Explanation and Codes

Linh Tran

2/5/2021

Data Cleansing for 'accounts' dataset

The first step of this part is to import the data, change the data type of some variables and add more column for better analytics. The next step is to clean the data by checking for duplicate, outliers so that we have a dataset that is in good shape and ready for data analytics.

Data Importing:

Load tidyverse library

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#importing accounts dataset, changing the data type of some variables
accounts <- read.csv('accounts.csv')
accounts$account_created_time <- as.POSIXct(accounts$account_created_time, format = "%Y-%m-%d %H:%M:%S")
accounts$is_deleted <-  as.logical(accounts$is_deleted)
accounts$main_industry <- as.factor(accounts$main_industry)
accounts$sub_industry <- as.factor(accounts$sub_industry)
accounts$type <- as.factor(accounts$type)
accounts$support_tier <- as.factor(accounts$support_tier)

#Delete empty 'continent' and 'country' columns
accounts <- accounts[,-c(4,5)]

#Add more columns for data analytics:
accounts$month_yr <- as.factor(format(accounts$account_created_time, '%Y-%m'))
accounts$hour <- as.factor(format(accounts$account_created_time,'%H'))
accounts$account_created_date <- as.factor(format(accounts$account_created_time, '%Y-%m-%d'))

head(accounts)
```

```
##           account_id account_created_time is_deleted
## 1 0015J00000AZd05QAD  2020-10-13 08:25:56      FALSE
## 2 0015J000002mXt3QAE  2020-03-12 14:24:50      FALSE
## 3 0015J000002mXa6QAE  2020-03-12 14:23:58      FALSE
## 4 0015J000002mWwQQAU  2020-03-12 14:20:19      FALSE
## 5 0015J000002mWvqQAE  2020-03-12 14:20:19      FALSE
## 6 0015J000002mWeXQAU  2020-03-12 14:20:06      FALSE
##                          main_industry sub_industry       type support_tier
## 1 Information Technology and Services              Whitelabel
## 2                                                    Prospect
## 3                                                    Prospect
## 4                                                    Prospect
## 5                                                    Prospect
## 6                                                    Prospect
##   month_yr hour account_created_date
## 1  2020-10   08           2020-10-13
## 2  2020-03   14           2020-03-12
## 3  2020-03   14           2020-03-12
## 4  2020-03   14           2020-03-12
## 5  2020-03   14           2020-03-12
## 6  2020-03   14           2020-03-12
```

Data Cleansing:

Check for duplicate observations

```
duplicate <-accounts[duplicated(accounts),]

duplicate
```

```
##  [1] account_id           account_created_time is_deleted
##  [4] main_industry        sub_industry         type
##  [7] support_tier         month_yr             hour
## [10] account_created_date
## <0 rows> (or 0-length row.names)
```

There is no duplicate rows in *'accounts'* table.

Check for outliers

The number of new accounts created by month of the year:

```
month_yr_new_account <- accounts %>%
                         group_by(month_yr) %>%
                         summarise(number_of_accounts = n_distinct(account_id))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
month_yr_new_account
```

```
## # A tibble: 9 x 2
##   month_yr number_of_accounts
##   <fct>                <int>
```

```
## 1 2020-03                4048
## 2 2020-06                 175
## 3 2020-07                 268
## 4 2020-08                1373
## 5 2020-09                1109
## 6 2020-10                 763
## 7 2020-11                 705
## 8 2020-12                 648
## 9 2021-01                 457
```

There is a significant amount of new account in March, 2020 => check the number of new accounts created by days in March, 2020:

```
march_accounts <- accounts %>%
                    filter(month_yr=='2020-03') %>%
                    group_by(account_created_date) %>%
                    summarise(number_of_accounts = n_distinct(account_id))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
march_accounts
```

```
## # A tibble: 15 x 2
##    account_created_date number_of_accounts
##    <fct>                             <int>
##  1 2020-03-12                         3947
##  2 2020-03-13                            4
##  3 2020-03-14                            5
##  4 2020-03-15                            3
##  5 2020-03-16                            3
##  6 2020-03-17                           14
##  7 2020-03-18                           11
##  8 2020-03-19                           11
##  9 2020-03-20                            8
## 10 2020-03-21                            3
## 11 2020-03-22                            6
## 12 2020-03-23                            6
## 13 2020-03-24                           15
## 14 2020-03-25                            7
## 15 2020-03-26                            5
```

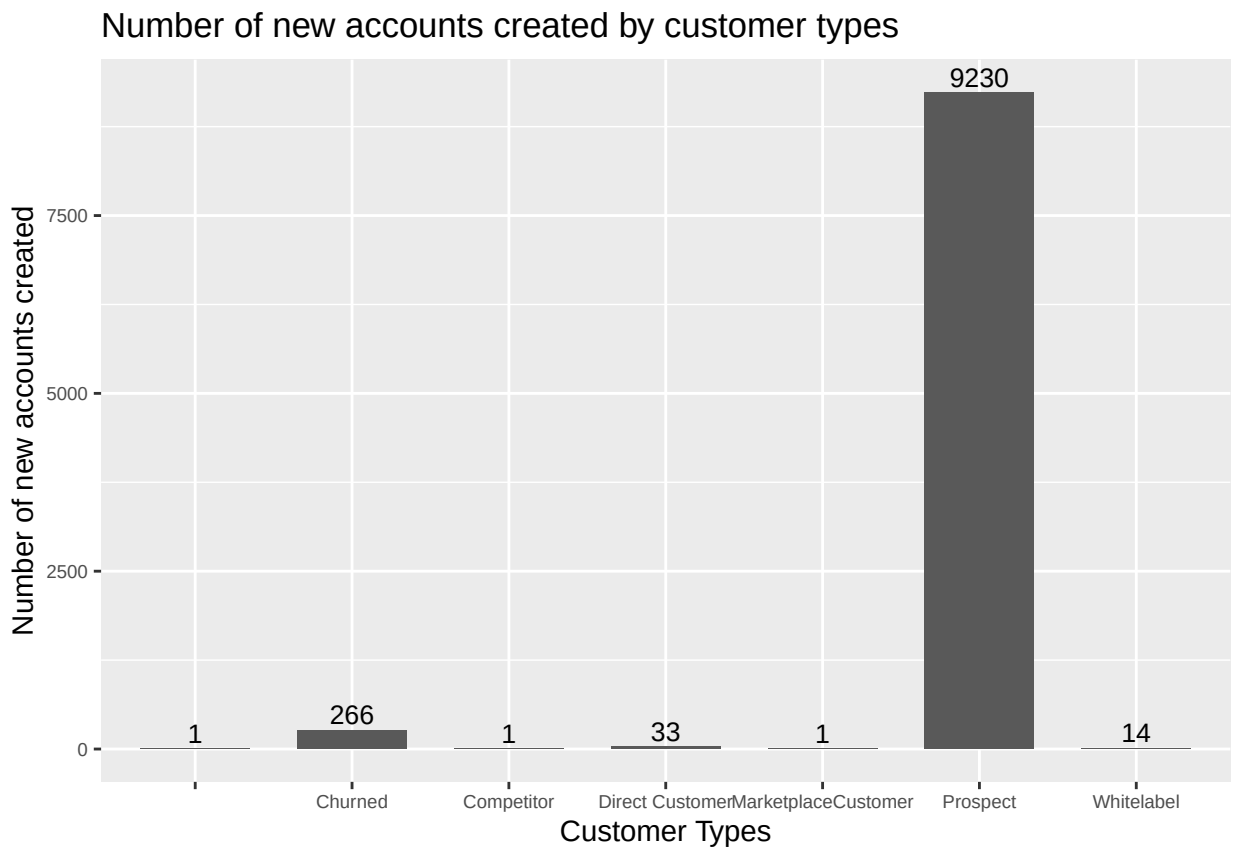
In 1 day: *12/03/2020*, there were 3,947 accounts created => this is an outlier, created by either a super duper successful day or an error in data inputting. => in order to have better analytics results, especially with date-time analytics, this report will exclude the accounts created in 12th of March 2020 on date-time analytics only. The other analytics that don't related to date-time will be performed with full data-set.

Data analytics for 'accounts' dataset

The number of new accounts created by the Customer Type

```
accounts %>%
  group_by(type) %>%
  summarise(number_of_users = n_distinct(account_id)) %>%
  ggplot(aes(x=type, y=number_of_users)) +
  geom_bar(stat="identity",width = 0.7) +
  theme(axis.text = element_text(size = 7)) +
  geom_text(aes(label=number_of_users), vjust=-0.3, size=3.5) +
  ggtitle("Number of new accounts created by customer types") +
  labs(y ="Number of new accounts created", x = "Customer Types" )
```

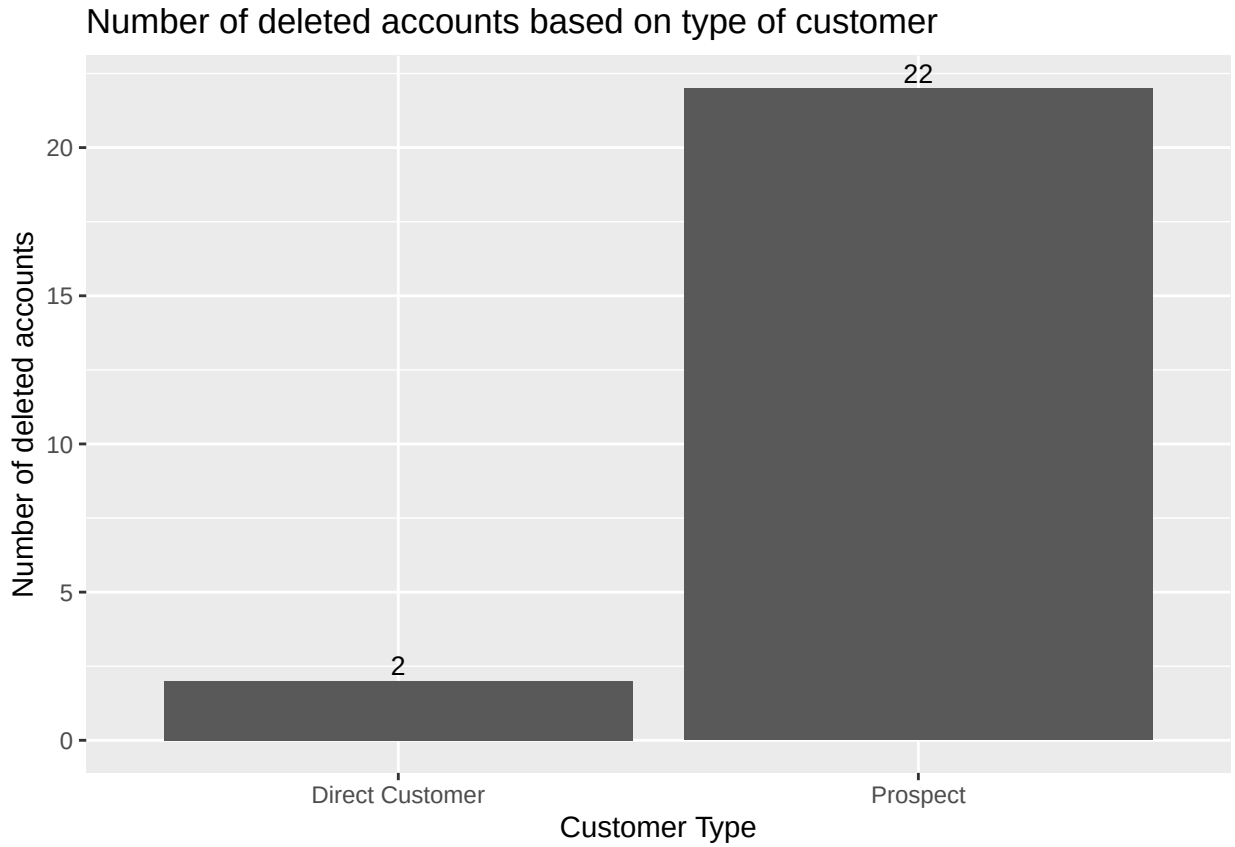
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



The number of new accounts that were deleted by Customer Type

```
accounts %>%
  group_by(type) %>%
  filter(is_deleted == 'TRUE') %>%
  summarise(number_of_users = n_distinct(account_id)) %>%
  ggplot(aes(x=type,y=number_of_users)) +
  geom_bar(stat ="identity")+
  geom_text(aes(label=number_of_users), vjust=-0.3, size=3.5)+
  ggtitle("Number of deleted accounts based on type of customer") +
  labs(y = "Number of deleted accounts", x = "Customer Type")
```

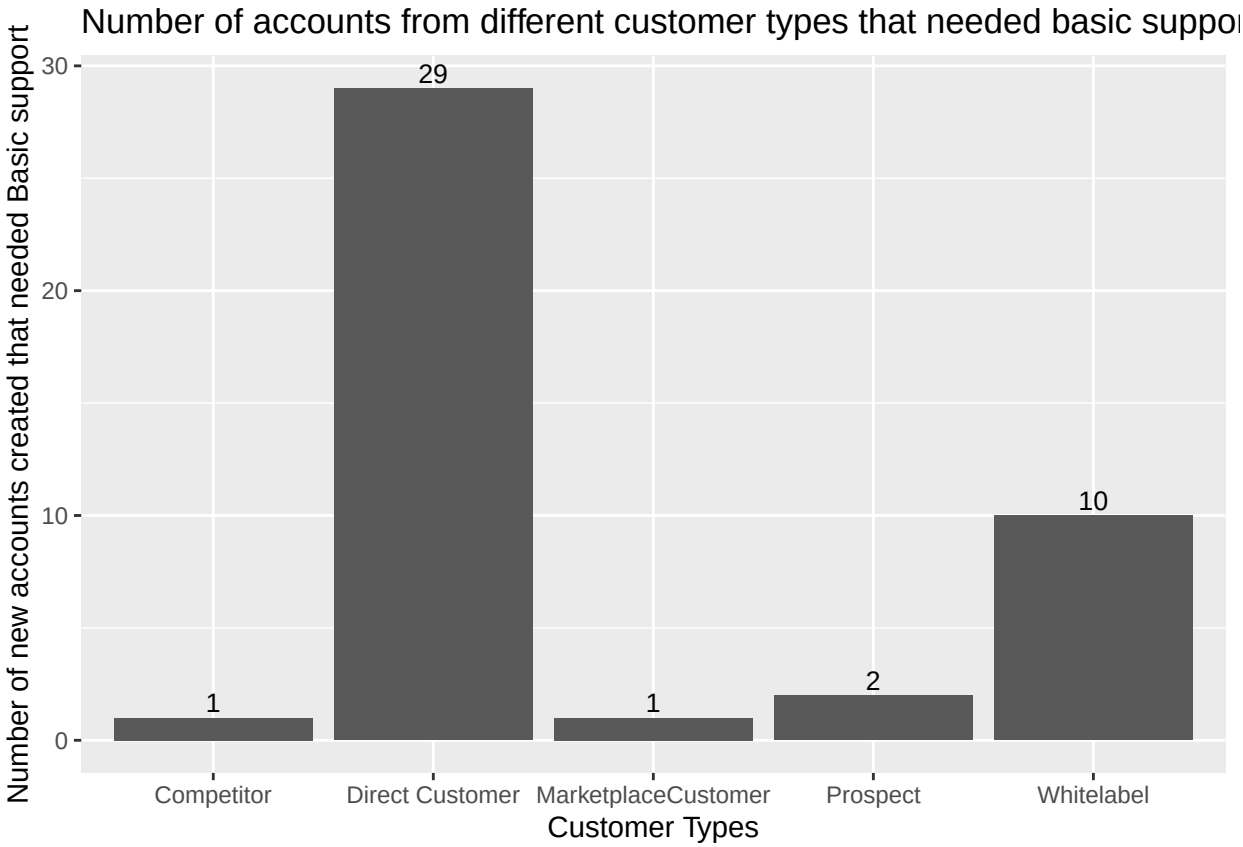
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



The number of new accounts that needed the basic support tier by Customer Type

```
basic_support_tier <- accounts %>%
  select(account_id,account_created_date,hour,is_deleted,type,support_tier) %>%
  filter(support_tier == 'Basic')
basic_support_tier %>%
  group_by(type) %>%
  summarise(number_of_users = n_distinct(account_id)) %>%
  ggplot(aes(x = type, y = number_of_users)) +
  geom_bar(stat="identity") +
  geom_text(aes(label=number_of_users), vjust=-0.3, size=3.5)+
  ggtitle("Number of accounts from different customer types that needed basic support") +
  labs(y ="Number of new accounts created that needed Basic support", x = "Customer Types" )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

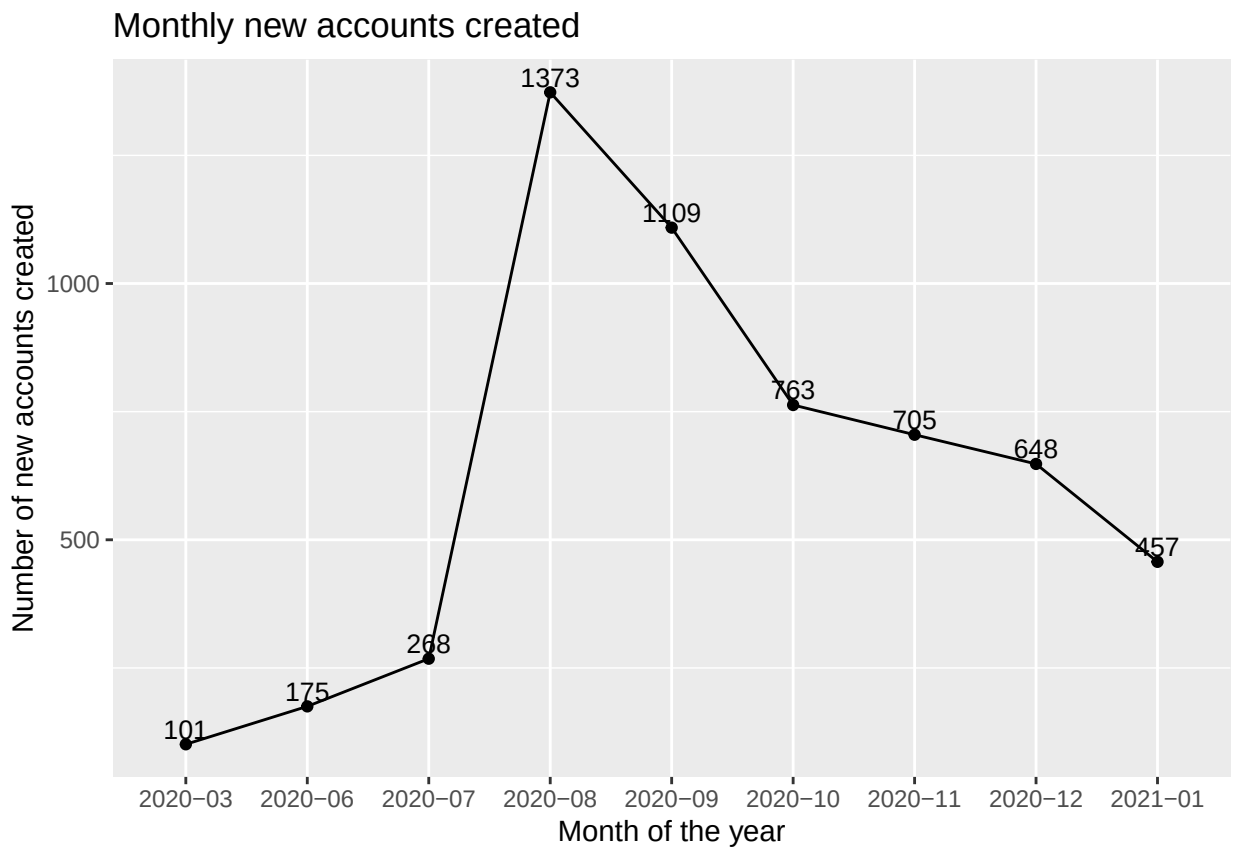


The number of new accounts created in each month during the data period

Since there is an outlier in the data in 12th of March, 2020, this part of the report will exclude the records of accounts created in 12th of March 2020 for better analysis.

```
#Filter out the account created in 12th of March 2020
accounts <- accounts %>%
  filter(!(account_created_date == '2020-03-12'))
#Calculate the number of new accounts created by month and create line chart
accounts %>%
  group_by(month_yr) %>%
  summarise(number_of_accounts = n_distinct(account_id)) %>%
  ggplot(aes(x=month_yr, y=number_of_accounts, group = 1)) +
  geom_line( color="black") +
  geom_point() +
  geom_text(aes(label=number_of_accounts), vjust=-0.3, size=3.5) +
  ggtitle("Monthly new accounts created") +
  labs(y = "Number of new accounts created", x = "Month of the year")
```

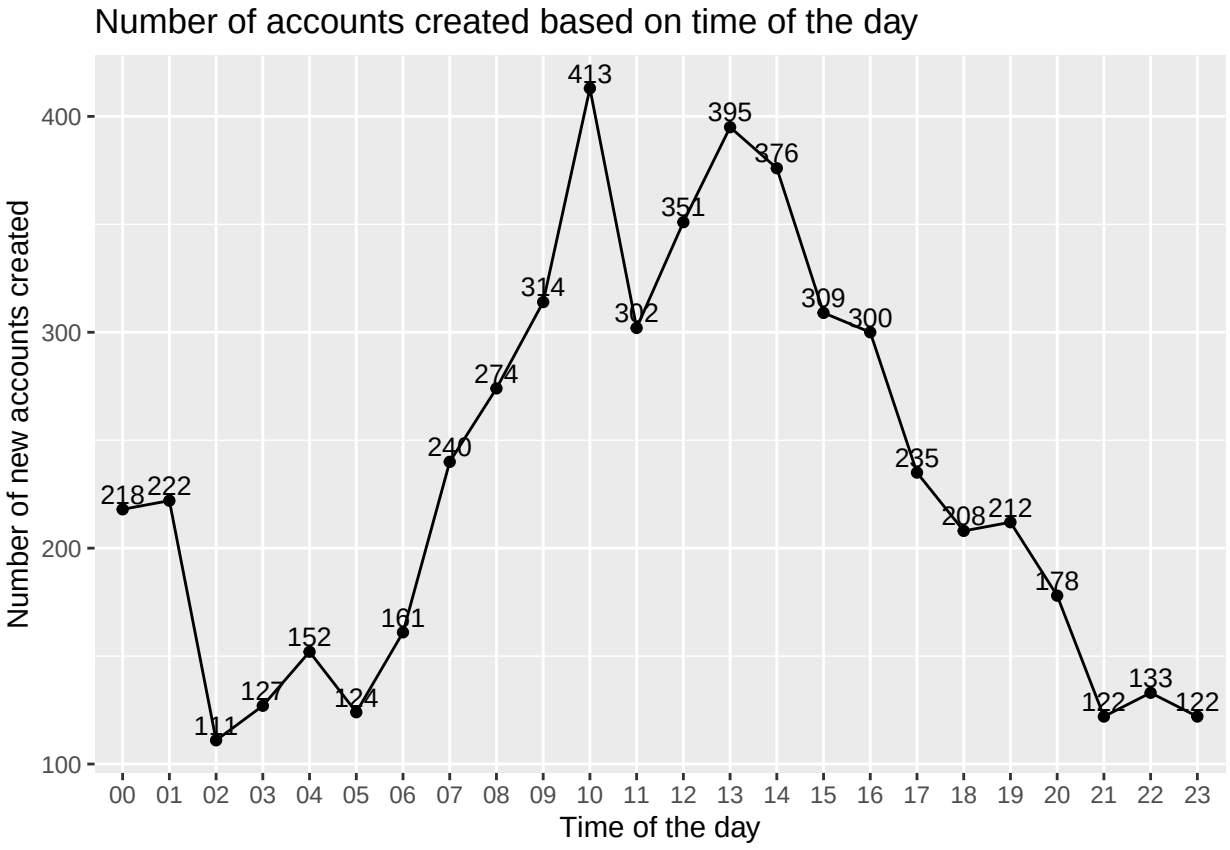
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



The number of new accounts created by time in a day

```
accounts %>%
  group_by(hour) %>%
  summarise(number_of_accounts = n_distinct(account_id)) %>%
  ggplot(aes(x=hour, y=number_of_accounts, group = 1)) +
  geom_line( color="black") +
  geom_point() +
  geom_text(aes(label=number_of_accounts), vjust=-0.3, size=3.5) +
  ggtitle("Number of accounts created based on time of the day") +
  labs(y = "Number of new accounts created", x = "Time of the day")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Data Cleansing for 'account_metrics' table

The first step of this part is to import the data, change the data type of some variables and add more column for better analytics. The next step is to clean the data by checking for duplicate, outliers so that we have a dataset that is in good shape and ready for data analytics.

Data importing:

```
#Import csv file
account_metrics <- read.csv('account_metrics.csv')
#Change the data types of some variables
account_metrics$latest_service_type <- as.factor(account_metrics$latest_service_type)
account_metrics$latest_service_created <- as.POSIXct(account_metrics$latest_service_created, format = "
account_metrics$month_yr <- as.factor(format(account_metrics$latest_service_created, '%Y-%m'))
account_metrics$latest_service_created_year <- as.factor(format(account_metrics$latest_service_created,
#Change the name of the variables for easier analytics
account_metrics <- account_metrics %>%
  rename(
    kafka = active_kafka_services,
    pg = active_pg_services,
    elasticsearch = active_elasticsearch_services,
    redis = active_redis_services,
    mysql = active_mysql_services,
    grafana = active_grafana_services,
    influxdb = active_influxdb_services,
```

```
    m3 = active_m3_services,
    ksql = active_ksql_services,
    kafka_connect = active_kafka_connect_services,
    cassandra = active_cassandra_services,
    kafka_mirrormaker = active_kafka_mirrormaker_services
  )
nrow(account_metrics)
```

```
## [1] 1357
```

There are 1,357 observations that was imported from csv file.

Data Cleansing:

Check for duplicate values:

```
duplicate2 <- account_metrics[duplicated(account_metrics),]
duplicate2
```

```
##  [1] account_id                 all_projects
##  [3] active_projects            all_services
##  [5] active_services            stopped_services
##  [7] kafka                      pg
##  [9] elasticsearch              redis
## [11] mysql                      grafana
## [13] influxdb                   m3
## [15] ksql                       kafka_connect
## [17] cassandra                  kafka_mirrormaker
## [19] clouds_in_use              clouds_in_use_names
## [21] cloud_regions_in_use       cloud_regions_in_use_names
## [23] gcp_projects               gcp_services
## [25] aws_projects               aws_services
## [27] azure_projects             azure_services
## [29] do_projects                do_services
## [31] upcloud_projects           upcloud_services
## [33] custom_do_projects         custom_do_services
## [35] packet_projects            packet_services
## [37] latest_service_type        latest_service_created
## [39] month_yr                   latest_service_created_year
## <0 rows> (or 0-length row.names)
```

There is no duplicated records in this dataset.

Check the accuracy of numeric columns:

During data analytics process, I made 3 assumptions:

1. In an observation, the number in 'active_projects' column should be equal to the sum of the numbers in 'gcp_projects', 'aws_projects', 'azure_projects', 'do_projects', 'upcloud_projects', 'custom_do_projects', and 'packet_projects' columns.
2. In an observation, the number in 'all_services' should be equal to the sum of the numbers in 'active_services' and 'stopped_services' columns.

3. In an observation, the number in 'active_services' should be equal to the sum of the numbers in 'kafka', 'pg', 'elasticsearch', 'redis', 'mysql', 'grafana', 'influxdb', 'm3', 'ksql', 'kafka_connect', 'cassandra', and 'kafka_mirrormaker' columns (all of the service products columns).

Any observations that do not meet the requirements of these assumptions are considered the inaccurate observation and should be exclude from the data.

In order to test these requirements, I created 3 new variables:

- *'check1'*: created by take the number in 'active_projects' minus the numbers in 'gcp_projects', 'aws_projects', 'azure_projects', 'do_projects', 'upcloud_projects', 'custom_do_projects', and 'packet_projects' columns.
- *'check2'*: created by take the number in 'all_services' minus the numbers in 'active_services' and 'stopped_services' columns
- *'check3'* created by take the number in 'active_services' minus the numbers in 'kafka', 'pg', 'elasticsearch', 'redis', 'mysql', 'grafana', 'influxdb', 'm3', 'ksql', 'kafka_connect', 'cassandra', and 'kafka_mirrormaker' columns (all of the service products columns).

Then I'll filter out all the observation that have the value of either 1 or more of these 3 columns equal 0.

```
#Create 3 new checking variables
account_metrics <- account_metrics %>%
                    select_all() %>%
                    mutate(check1=active_projects-gcp_projects-aws_projects-azure_projects-do_projects-
                    mutate(check2=all_services - active_services - stopped_services) %>%
                    mutate(check3=active_services-kafka-pg-elasticsearch-redis-mysql-grafana-influxdb-m
#filter out the rows that do not meet the requirements
account_metrics <- account_metrics %>%
                      filter(!(check1 != 0|check2 != 0| check3 != 0))
nrow(account_metrics)
```

```
## [1] 1312
```

Now there are only 1,312 observations left. 45 inaccurate observations were filtered out of the dataset.

Correct the inaccurate JSON key:

During the process of getting to know the data, I noticed the inaccurate JSON key for the empty value of 'clouds_in_use_names' and 'cloud_regions_in_use_names' columns. In order to deal with JSON columns, this incorrect key is needed to be replaced.

```
account_metrics$clouds_in_use_names[account_metrics$clouds_in_use_names == '{"v":]}'] <- '{"v":[{"v":"NA
account_metrics$cloud_regions_in_use_names[account_metrics$cloud_regions_in_use_names == '{"v":]}'] <-
head(account_metrics$clouds_in_use_names)
```

```
## [1] "{\"v\":[{\"v\":\"NA\"}]}" "{\"v\":[{\"v\":\"NA\"}]}"
## [3] "{\"v\":[{\"v\":\"NA\"}]}" "{\"v\":[{\"v\":\"NA\"}]}"
## [5] "{\"v\":[{\"v\":\"NA\"}]}" "{\"v\":[{\"v\":\"NA\"}]}"
```

```
head(account_metrics$cloud_regions_in_use_names)
```

```
## [1] "{\"v\":[{\"v\":\"NA\"}]}" "{\"v\":[{\"v\":\"NA\"}]}"
## [3] "{\"v\":[{\"v\":\"NA\"}]}" "{\"v\":[{\"v\":\"NA\"}]}"
## [5] "{\"v\":[{\"v\":\"NA\"}]}" "{\"v\":[{\"v\":\"NA\"}]}"
```

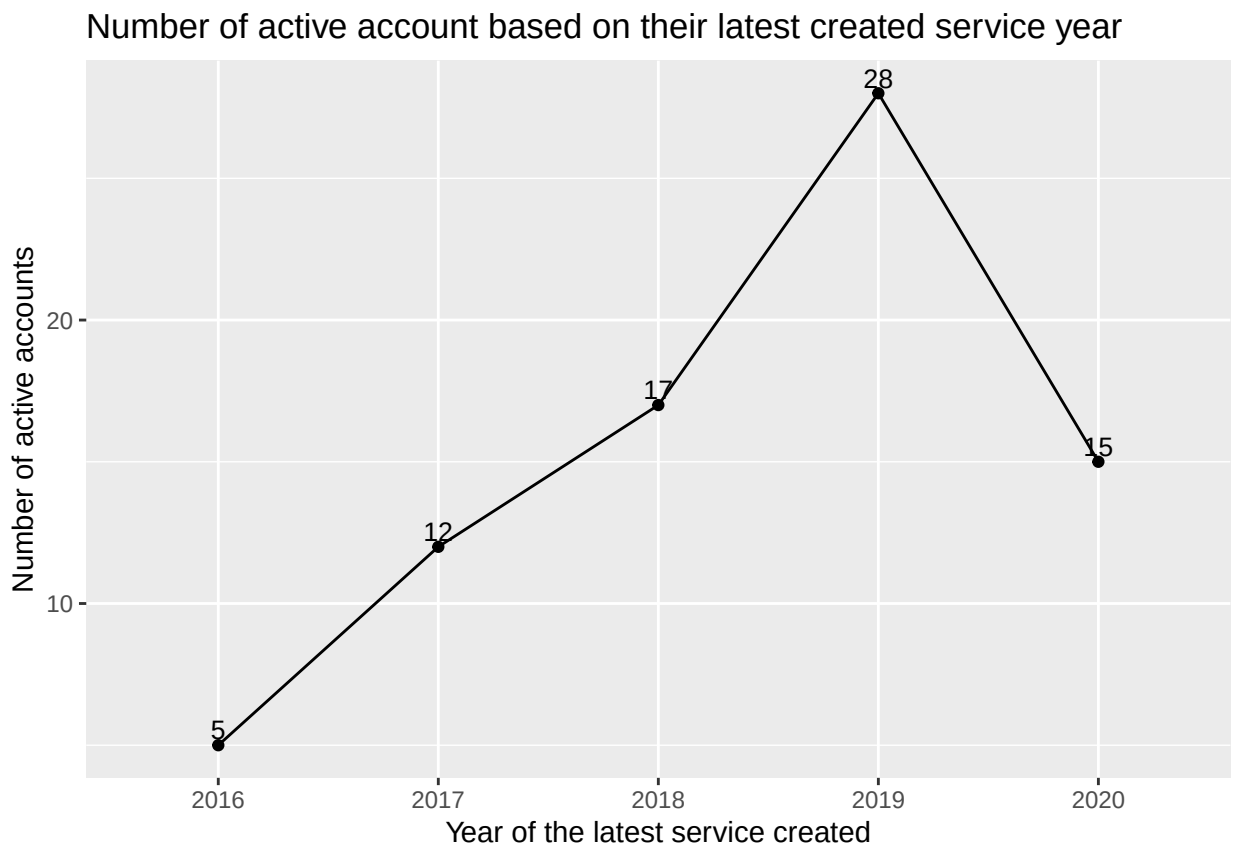
Data Analytics for 'account_metrics' dataset

Calculate number of active accounts based on the latest created service year

This calculation assumes that an active account is an account that either have the number in 'active_projects' bigger than 0 or the number in 'active_services' bigger than 0.

```
account_metrics %>%
  filter(active_projects > 0 | active_services > 0) %>%
  group_by(latest_service_created_year)%>%
  summarise(number_of_accounts = n_distinct(account_id))%>%
  ggplot(aes(x=latest_service_created_year, y=number_of_accounts, group = 1)) +
  geom_line( color="black") +
  geom_point() +
  geom_text(aes(label=number_of_accounts), vjust=-0.3, size=3.5) +
  ggtitle("Number of active account based on their latest created service year") +
  labs(y = "Number of active accounts", x = "Year of the latest service created")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

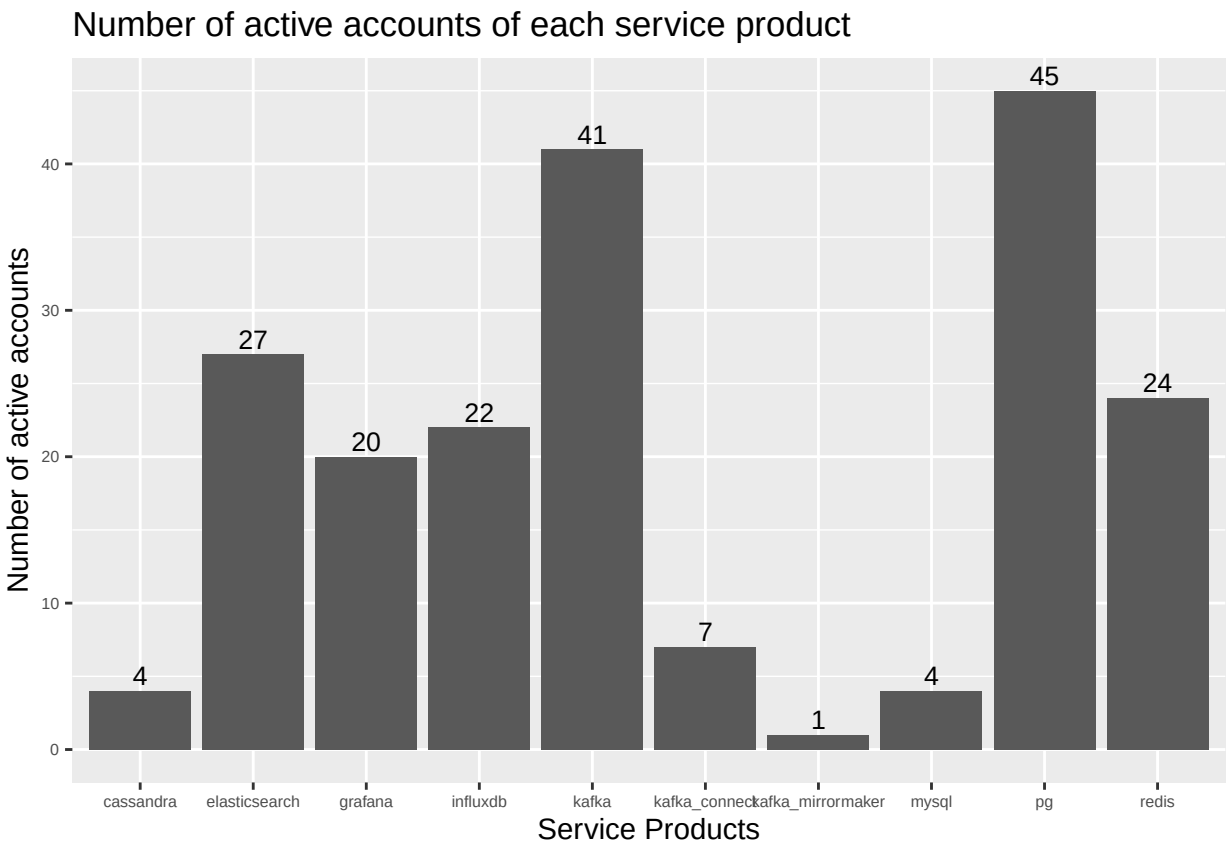


Calculate the number of active accounts of each service products

```
services <- gather(subset(account_metrics, select = c(7:18)) ,key = "key", value = "value", 1:12)
services %>%
  filter(value > 0) %>%
  group_by(key) %>%
```

```
  summarise(number_of_accounts = n()) %>%
  ggplot(aes(x=key, y=number_of_accounts)) +
  geom_bar(stat="identity") +
  theme(axis.text = element_text(size = 6)) +
  geom_text(aes(label=number_of_accounts), vjust=-0.3, size=3.5) +
  ggtitle("Number of active accounts of each service product") +
  labs(y = "Number of active accounts", x= "Service Products")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Dealing with JSON columns: calculate the number of active accounts of each cloud and the number of each active cloud account of each region

Handling 'clouds_in_use_names' column that contain JSON keys:

```
#Load 'jsonlite' library
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
```

```
## The following object is masked from 'package:purrr':
##
##     flatten
```

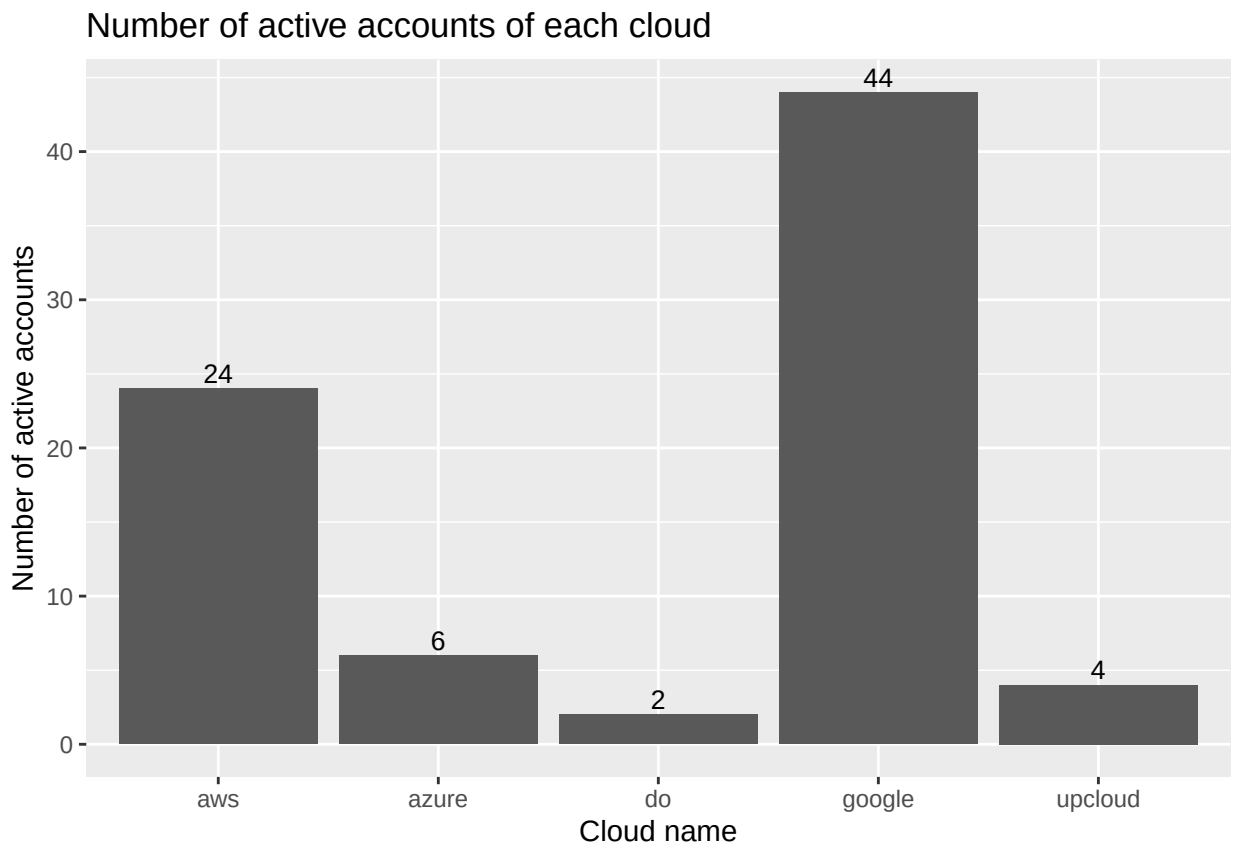
```
#Parse the data from JSON key
cloudjson <- account_metrics$clouds_in_use_names
cloud_name <- data.frame(account_metrics$account_id, account_metrics$clouds_in_use_names)
cloud_name <- cloud_name %>%
                mutate(cloudjson  = map(cloudjson , ~ fromJSON(.) %>% as.data.frame())) %>%
                unnest(cloudjson ) %>%
                rename(
                  account_id = account_metrics.account_id,
                  cloud_name = v)
cloud_name <- cloud_name[,-2]
head(cloud_name)
```

```
## # A tibble: 6 x 2
##   account_id         cloud_name
##   <chr>              <chr>
## 1 0015J000002mYOPQA2 <NA>
## 2 0015J000002mfDsQAI <NA>
## 3 0015J000002mYTxQAM <NA>
## 4 0015J000008eNF0QAM <NA>
## 5 0015J000008eNayQAE <NA>
## 6 0015J000002mYDUQA2 <NA>
```

Calculating the number of active account users of each cloud:

```
cloud_name %>%
  group_by(cloud_name) %>%
  summarise(number_of_users = n_distinct(account_id))%>%
  drop_na(cloud_name) %>%
  ggplot(aes(x = cloud_name, y = number_of_users)) +
  geom_bar(stat ="identity")+
  geom_text(aes(label=number_of_users), vjust=-0.3, size=3.5)+
  ggtitle("Number of active accounts of each cloud") +
  labs(y = "Number of active accounts", x = "Cloud name")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Handling 'cloud_regions_in_use_names' column that contain JSON keys:

```
regionjson <- account_metrics$cloud_regions_in_use_names
cloud_region <- data.frame(account_metrics$account_id, account_metrics$cloud_regions_in_use_names)
cloud_region <- cloud_region %>%
  mutate(regionjson = map(regionjson, ~ fromJSON(.) %>% as.data.frame())) %>%
  unnest(regionjson) %>%
  rename(
    account_id = account_metrics.account_id,
    region_name = v)
cloud_region <- cloud_region[,-2]
head(cloud_region)
```

```
## # A tibble: 6 x 2
##   account_id         region_name
##   <chr>              <chr>
## 1 0015J000002mYOPQA2 <NA>
## 2 0015J000002mfDsQAI <NA>
## 3 0015J000002mYTxQAM <NA>
## 4 0015J000008eNF0QAM <NA>
## 5 0015J000008eNayQAE <NA>
## 6 0015J000002mYDUQA2 <NA>
```

Calculating the number of active cloud account users of each region:

```
region_users_count <- cloud_region %>%
                        group_by(region_name) %>%
                        summarise(number_of_accounts = n_distinct(account_id))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

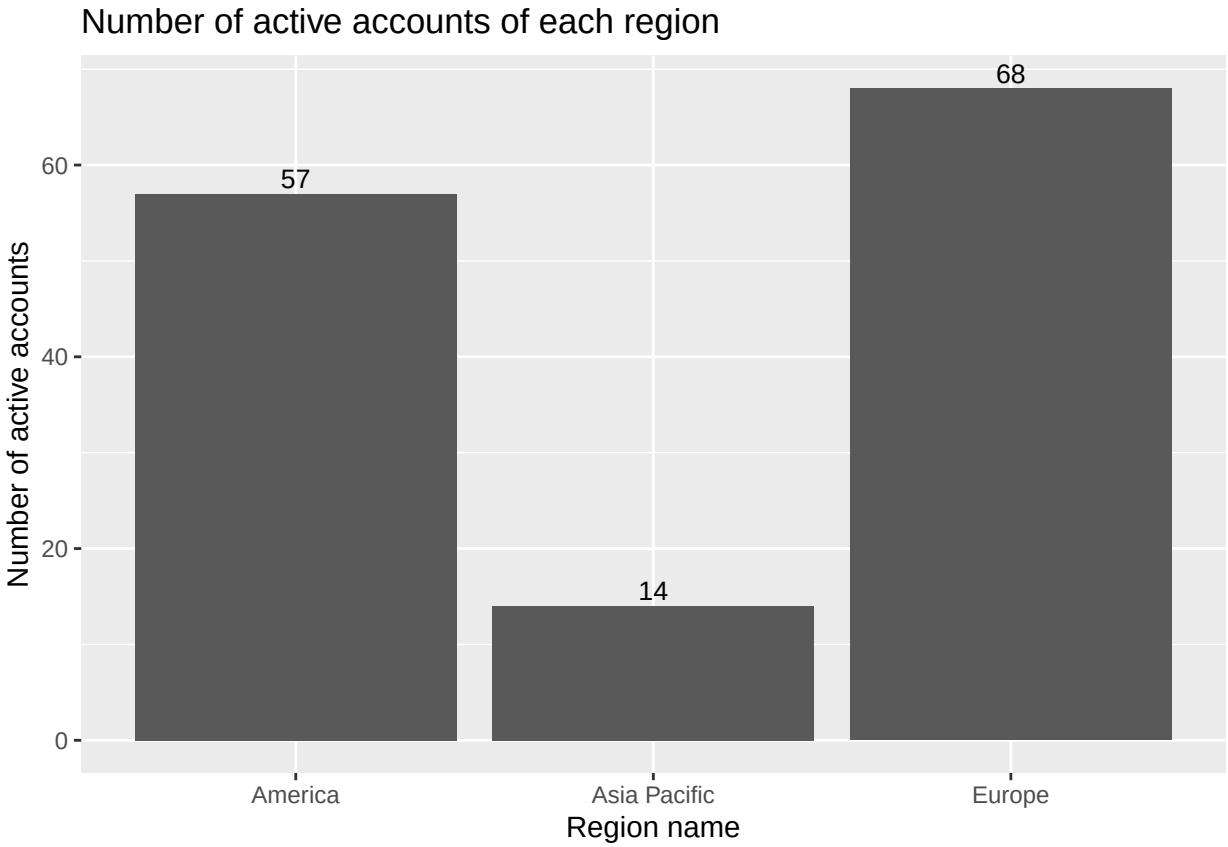
```
region_users_count
```

```
## # A tibble: 34 x 2
##    region_name     number_of_accounts
##    <chr>                        <int>
##  1 ap-northeast-1                   1
##  2 ap-south-1                       1
##  3 asia-east1                       4
##  4 asia-northeast1                  1
##  5 asia-southeast1                  5
##  6 asia-southeast2                  2
##  7 Central US                       1
##  8 East US                          2
##  9 eu-central-1                     6
## 10 eu-north-1                       1
## # ... with 24 more rows
```

There are 34 different regions that I could get from JSON key and a lot of them are repitition, inconsistent way of naming regions. I decided to group them together and have only 3 big region groups: America, Asia Pacific, and Europe.

```
region_users_count %>%
  mutate(regions = case_when(
    region_name %in% c("ap-northeast-1","ap-south-1","asia-east1","asia-northeast1","asia-southeast1","a
    region_name %in% c("Central US","East US","northamerica-northeast1","nyc","sfo","southamerica-east1
    region_name %in% c("eu-central-1","eu-north-1","eu-west-1","eu-west-2","europe-north1","europe-west
    TRUE ~ NA_character_
  )) %>%
  drop_na(region_name) %>%
  group_by(regions) %>%
  summarise(number_of_accounts = sum(number_of_accounts))%>%
  ggplot(aes(x = regions, y = number_of_accounts)) +
  geom_bar(stat ="identity")+
  geom_text(aes(label=number_of_accounts), vjust=-0.3, size=3.5)+
  ggtitle("Number of active accounts of each region") +
  labs(y = "Number of active accounts", x = "Region name")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Calculate number of active accounts categorized by the percentage of their active projects

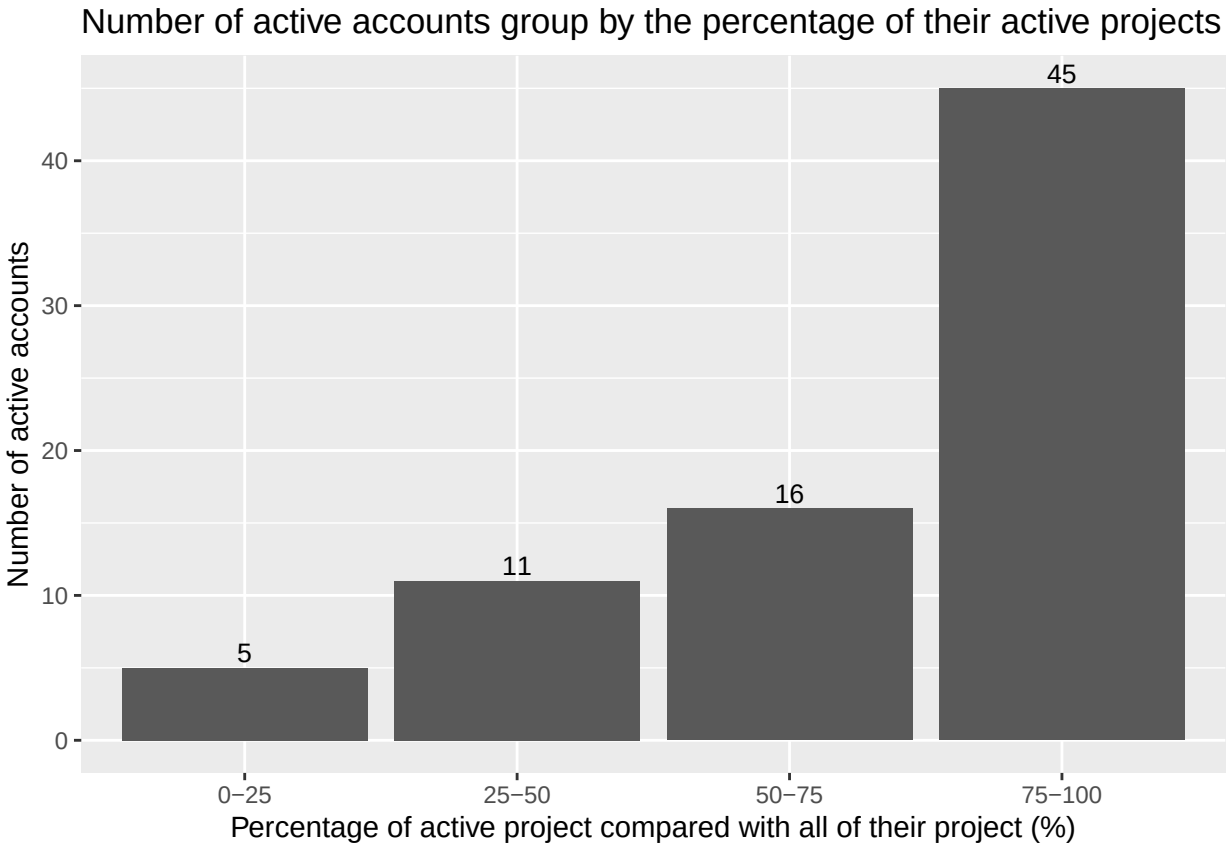
The percentage of an account's active project is calculated by dividing their active projects by their all projects.

```
account_metrics <- account_metrics %>%
                    mutate(percentage_of_active_project = active_projects/all_projects * 100) %>%
                    mutate(percentage_of_active_service = active_services/all_services * 100)

account_metrics$groups_percentage_active_project <- cut(account_metrics$percentage_of_active_project, c
levels(account_metrics$groups_percentage_active_project) = c("0-25", "25-50", "50-75", "75-100")

account_metrics %>%
  group_by(groups_percentage_active_project)%>%
  drop_na(groups_percentage_active_project) %>%
  summarise(number_of_accounts = n_distinct(account_id)) %>%
  ggplot(aes(x=groups_percentage_active_project, y=number_of_accounts)) +
  geom_bar(stat="identity") +
  geom_text(aes(label=number_of_accounts), vjust=-0.3, size=3.5) +
  ggtitle("Number of active accounts group by the percentage of their active projects") +
  labs(y = "Number of active accounts", x= "Percentage of active project compared with all of their pro
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



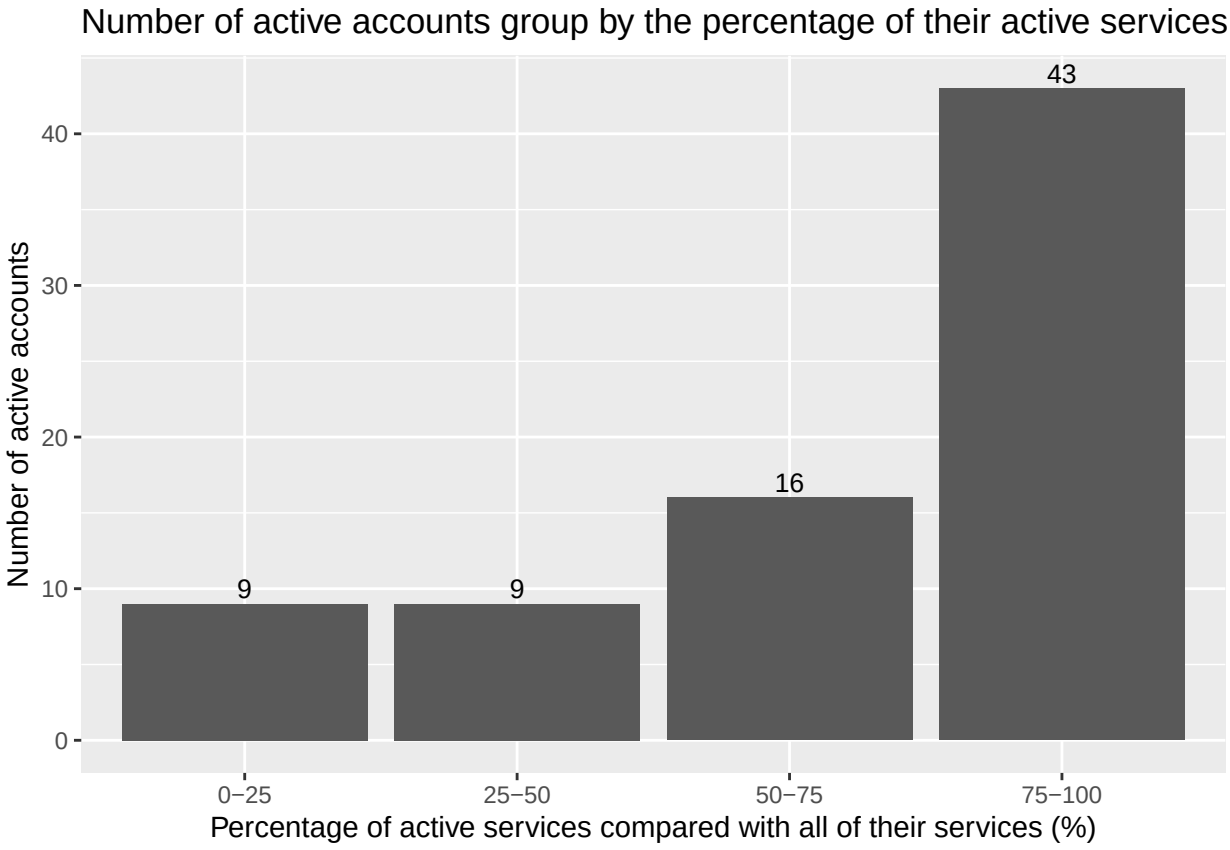
Calculate number of active accounts categorized by the percentage of their active services

The percentage of an account's active services is calculated by dividing their active services by their all services.

```
account_metrics$groups_percentage_active_service <- cut(account_metrics$percentage_of_active_service, c
levels(account_metrics$groups_percentage_active_service) = c("0-25", "25-50", "50-75", "75-100")

account_metrics %>%
  group_by(groups_percentage_active_service)%>%
  drop_na(groups_percentage_active_service) %>%
  summarise(number_of_accounts = n_distinct(account_id)) %>%
  ggplot(aes(x=groups_percentage_active_service, y=number_of_accounts)) +
  geom_bar(stat="identity") +
  geom_text(aes(label=number_of_accounts), vjust=-0.3, size=3.5) +
  ggtitle("Number of active accounts group by the percentage of their active services") +
  labs(y = "Number of active accounts", x= "Percentage of active services compared with all of their se
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Merging two datasets

Inner Join two datasets: use the 'accounts' dataset before exclude the observations on 12th of March, 2020

```
#Re-import 'accounts' dataset
accounts <- read.csv('accounts.csv')

accounts$account_created_time <- as.POSIXct(accounts$account_created_time, format = "%Y-%m-%d %H:%M:%S"
accounts$is_deleted <-  as.logical(accounts$is_deleted)
accounts$main_industry <- as.factor(accounts$main_industry)
accounts$sub_industry <- as.factor(accounts$sub_industry)
accounts$type <- as.factor(accounts$type)
accounts$support_tier <- as.factor(accounts$support_tier)
accounts <- accounts[,-c(4,5)]
accounts$month_yr <- as.factor(format(accounts$account_created_time, '%Y-%m'))
accounts$hour <- as.factor(format(accounts$account_created_time,'%H'))
accounts$account_created_date <- as.factor(format(accounts$account_created_time, '%Y-%m-%d'))

#Inner Join 2 datasets:
innerjoin <- merge(account_metrics,accounts, by = 'account_id')
nrow(innerjoin)
```

```
## [1] 498
```

There are 498 observations that match the 'account_id' between 2 dataset

Check if there is any observations in the innerjoin data that have active projects or active services

```
active_join <- innerjoin %>%
                filter(active_projects > 0 | active_services > 0)
active_join
```

```
##  [1] account_id                       all_projects
##  [3] active_projects                  all_services
##  [5] active_services                  stopped_services
##  [7] kafka                            pg
##  [9] elasticsearch                    redis
## [11] mysql                            grafana
## [13] influxdb                         m3
## [15] ksql                             kafka_connect
## [17] cassandra                        kafka_mirrormaker
## [19] clouds_in_use                    clouds_in_use_names
## [21] cloud_regions_in_use             cloud_regions_in_use_names
## [23] gcp_projects                     gcp_services
## [25] aws_projects                     aws_services
## [27] azure_projects                   azure_services
## [29] do_projects                      do_services
## [31] upcloud_projects                 upcloud_services
## [33] custom_do_projects               custom_do_services
## [35] packet_projects                  packet_services
## [37] latest_service_type              latest_service_created
## [39] month_yr.x                       latest_service_created_year
## [41] check1                           check2
## [43] check3                           percentage_of_active_project
## [45] percentage_of_active_service     groups_percentage_active_project
## [47] groups_percentage_active_service account_created_time
## [49] is_deleted                       main_industry
## [51] sub_industry                     type
## [53] support_tier                     month_yr.y
## [55] hour                             account_created_date
## <0 rows> (or 0-length row.names)
```

There is no accounts in the inner join table that have either active project or active services.

Latest service created time vs. account created time

The records with accounts created time after the latest service created time

```
#Filter out the accounts created in 12th of March 2020
accounts <- accounts %>%
  filter(!(account_created_date == '2020-03-12'))
innerjoin <- merge(account_metrics,accounts, by = 'account_id')
innerjoin %>%
  filter(latest_service_created < account_created_time)%>%
  nrow()
```

```
## [1] 141
```

There are 141 records of accounts that have account created after the latest service created.

The records with accounts created time after the latest service created time

```
innerjoin %>%
  filter(latest_service_created > account_created_time)%>%
  nrow()
```

```
## [1] 30
```

There are 30 records of accounts that have account created before the latest service created.

Thanks for reading