Problem Set 9 Solution

Andreas Bender, Philipp Kopper, Sven Lorenz

25 January 2022

Read chapter 13 on relational data in R4DS. You can skip the exercises.

Exercise 1

In this exercise we investigate data sets on baseball. In particular, we are interested in creating player statistics. You will combine different data sets to gather all the information that you are looking for. Save your resulting objects with the same naming convention as last week. That means your solution for exercise 1a) should be saved as `ex1a.Rds` in the folder `solutions`. Use the `all.equal` function to check your solutions independently.

```r
dir.create("solutions")
```

```
## Warning in dir.create("solutions"): 'solutions' already exists
```

a) Load the `Lahman` package and have a look at the `Pitching` dataset. Use the `?` function to get an overview on the following data sets: `Pitching`, `People`. Filter the `Pitching` data so that it only contains the data from the years 2018 and 2019. What is the primary key for this data set? Work with this filtered data set throughout the problem set.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(Lahman)
library(ggplot2)
theme_set(theme_bw())

Pitching <- Lahman::Pitching

Pitching <- Pitching %>%
  filter(yearID %in% c(2018, 2019))

Pitching %>%
  count(playerID, yearID, teamID, stint) %>%
  filter(n > 1)
```

```
## [1] playerID yearID   teamID   stint    n
## <0 rows> (or 0-length row.names)
```

```
# primary key playerID, yearID, teamID, stint
saveRDS(Pitching, "solutions/ex1a.Rds")
```

b) Group the data by player, year and team. Count the number of wins, losses, games, and home runs for each player, year and team. Store the resulting data in an object called `pitching`.

```
pitching <- Pitching %>%
  group_by(playerID, yearID, teamID) %>%
  summarise(
    wins     = sum(W),
    losses   = sum(L),
    games    = sum(G),
    homeruns = sum(HR))
```

```
## `summarise()` has grouped output by 'playerID', 'yearID'. You can override using
## the `.groups` argument.
```

```
saveRDS(pitching, "solutions/ex1b.Rds")
```

c) What is the primary key of the `pitching` data set?

```
pitching %>%
  count(playerID, yearID, teamID) %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 4
## # Groups:   playerID, yearID [0]
## # ... with 4 variables: playerID <chr>, yearID <int>, teamID <fct>, n <int>
```

```
  # shortest primary key

pitching %>%
  count(playerID, yearID) %>%
  filter(n > 1)
```

```
## # A tibble: 179 x 3
## # Groups:   playerID, yearID [179]
##    playerID  yearID     n
##    <chr>      <int> <int>
##  1 adamsau01   2019     2
##  2 adamsau02   2019     2
##  3 alanirj01   2019     2
##  4 allenlo01   2019     2
##  5 anderni01   2019     2
##  6 andrima01   2018     2
##  7 archech01   2018     2
##  8 armstsh01   2019     2
##  9 avilalu01   2018     2
## 10 axforjo01   2018     2
## # ... with 169 more rows
```

```
# each player has a maximum of 2 entries / rows for the two years
# some only have 1 entry / row for one of the two years
```

d) Have a look at the `People` data set from the `Lahman` package. What is the primary key here?

```r
people <- Lahman::People

people %>%
  count(playerID) %>%
  filter(n > 1)
```

```
## [1] playerID n
## <0 rows> (or 0-length row.names)
```

```r
# primary key: playerID
```

e) For each player in your `pitching` data set, we now want to have information on their birthplace. Join the appropriate data to your pitching table. Keep all rows from the pitching data and all columns from both tables.

```r
pitching_people <- pitching %>%
  left_join(people, by = "playerID")
saveRDS(pitching_people, "solutions/ex1e.Rds")
```

f) We now want to summarize the pitching information for each player regardless of year and team. Group the `pitching` data by `playerID` and add up the information about wins, losses, games and home runs.

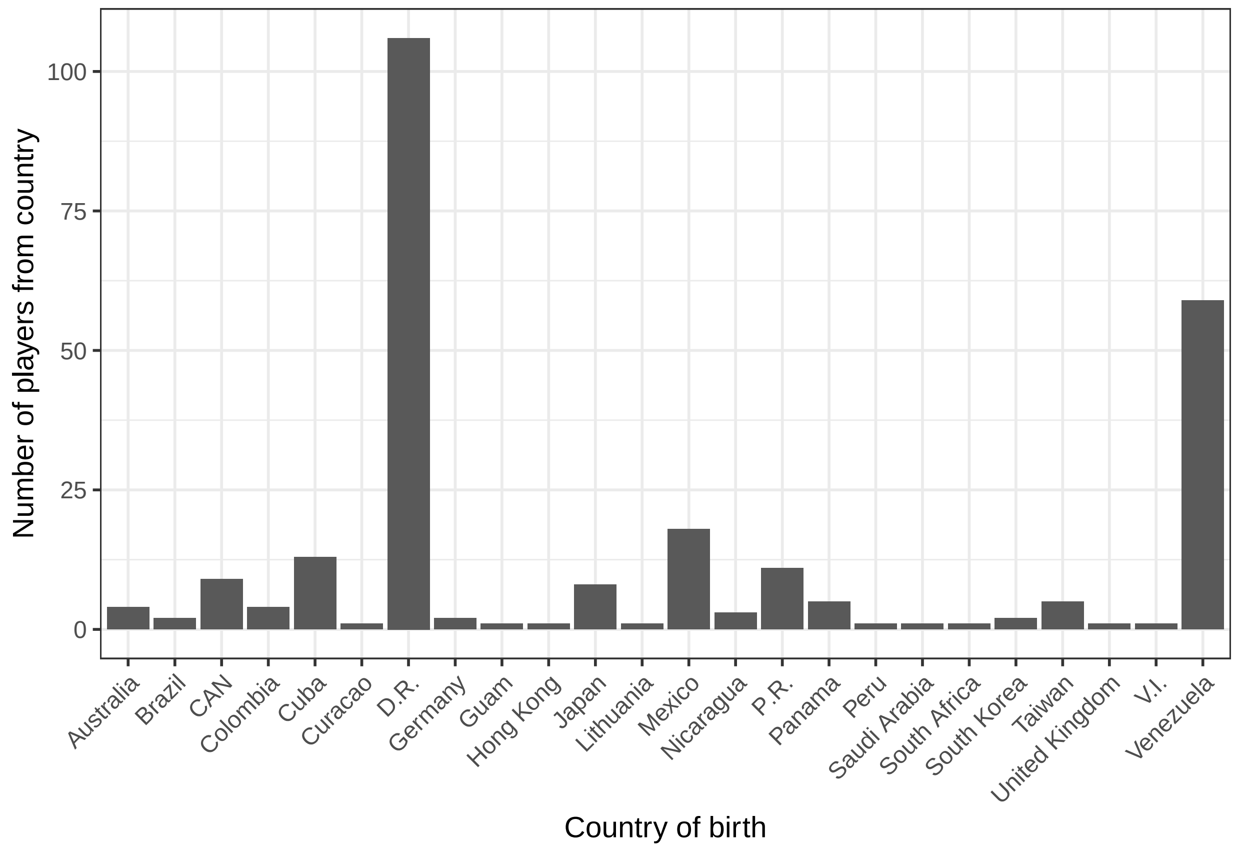
```r
pitching_sum <- pitching %>%
  group_by(playerID) %>%
  summarise(
    wins      = sum(wins),
    losses    = sum(losses),
    games     = sum(games),
    homeruns  = sum(homeruns)
  )
saveRDS(pitching_sum, "solutions/ex1f.Rds")
```

g) Now join this summarised pitching data set with the people data set as in task e). Based on this data set, create a data frame that for each country contains the information about the number of players from the respective country (excluding the USA). Sort the data such that the country with most players is at the top and the country with the least players is at the bottom. Use `ggplot` to create a bar plot that shows the number of players per country (Hint: use `geom_col`). Make sure that the country names in the axis labels don't overlap. **Bonus**: Recreate the above plot such that the bars are ordered from highest to lowest.

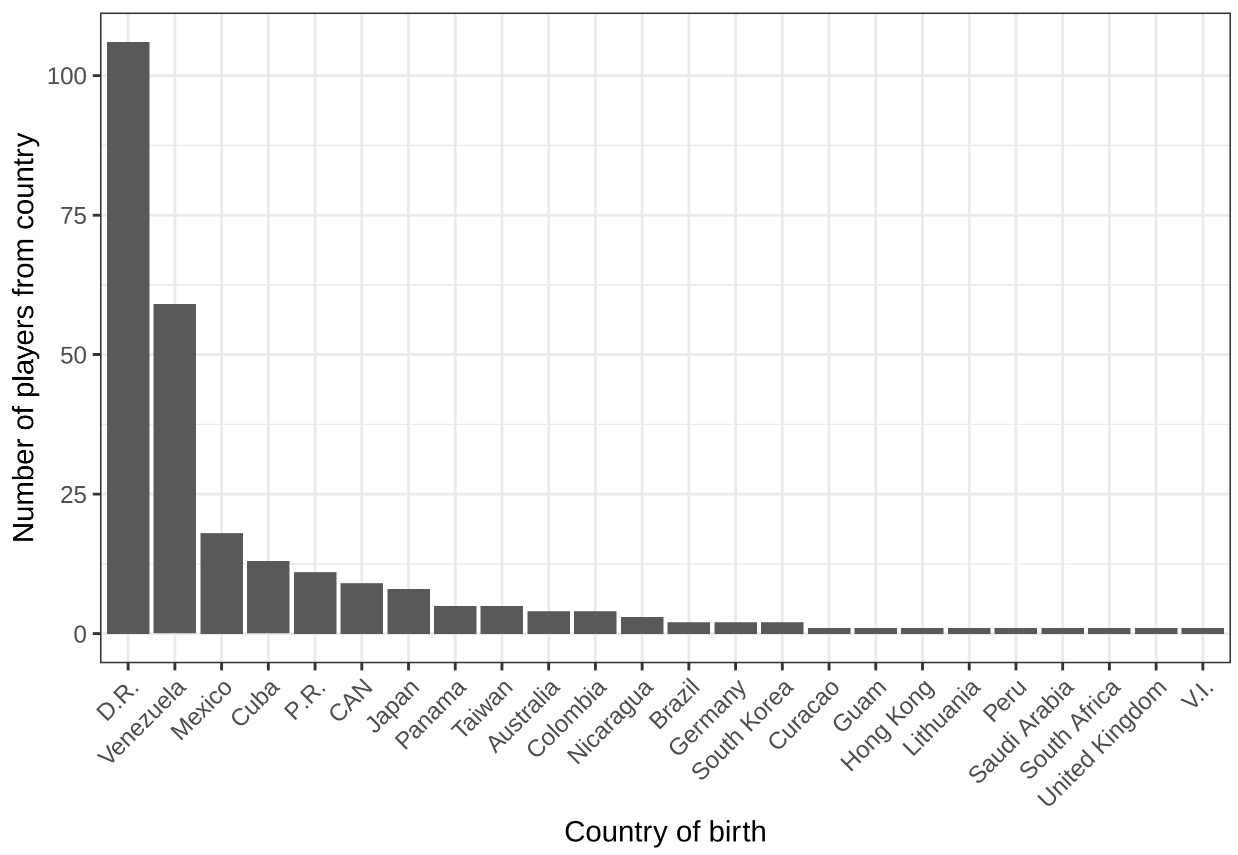
```r
pitching_people2 <- pitching_sum %>%
  left_join(people, by = "playerID") %>%
  select(-starts_with("death"))

not_usa <- pitching_people2 %>%
  filter(birthCountry != "USA") %>%
  group_by(birthCountry) %>%
  summarize(n = n()) %>%
  arrange(desc(n))
not_usa %>% slice(1:3)
```

```
## # A tibble: 3 x 2
##   birthCountry     n
##   <chr>        <int>
## 1 D.R.           106
## 2 Venezuela       59
## 3 Mexico          18
```

```r
# the three countries are D.R., Venezuela and Mexico

p1 <- ggplot(data = not_usa) +
  geom_col(aes(x = birthCountry, y = n)) +
  xlab("Country of birth") +
  ylab("Number of players from country") +
  coord_flip()

# alternative axis
p2 <- ggplot(data = not_usa) +
  geom_col(aes(x = birthCountry, y = n)) +
  xlab("Country of birth") +
  ylab("Number of players from country") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
p2
```



```r
# Bonus
p3 <- ggplot(data = not_usa) +
  geom_col(aes(x = reorder(birthCountry, -n), y = n)) +
  xlab("Country of birth") +
  ylab("Number of players from country") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
p3
```



```r
saveRDS(not_usa, "solutions/ex1g.Rds")
ggsave("solutions/ex1g.png", p3)
```

```
## Saving 6.5 x 4.5 in image
```

Exercise 2

We collected data on the coffee drinking preferences in the office. The people working from the office buy a variation of beans of which everyone can take some. We stored these information in the object `people`. The `data.frame` consists of the Name of the consumer, the information how many cups they drink, where the coffee is from (which is in our scenario identical to its name) and what kind of coffee drink they prefer. Note that there are two special cases: Julia drinks her own coffee of which we do not know a lot and Martin does not drink coffee at all. `coffee` features information on the beans that the people drink, precisely on the roast (light to dark) and its price (per kilogram). Next to these, we have a `data.frame` that explains how many shots of espresso we need for each of the drinks (`drinks`). Additionally, it tells how much milk (in ml) one needs for the drinks. Note that for one espresso shot you typically need 18g of coffee. A litre of milk costs 1 euro.

```r
people <- data.frame(
  name = c("Andreas", "Philipp", "Sven", "Martin",
           "Moritz", "Julia", "Jana", "Klaus",
           "Maria", "Leonie", "Frederick", "Ute"),
  n_cups = c(3, 5, 1, NA,
             3, 4, 1, 1,
             6, 5, 2, 1),
  origin = c("Ethiopia", "Honduras", "Costa Rica", NA,
             "Vietnam", "Brazil", "Honduras", "Honduras",
             "Costa Rica", "Vietnam", "Colombia", "Costa Rica"),
  drink = c("Cappucino", "Flat White", "Cappucino", NA,
            "Espresso", "Latte", "Cappucino", "Cappucino",
            "Espresso doppio", "Flat White", "Cappucino", "Espresso doppio"))

coffee <- data.frame(
  origin = c("Ethiopia", "Honduras", "Costa Rica", "Vietnam", "Colombia"),
  roast = c("decaf", "medium", "dark", "light", "medium"),
  price = c(18.95, 26.15, 17.99, 29.00, 21.40))

drinks <- data.frame(
  drink = c("Espresso", "Espresso doppio", "Cappucino", "Cafè Crema",
            "Latte", "Flat White",  "Americano"),
  shots = c(1, 2, 1, 1, 1, 2, 1),
  milk = c(0, 0, 100, 0, 150, 100, 0)
)
```

a) Compute how many shots of each coffee are pulled on a regular office day. How much do the collegues spend per day on coffee beans? How much for milk?

```
coffee %>%
  inner_join(people) %>%
  inner_join(drinks) %>%
  mutate(total_shots = shots * n_cups, total_milk = n_cups * milk) %>%
  mutate(money_spent_coffee = price * 0.018 * total_shots,
         money_spent_milk = (total_milk / 1000)) %>%
  summarise(spendings_coffee = sum(money_spent_coffee),
            spendings_milk = sum(money_spent_milk))
```

```
## Joining, by = "origin"
## Joining, by = "drink"
```

```
##   spendings_coffee spendings_milk
## 1          19.0854            1.8
```

b) Compute how many kilograms of each roast of coffee are consumed over the course of the week.

```
coffee %>%
  inner_join(people) %>%
  inner_join(drinks) %>%
  mutate(total_shots = shots * n_cups) %>%
  mutate(total_coffee = 0.018 * total_shots * 5) %>%
  group_by(roast) %>%
  summarise(total_coffee = sum(total_coffee))
```

```
## Joining, by = "origin"
## Joining, by = "drink"
```

```
## # A tibble: 4 x 2
##   roast  total_coffee
##   <chr>         <dbl>
## 1 dark           1.35
## 2 decaf          0.27
## 3 light          1.17
## 4 medium         1.26
```

c) Identify the individual costs of coffee consumption per day. Note that Julia pays for her own coffee, so she has an individual contribution of 0 Euros. As most people have the feeling that the distribution of coffee consumption is extremely uneven, they want to express this unequality by a number, namely the Gini coefficient. Use the `ineq` package to compute the Gini coefficient (`ineq` function) of this distribution. Interpret the value. Also draw the Lorenz curve. You find a comprehensive overview on the Gini coefficient and the Lorenz curve here, if you did not attend Statistics 1 this semester or as a refresher. Note that the package does most of the work for you in computing the Gini cofficient and drawing the curve. Focus on the preprocessing.

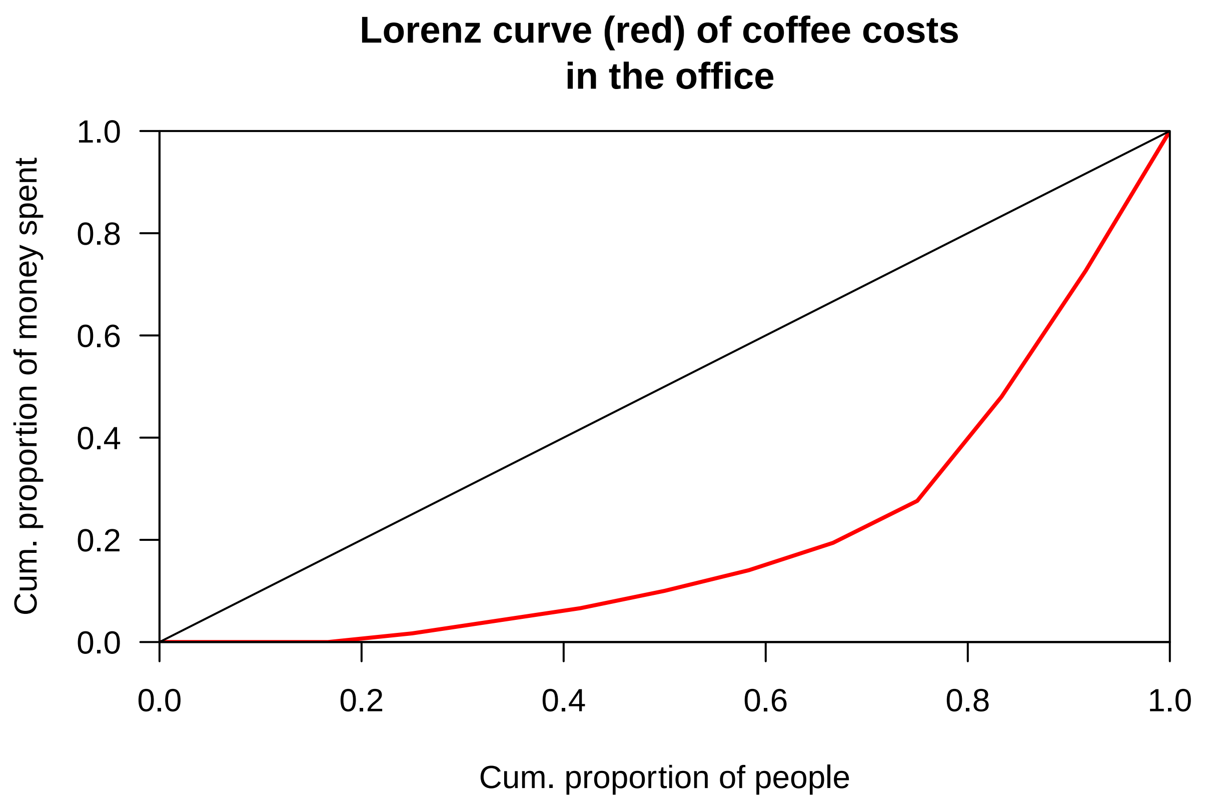
```r
library(ineq)
coffee_costs <-
  people %>%
  full_join(coffee) %>%
  left_join(drinks) %>%
  mutate(total_shots = shots * n_cups) %>%
  mutate(money_spent = price * 0.018 * total_shots) %>%
  select(name, money_spent) %>%
  mutate(money_spent = ifelse(name %in% c("Martin", "Julia"), 0,
                              money_spent)) %>%
  arrange(-money_spent)
```

```
## Joining, by = "origin"
## Joining, by = "drink"
```

```r
ineq(coffee_costs$money_spent, type = "Gini")
```

```
## [1] 0.5762441
```

```r
plot(Lc(coffee_costs$money_spent), col = "red",
     main = "Lorenz curve (red) of coffee costs \n in the office",
     ylab = "Cum. proportion of money spent",
     xlab = "Cum. proportion of people")
```



Session Info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ineq_0.2-13   ggplot2_3.3.5 Lahman_9.0-0  dplyr_1.0.7
```

```
##
## loaded via a namespace (and not attached):
##  [1] highr_0.9         pillar_1.6.4      compiler_4.1.2    tools_4.1.2
##  [5] digest_0.6.29     evaluate_0.14     lifecycle_1.0.1   tibble_3.1.6
##  [9] gtable_0.3.0      pkgconfig_2.0.3   rlang_0.99.0.9003 cli_3.1.0.9000
## [13] DBI_1.1.1         yaml_2.2.1        xfun_0.29         fastmap_1.1.0
## [17] withr_2.4.3       stringr_1.4.0     knitr_1.37        systemfonts_1.0.3
## [21] generics_0.1.1    vctrs_0.3.8       grid_4.1.2        tidyselect_1.1.1
## [25] glue_1.6.0        R6_2.5.1          textshaping_0.3.6 fansi_0.5.0
## [29] rmarkdown_2.11    farver_2.1.0      purrr_0.3.4       magrittr_2.0.1
## [33] scales_1.1.1      ellipsis_0.3.2    htmltools_0.5.2   assertthat_0.2.1
## [37] colorspace_2.0-2  ragg_1.2.1        labeling_0.4.2    utf8_1.2.2
## [41] stringi_1.7.6     munsell_0.5.0     crayon_1.4.2
```