

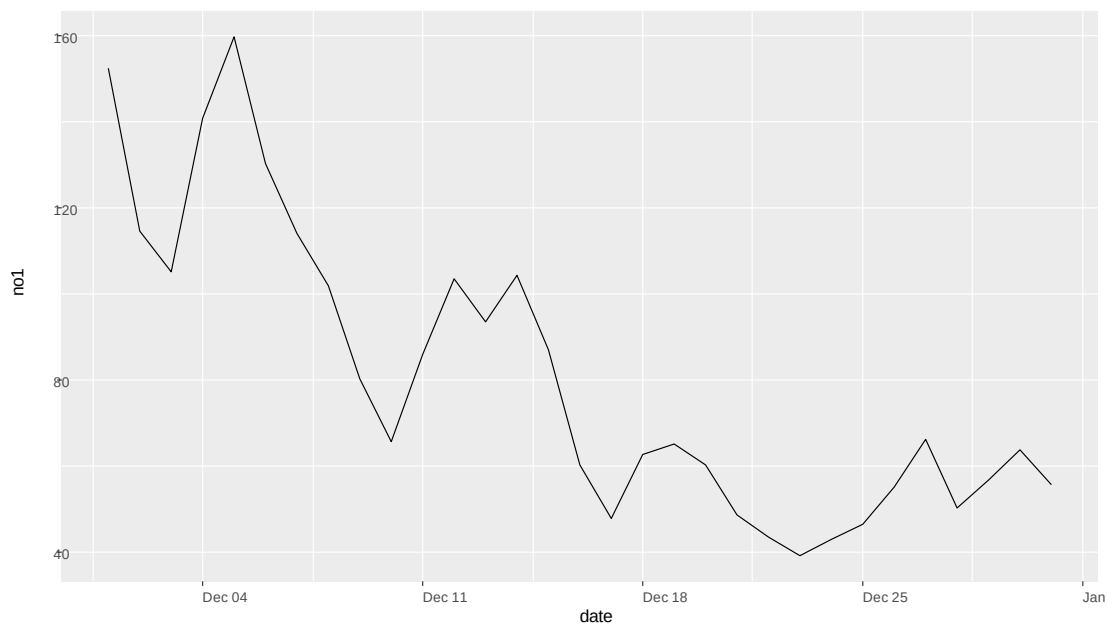
I. Electricity prices (weight 60 %)

The Excel file [ElPrices.xlsx](#) contains daily day-ahead electricity prices for 2023 provided by NordPool. The first column is the date; the remaining columns are geographical areas sharing a common electricity price. In this exercise we concentrate on the five Norwegian electricity regions, indicated on the map to the right. The file has twelve sheets, one for each month of the year.

a) Import the electricity prices for December. Clean the data set so you only have the date and the Norwegian electricity prices (regions “no1” to “no5”). Construct a time series plot of the electricity price for region “no1”.

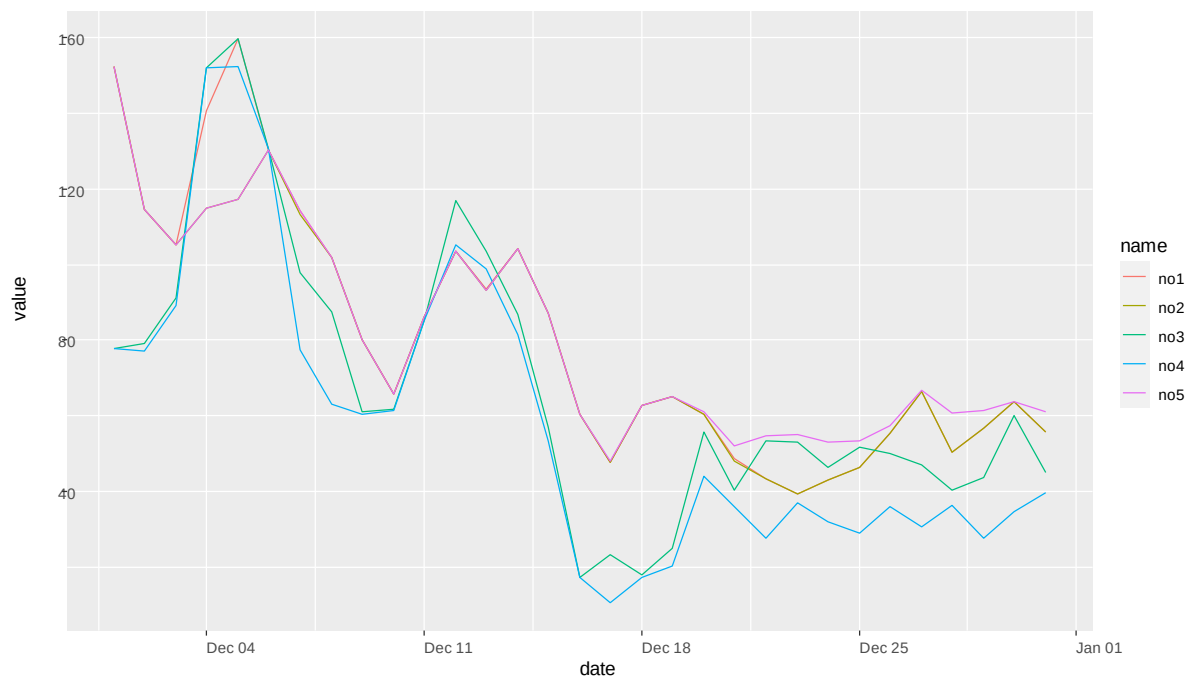
```
library(tidyverse)
library(readxl)
setwd("...")
el_december <- read_excel("ElPrices.xlsx") |>
  select(date, starts_with('no'))
```

```
el_december |>
  ggplot(aes(x=date, y=no1)) +
  geom_line()
```



b) Explain the difference between long and short data. Transform the data into long format and construct a time series plot where you show the prices for all five regions in the same plot.

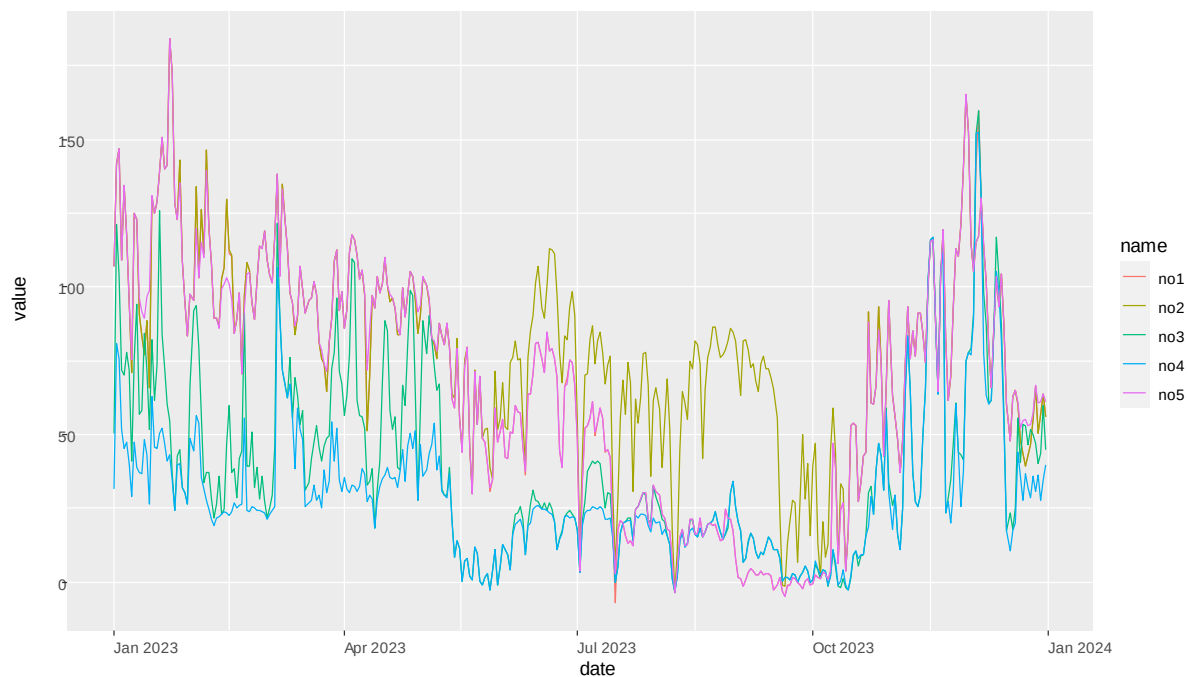
```
el_december |>
  pivot_longer(cols = starts_with('no')) |>
  ggplot(aes(x=date, y=value, color=name)) +
  geom_line()
```



c) Construct a loop to read and combine the data from all the twelve monthly sheets into a common data set. Show the daily data for each region for all of 2023 in a time series plot.

```
elprices <- NULL
for (m in 1:12) {
  print(month.name[m])
  mprices <- read_excel("ElPrices.xlsx", sheet = month.name[m])
  elprices <- bind_rows(elprices, mprices)
}

elprices |>
  pivot_longer(cols = starts_with('no')) |>
  ggplot(aes(x=date, y=value, color=name)) +
  geom_line()
```



There are several ways to make the loop. Most efficient is using `apply` or `purr`, but for 12 iterations this is not important. Manually importing each sheet gives some credit but not full score.

d) Consider region “no1”. Compute the average price by month and day of the week (Monday etc.). Present the results in a table with months as rows, weekdays as columns.

```
elprices |>
  select(date, no1) |>
  mutate(day = wday(date, label = T, abbr = F, week_start = 1),
         month = month(date, label = T, abbr = F)) |>
  group_by(month, day) |>
  summarise(elprice = mean(no1)) |>
  pivot_wider(names_from = day,
              values_from = elprice)
```

A tibble: 12 x 8

Groups: month [12]

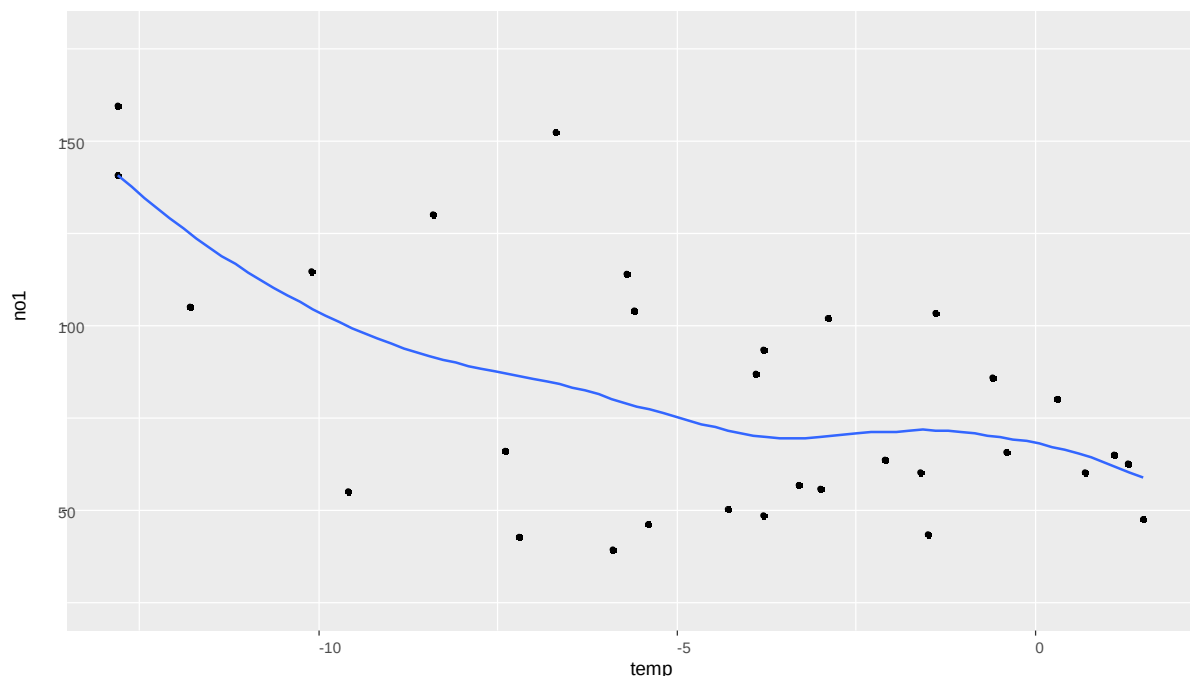
month	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
<ord>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1 January	133.	133.	114.	120.	122.	108.	96.1
2 February	109.	115.	106.	110.	92.9	97.9	104.
3 March	103.	100.	111.	99.9	96.2	89.8	97.3
4 April	93.3	94.6	101.	101.	101.	93.9	91.7
5 May	72.7	71.0	68.5	76.2	72.3	57.1	48.8
6 June	66.5	65.4	71.2	68.0	65.9	52.3	47.1
7 July	33.8	38.5	36.3	36.7	36.6	30.4	16.4
8 August	18.2	18.0	18.2	18.3	19.4	14.9	15.4
9 September	0.912	-0.175	-0.185	1.69	2.86	0.418	0.975
10 October	52.0	50.2	30.3	30.6	39.2	30.5	21.4
11 November	93.0	102.	92.3	96.7	77.8	77.5	79.3
12 December	84.0	96.0	87.6	79.3	88.4	71.7	63.5

e) For every month and each of the five regions, find the day with the highest electricity price.

```
elprices |>
  select(date, starts_with('no')) |>
  pivot_longer(cols = starts_with('no')) |>
  mutate(month = month(date, label = T, abbr = F)) |>
  rename(region = name,
         elprice = value) |>
  group_by(region, month) |>
  slice_max(elprice) |>
  select(date, month, region) |>
  pivot_wider(names_from = region,
             values_from = date)
```

f) The file `temperatures.csv` is a csv file with daily average temperatures at Blindern, Oslo, for each day of December 2023. Import the data and combine them with the December electricity prices used in question a). Construct a scatter plot with the relationship between average daily temperature and electricity price in region "no1". Does there seem to be a relationship between the two variables?

```
temperatures <- read_csv("temperatures.csv") |>
  mutate(date = ymd(20231200+day))
el_december |>
  left_join(temperatures) |>
  ggplot(aes(x=temp, y=no1)) +
  geom_point() +
  geom_smooth()
```

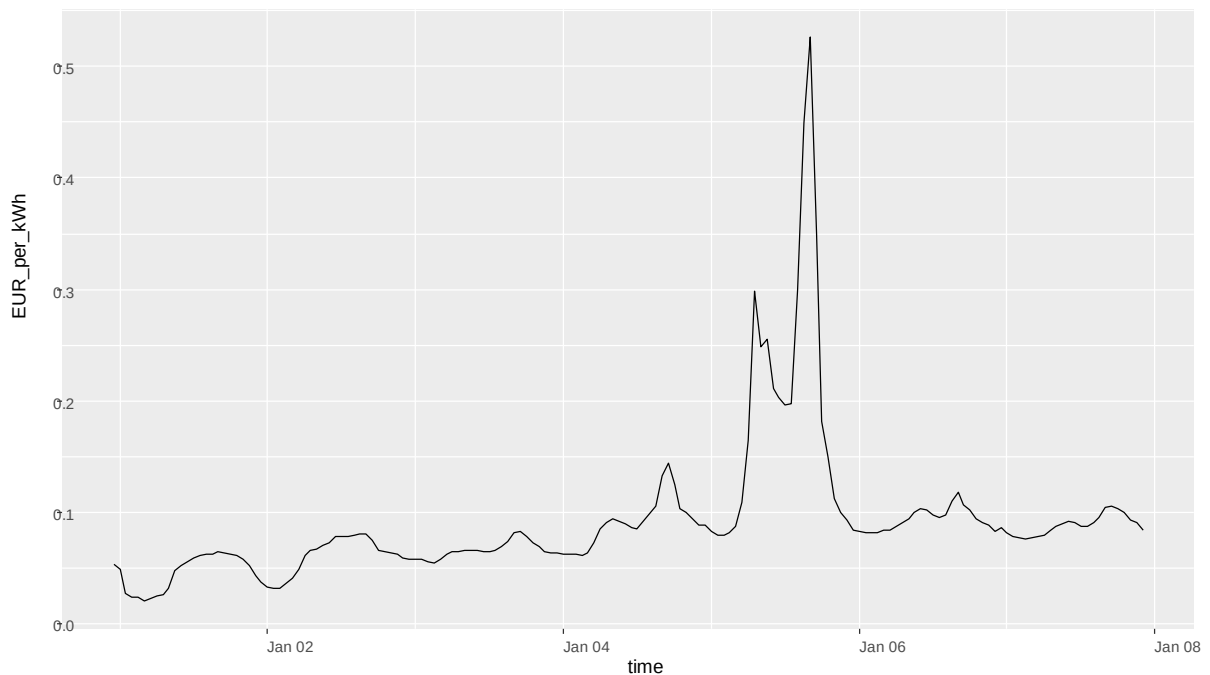


The two tibbles should be joined with a proper join (or merge) for full score.

g) The web site “Hva koster strømmen” (“What does the power cost?”) provides hourly electricity prices that can be downloaded using an API call. The link https://www.hvakosterstrommen.no/api/v1/prices/2024/01-02_NO1.json provides hourly prices for January 2nd 2024 for region no1 in JSON format. So the last part of the link contains month, day, and region.

Explain what characterized the JSON format. Make a loop that downloads the hourly data for the first week of January 2024 (i.e. 1st to 7th) for region “no1”, converts the data to a tibble, and combines all the data in a single data set. Construct a time series plot of the hourly electricity price in Euros.

```
library(jsonlite)
hour_data <- NULL
for (d in 1:7) {
  link <- paste("https://www.hvakosterstrommen.no/api/v1/prices/2024/01-",
    d, "_NO1.json", sep="")
  hour_data <- bind_rows(hour_data, fromJSON(link))
}
hour_data <- fromJSON(link)
hour_data |>
  mutate(time = ymd_hms(time_start)) |>
  ggplot(aes(x=time, y=EUR_per_kWh)) +
  geom_line()
```



JSON (JavaScript Object Notation) is a non-rectangular format for storing and transmitting data in plain text. Popular in APIs.

II. Simulating the St. Petersburg paradox (weight 25 %)

The St. Petersburg paradox is a game initially discussed by Bernoulli in the 18th century. A fair coin is flipped. If it turns heads up, the game ends and the player gets 1 ducat (a coin). If the coin turns tails up, the coin is flipped again. If it then turns heads up, the game ends and the player gets 2 ducats. If it turns up tails the game continues, now with stakes 4. Hence every time the coin turns tails up, the game continues with the stakes doubled. When the coin ends heads up the game ends.

a) Write a function that plays the game (until the coin turns up heads) and returns the number of times the coin was tossed. Call the function once and compute the gains to the player.

```
library(tidyverse)
tails = function() {
  return(runif(1)>.5)
}
petersburg <- function() {
  throws <- 1
  while(tails()) {
    throws <- throws+1
  }
  return(throws)
}
set.seed(12345)
gains <- 2^(petersburg()-1)
```

The condition for the while loop should be where the coin toss is evaluated. Solutions with infinite loops and break statements is spaghetti code that is not given full score.

b) Assume the player has a CRRA utility function, that is a function of the form

$$u(c) = \begin{cases} \frac{c^{1-\eta} - 1}{1-\eta} & \eta \neq 1 \\ \ln(c) & \eta = 1 \end{cases}$$

where $\eta \geq 0$ is the coefficient of relative risk aversion. Use Monte Carlo integration to find the expected utility of playing the St. Petersburg game for $\eta = \frac{1}{2}$, 1, and 2.

```
u <- function(c, eta) {
  if(eta==1) {
    return(log(c))
  } else {
    return((c^(1-eta)-1)/(1-eta))
  }
}

gains <- replicate(1e4, 2^(petersburg()-1))

mean(u(gains,.5))
mean(u(gains,1))
```

```
mean(u(gains,2))
```

```
[1] 1.373128
```

```
[1] 0.6907212
```

```
[1] 0.3333553
```

c) Find the security equivalent of playing the St. Petersburg game for players with risk aversion $\eta = \frac{1}{2}, 1$, and 2. The security equivalent is the non-stochastic consumption level that yields the same utility as the expected utility of playing the game.

```
sec_equiv <- function(Ec, eta) {  
  if(eta==1) {  
    return(exp(Ec))  
  } else {  
    return(  
      (((1-eta)*Ec+1)^(1/(1-eta)))  
    )  
  }  
}
```

```
sec_equiv(mean(u(gains,.5)), .5)
```

```
sec_equiv(mean(u(gains,1)), 1)
```

```
sec_equiv(mean(u(gains,2)), 2)
```

```
[1] 2.844498
```

```
[1] 1.995154
```

```
[1] 1.500049
```

Should comment that the security equivalent is higher for lower risk aversion. This is a risky bet, so less valuable the more risk averse one is.

The security equivalent can also be found by solving an equation instead of inverting the utility function. The inversion is simpler, but both are acceptable.

d) Consider now a risk neutral player, that is one with $\eta = 0$ and hence utility function $u(c) = c$. Find this player's expected utility from playing the game. Repeat your computation 10 times. Compare the results with 10 repeated computations for a player with $\eta = 2$. Explain the difference in precision level between the two sets of computations.

```
for (i in 1:10) {
  gains <- replicate(1e4, 2^(petersburg()-1))
  print(mean(u(gains,0)))
  print(mean(u(gains,2)))
}
```

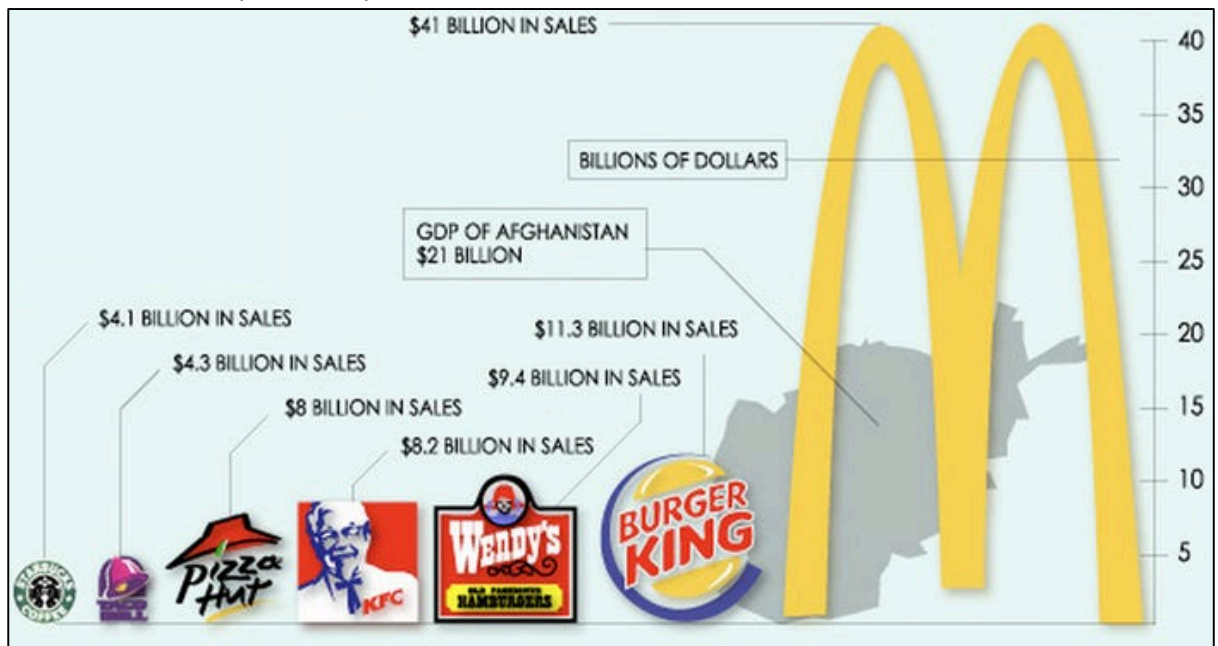
```
[1] 12.7899
[1] 0.3272431
[1] 9.0621
[1] 0.3367156
[1] 7.0164
[1] 0.3353947
[1] 12.0226
[1] 0.3328649
[1] 6.9525
[1] 0.3292855
[1] 7.3975
[1] 0.3344645
[1] 8.2471
[1] 0.3342133
[1] 5.4645
[1] 0.3382289
[1] 8.6739
[1] 0.3368841
[1] 10.7299
[1] 0.331484
```

In each repetition, the expected utility with risk aversion 2 gives about the same number (about 0.33), but some variation in this case as only 1000 replications are used.

With risk aversion 0 the number vary wildly, ranging from 5 to 13. It is well known and easy to show that the expected gains in the St. Petersburg paradox is infinity (which is why it is a paradox). Hence we sample from a distribution with expectation infinity. This means that the average of the draws does not converge to a finite value. Regularly very large values occur, disturbing the calculated average.

III. Visualizing data (weight 15 %)

- a) Consider the following visualization of sales in some big fast-food chains. Discuss good and bad properties of the visualization. Use the numbers provided in the graph to produce a new visualization and explain how your visualization is better than the one below.

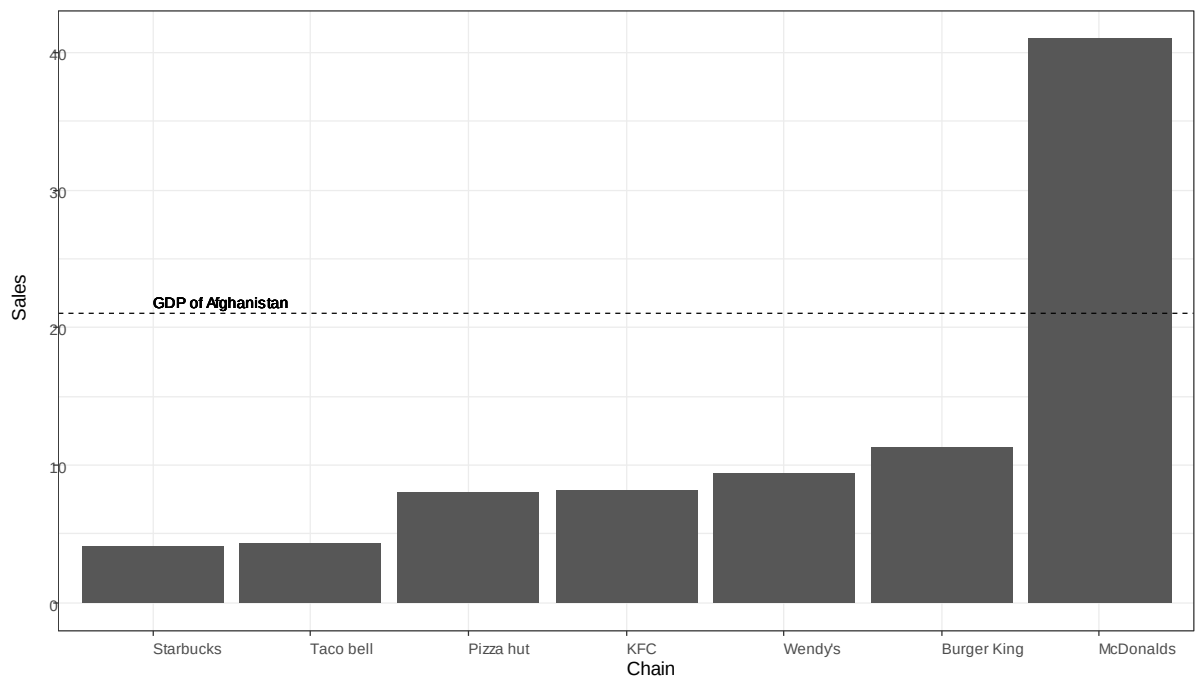


There are few good properties, the best may be that the numbers are on top of the “bars”. The major problem is that the logos vary both in height and width, so the area is the square of the number that is represented. The eye will typically judge the area, not the height, hence this gives a flawed impression. The different shapes with varying degree of transparency can also distort the reading of the graph. The comparison to the GDP of Afghanistan is odd.

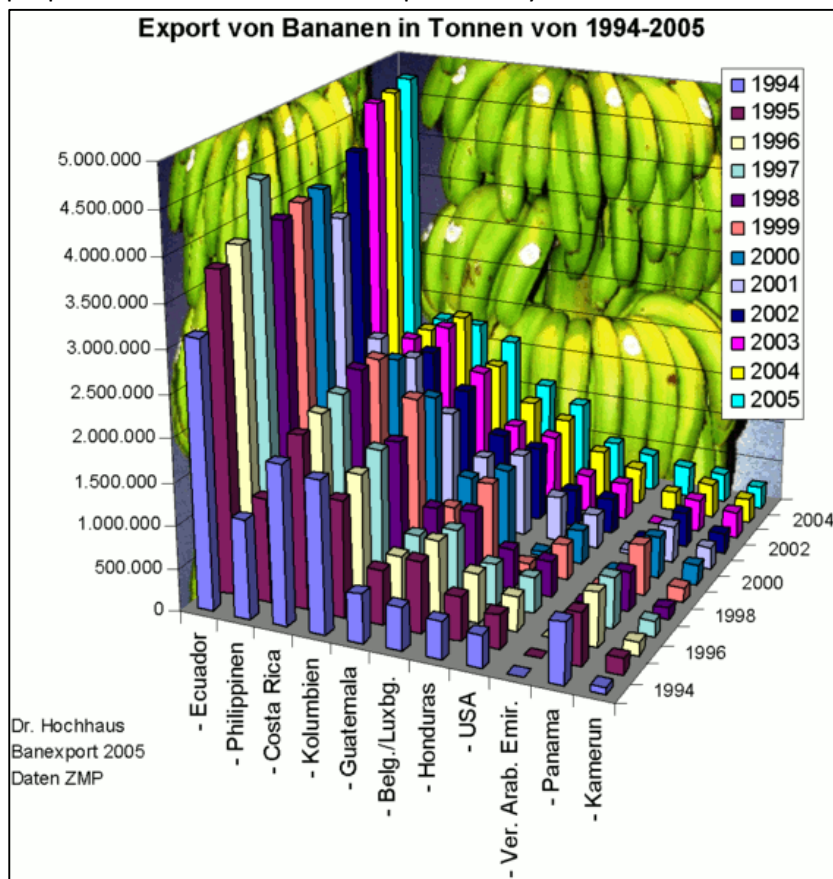
One possibility is as follows, where also Afghanistan is singled out as something different from fast food chains:

```
library(tidyverse)
```

```
fastfood <- tibble(
  names = c("Starbucks", "Taco bell", "Pizza hut", "KFC", "Wendy's",
            "Burger King", "McDonalds"),
  sales = c(4.1, 4.3, 8, 8.2, 9.4, 11.3, 41)
)
fastfood |>
  ggplot(aes(x=reorder(names, sales), y=sales)) +
  geom_col() +
  geom_hline(yintercept = 21,
            linetype = 'dashed') +
  geom_text(x=1, y=22, label = "GDP of Afghanistan", size = 3) +
  theme_bw() +
  labs(x='Chain', y='Sales')
```



- b) Consider next the following visualization of sales of bananas. Discuss good and bad properties of the visualization. Explain how you would show these numbers.



The main problem is that it is difficult to read the bars, as is usual with 3D bars. Also, the bananas in the back do not help. The colors of the bars do little to improve readability. A better graph would be a series of time series plots, one for each country. Depending on the message, it could also be better to express exports in logs.