can inspect both at the same time in the *Residuals vs. Leverage* plot.



We can see that there are no clear outliers. To assess points with high leverage we first have to compute the threshold as: $2 * (3 + 1)/200 = 0.04$. Note that while we have two independent variables in our model, we actually have three predictors due to our categorical variable. Thus we have to compute with $p = 3$ instead of $p = 2$. We can see that there are a number of points that exceed this value. The question is, why do these values exist? Sometimes these are measurement errors, extreme values or unusual combinations that come down to the researcher recording the wrong values into the data set. In these cases we can try to fix the errors or remove the observations from the data. As we have several values with high leverage, this seems highly unlikely. But if we had not simulated the data ourselves and knew that there is no error, we should at least check. What seems more probable though, and is often the actual root of high leverage, is that there are variables missing from the model that could explain the high values. In this case the values should be lower after we include

the missing variables into the model. We will return to this later.

We also may not have a serious problem here. Leverage on its own does not have to be problematic. Some data points will always be more influential than others and remember that the cutoff is always just a rule of thumb. As said above, it is the combination of unusual values that tends to get problematic. We can check for observations that are outliers and have high leverage visually in our plot. Problematic observations tend to gather in the upper and lower right corners. As neither are populated for our model, we can not really conclude that we have overly influential data points.

XXX IS LEVERAGE CORRECTLY COMPUTED? XXX

7.3.5 No (multi)collinearity

The final assumption we will discuss here, is the absence of (high) collinearity between the predictor variables. Collinearity is present, if two independent variables are highly correlated with each other. This can become a problem as it gets harder to individually estimate the effects for both variables on the outcome as the collinear variables vary together at the same time.

Often collinearity can already be spotted in the correlation matrix. Considering our matrix above we saw no clear indication that `hours` and `contact` are correlated. But the problem can get more complicated if we include three or more independent variables in our model. While none of the pairs of variables may be highly correlated, correlation may exist for a set of three or more of those variables. In these cases we speak of *multicollinearity*. We can not spot this in a correlation matrix, but there is an easy to use measure available.

The *variance inflation factor* (VIF) can be used to inspect (multi)collinearity between two or more independent variables accurately. A VIF of 1 would indicate no collinearity. For real world data this is almost never true as some amount of collinearity always exists. But in general we can say that the VIF should be near 1 and should not exceed a value of 5.

Let us compute the measure for our model with `hours_centered` and `contact`:

```
## Warning: package 'car' was built under R version 4.2.3
```

```
## hours_centered        contact
##       1.004352       1.004352
```

Both values are very close to 1 so we can conclude that we did not violate the assumption. But what could we do, if we did? One approach is to just delete one of the highly correlated independent variables from the model. As they vary together, it may be save to exclude one of them without losing to much information. Another approach would be to combine both variables into a new

measure. Let us imagine that besides `contact` we would have another variable in our model, measuring how well a student feels supported by their lecturer in writing the paper. We could also imagine both variables being strongly correlated as they measure comparable concepts. We could then either drop one of the variables, maybe losing some information in the process, or we could combine both into a new variable which measures the form and the feeling of support at the same time, maybe leading to a more accurate estimate while at the same time eliminating the problem of collinearity. Which one is the right solution depends on the specific case.

7.4 Returning to our research question

When we tested for linearity above, we saw a mild pattern in the data which is not explainable by a violation of the assumption of linearity and thus could be an indication of a missing relevant explanatory variable in our model. Some of the other tests also supported this notion. The Q-Q plot showed us that the residuals have some slight deviations from normality in the tails. While these deviations are small enough to not cause concern on their own, taken together with the residuals vs. fitted plot this gives more weight to the suspicion that some important variable is missing. We also identified some of observations with high leverage. While we can rule out errors in our data, the high leverage could also be explainable by a missing variable.

But which variable could be missing from the model? If our DAG is correct, we can rule out `attendance` and `previous_grades`. We did assume that `contact` is a confounder for `hours` and `grade` and thus included it in our model. Of course we could also miss a variable that is not in our data at all, or even one that is not measureable. As we did simulate the data, we know this is not true, but with real world data this is always a possibility.

Let us think about the `contact`variable once more. We did assume, that the more personal the contact, the more efficiently the time working on the paper can be used. And here may lie the problem. The way we included `contact` in the model is not the way we reasoned in our DAG. It would be correct if we assumed that the more personal the contact, the less time has to be invested. But we already saw in the scatterplot matrix that there is no such relationship between the variables. To specify the effect of `contact` in the model correctly, reflecting the idea of a more efficient use of time the closer the contact was, we have to include it as an interaction with `hours`.

7.4.1 Interactions

In an *interaction*, we assume that the effect of one variable differs based on the value of another variable. Let us return to the formula for a multiple regression with two variables:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \epsilon$$

Here we assume that the value of y varies with the value of x_1 and x_2 as indicated by the coefficients β_1 and β_2.

But we could also follow the notion that the value of x_1 influences y differently based on the value of x_2. For example the effect of x_1 on y could be higher when x_2 also has a high value. This is an interaction and is reflected in the formula by adding an additional multiplicative term between the two dependent variables with an additional associated coefficient:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_1 * x_2 + \epsilon$$

To get a better understanding of this, let us return to our model and add an interaction between `hours_centered` and `contact`.

```
##
## Call:
## lm(formula = grade ~ hours_centered + contact + hours_centered *
##     contact, data = grades)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.77816 -0.72882 -0.08719  0.56140  2.53271
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      3.44466    0.10345  33.298  < 2e-16 ***
## hours_centered                  -0.02893    0.01535  -1.885  0.06098 .
## contactE-Mail                   -0.46775    0.16729  -2.796  0.00569 **
## contactIn Person                -1.01493    0.15171  -6.690 2.33e-10 ***
## hours_centered:contactE-Mail    -0.02377    0.02657  -0.895  0.37204
## hours_centered:contactIn Person -0.05017    0.02442  -2.055  0.04125 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9253 on 194 degrees of freedom
## Multiple R-squared:   0.28,  Adjusted R-squared:  0.2615
## F-statistic: 15.09 on 5 and 194 DF,  p-value: 1.63e-12
```

How can we interpret these results? While the estimates for the intercept and for having e-mail or personal contact in comparison to having no contact at all have barely changed, the coefficient for the amount of hours invested substantially shrunk to almost half its former value. Untll now, we assumed that the effect

of `hours` would be the same for each student. Now that we have included an interaction we assume that the effect of `hours` differs, based on the form of contact a student had.

Let us rewrite our formula for $\hat{y}$ including the interaction. As we are interacting with a categorical variable with three categories, we have to add two interaction terms. The first for the effect of invested hours when e-mail contact was made and the second for the effect of hours when contact was made in person, in both cases compared to having had no contact.

$$\hat{y} = b_0 + b_{hours\_centered} * x_{hours\_centered} + b_{E-Mail} * x_{E-Mail} + b_{InPerson} * x_{InPerson} + b_{hours\_E-Mail} * x_{hours\_c}$$

Let us now also add the coefficients from the model:

$$\hat{y} = 3.44466 - 0.02893 * x_{hours\_centered} - 0.46775 * x_{E-Mail} - 1.01493 * x_{InPerson} - 0.02377 * x_{hours\_centered} * x_{I}$$

We can now consider the three possible forms of contact one by one.

What happens, when a student had no contact? To explore this, we return to the regression formula and equal x_{E-Mail} and $x_{InPerson}$ to 0, which means that no contact was made beforehand. Note that for now we do not care about the actual value of `hours_centered`.

$$\hat{y} = 3.44466 - 0.02893 * x_{hours\_centered} - 0.46775 * 0 - 1.01493 * 0 - 0.02377 * x_{hours\_centered} * 0 - 0.05017 * x_{hours}$$

This shortens to:

$$\hat{y} = 3.44466 - 0.02893 * x_{hours\_centered}$$

For a student who did not make contact, we would estimate the final grade as the intercept minus 0.02893 per hour invested more than the mean of `hours_centered`. Having equaled x_{E-Mail} and $x_{InPerson}$ to 0 not only "switched off" the effects of `contact` but also removed the interaction effects from the equation, the estimated effect for `hours_centered` is only its coefficient of -0.02893.

What happens, when a student had e-mail contact?

$$\hat{y} = 3.44466 - 0.02893 * x_{hours\_centered} - 0.46775 * 1 - 1.01493 * 0 - 0.02377 * x_{hours\_centered} * 1 - 0.05017 * x_{hours}$$

This shortens to:

$$\hat{y} = 3.44466 - 0.02893 * x_{hours\_centered} - 0.46775 - 0.02377 * x_{hours\_centered}$$

and further to:

$$\hat{y} = 2.97691 - 0.0527 * x_{hours\_centered}$$

The intercept is reduced by the coefficient of having e-mail contact, but what is actually of interest here is the effect that `hours_centerd` has. For a student who had e-mail contact, each hour invested above the mean reduces the estimated grade by -0.0527.

We can compute the same for a student with personal contact:

$$\hat{y} = 3.44466 - 0.02893 * x_{hours\_centered} - 0.46775 * 0 - 1.01493 * 1 - 0.02377 * x_{hours\_centered} * 0 - 0.05017 * x_{hours\_centered} * 1$$

$$\hat{y} = 3.44466 - 0.02893 * x_{hours\_centered} - 1.01493 - 0.05017 * x_{hours\_centered}$$

$$\hat{y} = 2.42973 - 0.0791 * x_{hours\_centered}$$

For a student who had in person contact, each hour invested above the mean reduces the estimated grade by -0.0791.

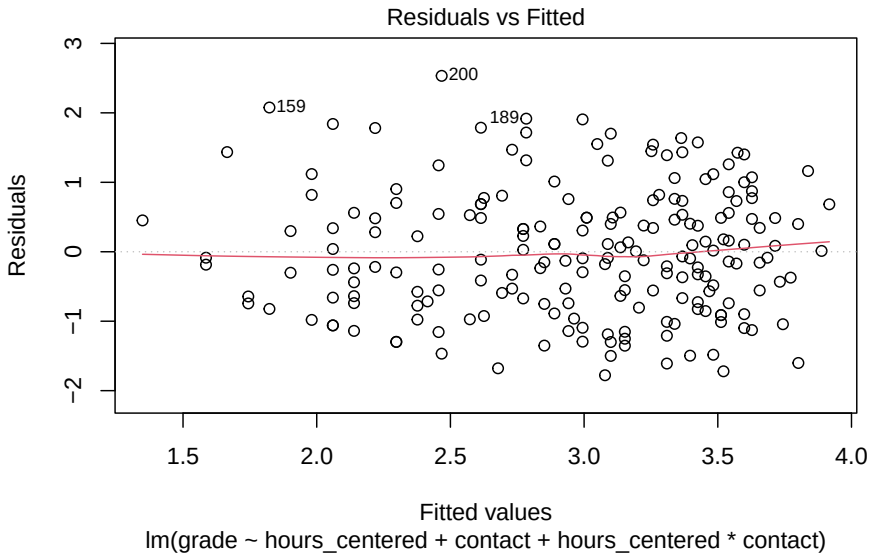
In practice we would have reached these conclusions more quickly by inspecting the output from our model and just subtracting the corresponding interaction effect from the effect for `hours_centered`, but it is important to understand what happens in the formula to get a full grasp on linear regression models.

In the model without the added interaction we concluded that on average each hour invested above the mean decreases the final grade by about -0.5. Now we see that the effect of hours depends on the form of contact had. This reflects the theoretical assumption from our DAG that time can be used more efficiently the more personal the form of contact was.

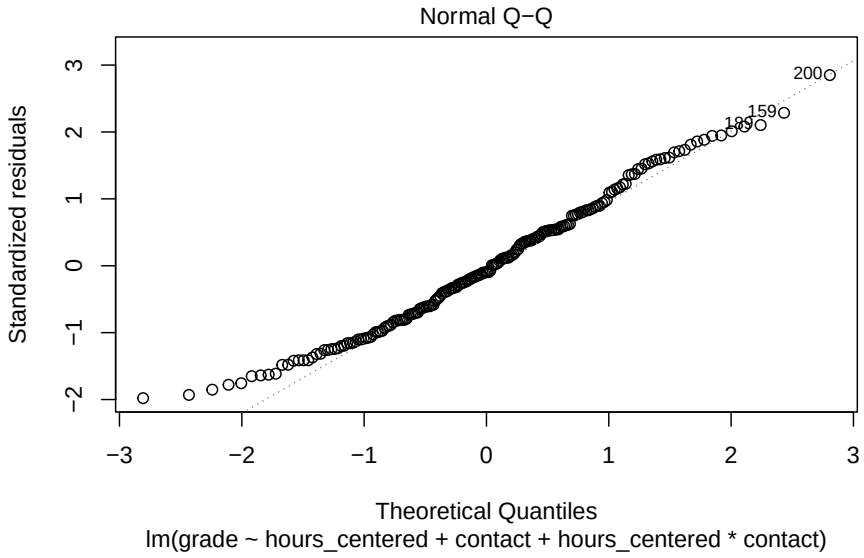
The same DAG that informed our best model from last week now lead us to including the interaction. This underlines the importance of thorough theoretical thinking before starting to model. It was not even the case that the DAG was wrong, but our conclusions we draw from it were at least not completely right. If we had invested more time, we could have build the correct model directly. In our case we first needed the regression diagnostics to tell us that something might be off before we figured out our error.

7.4.2 Regression diagnostics (revisited)

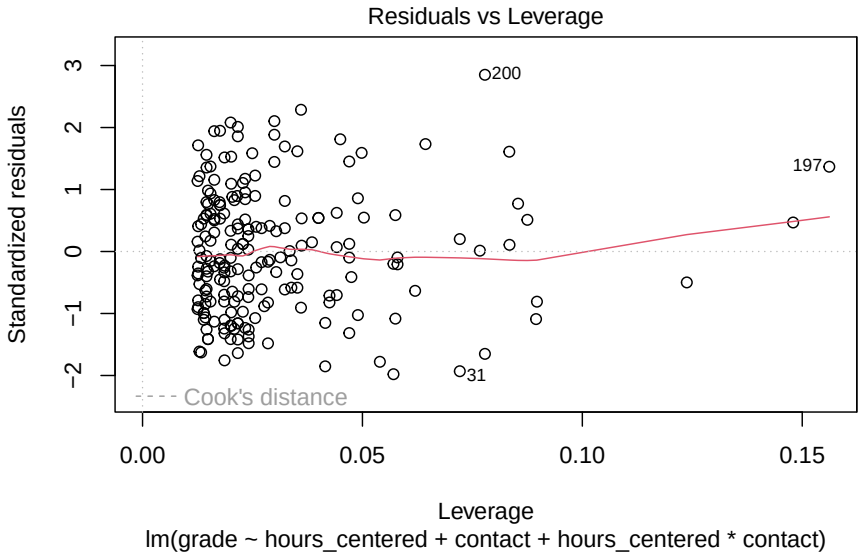
We now have a theoretically sound model, but did we also solve the problems indicated in the regression diagnostics?



The residuals vs. fitted plot now shows an almost straight horizontal line with no clear visible patterns. This indicates that the problem we saw with our former model actually came down to a missing variable, or to be more precise a missing term in our case.



The Q-Q plot now also shows more normally distributed residuals. While there are still some small deviations at the lower tail these do not indicate a remaining severe problem. The deviations are smaller than before and also not drastic in absolute terms. Also, as stated above, violating the normality assumption is less problematic with a high n and few variables in the model, which is still true for our case.



Adding the interaction term actually increased the leverage of the more influential observations. When we recompute the threshold as $2 * (5 + 1)/200 = 0.06$, we see that there still some observations with values higher than this. What has not changed, is that there are no clusters of observations in the lower or upper right corners. Overall we can conclude, that this problem is present but negligible. When there are observations that are problematic on both values at the same time, these also get marked by a red dashed line in the plot. This is also not present.

7.5 Conclusion

Over the last three sessions we have learned what a linear regression is, how its formula worked, how to interpret the results for different kinds of variables as well as how to check and correct violations of its underlying assumptions.

At the same time we built a model, which in its final version is able to accurately estimate our effect of interest. But we had one immeasurable advantage: We simulated the data ourselves and thus knew where the journey was going to end up from the start. We new our DAG was correct because the data was simulated in this way and we also knew that there was going to be a interaction effect to solve the remaining diagnostic problems. Sneaky, right? But in real world data, we do not have these advantages. Our DAGs can be incorrect and we may or may not find the missing part of the puzzle that elevates an OK model to a great one. All we can do is think, explore our data, think again, run

diagnostics, think again and maybe most importantly do not give up along the way.

In the next session we will return to our NBA data and try to apply everything that we have learned over the last sessions.

Chapter 8

Linear Regression - Application

After we have learned the ins and outs of linear regression we will now return to our NBA data. We already saw, that there was an interesting relationship between the points a player makes per game and the salary he receives in session 2. In session 4 we also built a DAG that reflects our assumptions about the data generating process. Based on the DAGs implications we can now build a linear regression model and try to estimate the effect of interest as accurately as possible.

8.1 Objectives

- Estimate the effect of interest, scored point on salary, using a linear regression
- Applying diagnostics to the model and correct mistakes
- Interpret the final model

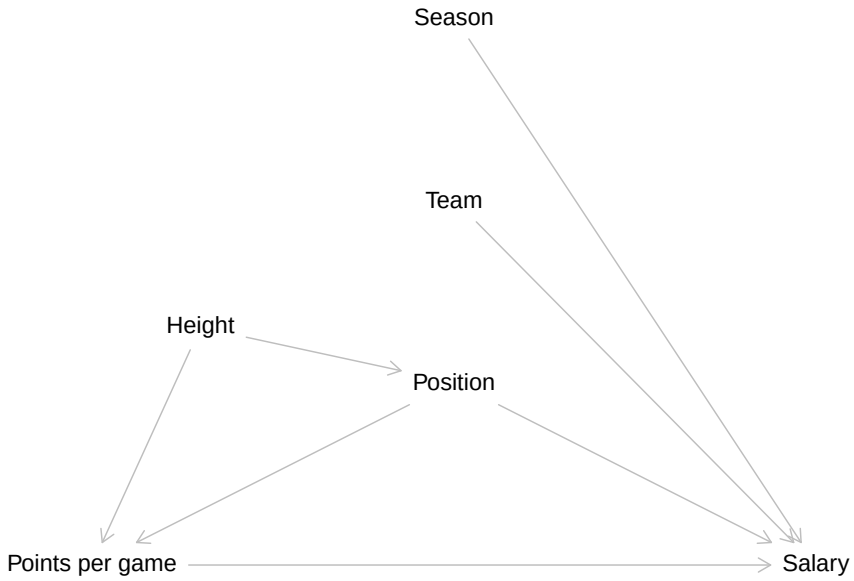
8.2 R functions covered this week

- `lm()` XXX FILL IN XXX

8.3 Research question

Picking up from session 2 & 4, our research question is still to get an unbiased estimate for the effect from points scored on the salary an NBA player receives.

We already constructed a DAG that reflects our assumptions for the underlying data generating process. Let us revist this briefly:



The implications of our DAG were that we only have to control for the position a player occupies to get an unbiased estimate for our effect of interest. The path that passes the body height is already closed by controlling for position and the team a player plays for as well as the season an observation was recorded in do not lie on an paths from our main independent to our dependent variable. Based on this we construct our model.

Now let us get to it and load the NBA data we prepared in week 2.

```
library(tidyverse)
load("../datasets/nba/data_nba.RData")
```

8.4 Simple linear regression in R

To conduct a multiple linear regression in R, we can use the built-in *base R* function `lm()`, short for *linear model*. The function is straightforward to use. As the first argument we write the regression formula in R's *formula syntax*.

We start building the formula by writing the name of our `dependent_variable` followed by a *tilde* `~`. You can read this as an = or as "regress the dependent variable on". After the tilde we add our first `indepedent variable` by again writing out its name. If we have multiple independent variables in our model -

when we are running a *multiple linear regression* - we can add those by writing a + followed by the name of the variable to be added.

As a second argument, the function needs the name of the object that holds our data.

The goal of our research question is to estimate the effect of the points per game on the received salary. So to regress `salary` on `career_PTS`, we just write:

```
lm(salary ~ career_PTS, data = data_nba)
```

```
##
## Call:
## lm(formula = salary ~ career_PTS, data = data_nba)
##
## Coefficients:
## (Intercept)   career_PTS
##     -851914       552843
```

This gives us a short output. The first line just echoes our code used to run the regression. We have seen this in the last session already, but now we know what the meaning was. After this we have a short block with the estimated coefficients. As we have run a simple linear regression, we only get the intercept and the coefficient for the sole independent variable used in the model. If we would have run a multiple linear regression, the result would basically look the same, only with more coefficients to display.

Before we dive into the results, we should talk about how to receive a more verbose output that does not hide all the other vital information that is associated with the model.

The easiest way is to use the base R function `summary()`. This is a generic R function that returns different summaries, depending on the object it is used on. We can for example use it on a data frame or tibble to get some descriptive statistics for the included variables. For example, we can information on the distribution of points per game by writing:

```
summary(data_nba$career_PTS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   5.100   8.000   8.908  12.000  30.100
```

When we use `summary()` on a model object, like the one created by `lm()`, we get a different output. Before we apply this we should save our model in an object. This is good practice in most cases as we can now apply all additional analysis of the model on this object and we do not have to rerun the model every time.

```
m1 <- lm(salary ~ career_PTS, data = data_nba)
```

We can now apply `summary()` on the object `m1`, short for "model 1":

```
summary(m1)
```

```
##
## Call:
## lm(formula = salary ~ career_PTS, data = data_nba)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -14788659  -2023969   -434599   1311807  24326060
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -851915      76417  -11.15   <2e-16 ***
## career_PTS    552843       7453   74.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3732000 on 9726 degrees of freedom
## Multiple R-squared:  0.3613, Adjusted R-squared:  0.3612
## F-statistic:  5502 on 1 and 9726 DF,  p-value: < 2.2e-16
```

This is the output we saw over the last weeks and it includes extended and better readable coefficient block as well as the information on the residuals and the model fit.

An alternative method of displaying the coefficients in a regular tibble format, is to use `tidy()` from the `broom` package.

```
library(broom)
```

```
## Warning: package 'broom' was built under R version 4.2.3
```

```
tidy(m1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) -851914.    76417.     -11.1 1.09e-28
## 2 career_PTS   552843.     7453.      74.2 0
```

8.4.1 Interpretation

While we know our model is not complete yet, let us still inspect the results. For each point a player scores per game, his salary rises by about 552,000$. We see a clear positive and substantial effect. Let us also inspect the intercept. This tells us that a player who makes no points per game has to pay the team about 850,000. Wait, this does not make sense… To make the intercept more readily interpretable we should again center our metric dependent variable `career_PTS` on its mean.

```
mean(data_nba$career_PTS)
```

```
## [1] 8.907679
```

```
data_nba <- data_nba %>%
  mutate(PTS_centered = career_PTS - mean(career_PTS))
```

As we have now centered the independent variable of interest on its mean of 8.9 we can rerun the model.

```
m1 <- lm(salary ~ PTS_centered, data = data_nba)

summary(m1)
```

```
##
## Call:
## lm(formula = salary ~ PTS_centered, data = data_nba)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -14788659  -2023969   -434599   1311807  24326060
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4072633      37840  107.63   <2e-16 ***
## PTS_centered   552843       7453   74.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3732000 on 9726 degrees of freedom
## Multiple R-squared:  0.3613, Adjusted R-squared:  0.3612
## F-statistic:  5502 on 1 and 9726 DF,  p-value: < 2.2e-16
```

The coefficient for points per game has not changed but its interpretation has. For each point per game over the mean of 8.9 points per game, the salary is

estimated to increase by about $552,000$\$. At the same time, for each point below the mean the salary is estimated to decrease by the same amount. The intercept now shows us the estimated salary of a player who scores 8.9 points per game which is slightly upwards of $4,000,000$\$ This makes way more sense.

This model model already achieved a considerable R^2 of 0.36. About 36% of the variance in salaries is explained by the points per game a player scores.

8.5 Multiple linear regression in R

The DAG we have constructed above based on our research question indicated that we also have to include the position a player occupies in our model. We can add additional independent variables to the formula used in `lm()` with a `+` and the name of the additional variable(s). This works the same way for all types of variables, i.e. metric, dummies or categorical variables. So let us do this now by adding the 5 dummies we constructed for the positions:

```
m2 <- lm(salary ~ PTS_centered + position_center + position_sf +  position_pf + positi

summary(m2)
```

```
##
## Call:
## lm(formula = salary ~ PTS_centered + position_center + position_sf +
##     position_pf + position_sg + position_pg, data = data_nba)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -14511723  -1950255   -372906   1358768  24433660
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3679728     114300  32.193  < 2e-16 ***
## PTS_centered      568019       7474  75.994  < 2e-16 ***
## position_center  1380246     114539  12.050  < 2e-16 ***
## position_sf       125384     102174   1.227 0.219790
## position_pf       206505      94882   2.176 0.029547 *
## position_sg      -331033      96841  -3.418 0.000633 ***
## position_pg      -114552     117486  -0.975 0.329572
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3652000 on 9721 degrees of freedom
## Multiple R-squared:  0.3888, Adjusted R-squared:  0.3884
## F-statistic:  1031 on 6 and 9721 DF,  p-value: < 2.2e-16
```

8.5.1 Interpretation

We still see a clear positive effect of points per game on the received salary after controlling for the position a player occupies. Among those centers are by far the top earners, making about $1,400,00$ more than players on other positions. Most other positions show relatively small effects on the earnings. Power and small forwards earn somewhat more than other positions on average while point and especially shooting guards earn less.

We can now compare two fictive cases of a center and a point guard who each make about 20 points per game. What is the estimated salary for them?

As we have extensively worked with the formulas over the last sessions, we can now keep it short and calculate the estimate directly. Remember that we centered the points per game on the mean of about 8.9, so making 20 per game would mean making about 11.1 than the average player. We will keep it simple here and calculate with 11.

$$\hat{y_{center\_20}} = 3679728 + 568019 * 11 + 1380246 = 11,308,183$$

$$\hat{y_{pg\_20}} = 3679728 + 568019 * 11 - 114552 = 9,813,385$$

Despite making the same amount of points per game for their team, the model estimates that a point guard earns about $1,500,000$ less compared to a center.

8.5.2 Sidenote: Adding interactions

We will not use interactions in this session but we briefly want to state how we could add them in the formula syntax.

Remember that interactions are multiplicative terms in our regression formula. Adding them to the R formula syntax works the same way. We add the new term with a `+` and use a `*` between the two variables that we want to interact.

Here is a non running toy example where we interact two x-variables:

```
lm(y ~ x1 + x2 + x1 * x2, data = some_data)
```

8.6 Regression Diagnostics

So how does our model perform? Did we meet all the regression assumptions that were introduced last week?

To access the visual tests we used last session, we can just use the base R function `plot()`, applied to the model object. If we just write `plot(m2)`, the

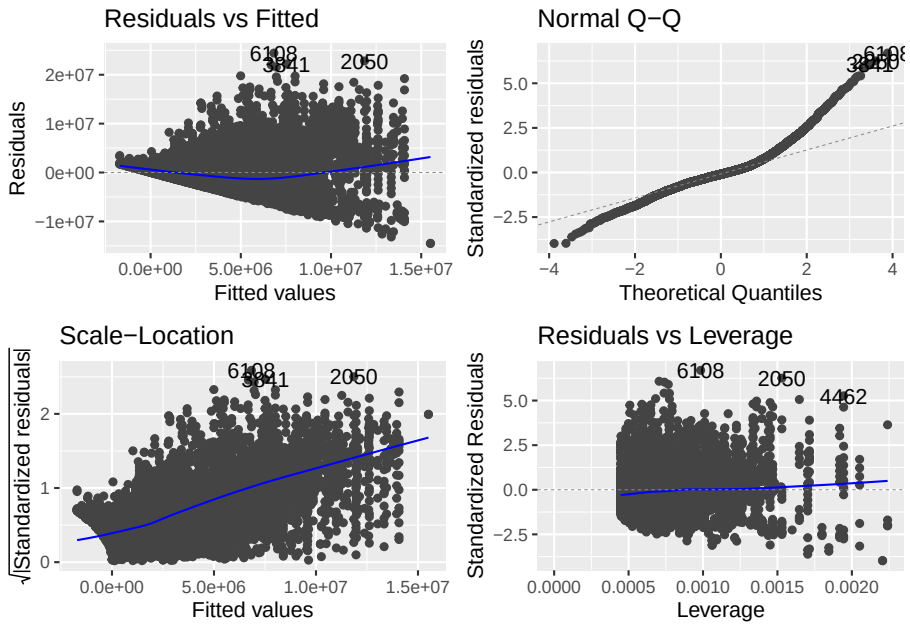
console asks us to press `ENTER` to go through each plot one by one. We can also add a number as a second argument, specifying which plot we want to see. For example, `plot(m2, 1)` gives us the residuals vs. fitted plot.

But there is an easier way to see all four plots at once. The package `ggfortify` expands the functionalities of `ggplot2` so that we can use it's `autoplot()` function to automatically plot all four visual tests of interest. An added benefit, depending on your taste, is that the plots are rendered in the style of `ggplot2`.

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.2.3
```

```
autoplot(m2)
```



The residuals vs. fitted does not show us a more or less straight line but starts mildly positive, then dips below 0 and rises again for higher estimated salaries. This could indicate at least two thing. Either we have missed an important independent variable, like in the last session, or we are actually dealing with some amount on non-linearity. If it is non-linearity, it is still *mild* non-linearity, but maybe we should still inspect this.

The Q-Q plot this time shows that the actual residuals are far from being distributed normally. While we can never expect a perfectly normal distribution, here the deviations are striking, especially for high residuals.

The scale-location plot is used to address the assumption of homoscedasticity. What we want to see, is a straight line with data points equally distributed around it. This clearly is not the case here. As it is, the plot indicates that we may be able to estimate small salaries reasonably well but that the higher the estimate, the more unreliable our model gets.

The residuals vs. leverage plot also indicates some problems. There are some observations that have larger or smaller standardized residuals compared to the thresholds of 3 and -3. The threshold for leverage is computed as $2 * (6 + 1)/9728 = 0.001439145$. We also see some observations with higher values. While both are rules of thumb and may not necessarily point to severe problems by themselves, things can get problematic when there are observation that do not meet the thresholds for both measures at the same time. This is indicated by clusters in the lower or upper right corners. This time we can observe this in the lower right.

We should also test for multicollinearity. We can compute the VIF measures using a function from the package `car`.

```
library(car)

vif(m2)
```

```
##    PTS_centered position_center     position_sf     position_pf     position_sg
##        1.050400        2.035722        1.686736        1.512765        1.565004
##     position_pg
##        1.916933
```

The values for any variable should not exceed 5 and should be closer to 1. Our value for points shows no signs of multicollinearity. The values for the position dummies have somewhat higher values, which makes sense. While there are some players that play multiple positions, for most a value of 1 on one position predicts the other positions as having a value of 0. But as we are still far away from the threshold of 5, there is no need for concern here.

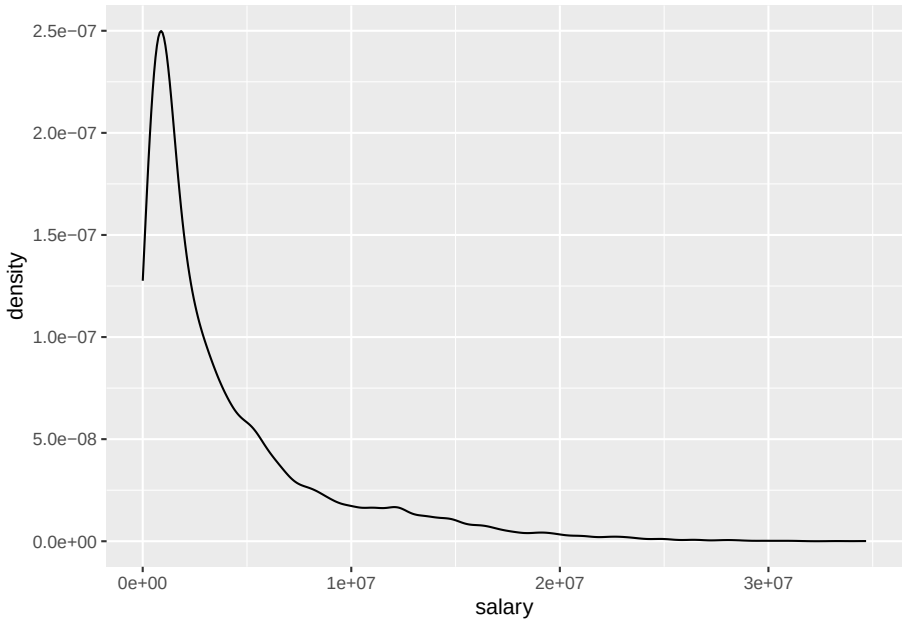
Overall, we have problems! While we do not see signs of problematic multicollinearity, all other tests indicated clear and in parts severe problems. We have to put in some more work before we can be confident that our model accurately predicts the effect of points per game on the received salary.

Before we start addressing the problems, we should not that the four plots are highly interactive. It is entirely possible that solving one of the problems also solves the others or, for added fun, even generates new ones. This means that we should refrain from turning too many dials at once and rather change the model one step at a time, see if it improves things and then address remaining problems in the same way.

8.6.1 Skewed outcome variable

The deviation from normality and the clearly present heteroscedasticity could both point to the same problem, namely a skewed dependent variable. Let us examine its distribution first.

```
data_nba %>%
  ggplot(aes(x = salary)) +
  geom_density()
```



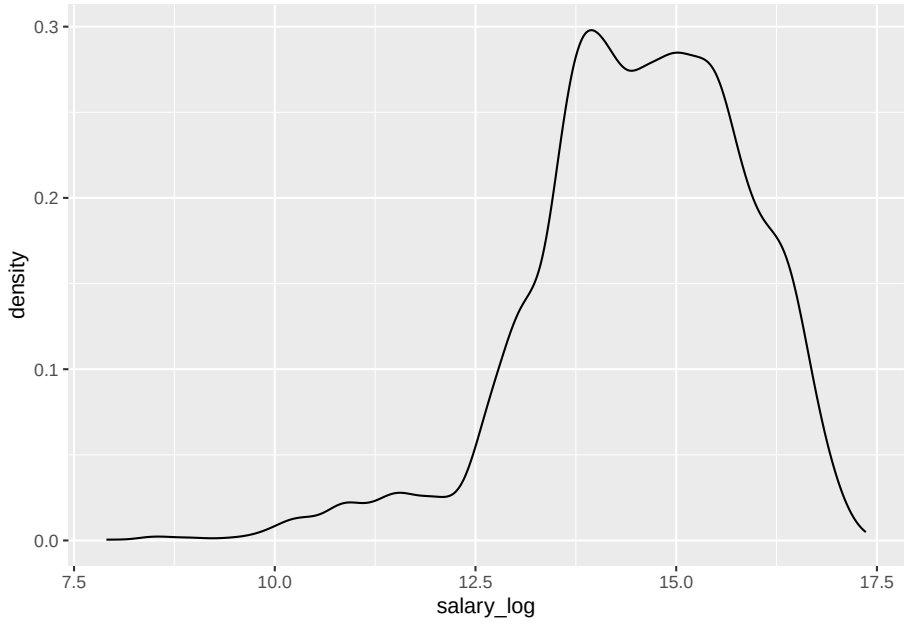
Our outcome variable is not only skewed, it is **highly skewed**. While there are many salaries in the "lower" six to seven digits regions, we also see some extremely high wages up to about $35,000,000$\$. The higher the salary the fewer observations we have. That is why we see such a long flat tail to the right.

This distribution actually is relatively common for income data. In most surveys of the general population we have many people receiving relatively low incomes while fewer individuals receive higher or extremely high incomes. It is still interesting that this also holds true for a population of high earners such as NBA players. Inequality is relative. Compared to the general population almost all our players would be somewhere in the long tail to the right. Compared to their own population we still see highly substantial differences in outcomes.

We can transform the dependent variable to a different scale to get a less skewed distribution. A common transformation for income data is to take the *logarithmus naturalis* of the actual value and then use this as our dependent variable.

To achieve the transformation we can simply use the base R function `log()` which as its default computes the ln.

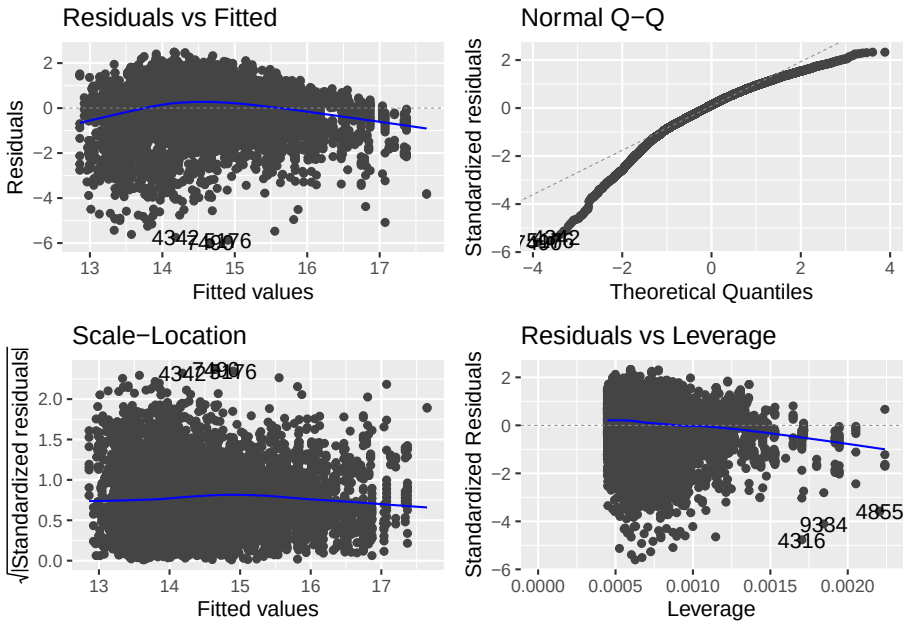
```
data_nba <- data_nba %>%
  mutate(salary_log = log(salary))

data_nba %>%
  ggplot(aes(x = salary_log)) +
  geom_density()
```



While the distribution of the transformed variable also is somewhat skewed, now to the left, overall it is much more evenly distributed.

We should use the new variable as our outcome an check the tests again.

```
m3 <- lm(salary_log ~ PTS_centered + position_center + position_sf +  position_pf + position_sg +

autoplot(m3)
```



Looking at the scale-location plot first, we can now see a straight line with our residuals fairly evenly distributed around it. Thus we now longer see any signs of heteroscedasticity. The Q-Q plot now also indicates a somewhat more normal distribution of our residuals but there are substantial deviations still. While high residuals now appear to more or less follow the normal distribution, small residuals now deviate stronger than they have before. This reflects the transformation and its distribution, which now has long tail on the left and not on the right anymore. Turning to the residuals vs. leverage plot we still see some observations that do not meet the respective thresholds. At the same time, there appear to be less that simultaneously have high absolute standardized residuals and high leverage. The residuals vs. fitted plot now also shows a more even distribution while the signs on non-linearity remain. We do not have to recompute the VIF measure as we did not change any independent variables in the model.

8.6.2 Non-linearity

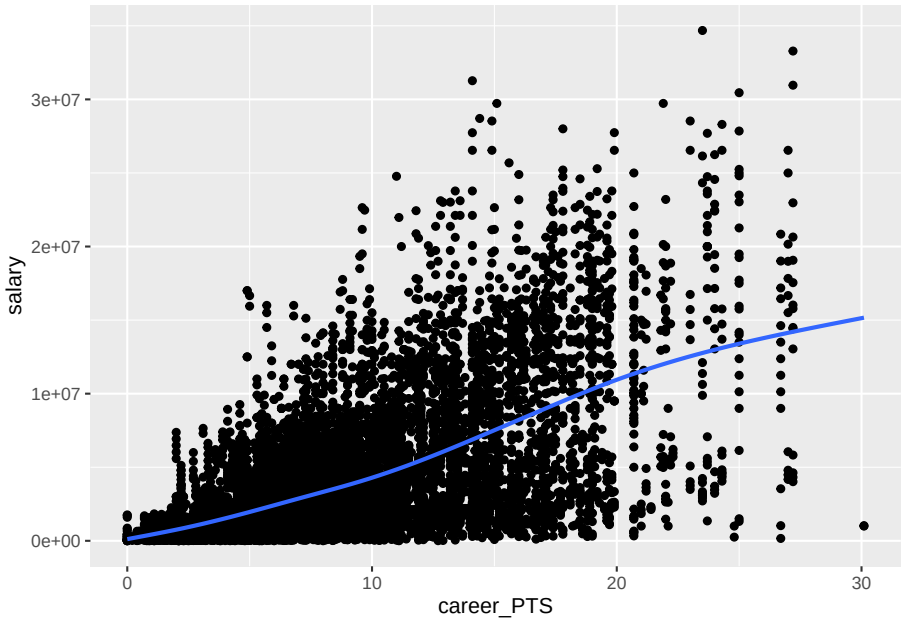
Let us now address the non-linearity that is still indicated in the first plot. We can approach non-linear relationships in our inherently linear model by adding non-linear transformations of a dependent variable to the model. But before we start squaring random variables, we should think about what could be non-linear in our case. We can rule out our dummy variables for position. This leaves the points scored. The model already tells us that our suspicion that salary rises with the points scored could be true. But maybe this relationship is

not linear over its whole range. If you already are among high scorers, scoring one or two points more than your peers may not be such a substantial difference and thus may not have the same strong effect on salary.

We should first inspect the relationship between both variables again. This time we add a LOWESS curve to the plot. This is often helpful in detecting non-linearity as the curve can change its slope over the range of the dependent variable. This is also the default for `geom_smooth()`.

```
data_nba %>%
  ggplot(aes(x = career_PTS, y = salary)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

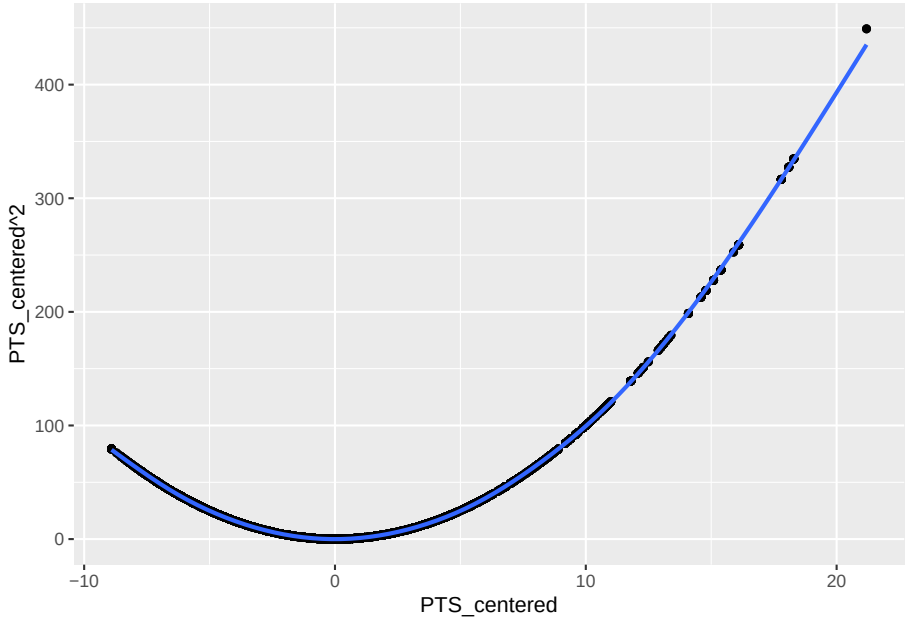


While this is not as clear as we hoped, the line may still indicate some mild non-linearity as it flattens somewhat for really high point values. Also we have to keep in mind that the non-linearity may be stronger when we control for additional variables, as our position dummies.

One common way to address the non-linearity is taking the square of the dependent variable in question. We should not square our centered points variable though. Let us inspect what would happen if we squared it.

```
data_nba %>%
  ggplot(aes(x = PTS_centered, y = PTS_centered ^ 2)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

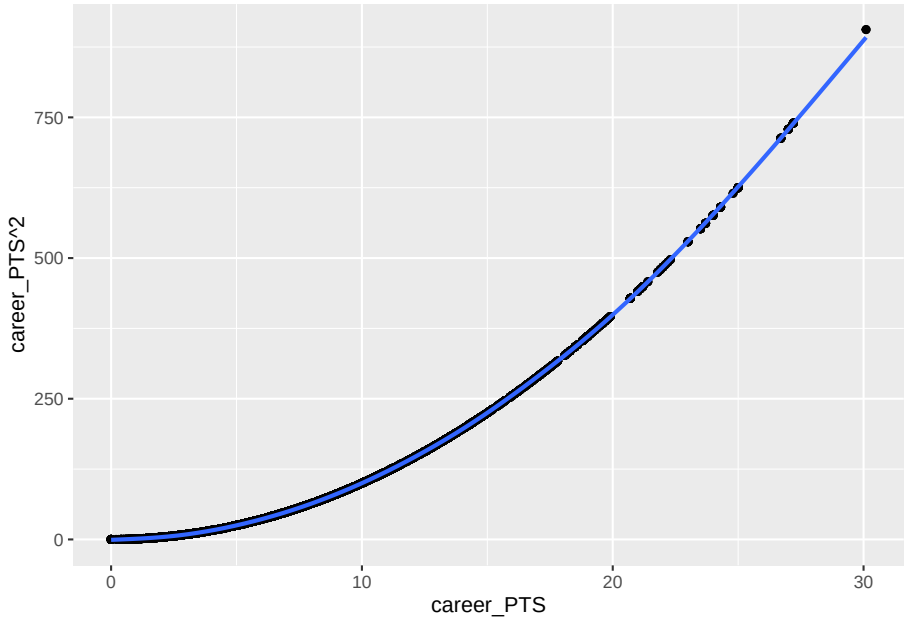


As the square of negative values is positive we would basically introduce the assumption into the model that there is non-linearity for low and high scorers and that the effect will be in the same direction. While our assumption is that there are diminishing returns between being a high scorer and a **really** high scorer, we do not assume that making more points if you are among the lower scorers should have the same effect. If at all, in these regions additional points could have an even larger effect.

Because of this we should return to the uncentred version of our variable. What happens if we square this?

```
data_nba %>%
  ggplot(aes(x = career_PTS, y = career_PTS ^ 2)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



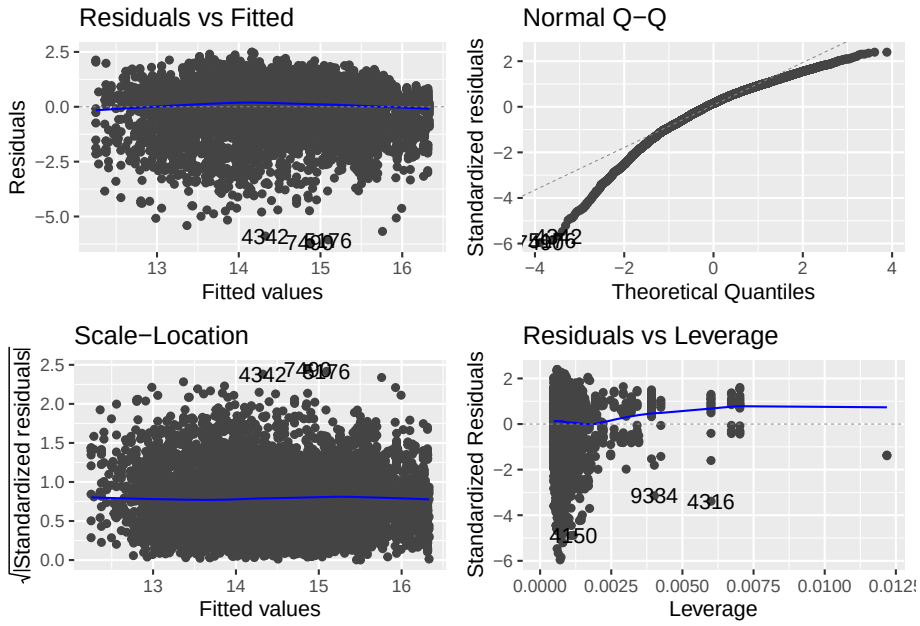
This is what we wanted, a transformation that expects a stronger difference the higher the score value is. For the final model we will thus work with the uncentred variable. We included the centered version because it is more straightforward to interpret. This is not really a concern anymore because dreams of easy interpretability are long gone after transforming two variables.

We could again transform the variable in our data and thus add a second version, but we can also do so directly in the formula syntax. When we use the function `I()` we tell R to interpret anything within the parentheses as a mathematical expression. This is what we will do below. Note that we add the point variable as its untransformed and transformed versions. The first represents the linear parts and the second the non-linear parts of the effect.

```
m4 <- lm(salary_log ~ career_PTS + I(career_PTS^2) + position_center + position_sf + position_pf
```

We can now reassess the tests for the last model.

```
autoplot(m4)
```



The line in the residuals vs. fitted plot got more straight. It seems that we actually captured the mild non-linearity that was present before by adding the squared points value to our model. The scale-location plot also still indicates no more problems of heteroscedasticity. Contrary, the data points now are even more evenly distributed compared to `m3`. The Q-Q plot also has not substantially changed, still showing non-normally distributed residuals. We can not really fix this now, but we also learned that this test is less consequential if we have a large n. Turning to the residuals vs. leverage plot, we still see several points that do not meet the thresholds but at the same time we do not see any points with high values for both. Overall there seem to be no overly influential points which we had to address. Let us also reexamine the VIF.

```
vif(m4)
```

```
##      career_PTS I(career_PTS^2) position_center     position_sf     position_pf
##       12.135096       11.883255        2.040563        1.709312        1.520284
##     position_sg     position_pg
##        1.569463        1.949078
```

We now see high VIF values for both versions of our point variable, which is the only substantial change. Did we introduce a new problem? If we take the measure at face value, yes. But if we think about it, no. All this means is that both versions of our variable are highly correlated. Of course they are. One is computed from the other. We can perfectly predict the value of `career_PTS^2`

from career_PTS. There is collinearity by design. If we want to assess multicollinearity we should apply the function to m3. If would have used interactions, the situation would be similar. This is just a small reminder that all our tests do not work without thinking about what we are actually doing.

8.7 Returning to our research question

As we now settled on m4 as our best model, it is time to discuss what we actually found out about the effect of scored points on the received salary.

```
summary(m4)
```

```
##
## Call:
## lm(formula = salary_log ~ career_PTS + I(career_PTS^2) + position_center +
##     position_sf + position_pf + position_sg + position_pg, data = data_nba)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1886 -0.5744  0.1583  0.7318  2.4876
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     12.2544680  0.0428953 285.684  < 2e-16 ***
## career_PTS       0.3124511  0.0072287  43.224  < 2e-16 ***
## I(career_PTS^2) -0.0071879  0.0003113 -23.092  < 2e-16 ***
## position_center  0.5482072  0.0326289  16.801  < 2e-16 ***
## position_sf      0.1067444  0.0292656   3.647 0.000266 ***
## position_pf      0.1214253  0.0270641   4.487 7.32e-06 ***
## position_sg     -0.0025577  0.0275935  -0.093 0.926150
## position_pg      0.0312286  0.0337077   0.926 0.354234
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.039 on 9720 degrees of freedom
## Multiple R-squared:  0.3978, Adjusted R-squared:  0.3973
## F-statistic: 917.1 on 7 and 9720 DF,  p-value: < 2.2e-16
```

The more points a player scores, the higher the salary is estimated. At the same time we have identified a non-linear aspect to this relationship. The non-linear effect is small but negative. This indicates diminishing returns for high scorers. The higher the score, the less positive the effect of additional points is.

The problem after two transformations of involved variables is, that interpretation has lost all its intuitiveness. The effects now describe the change in

logarithmised salaries. While we can still easily assess if an effect is positive or negative we do not really know what the magnitude of an effect is. For this we have to reverse the transformation.

Let us revisit our example of a center and a point guard making 20 points per game from above. Note that we also have to take the square of points for the non-linear term.

$$\widehat{y_{center\_20}} = 12.2544680 + 0.3124511 * 20 - 0.0071879 * 20^2 + 0.5482072 = 16.17654$$

$$\widehat{y_{pg\_20}} = 12.2544680 + 0.3124511 * 20 - 0.0071879 * 20^2 + 0.0312286 = 15.659565$$

To find out what this means in hard dollars, we have to reverse the logarithmus naturalis. We can do this by calculating $e^{\hat{y}}$. R gives us the function `exp()` to do just that.

```
exp(16.17654)
```

```
## [1] 10601860
```

```
exp(15.659565)
```

```
## [1] 6322119
```

For our center who scored 20 points we thus estimate a salary of $10,601,860$\$, for a point guard with the same amounts of points $6,322,119$\$. These estimates are not only lower than the ones derived above, the difference between the positions is also more pronounced. For a point guard, scoring additional hoops does not have the same payoff compared to a center. While the latter receive a higher pay in general they also receive more per additional point scored.

As the name suggests and as we have seen in our exploratory data analysis, point guards make a lot of points. At the same time they earn considerable less. Above we see an estimated difference of over $4,000,000$\$ between high scoring centers and point guards. The question remains why this is the case. Maybe there are some additional variables we had to consider to fully unravel this. For example it is reasonable to expect some effects on salary that are not connected to the performance in terms of scoring. Both positions fill different roles in basketball game. Centers, besides scoring, have to get rebounds and facilitate turnarounds. Point guards on the other hand have the role to build opportunities for their team and pass to other players. Both measures of performance were not considered in our model but are highly valuable to any team.

Also there is something we can call “starfactor” or “flashiness”. A center is much more visible in the game, takes big jumps and dunks. Maybe a center is not only more valuable in terms of performance but also in being an attracting force for fans. This could change the relationship between points and salary for players with a high “starpower”. Such a variable would be hard to measure, but could maybe solve the parts of the puzzle that still remain.

8.8 Moving on

But there is another possibility. We built the best model based on our DAG, but maybe the DAG is not entirely correct. Maybe we made some faulty assumptions or maybe we missed a variable with an important role in the data generating process. In the next session we will see, how we can come up with a different DAG that incorporates the same variables but makes some different assumptions.

Chapter 9

Linear Regression - Exercises

9.1 Exercises of Linear Regression

In this exercise, you will use the Boston Housing Dataset to explore the relationship between housing prices and various features of the houses and their surroundings. The dataset contains 506 observations and 15 columns. The last column, MEDV, is the median value of owner-occupied homes in $1000's. This is the target variable that you will try to predict using linear regression models. The other 14 columns represent different features of the houses and their surroundings, such as crime rate, nitric oxides concentration, pupil-teacher ratio, etc.

DRAFT

1. **Simple linear regression**: Use simple linear regression to predict the median value of owner-occupied homes (`MEDV`) based on a single predictor variable which is the average number of rooms per dwelling (`RM`). Fit the model.

```
# loading dataset
library(readxl)
BostonHousing <- read_excel("../datasets/boston.xlsx")
# applying linear regression
model_l <- lm(medv ~ rm, data = BostonHousing)
```

2. **Multiple linear regression**: Use multiple linear regression to predict `MEDV` based on multiple predictor variables, such as `RM`, `CRIM` (per capita

crime rate by town), and `LSTAT` (% lower status of the population). Fit the model, generate predictions, and calculate the R-squared value to assess the model's performance.

```
# applying linear regression
model_m <- lm(medv ~ rm + crim + lstat, data = BostonHousing)
```

3. **Model summary**: Generate summary of a both regression model, including information about the coefficients, standard errors, t-values, and p-values. Interpret the results. Calculate the R-squared value to assess both model's performance. Which model has better goodness of fit (R-Squared) ?

```
summary(model_l)
```

```
##
## Call:
## lm(formula = medv ~ rm, data = BostonHousing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.346  -2.547   0.090   2.986  39.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -34.671      2.650  -13.08   <2e-16 ***
## rm             9.102      0.419   21.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.616 on 504 degrees of freedom
## Multiple R-squared:  0.4835, Adjusted R-squared:  0.4825
## F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(model_m)
```

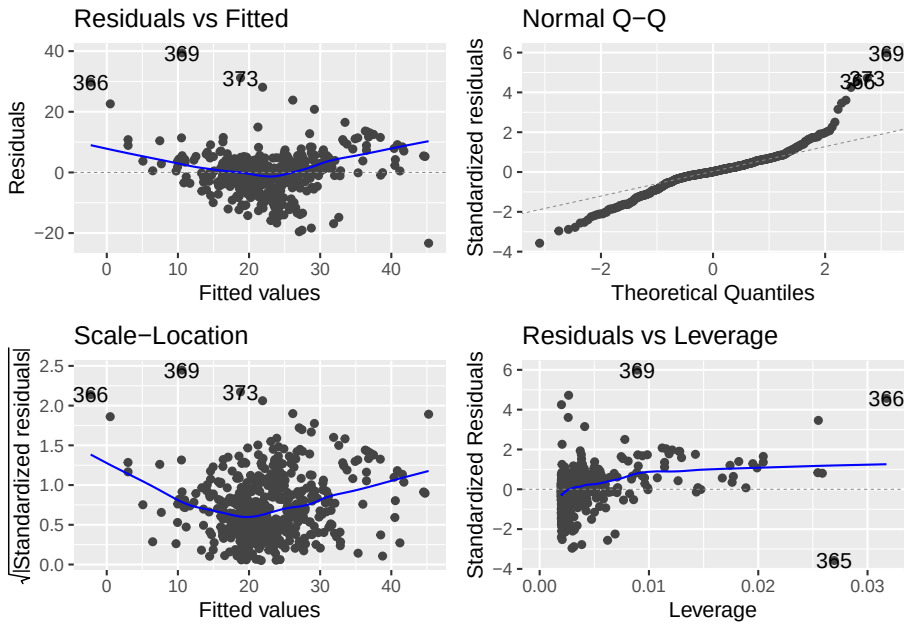
```
##
## Call:
## lm(formula = medv ~ rm + crim + lstat, data = BostonHousing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.925  -3.566  -1.157   1.906  29.024
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.56225    3.16602  -0.809  0.41873
## rm           5.21695    0.44203  11.802  < 2e-16 ***
## crim        -0.10294    0.03202  -3.215  0.00139 **
## lstat       -0.57849    0.04767 -12.135  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.49 on 502 degrees of freedom
## Multiple R-squared:  0.6459, Adjusted R-squared:  0.6437
## F-statistic: 305.2 on 3 and 502 DF,  p-value: < 2.2e-16
```

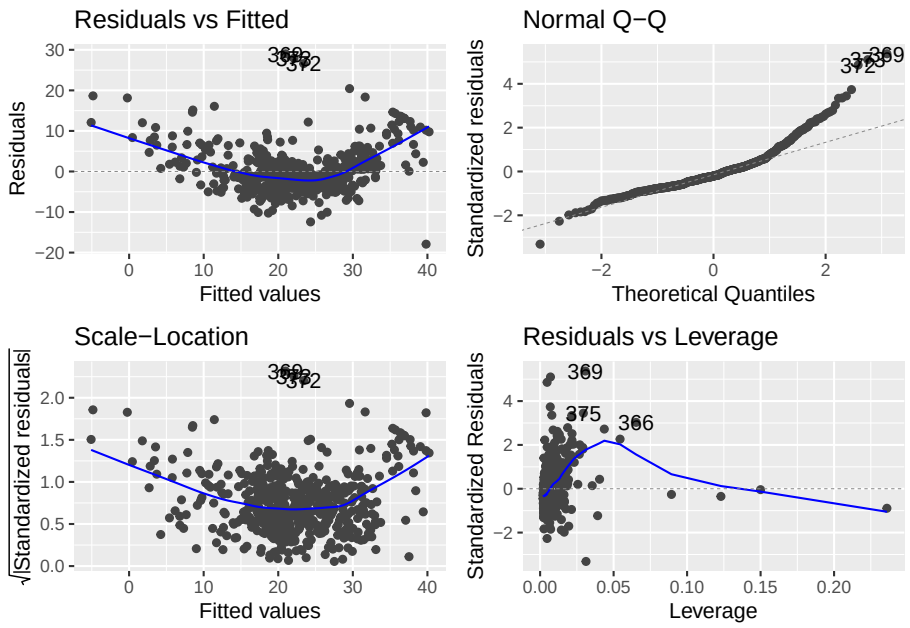
XXX–INTERPRETATION HERE–XXX

4. **Model diagnostics**: Create diagnostic plots to assess the assumptions of the models. Check whether your regression models from Tasks 1 and 2 satisfy the assumptions of linearity, multicollinearity, normality and homoscedasticity.Use numerical method variance inflation factor (VIF) to detect any multicollinearity issues among your predictor variables.

```
library(ggfortify)
library(car)
autoplot(model_1)
```



```
autoplot(model_m)
```




```
vif(model_m)
```

```
##       rm     crim    lstat
## 1.616468 1.271372 1.941883
```

XXX—INTERPRETATION HERE—XXX

Chapter 10

Mediation

In this week, we will introduce to the concept of mediation analysis.

Mediation is a way of using linear regression with a different interest in mind. So far, we have been interested in estimating the effect of an x variable on a y variable. In our example using NBA data, we were interested in the effect of 'points scored' by a player on player salary. We learned how to "adjust" or "condition" the effect by adding covariates to the model (i.e. control variables). Going back to out week on DAGs, we saw that we want to control for so-called confounders, i.e. those variables which have an effect on both x and y. The "adjusted" effect is the "total causal effect" of x on y. Mediation now is all about trying to explain "how" x affect y. Mediation analysis is about identifying the mechanism between x and y. In other words, what share of the total effect can be explained by something else (i.e. the mediator). When considering mediation, we can distinguish 3 types of effects:

- total effect (Effect of x on y (conditional on z), without mediator)
- direct effect (Effect of x on y (conditional on z), left after also adjusting for the mediator))
- indirect effect (The effect of x on y (conditional on z) which only goes through the mediator.

Go back to week X for a refresher.

Let's turn to our NBA data to make this more tangible. Let's assume now that we are interested in how a player's position on the team affects his salary. Maybe guards feel discriminated because they earn less.

First, we identify the year of the season as a confounder which we want to adjust. Tactics over the years changed, so the year when games took place had an influence on whether a player would be assigned to the guard or forward position, for example. Also, the NBA become more popular, more fans means more

money, and higher salaries for players. As a result, we will include "season_year" as a control variable.

Second, we have a hypothesis that the points scored on average affect the salaries. In the end, fans come to see spectacular dunks and shots, not rebounds and passes. Certain positions play further away from the basket (e.g. guards) and thus have a harder time scoring. They also are supposed to pass the ball to the big guys (usually centers). So maybe, the fact guards and forwards simply score less than centers explains their salary gap. We can test this using mediation.

To visualize this mini-theory and identification strategy, let's have a look at this DAG.

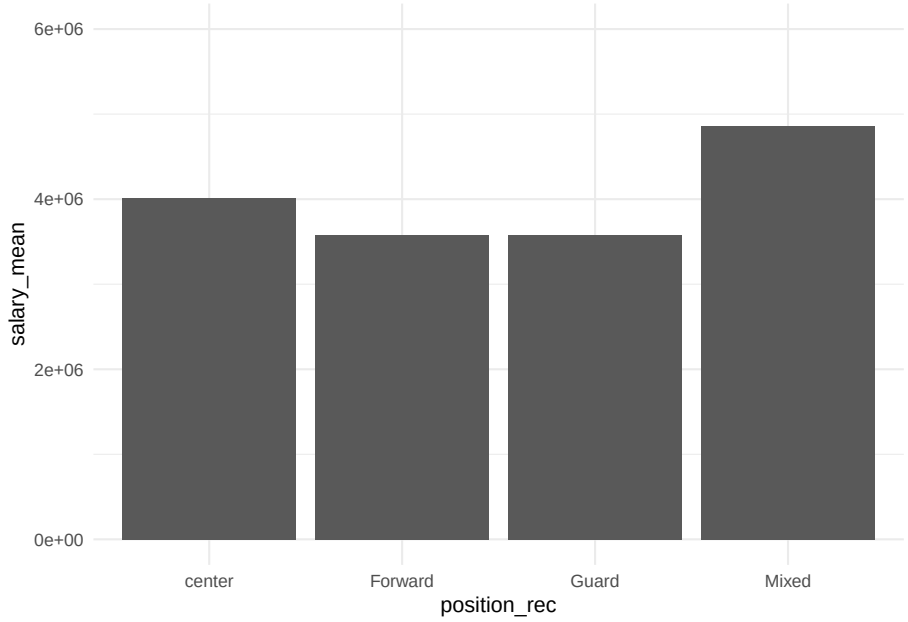
–> Jasper: visual here with the DAG from my powerpoint

Now let's test this out using R.

- First, we load the data.
- Second, we reduce the position variable to 4 categories: Center, Guards, Forwards, and Mixed. As we will see, it is actually not common that players take on several roles even within the same season. This likely affects their salary as well. Players that can play multiple positions may get more playing time and as a result, score more. We are interested, however, in the difference between "pure" guards, forwards and centers, so we will run the analysis excluding "mixed" position players.
- Third, we check whether there are any differences in average salaries by position using a graph and a table
- Fourth, we run a linear regression, regressing position (as factor variable) on salary while adjusting for year of season.

```
# load packages

load(file = "../datasets/nba/data_nba.RData")



# recode "position" variables"
data_nba <- data_nba %>%
  mutate(position_num = rowSums(across(c("position_center", "position_pf", "position_si
                                         "position_pg", "position_sg"))),
         position_rec =
           case_when(
             position_center==1 & position_num==1 ~ "center",
             position_pf==1 & position_num==1 ~ "Forward",
             position_sf==1 & position_num==1 ~ "Forward",
             position_pg==1 & position_num==1 ~ "Guard",
```

```
            position_sg==1 & position_num==1 ~ "Guard",
            position_sg==1 & position_pg==1 & position_num==2 ~ "Guard",
            position_pf==1 & position_sf==1 & position_num==2 ~ "Forward",
            TRUE ~ "Mixed"))

# pot mean salary by position
data_nba %>% group_by(position_rec) %>%
  summarize(salary_mean = mean(salary, rm.na =T)) %>%
  ggplot() +
  geom_bar(aes(x=position_rec, y=salary_mean), stat="identity") +
  ylim(0,6000000) +
  theme_minimal()
```




```
# linear regression
summary(lm(salary ~ as.factor(position_rec), data=data_nba, subset=(position_rec!="Mixed")))
```

```
##
## Call:
## lm(formula = salary ~ as.factor(position_rec), data = data_nba,
##     subset = (position_rec != "Mixed"))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4011485 -2826365 -1708014  1102456 31105006
```

```
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     4016014     126850  31.659  < 2e-16 ***
## as.factor(position_rec)Forward  -439249     158703  -2.768  0.00566 **
## as.factor(position_rec)Guard    -438470     149803  -2.927  0.00343 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4402000 on 6382 degrees of freedom
## Multiple R-squared:  0.001519,   Adjusted R-squared:  0.001206
## F-statistic: 4.855 on 2 and 6382 DF,  p-value: 0.007821
```

```
# adjust for confounder
summary(lm(salary ~ as.factor(position_rec) + as.factor(season_start), data=data_nba, s
```

```
##
## Call:
## lm(formula = salary ~ as.factor(position_rec) + as.factor(season_start),
##     data = data_nba, subset = (position_rec != "Mixed"))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5983717 -2733737 -1557867  1229567 29209309
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     2614310     272704   9.587  < 2e-16 ***
## as.factor(position_rec)Forward  -510295     156308  -3.265 0.001102 **
## as.factor(position_rec)Guard    -523632     147538  -3.549 0.000389 ***
## as.factor(season_start)1999       22191     334992   0.066 0.947186
## as.factor(season_start)2000      705147     348782   2.022 0.043245 *
## as.factor(season_start)2001      818596     352465   2.322 0.020238 *
## as.factor(season_start)2002     1035391     354870   2.918 0.003539 **
## as.factor(season_start)2003     1069793     359704   2.974 0.002950 **
## as.factor(season_start)2004     1117189     358397   3.117 0.001834 **
## as.factor(season_start)2005     1090880     354570   3.077 0.002102 **
## as.factor(season_start)2006      958637     352167   2.722 0.006504 **
## as.factor(season_start)2007     1654377     357706   4.625 3.82e-06 ***
## as.factor(season_start)2008     1944946     357392   5.442 5.46e-08 ***
## as.factor(season_start)2009     1832838     359404   5.100 3.50e-07 ***
## as.factor(season_start)2010     1748651     360028   4.857 1.22e-06 ***
## as.factor(season_start)2011     1582423     353355   4.478 7.66e-06 ***
## as.factor(season_start)2012     1651497     346695   4.764 1.94e-06 ***
## as.factor(season_start)2013     2121273     361746   5.864 4.75e-09 ***
```

```
## as.factor(season_start)2014     1243733     331916   3.747 0.000180 ***
## as.factor(season_start)2015     1804493     332700   5.424 6.05e-08 ***
## as.factor(season_start)2016     2645083     326043   8.113 5.89e-16 ***
## as.factor(season_start)2017     3382563     324615  10.420  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4327000 on 6363 degrees of freedom
## Multiple R-squared:  0.03774,    Adjusted R-squared:  0.03456
## F-statistic: 11.88 on 21 and 6363 DF,  p-value: < 2.2e-16
```

Looking at the regression ouput, we see that, indeed, centers make more than guards and forwards. Actually, a lot less, approx. 500.000 USD less on average.

Let's ignore for the moment that this is a terrible model. The r-squared is only 0.03 which means that position and season only explain ~ 3% of variation in salaries.

Now, we want to know whether scoring explain this salary penalty. Here is a simple mediation.

```
# let'S compare models side by side in a regression table

#install.packages("modelsummary")
library(modelsummary)
```

```
## Warning: package 'modelsummary' was built under R version 4.2.3
```

```
models <- list(
  m1 = lm(salary ~ as.factor(position_rec) + as.factor(season_start), data=data_nba, subset=(posi
  m2 <- lm(salary ~ as.factor(position_rec) + career_PTS + as.factor(season_start), data=data_nba
)

modelsummary(models)
```

At first look, our hypothesis actually does not seem to hold. When we adjust for points scored, the gap between centers, forwards and guards. This means that guards don't make less because they make less points. Guards and forwards with average points, compared to centers with average points, make even less than centers, compared to when we do not account for points at all. This could mean that teams know that guards are important despite the fact that they score less.

Let's use a new package for "causal mediation analysis".

	m1	
(Intercept)	2 614 309.605	−939 410.632
	(272 704.206)	(221 569.259)
as.factor(position_rec)Forward	−510 295.017	−1 413 516.300
	(156 308.018)	(123 630.475)
as.factor(position_rec)Guard	−523 632.304	−2 000 142.285
	(147 537.586)	(118 260.027)
as.factor(season_start)1999	22 190.937	302 958.632
	(334 991.928)	(263 200.021)
as.factor(season_start)2000	705 146.970	685 025.470
	(348 782.404)	(273 995.766)
as.factor(season_start)2001	818 595.739	860 051.794
	(352 464.794)	(276 889.168)
as.factor(season_start)2002	1 035 390.857	1 185 765.987
	(354 869.752)	(278 787.933)
as.factor(season_start)2003	1 069 793.041	1 154 734.396
	(359 704.108)	(282 578.656)
as.factor(season_start)2004	1 117 189.300	1 335 647.564
	(358 397.141)	(281 570.162)
as.factor(season_start)2005	1 090 880.186	1 408 951.168
	(354 570.248)	(278 588.367)
as.factor(season_start)2006	958 636.479	1 404 509.355
	(352 166.838)	(276 745.293)
as.factor(season_start)2007	1 654 377.014	1 740 042.609
	(357 705.703)	(281 008.828)
as.factor(season_start)2008	1 944 945.963	1 844 901.044
	(357 392.398)	(280 763.910)
as.factor(season_start)2009	1 832 838.212	1 469 738.620
	(359 403.707)	(282 398.558)
as.factor(season_start)2010	1 748 650.500	1 381 348.241
	(360 027.993)	(282 890.255)
as.factor(season_start)2011	1 582 423.228	1 261 385.859
	(353 354.876)	(277 634.621)
as.factor(season_start)2012	1 651 497.174	1 339 040.290
	(346 694.638)	(272 400.865)
as.factor(season_start)2013	2 121 273.316	1 360 644.125
	(361 745.546)	(284 436.812)
as.factor(season_start)2014	1 243 732.606	1 141 626.786
	(331 915.955)	(260 750.740)
as.factor(season_start)2015	1 804 493.065	1 543 464.545
	(332 699.559)	(261 394.268)
as.factor(season_start)2016	2 645 083.101	2 527 969.077
	(326 042.835)	(256 138.668)
as.factor(season_start)2017	3 382 563.347	3 172 576.179
	(324 614.892)	(255 032.022)
career_PTS		545 303.023
		(8677.934)
Num.Obs.	6385	6385
R2	0.038	0.406
R2 Adj.	0.035	0.404
AIC	213 275.6	210 194.7
BIC	213 431.1	210 357.0
Log.Lik.	−106 614.789	−105 073.355
F	11.882	197.861
RMSE	4 319 963.19	3 393 398.83

```r
# now let's try a dedidcated package for mediation
library(mediation)

data_nba <- data_nba %>% mutate(position_rec = as.factor(position_rec),
                                season_start = as.factor(season_start),
                                salary = salary/10)

mean(data_nba$salary)
```

```
## [1] 407263.3
```

```r
# first path from x to mediator
path_a <- lm(career_PTS ~ position_rec + season_start, data=data_nba, subset=(position_rec!="Mixe

# Now full model, x to mediator to y
path_b <- lm(salary ~ position_rec + career_PTS + season_start, data=data_nba, subset=(position_r

# centers vs. forwards
results_mediation_forwards <- mediate(path_a, path_b,
                              treat = "position_rec", # x or main independent variable
                              mediator = "career_PTS",
                              treat.value = 1)

# centers vs. guards
results_mediation_guards <- mediate(path_a, path_b,
                              treat = "position_rec", # x or main independent variable
                              mediator = "career_PTS",
                              treat.value = 2)

summary(results_mediation_forwards)
```

```
##
## Causal Mediation Analysis
##
## Quasi-Bayesian Confidence Intervals
##
##                 Estimate 95% CI Lower 95% CI Upper p-value
## ACME             9.03e+04     7.11e+04     109770.6  <2e-16 ***
## ADE             -1.42e+05    -1.65e+05    -117872.5  <2e-16 ***
## Total Effect    -5.13e+04    -8.29e+04     -21008.8  <2e-16 ***
## Prop. Mediated -1.82e+00    -4.92e+00         -0.9  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Sample Size Used: 6385
##
##
## Simulations: 1000
```

```
summary(results_mediation_guards)
```

```
##
## Causal Mediation Analysis
##
## Quasi-Bayesian Confidence Intervals
##
##                 Estimate 95% CI Lower 95% CI Upper p-value
## ACME            9.06e+04     7.07e+04     1.09e+05  <2e-16 ***
## ADE            -1.42e+05    -1.65e+05    -1.17e+05  <2e-16 ***
## Total Effect   -5.11e+04    -8.12e+04    -2.22e+04  <2e-16 ***
## Prop. Mediated -1.79e+00    -4.36e+00    -9.10e-01  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Sample Size Used: 6385
##
##
## Simulations: 1000
```

```
# note: this dies not yet work. FOrwads and guards are the same. CHeck how to
# take different groups of the treatment variable. Problem is that the treatment
# variable is categorical.
```

The output looks different. Let's go through it step by step:

- ACME: Average Causal Mediation Effect - "indirect effect", this is the effect of position through points on salary.
- ADE: Average Direct Effect - This is the direct effect of position on salary. + Total Effect: ADE and ACME together are the "total effect"
- Prop. Mediated. This is the proportion of the total effect that is "explained" by the indirect effect.

Let's interpret this for our model:

- ACME: If only points could explain salaries, Guards would make, on average 900.000 USD more than centers (because they actually score more points)

- ADE: Adjusting for points, guards would make 1.4 Million USD less than centers
- Total Effect: Overall, guards actually make 500.000 USD less.
- Prop. Mediated. This is actually a negative value which makes no sense in this case. If points scored really "mediated" part of the total effect and had some explanatory power here, then we would see a positive percentage.

When you take a look at our first approach, where we just estimated both regression models side by side. The coefficient for position is the total effect when we don't include the mediator in the model. The coefficient is the direct effect when we do include the mediator. The indirect effect is the total effect minus the direct effect (-1.400.000 - (-500.000) = -900.000).

In sum, Guards are disadvantaged. They receive less money than center, although, if the points were considered, they should make even more. There must be other reasons why guards earn less. We need to go back to theory and come up with new hypothesis.

10.1 Find out more

- provide resources here
- references etc.

Chapter 11

Prediction - Theory

In prior weeks, you learned how to build a linear regression model. The main interest we pursued so far was to arrive at a good estimate of one independent variable (points scored in NBA basketball league) on an outcome (salary of NBA players).

With the help of DAGs, we identified relevant "counfounders" we should adjust for to "isolate' the effect of points as much as possible and reduce bias. COnfounders enter as covariates in in the model.

With the help of mediation analysis, we were then interested what "explains" or "mediates" the effect of x on y. We use additional variables which we assume operate as mechanisms of the causal effect of x on y and we test how much of the effect of x and y can be attributed to this mediator. In our example, we found that points scored do not really explain why guards earn less money in the NBA compared to centers.

All what we have done so far can be considered part of causal inference, i.e. understanding why an outcome varies. A different perspective is the perspective of PREDICTION.Prediction is at the heart of appraoches in "data science" and "machine learning".

In this scenario, we build regression models (or other models) simply to predict and outcome. The main interest is not to learn more about how the outcome can be explained but to predict something with it which we want to know. Machine learning then takes it a step further and simply iteratively select the best models among hundrets of options to arrive at the best possible prediction (more on that at the end of the class). First, we will learn how to predict values based on a linear regression model.

11.1 How prediction works

For a linear regression model, prediction is very straight forward. Linear regression is all about finding a straight line through a cloud of points that lie on as many dimensions as there are variables in model. The model provides you with an intercept (i.e. where the line touches the Y-Axis, and a slope; the increase in y given one unit increase in x. The unit is whatever the scale of the x variable is). The formula is y= b + ßx + €. Any prediction is the on the line. You know the intercept, you plug in a value for x and you get your predicted y.

-> visual here.

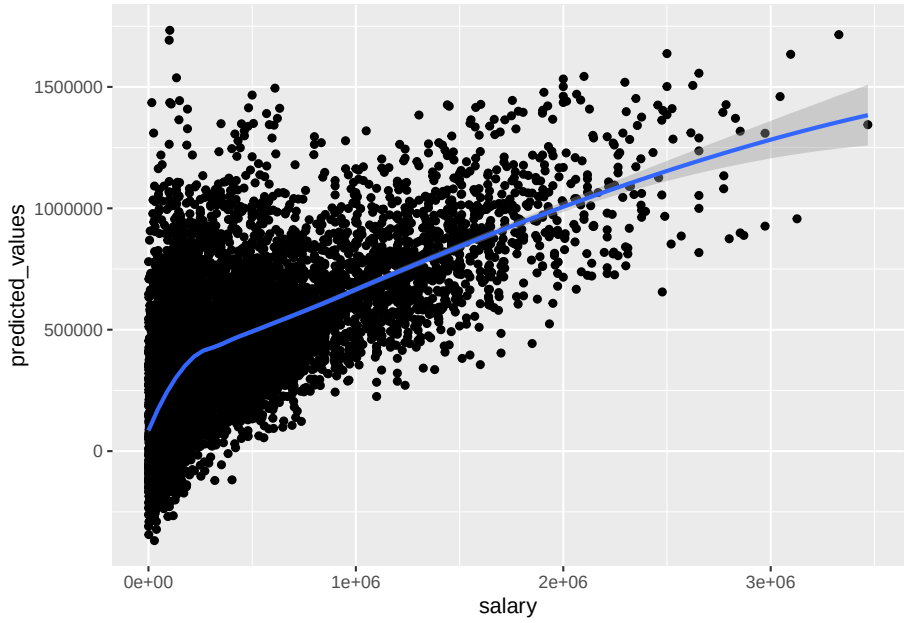
Let's apply this logic to our NBA data. Remember in prior weeks, we built a linear model to estimate the effect of points scored on average per game and salaries of players. Let's assume we now want to predict salaries of players and don't care too much about the scores. We can consider a range of variables which we think explain variation in salaries. The better we can capture variation in salaries between players, the more precise our prediction will be.

Now, you may rightfully ask: "Why do we want to predict salaries if we already know the actual salaries!?" Fair point. Prediction is commonly used to predict values which we don't have. Imagine there are some players that don't report their salaries. We could predict their salaries based on what we know from players who are similar to them in many other observable characteristics. Or imagine we want to predict the salary of a hypothetical player that does not exist. Imagine an average players would like to know how much he could earn more if player more like other players. We can predict that. Last example, imagine we want to forecast how much a player makes next year, depending on his past performance.

Machine learning is basically predicting outcomes that are no known based on very large datasets. You provide R with a million photos of animals, you build a model to explain which animal is a cat. The you apply the model to new data and the model predicts whether there is a cat in the photo. Of course, machine learning gets much more complicated quickly, however, the basic logic is the logic of prediction.

```
model1 <- lm(salary ~ career_PTS + position_rec + season_start +
             age, data = data_nba)

# base R way to get predicted values
data_nba$predicted_values <- model1$fitted.values
data_nba <- data_nba %>% dplyr::select('_id', name, salary, predicted_values, everythi

data_nba %>%
  ggplot(aes(x=salary, y=predicted_values)) +
  geom_point() +
  geom_smooth(method = "loess")
```

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




```
# new tidyverse way to get predicted values
library(broom)

# broom package has nice features to work with models

# tidy() converts the model output into a dataframe, makes it easy to process further, e.g. make
tidy(model1)
```

```
## # A tibble: 25 x 5
##    term                estimate std.error statistic  p.value
##    <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)         -568370.    29308.    -19.4  3.15e-82
##  2 career_PTS            53791.      731.     73.6  0
##  3 position_recForward -140645.    12779.    -11.0  5.24e-28
##  4 position_recGuard   -208234.    12166.    -17.1  9.89e-65
##  5 position_recMixed   -103661.    12104.     -8.56 1.26e-17
##  6 season_start1999      29357.    22884.      1.28 2.00e- 1
##  7 season_start2000      77209.    23559.      3.28 1.05e- 3
##  8 season_start2001     106436.    23625.      4.51 6.71e- 6
##  9 season_start2002     130348.    23627.      5.52 3.54e- 8
```

```
## 10 season_start2003     135221.    23648.      5.72 1.11e- 8
## # i 15 more rows
```

```
# glance provides meta-level info like r-squared etc.
glance(model1)
```

```
## # 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     0.432         0.431 352361.      308.       0    24 -138041. 2.76e5 2.76e5
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
# augment creates a dataframe with the predicted values for everyobservation in the da

nba_salalary_predicted <- augment(model1)
```

The output above, get you the predicted values for the observations in the dataset. This is mostly used to evaluate the model itself. The bigger the difference between the predicted values and the actual values (so-called residuals), the worse the model.

Now, let's predict the salary of hypothetical players:

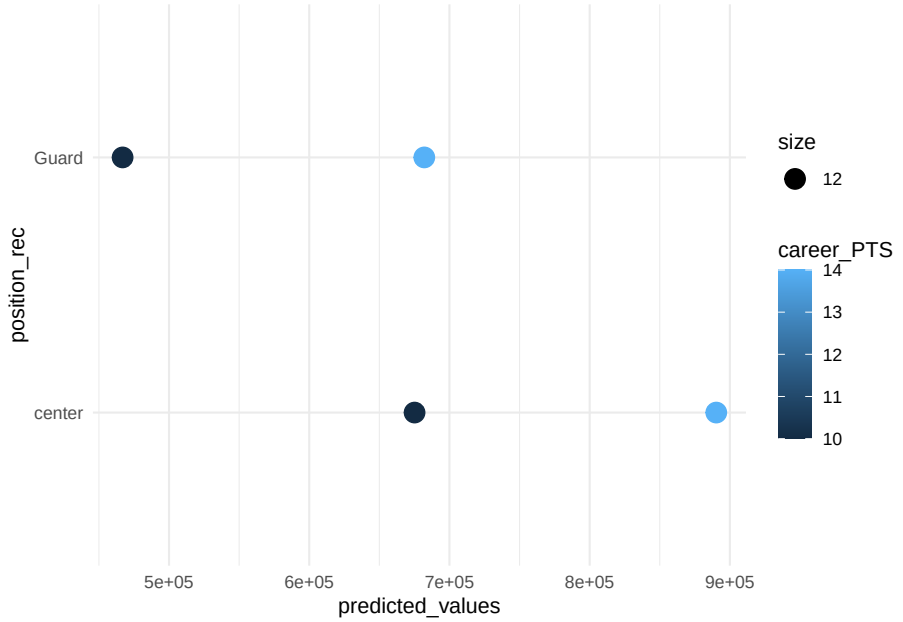
```
table(data_nba$position_rec)
```

```
##
##  center Forward   Guard   Mixed
##    1204    2130    3051    3343
```

```
# create all combination of the control variables which you want to predict
prediction.data <- tibble(
  position_rec = c("center", "center", "Guard", "Guard"),
  career_PTS = c(10, 14, 10, 14),
  age = c(20,20,20,20),
  season_start = c("2017", "2017", "2017", "2017")
)

# apply the model to the "new" dataset
predict(model1,
        prediction.data)
```

```
##        1        2        3        4
## 675129.3 890294.2 466895.1 682059.9
```

```
prediction.data$predicted_values <- predict(model1,
        prediction.data)

prediction.data %>% ggplot() +
  geom_point(aes(position_rec, predicted_values,
                 color=career_PTS,
                 size=12)) +
  coord_flip() +
  theme_minimal()
```



We can see that there is a huge salary increase predicted for just making two more baskets (i.e. 4 points) on average each game, for both guards and centers. We also see that centers make more money generally.

There is another way to get predicted values using the `margins()` function.

```
library(margins)
# This gets you the average change in predicted value for a unit-increase in the all model variab
margins(model1)
```

```
##  career_PTS   age position_recForward position_recGuard position_recMixed
##       53791 16809             -140645           -208234           -103661
##  season_start1999 season_start2000 season_start2001 season_start2002
##             29357            77209           106436           130348
```

```
##  season_start2003 season_start2004 season_start2005 season_start2006
##            135221           150576           174625           183653
##  season_start2007 season_start2008 season_start2009 season_start2010
##            206434           231505           204116           192220
##  season_start2011 season_start2012 season_start2013 season_start2014
##            183026           182509           199072           174359
##  season_start2015 season_start2016 season_start2017
##            206791           305228           369398
```

```
summary(margins(model1, at= list(position_rec= c("Guard", "center"),
                         career_PTS= c(10, 14))))
```

```
##              factor position_rec career_PTS         AME         SE       z
##                 age       1.0000    10.0000  16809.4290   822.1461 20.4458
##                 age       1.0000    14.0000  16809.4290   821.2659 20.4677
##                 age       2.0000    10.0000  16809.4290   821.3697 20.4651
##                 age       2.0000    14.0000  16809.4290   810.1705 20.7480
##          career_PTS       1.0000    10.0000  53791.2088   673.5739 79.8594
##          career_PTS       1.0000    14.0000  53791.2088   777.6596 69.1706
##          career_PTS       2.0000    10.0000  53791.2088   672.1461 80.0290
##          career_PTS       2.0000    14.0000  53791.2088   869.2152 61.8848
##  position_reccenter       1.0000    10.0000 208234.2670 12165.6828 17.1165
##  position_reccenter       1.0000    14.0000 208234.2670 12165.6828 17.1165
##  position_reccenter       2.0000    10.0000 208234.2670 12165.6828 17.1165
##  position_reccenter       2.0000    14.0000 208234.2670 12165.6828 17.1165
##    season_start1999       1.0000    10.0000  29356.8409 22884.2755  1.2828
##    season_start1999       1.0000    14.0000  29356.8409 22884.2755  1.2828
##    season_start1999       2.0000    10.0000  29356.8409 22884.2755  1.2828
##    season_start1999       2.0000    14.0000  29356.8409 22884.3468  1.2828
##    season_start2000       1.0000    10.0000  77209.1717 23559.4567  3.2772
##    season_start2000       1.0000    14.0000  77209.1717 23559.4567  3.2772
##    season_start2000       2.0000    10.0000  77209.1717 23559.4567  3.2772
##    season_start2000       2.0000    14.0000  77209.1717 23559.4567  3.2772
##    season_start2001       1.0000    10.0000 106435.5787 23624.8634  4.5052
##    season_start2001       1.0000    14.0000 106435.5787 23624.8634  4.5052
##    season_start2001       2.0000    10.0000 106435.5787 23624.8634  4.5052
##    season_start2001       2.0000    14.0000 106435.5787 23624.8621  4.5052
##    season_start2002       1.0000    10.0000 130348.2150 23627.2203  5.5169
##    season_start2002       1.0000    14.0000 130348.2150 23627.2203  5.5169
##    season_start2002       2.0000    10.0000 130348.2150 23627.2203  5.5169
##    season_start2002       2.0000    14.0000 130348.2150 23627.2203  5.5169
##    season_start2003       1.0000    10.0000 135220.5712 23648.3086  5.7180
##    season_start2003       1.0000    14.0000 135220.5712 23648.3086  5.7180
##    season_start2003       2.0000    10.0000 135220.5712 23648.3086  5.7180
##    season_start2003       2.0000    14.0000 135220.5712 23648.3047  5.7180
```

```
##    season_start2004       1.0000    10.0000 150575.6148 23423.7814  6.4283
##    season_start2004       1.0000    14.0000 150575.6148 23423.7814  6.4283
##    season_start2004       2.0000    10.0000 150575.6148 23423.7814  6.4283
##    season_start2004       2.0000    14.0000 150575.6148 23416.9662  6.4302
##    season_start2005       1.0000    10.0000 174625.3789 23247.7154  7.5115
##    season_start2005       1.0000    14.0000 174625.3789 23247.7154  7.5115
##    season_start2005       2.0000    10.0000 174625.3789 23247.7154  7.5115
##    season_start2005       2.0000    14.0000 174625.3789 23240.9439  7.5137
##    season_start2006       1.0000    10.0000 183653.2187 23057.8278  7.9649
##    season_start2006       1.0000    14.0000 183653.2187 23071.2667  7.9603
##    season_start2006       2.0000    10.0000 183653.2187 23071.2569  7.9603
##    season_start2006       2.0000    14.0000 183653.2187 23071.2471  7.9603
##    season_start2007       1.0000    10.0000 206434.1344 23336.8298  8.8459
##    season_start2007       1.0000    14.0000 206434.1344 23336.8298  8.8459
##    season_start2007       2.0000    10.0000 206434.1344 23336.8298  8.8459
##    season_start2007       2.0000    14.0000 206434.1344 23330.0326  8.8484
##    season_start2008       1.0000    10.0000 231505.0635 23507.2919  9.8482
##    season_start2008       1.0000    14.0000 231505.0635 23507.2919  9.8482
##    season_start2008       2.0000    10.0000 231505.0635 23507.2919  9.8482
##    season_start2008       2.0000    14.0000 231505.0635 23500.4440  9.8511
##    season_start2009       1.0000    10.0000 204115.6638 23475.4934  8.6948
##    season_start2009       1.0000    14.0000 204115.6638 23489.1731  8.6898
##    season_start2009       2.0000    10.0000 204115.6638 23489.1731  8.6898
##    season_start2009       2.0000    14.0000 204115.6638 23489.1586  8.6898
##    season_start2010       1.0000    10.0000 192219.8545 23548.5054  8.1627
##    season_start2010       1.0000    14.0000 192219.8545 23548.5054  8.1627
##    season_start2010       2.0000    10.0000 192219.8545 23548.5054  8.1627
##    season_start2010       2.0000    14.0000 192219.8545 23555.3511  8.1603
##    season_start2011       1.0000    10.0000 183025.8381 23534.7356  7.7768
##    season_start2011       1.0000    14.0000 183025.8381 23534.7356  7.7768
##    season_start2011       2.0000    10.0000 183025.8381 23534.7356  7.7768
##    season_start2011       2.0000    14.0000 183025.8381 23534.7356  7.7768
##    season_start2012       1.0000    10.0000 182509.4584 23219.2518  7.8603
##    season_start2012       1.0000    14.0000 182509.4584 23219.2518  7.8603
##    season_start2012       2.0000    10.0000 182509.4584 23219.2518  7.8603
##    season_start2012       2.0000    14.0000 182509.4584 23212.4961  7.8626
##    season_start2013       1.0000    10.0000 199072.4713 24314.5882  8.1874
##    season_start2013       1.0000    14.0000 199072.4713 24314.5882  8.1874
##    season_start2013       2.0000    10.0000 199072.4713 24314.5882  8.1874
##    season_start2013       2.0000    14.0000 199072.4713 24321.6667  8.1850
##    season_start2014       1.0000    10.0000 174358.9208 22746.2703  7.6654
##    season_start2014       1.0000    14.0000 174358.9208 22733.0380  7.6698
##    season_start2014       2.0000    10.0000 174358.9208 22733.0380  7.6698
##    season_start2014       2.0000    14.0000 174358.9208 22733.0380  7.6698
##    season_start2015       1.0000    10.0000 206791.4964 22900.6232  9.0300
##    season_start2015       1.0000    14.0000 206791.4964 22887.2931  9.0352
```

```
##    season_start2015       2.0000    10.0000 206791.4964 22887.2931  9.0352
##    season_start2015       2.0000    14.0000 206791.4964 22887.3012  9.0352
##    season_start2016       1.0000    10.0000 305227.6577 22617.9191 13.4949
##    season_start2016       1.0000    14.0000 305227.6577 22617.9191 13.4949
##    season_start2016       2.0000    10.0000 305227.6577 22617.9191 13.4949
##    season_start2016       2.0000    14.0000 305227.6577 22617.9191 13.4949
##    season_start2017       1.0000    10.0000 369398.3511 22665.9479 16.2975
##    season_start2017       1.0000    14.0000 369398.3511 22665.9479 16.2975
##    season_start2017       2.0000    10.0000 369398.3511 22665.9479 16.2975
##    season_start2017       2.0000    14.0000 369398.3511 22655.0335 16.3054
##       p       lower       upper
##  0.0000  15198.0523  18420.8057
##  0.0000  15199.7774  18419.0806
##  0.0000  15199.5740  18419.2841
##  0.0000  15221.5240  18397.3341
##  0.0000  52471.0281  55111.3895
##  0.0000  52267.0240  55315.3936
##  0.0000  52473.8267  55108.5909
##  0.0000  52087.5784  55494.8392
##  0.0000 184389.9668 232078.5671
##  0.0000 184389.9668 232078.5671
##  0.0000 184389.9668 232078.5671
##  0.0000 184389.9668 232078.5671
##  0.1995 -15495.5149  74209.1967
##  0.1995 -15495.5149  74209.1967
##  0.1995 -15495.5149  74209.1967
##  0.1995 -15495.6546  74209.3364
##  0.0010  31033.4850 123384.8584
##  0.0010  31033.4850 123384.8584
##  0.0010  31033.4850 123384.8584
##  0.0010  31033.4850 123384.8584
##  0.0000  60131.6973 152739.4600
##  0.0000  60131.6973 152739.4600
##  0.0000  60131.6973 152739.4600
##  0.0000  60131.6998 152739.4575
##  0.0000  84039.7141 176656.7158
##  0.0000  84039.7141 176656.7158
##  0.0000  84039.7141 176656.7158
##  0.0000  84039.7141 176656.7158
##  0.0000  88870.7382 181570.4043
##  0.0000  88870.7382 181570.4043
##  0.0000  88870.7382 181570.4043
##  0.0000  88870.7457 181570.3968
##  0.0000 104665.8469 196485.3827
##  0.0000 104665.8469 196485.3827
##  0.0000 104665.8469 196485.3827
```

```
##  0.0000 104679.2044 196472.0252
##  0.0000 129060.6941 220190.0637
##  0.0000 129060.6941 220190.0637
##  0.0000 129060.6941 220190.0637
##  0.0000 129073.9659 220176.7919
##  0.0000 138460.7067 228845.7307
##  0.0000 138434.3670 228872.0704
##  0.0000 138434.3861 228872.0513
##  0.0000 138434.4052 228872.0322
##  0.0000 160694.7885 252173.4804
##  0.0000 160694.7885 252173.4804
##  0.0000 160694.7885 252173.4804
##  0.0000 160708.1107 252160.1581
##  0.0000 185431.6180 277578.5090
##  0.0000 185431.6180 277578.5090
##  0.0000 185431.6180 277578.5090
##  0.0000 185445.0396 277565.0874
##  0.0000 158104.5422 250126.7855
##  0.0000 158077.7306 250153.5971
##  0.0000 158077.7306 250153.5971
##  0.0000 158077.7590 250153.5687
##  0.0000 146065.6320 238374.0769
##  0.0000 146065.6320 238374.0769
##  0.0000 146065.6320 238374.0769
##  0.0000 146052.2146 238387.4943
##  0.0000 136898.6039 229153.0723
##  0.0000 136898.6039 229153.0723
##  0.0000 136898.6039 229153.0723
##  0.0000 136898.6039 229153.0723
##  0.0000 137000.5611 228018.3557
##  0.0000 137000.5611 228018.3557
##  0.0000 137000.5611 228018.3557
##  0.0000 137013.8020 228005.1148
##  0.0000 151416.7540 246728.1885
##  0.0000 151416.7540 246728.1885
##  0.0000 151416.7540 246728.1885
##  0.0000 151402.8805 246742.0621
##  0.0000 129777.0502 218940.7913
##  0.0000 129802.9850 218914.8566
##  0.0000 129802.9850 218914.8566
##  0.0000 129802.9850 218914.8566
##  0.0000 161907.0996 251675.8931
##  0.0000 161933.2262 251649.7666
##  0.0000 161933.2262 251649.7666
##  0.0000 161933.2103 251649.7824
##  0.0000 260897.3510 349557.9645
```

```
##  0.0000 260897.3510 349557.9645
##  0.0000 260897.3510 349557.9645
##  0.0000 260897.3510 349557.9645
##  0.0000 324973.9095 413822.7927
##  0.0000 324973.9095 413822.7927
##  0.0000 324973.9095 413822.7927
##  0.0000 324995.3013 413801.4009
```

The margins function is handy to calculate predictions for different groups. It automatically can hold other control variables at their mean or at their observed value.

Now, let's also calculate confidence and prediction intervals. For background, watch these short videos to understand what they are [hyperlink] [hyperlink]. Confidence intervals basically tell how the following: "If we repeated our study on a different sample of people with the same sample size, then the estimate which we have (for example, a mean) would we within the confidence interval 95% of the time. This means in 5% of cases, our study would arrive at a lower or higher mean. The formula for confidence is not very intuitive:

-> formulate here: Margin of error = z * (standard deviation/ square-root of sample size)

Let's not worry about why this formula works, but let's focus on its ingredients: Sample size (N) is the number of people in our data; Standard Deviation is a measure for how much individual people deviate from the mean on average, in other words, how much the data spreads around the mean. and z is 1.96 and is derived from probability theory (i.e. in a normal distribution, there is a certain known likelihood that means fall within a range when re-sampling populations). In other words, the confidence intervals tells us how "confident" we can be that our estimate is within the range 95% of times.

Prediction intervals are very similar but only apply to predictions for specfific values. It gives us a measure for "confident" we can be that our prediction would be within the prediction interval (95% of times).

The 95% is an arbitrarily set value which is a standard in research. However, we can also set it at 99% or 90%.

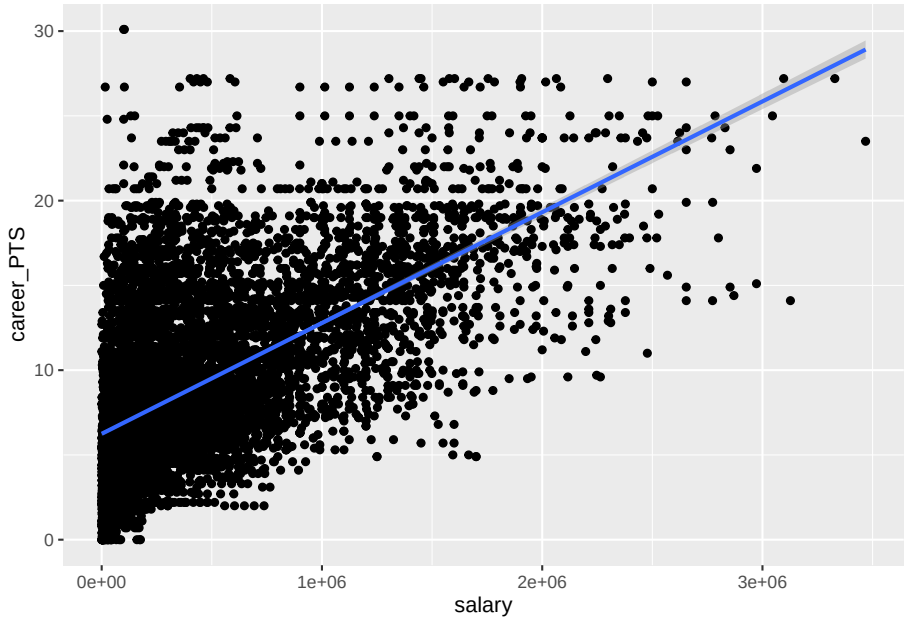
Let's apply this to our data.

```
# Just for illustration, let's take a simple model
model2 <- lm(salary ~ career_PTS, data = data_nba)
preds <- predict(model2)
preds[1:10]
```

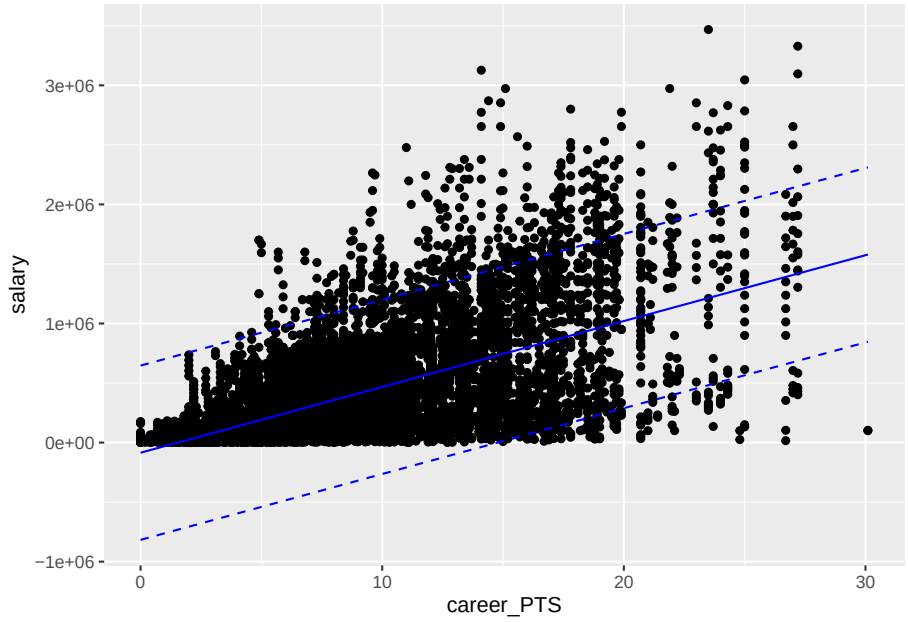
```
##        1        2        3        4        5        6        7        8
## 721959.3 346026.1 346026.1 346026.1 346026.1 346026.1 346026.1 346026.1
```

```
##         9        10
## 346026.1 346026.1
```

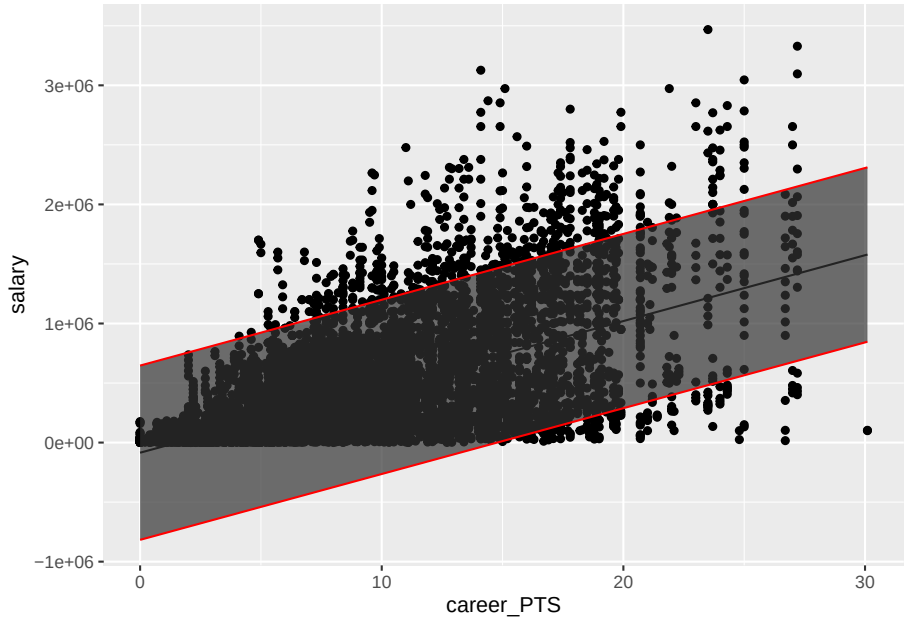
```
# using geom_smooth, method=lm will automatically plot the confidence intervals
data_nba %>%
  ggplot(aes(x=salary, y=career_PTS)) +
  geom_point() +
  geom_smooth(method = "lm")
```




```
# let's get prediction intervals and add them to our dataset
data_nba_predict <- cbind(data_nba, predict(model2, interval = c("prediction")))

data_nba_predict %>%
  ggplot(aes(x= career_PTS, y=salary)) +
  geom_point() +
  geom_line(aes(x=career_PTS, y=fit),
            col="blue") +
  geom_line(aes(y=lwr),
                col="blue",
                linetype="dashed") +
  geom_line(aes(y=upr),
                col="blue",
                linetype="dashed")
```



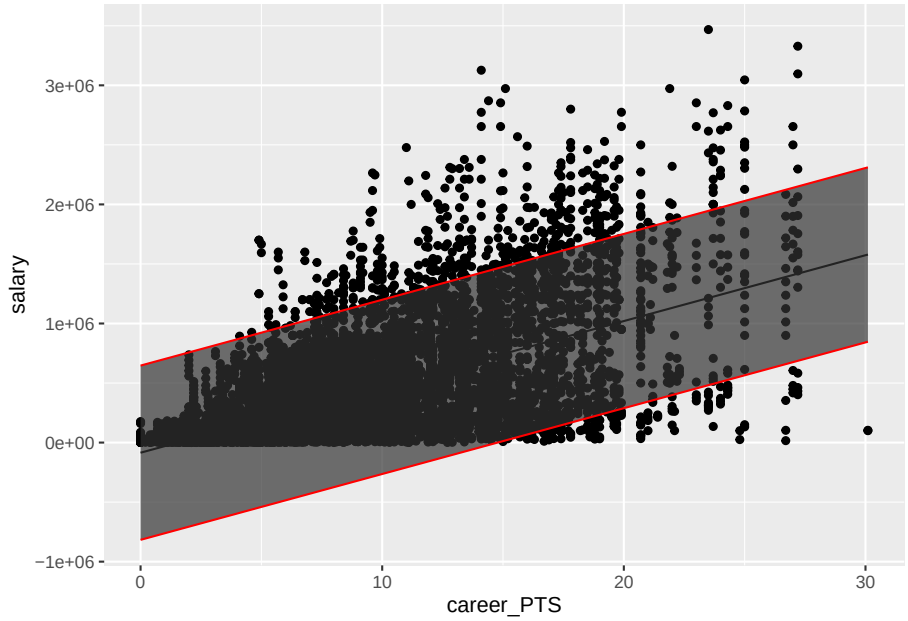

```
# same using geom_ribbon
data_nba_predict %>%
ggplot(aes(x=career_PTS,y=salary))+
  geom_point()+
  geom_line(aes(x=career_PTS,y=fit))+
  geom_ribbon(aes(ymax=upr,ymin=lwr),color="red",alpha=0.7)
```




```
## do this with tidy approach

# get confidnce interval for estimate
tidy(model2, conf.int = TRUE)
```

```
## # A tibble: 2 x 7
##   term        estimate std.error statistic  p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)  -85191.     7642.     -11.1 1.09e-28 -100171.   -70212.
## 2 career_PTS    55284.      745.      74.2 0          53823.    56745.
```

```
# get predictive intervals
augment(model2, interval= "prediction") %>%
  ggplot(aes(x=career_PTS,y=salary))+
  geom_point()+
  geom_line(aes(x=career_PTS,y=.fitted))+
  geom_ribbon(aes(ymax=.upper,ymin=.lower),color="red",alpha=0.7)
```



Now, let's compare the prediction interval for 2 differnt models.To compare the performance of models, you can use the r-squared (which measures how much of the variation in the outcome can be explained by your set of independent variables) and the mean squared error. For more background on both measures see here: https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/

The Mean squared error (MSE) represents the error of the estimator or predictive model created based on the given set of observations in the sample. It measures the average squared difference between the predicted values and the actual values, quantifying the discrepancy between the model's predictions and the true observations. The lower the MSE, the better the model predictive accuracy, and, the better the regression model is.

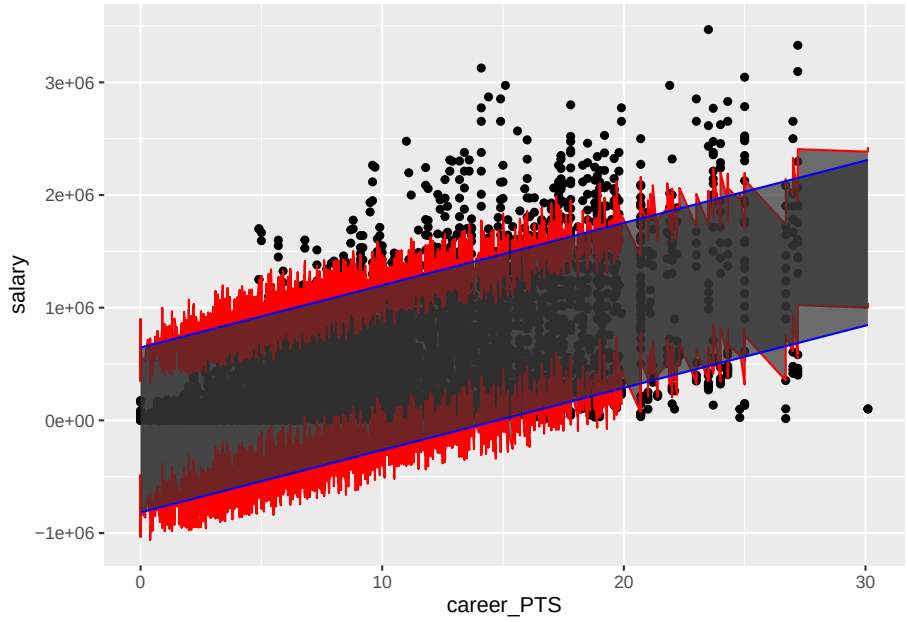
```
model1 <- lm(salary ~ career_PTS + position_rec + season_start +
             age, data = data_nba)
model2 <- lm(salary ~ career_PTS, data = data_nba)

# compared R-squared/ adjusted R-squared
glance(model1)
```

```
## # 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     0.432         0.431 352361.      308.       0    24 -138041. 2.76e5 2.76e5
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
glance(model2)
```

```
## # 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     0.361         0.361 373217.     5502.       0     1 -138612. 2.77e5 2.77e5
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
# compare prediction intervals
model1_predicted <- augment(model1, interval = "prediction")
model2_predicted <- augment(model2, interval = "prediction")

  ggplot(aes(x=career_PTS,y=salary), data = model1_predicted)+
  geom_point()+
  # geom_line(aes(x=career_pts,y=.fitted),
  #           color ="red", data = model1_predicted) +
  # geom_line(aes(x=career_pts,y=.fitted),
  #          color ="blue", data = model2_predicted) +
  geom_ribbon(aes(ymax=.upper,ymin=.lower),
              color="red",
              alpha=0.7,
              data = model1_predicted) +
  geom_ribbon(aes(ymax=.upper,ymin=.lower),
              color="blue",
              alpha=0.7,
              data = model2_predicted)
```




```
# see if there is a better way to do this

library(Metrics) # using rmse from Metrics library
# compare mean squared error
  rmse(model1_predicted$.fitted, data_nba$salary)
```

```
## [1] 351908.3
```

```
  rmse(model2_predicted$.fitted, data_nba$salary)
```

```
## [1] 373178.4
```

Looking at r2, the prediction intervals and rmse, model 1 clearly performs better than model 2.

11.2 Intro to Machine learning

We have now arrived at the entry gates to machine learning. We will conduct a very basic and simple machine learning routine using linear regression. The main difference between simple prediction and Machine Learning is that the sample is first divided into a training and a test dataset at random. The model

is than tuned based on, let's say, 80% of the sample. When the model is ready, it is tested based on the 20% remaining sample. The prediction produced with the model are then compared with the actual values in the test dataset. There is of course more nuance to all this, but this is basically the idea.

Let's use the "caret" package, a common package for machine learning.

```
library(caret)

# Create a train and test split
set.seed(123)  # For reproducibility
train_indices <- createDataPartition(data_nba$salary, p = 0.7, list = FALSE)
train_data <- data_nba[train_indices, ]
test_data <- data_nba[-train_indices, ]

# Create a train control object
ctrl <- trainControl(method = "none")

# Train a linear regression model using caret
model1 <- train(
  salary ~ career_PTS + position_rec + season_start + age,
  data = train_data,
  method = "lm",
  trControl = ctrl
)

model2 <- train(
  salary ~ career_PTS,
  data = train_data,
  method = "lm",
  trControl = ctrl
)

# Make predictions on the test set
predicted_salaries_m1 <- predict(model1, newdata = test_data)
predicted_salaries_m2 <- predict(model2, newdata = test_data)


# Calculate prediction errors (e.g., root mean squared error)
rmse1 <- sqrt(mean((predicted_salaries_m1 - test_data$salary)^2))
rmse2 <- sqrt(mean((predicted_salaries_m2 - test_data$salary)^2))

# Print the prediction errors
print(rmse1)
```

```
## [1] 341746.4
```

```
print(rmse2)
```

```
## [1] 360297.3
```

Now machine learning usually involves several more steps: + we can optimize how the variables (in ML language called features) enter the model (Preprocessing; transformations; diagnostics, see week X) + we can optimize which variables should even enter the model ("feature selection", see e.g. lasso regression) + we can optimize how the predictions of the model get evaluated ("training") + we can optimize which estimator or algorithm best predicts the outcome (lm model is just one option among many) + for other algorithms (e.g. random forests), we can also "tune" the model using hyper-parameters

11.3 More resources:

- Provide list of online resources to dig deeper
- see `tidymodels()` for another package for Machine Learning

Chapter 12

Prediction - Application

Here goes some texts.

12.1 Exercises

1. **Variable selection**: Use variable selection methods, such as stepwise selection or best subset selection, to identify a subset of predictor variables that provide the best fit for a multiple linear regression model. Compare the performance of different models and discuss their relative strengths and weaknesses.
2. **Interaction effects**: Include interaction terms in a multiple linear regression model to capture the effect of two or more predictor variables interacting with each other. Interpret the coefficients of the interaction terms and discuss their implications from a social science perspective.
3. **Polynomial regression**: Use polynomial regression to model non-linear relationships between predictor variables and the response variable. Fit a polynomial regression model, generate predictions, and assess the model's performance.
4. **Model comparison**: Use model comparison methods, such as adjusted R-squared or AIC, to compare the performance of different linear regression models. Select the best model based on these criteria and discuss its strengths and weaknesses.
5. **Tidy models**: Use the `broom` package to convert linear regression models into tidy data frames using functions such as `tidy()`, `glance()`, and `augment()`. Manipulate and visualize the tidy data frames to gain insights into the models and their performance.
6. **Prediction intervals**: Generate prediction intervals for new observations using a fitted linear regression model. Interpret the prediction intervals

and discuss their usefulness for making predictions in a social science context.

Chapter 13

Outlook

13.1 Summary of what was covered in course

13.2 Other outcome variables

- generalized linear models
- logistic
- poisson

13.3 Data structures

- multilevel

13.4 Where to go next

- econometrics
- machine learning
- geo-data
- web-scraping