

Assignment 1

#Before we start

Running libraries

```
library(tidyverse)
```

```
library(lubridate)
```

```
library(magrittr)
```

Loading data

```
trips= read_csv("https://sds-aau.github.io/SDS-master/M1/data/trips.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Warning: 28 parsing failures.
```

```
##   row      col expected      actual                                     file
## 4141 date_end  valid date 2016-06-31 'https://sds-aau.github.io/SDS-master/M1/data/trips.csv'
## 7438 date_end  date like  9/17/15    'https://sds-aau.github.io/SDS-master/M1/data/trips.csv'
## 8720 date_start valid date 2011-31-11 'https://sds-aau.github.io/SDS-master/M1/data/trips.csv'
## 16113 date_start date like 20016-01-12 'https://sds-aau.github.io/SDS-master/M1/data/trips.csv'
## 26626 date_start valid date 2016-02-30 'https://sds-aau.github.io/SDS-master/M1/data/trips.csv'
## .....
## See problems(...) for more details.
```

```
colnames(trips)[1] = "obs"
```

```
people= read_csv("https://sds-aau.github.io/SDS-master/M1/data/people.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
colnames(people)[1] = "obs"
```

```
country= read_csv("https://sds-aau.github.io/SDS-master/M1/data/countrylist.csv")
```

Looking at the data:

```
glimpse(trips)
```

```
## Rows: 46,510
```

```
## Columns: 11
```

```
## $ obs      <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
```

```
## $ username      <chr> "@lewellenmichael", "@lewellenmichael", "@lewellenmichael~
## $ country       <chr> "Mexico", "Mexico", "Mexico", "Jordan", "China", "Vietnam~
## $ country_code  <chr> "MX", "MX", "MX", "JO", "CN", "VN", "HK", "CN", "CN", "CN~
## $ country_slug  <chr> "mexico", "mexico", "mexico", "jordan", "china", "vietnam~
## $ date_end      <date> 2018-06-15, 2018-06-03, 2017-11-05, 2017-08-07, 2017-03--
## $ date_start    <date> 2018-06-04, 2018-05-31, 2017-11-01, 2017-07-24, 2017-02--
## $ latitude      <dbl> 21, 19, 21, 31, 40, 10, 22, 22, 22, 18, 7, 3, 11, 10, 13,~
## $ longitude     <dbl> -101, -99, -86, 35, 122, 106, 114, 114, 113, 109, 98, 101~
## $ place         <chr> "Guanajuato", "Mexico City", "Cancun", "Amman", "Yingkou"~
## $ place_slug    <chr> "mexico", "mexico-city-mexico", "cancun-mexico", "amman-j~
```

```
glimpse(people)
```

```
## Rows: 4,016
## Columns: 6
## $ obs          <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
## $ username     <chr> "@lewellenmichael", "@waylandchin", "@karan", "@skaboss2~
## $ followers    <dbl> 1, 0, 2, 0, 17, 3, 4, 2, 17, 2, 11, 11, 5, 8, 0, 9, 3, 5~
## $ following    <dbl> 2, 2, 1, 1, 426, 3, 9, 3, 23, 2, 17, 6, 9, 7, 1, 6, 3, 3~
## $ work_raw     <chr> "Software Dev, Startup Founder, Finance, Crypto, Product~
## $ education_raw <chr> "High School, Bachelor's Degree", NA, NA, NA, NA, NA, NA~
```

```
glimpse(country)
```

```
## Rows: 249
## Columns: 3
## $ alpha_2      <chr> "AF", "AX", "AL", "DZ", "AS", "AD", "AO", "AI", "AQ", "AG",~
## $ region       <chr> "Asia", "Europe", "Europe", "Africa", "Oceania", "Europe", ~
## $ sub_region   <chr> "Southern Asia", "Northern Europe", "Southern Europe", "Nor~
```

#Preprocessing

Trips: transform dates into timestamps (note: in Python, you will have to coerce errors for faulty dates)

```
trips %<>%
  mutate(date_end_time= as.numeric(as.POSIXct(date_end, format="%Y-%m-%d")), date_start_time=as.numeric
trips
```

```
## # A tibble: 46,510 x 13
##   obs username      country country_code country_slug date_end   date_start
##   <dbl> <chr>         <chr>    <chr>      <chr>      <date>     <date>
## 1     0 @lewellenmich~ Mexico    MX        mexico    2018-06-15 2018-06-04
## 2     1 @lewellenmich~ Mexico    MX        mexico    2018-06-03 2018-05-31
## 3     2 @lewellenmich~ Mexico    MX        mexico    2017-11-05 2017-11-01
## 4     3 @lewellenmich~ Jordan    JO        jordan     2017-08-07 2017-07-24
## 5     4 @waylandchin  China     CN        china      2017-03-18 2017-02-17
## 6     5 @waylandchin  Vietnam   VN        vietnam    2017-02-16 2016-09-02
## 7     6 @waylandchin  Hong Ko~  HK        hong-kong  2016-09-01 2016-08-02
## 8     7 @waylandchin  China     CN        china      2016-08-02 2016-07-31
## 9     8 @waylandchin  China     CN        china      2016-07-31 2016-07-03
## 10    9 @waylandchin  China     CN        china      2016-07-03 2016-06-03
```

```
## # ... with 46,500 more rows, and 6 more variables: latitude <dbl>,
## #   longitude <dbl>, place <chr>, place_slug <chr>, date_end_time <dbl>,
## #   date_start_time <dbl>
```

Calculate trip duration in days (you can use loops, list comprehensions or map-lambda-functions (python) to create a column that holds the numerical value of the day. You can also use the datetime package.)

```
trips %<>%
  mutate(trip_duration= date_end_time- date_start_time, trip_duration_days= trip_duration/86400)
```

Filter extreme (fake?) observations for durations as well as dates - start and end (trips that last 234565 days / are in the 17th or 23rd century) The minimum duration of a trip is 1 day! Hint: use percentiles/quantiles to set boundaries for extreme values - between 1 and 97, calculate and store the boundaries before subsetting. Rhint: Use percent_rank(as.numeric(variable)) to create percentiles

```
trips %<>%
  filter(!is.na(date_end_time))%>%
  filter(trip_duration_days >= 1 & trip_duration_days <= 97, date_end_time >= quantile(date_end_time, 0
```

Join the countrylist data to the trips data-frame using the countrycode as a key e. [Only for python users] Set DateTime index as the start date of a trip

```
trips%>%
  left_join(country, by= c("country_code" = "alpha_2")) %>%
  count(region)
```

```
## # A tibble: 6 x 2
##   region      n
##   <chr>    <int>
## 1 Africa   1375
## 2 Americas 11013
## 3 Asia    11887
## 4 Europe   13513
## 5 Oceania  1257
## 6 <NA>     1586
```

```
UK = data.frame("UK", "Europe", "Northern Europe")
names(UK) = c("alpha_2", "region", "sub_region")
country = rbind(country, UK)
```

```
trips%>%
  left_join(country, by= c("country_code" = "alpha_2")) %>%
  count(region)
```

```
## # A tibble: 6 x 2
##   region      n
##   <chr>    <int>
## 1 Africa   1375
## 2 Americas 11013
## 3 Asia    11887
## 4 Europe   15055
## 5 Oceania  1257
## 6 <NA>      44
```

```
trips%<>%
  left_join(country, by= c("country_code" = "alpha_2"))
```

1586 NAs before inserting the UK row into the country dataset, and only 44 after. This is due to the fact that the UK in the country dataset had the alpha_2 value of GB and in the trips dataset it had the country_code UK.

#people

How many people have at least a “High School” diploma? Hint: For this calculation remove missing value-rows or fill with “False”.

```
people %>%
  drop_na(education_raw) %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    451
```

First we drop the NAs. Thereafter we count the remaining observations as high school is presumed to be the lowest education in the category. From this we see that 451 people have at least a High School diploma.

How many “Startup Founders” have attained a “Master’s Degree”? Bonus: compared to people who don’t have a formal higher education (e.g. by using the “False” occurrences)?

```
people %>%
  filter(grepl("Startup Founder", work_raw)) %>%
  filter(grepl("Master", education_raw)) %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     53
```

```
people %>%
  filter(grepl("Startup Founder", work_raw)) %>%
  filter(!grepl("Master", education_raw)) %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    668
```

Here we use the grepl command to filter by both the work_raw and education_raw. The grepl lets us filter by strings containing certain text. By doing this we get that 53 startup founders have attained a Master’s Degree. By adding an “!” before the second grepl we can reverse the function so that we get all startup founders without a Master’s degree. This shows us that 668 startup founders does not have a master’s degree.

Who is the person with a Master’s Degree that has the highest number of followers? Bonus: Explore the individual further, what else can you find out?

```
people %>%
  filter(grepl("Master's Degree", education_raw)) %>%
  arrange(desc(followers)) %>%
  head(1) %>%
  select(-obs)
```

```
## # A tibble: 1 x 5
##   username followers following work_raw education_raw
##   <chr>      <dbl>      <dbl> <chr>      <chr>
## 1 @levelsio    2182        353 Software Dev, Startu~ High School, Bachelor's D~
```

To get the results we first filtered in the education_raw category by strings containing “Master’s Degree”. Afterwards we arranged by followers in descending order, so that the top result will be the person with most followers. Thereafter we write the “head(1)” function which excludes all results not in the top of the list. In the end we use the select function to exclude the ID of the person, because this is not relevant. The result shows us, that the person with the most followers is @levelsio. He has 2182 followers. He follows 353. He is a software Dev, a startup founder and a creative. Furthermore he has a high school diploma, a bachelor’s degree and a master’s degree.

#Trips

Which country received the highest number of trips? – And which the lowest?

```
trips %>%
  count(country, sort = TRUE) %>%
  filter(n %in% c(min(n), max(n)))
```

```
## # A tibble: 54 x 2
##   country      n
##   <chr>      <int>
## 1 United States 6539
## 2 180 40        1
## 3 45230         1
## 4 630 72        1
## 5 671 03        1
## 6 847 00        1
## 7 AB30         1
## 8 Abkhazia     1
## 9 AD100        1
## 10 Algeria     1
## # ... with 44 more rows
```

To find out which country received the most and fewest trips, we just count the number of times each country shows up in the trips datasets and then filter for the maximum and minimum value. The result shows that the most visited country is the United States with 6539, and 53 country share the price of being the lowest visited destination with one trip. Around thirteen of the countries whom has been visited the lowest amount has only numerical values in the country column, so those could have been dropped, but didnt because they still represent a trip just unclear to where.

Which region received the highest number of trips in 2017? Use the start of trips as a time reference.

```
trips %>%
  drop_na(region) %>%
  filter(date_start >= "2017-01-01", date_start <= "2017-12-31") %>%
  count(region, sort = TRUE)
```

```
## # A tibble: 5 x 2
##   region      n
##   <chr>    <int>
## 1 Europe   4659
## 2 Asia     3356
## 3 Americas 3197
## 4 Africa    414
## 5 Oceania   374
```

To access regions we need to join our trips and country datasets, but we already did that earlier and saved it as our new trips dataset, so no need to do that again. Then we drop the trips which have no region because they are not of significance to us. Then we filter for trips that started in 2017 and count the number of times each region was visited, and then we end up with, that the most visited region in 2017 was Europe with 4659 trips.

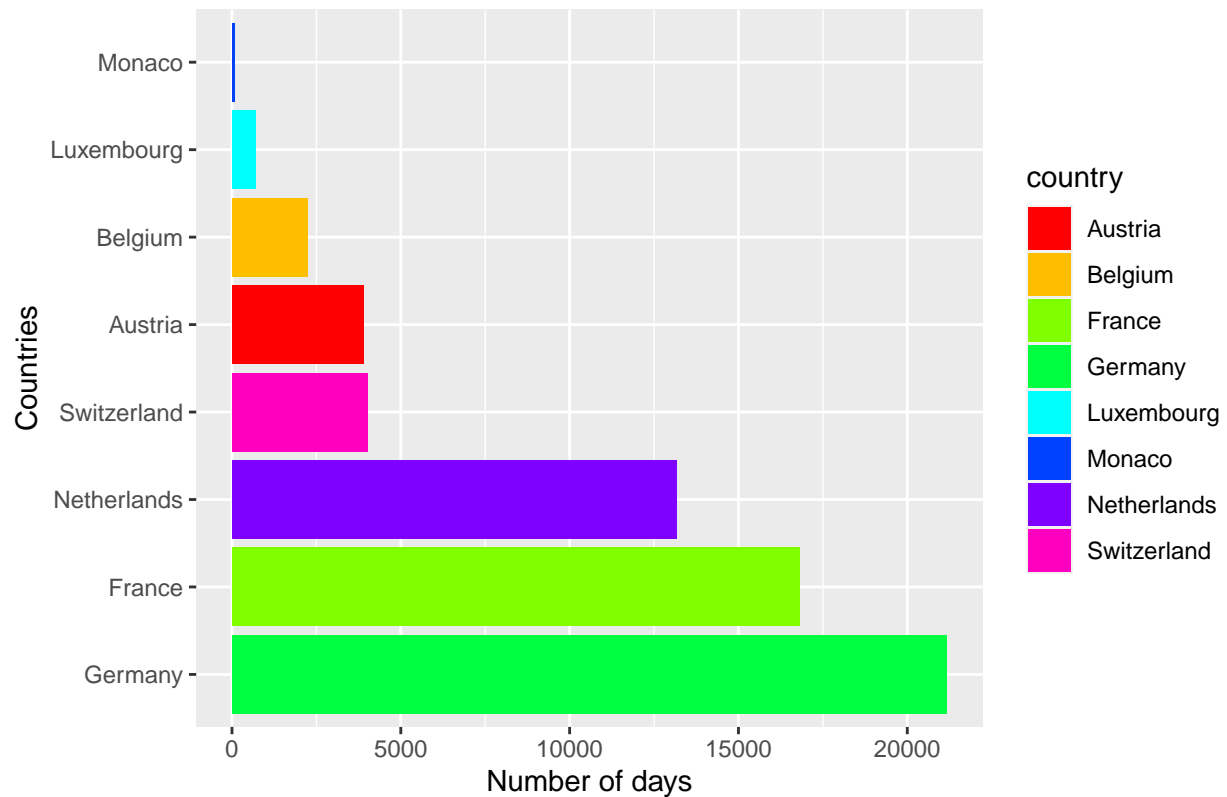
Which country in “Western Europe” did travelers spend least time? – Provide visualization

```
plot_trips = trips %>%
  drop_na(sub_region, trip_duration_days) %>%
  filter(sub_region == "Western Europe") %>%
  group_by(country) %>%
  summarise(total_time = sum(trip_duration_days))
```

To figure out which country in Western Europe travelers spent the least time we first drop the NAs from both sub_region and trip_duration_days as observations with no values in are of no interest to us. Then we filter for the sub_region Western Europe, group_by country and summarise the total number of days spent in each Western European country. The above results will now be presented visually.

```
plot_trips %>%
  ggplot(aes(x = total_time, y = reorder(country, -total_time), fill = country)) +
  geom_col() + scale_fill_manual(values = rainbow(8)) +
  ggtitle("Total number of days spent in Western European countries") +
  labs(x = "Number of days", y = "Countries")
```

Total number of days spent in Western European countries



The above bar chart show visually the number of days spent in each western european country, where Monaco clearly is where travelers have spent the least amount of time.

Do nomad Startup Founders tend to have shorter or longer trips on average?

```
trips %>%
  left_join(people, by = c("username" = "username")) %>%
  filter(grepl("Startup Founder", work_raw)) %>%
  summarise(avg_time = mean(trip_duration_days))
```

```
## # A tibble: 1 x 1
##   avg_time
##   <dbl>
## 1      13.7
```

```
trips %>%
  left_join(people, by = c("username" = "username")) %>%
  filter(!grepl("Startup Founder", work_raw)) %>%
  summarise(avg_time = mean(trip_duration_days))
```

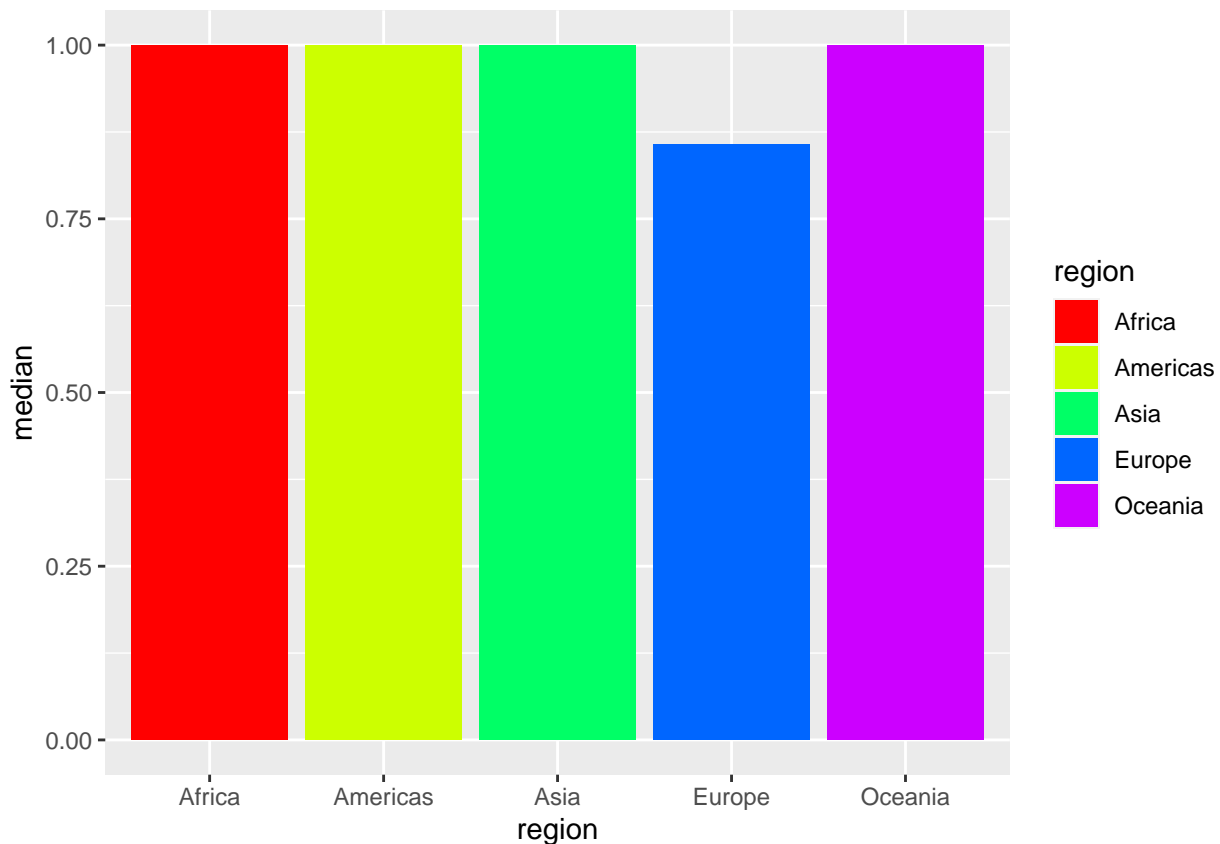
```
## # A tibble: 1 x 1
##   avg_time
##   <dbl>
## 1      14.3
```

On average every person who is or has been a Startup Founder spend 13.74 days per trip vs. 14.33 days per

trip for all other occupations. We are not sure if this is the result you guys wanted, because we had a hard time understanding what we would be comparing the startup founders average trip length to.

visualize over-time median trip duration overall (bonus: and split by world-region) The plot will look weird ^^. PyHint: Resample by week ('W') and calculate the size of observations. RHint: Use the floor_date function to reset dates by week.

```
trips %>%
  drop_na(region) %>%
  mutate(trip_duration_week = trip_duration_days/7) %>%
  group_by(region) %>%
  summarise(median = median(trip_duration_week)) %>%
  ggplot(aes(x = region, y = median, fill = region)) + geom_col() + scale_fill_manual(values = rainbow(5))
```



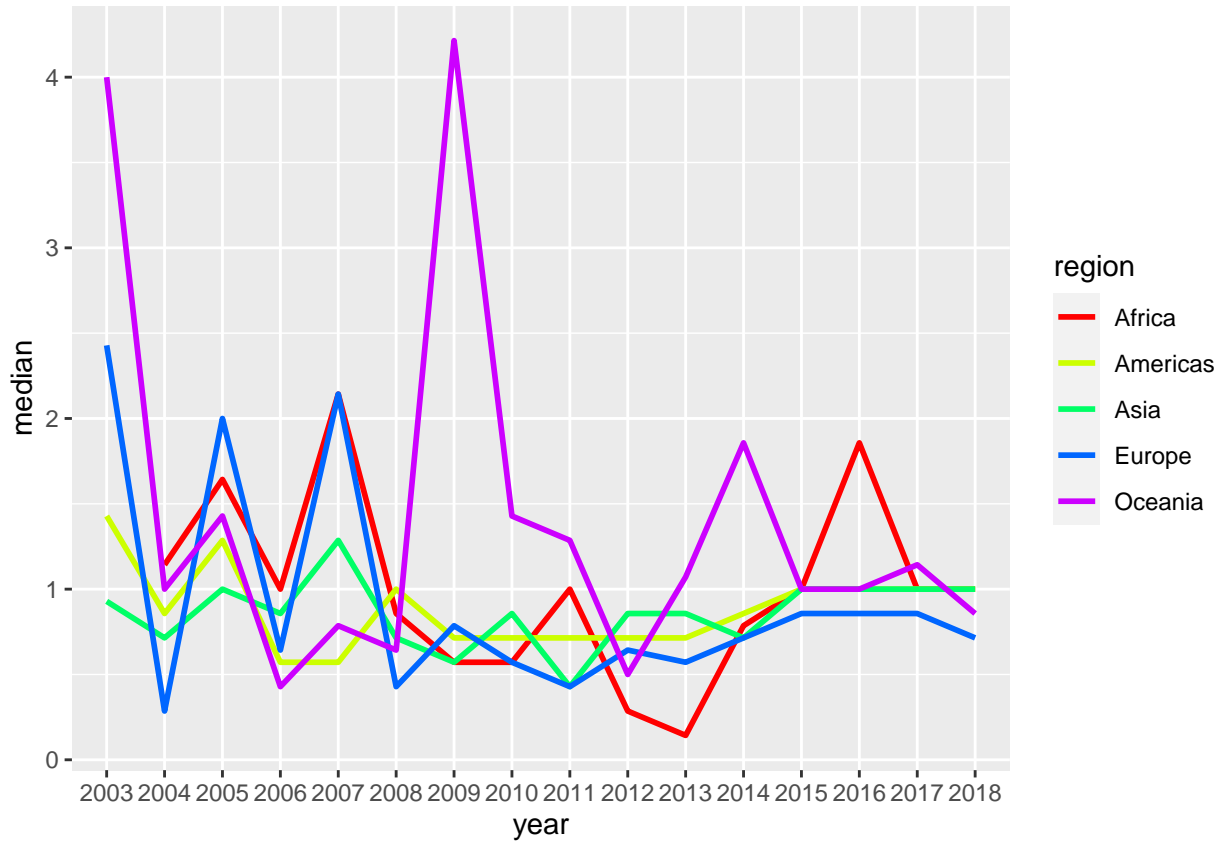
So we are not sure what we should use the floor_date function to, instead we just made our trip_duration_days column to trip_duration_week by dividing it with 7. Beforehand we dropped the NAs and then we grouped by region to calculate the median for every region and then we plotted it. This just doesnt visualise it over time, this only visualize the median by region over the entire time period. The assignment doesnt state in which interval to check the overall development in the median, so below we plot the median by region every year in the dataset.

```
trips %>%
  mutate(year = format(as.Date(trips$date_start, format="%Y-%m-%d"), "%Y")) %>%
  drop_na(region, year) %>%
  mutate(trip_duration_week = trip_duration_days/7) %>%
  group_by(region, year) %>%
```



```
summarise(median = median(trip_duration_week)) %>%
  ggplot(aes(x = year, y = median, color = region, group = region)) + geom_line(size = 1) + scale_color.
```

'summarise()' has grouped output by 'region'. You can override using the '.groups' argument.



We do this by making a column representing which year the trip took place, then grouping both by region and year, to get the medians and then plotting them using ggplot. We see that the median changes over time and mostly in the region of Oceania.