Independent Project.

Simon Mmari

2022-03-25

Define the question

The question is making conclusion on who is likely to click on the ads or not.

Metric for success

In order to work on the above problem, you need to do the following:

- Define the question, the metric for success, the context, experimental design taken and the appropriateness of the available data to answer the given question.
- Find and deal with outliers, anomalies, and missing data within the dataset.
- Perform univariate and bivariate analysis.
- From your insights provide a conclusion and recommendation.

Data Understanding (the context)

A Kenyan entrepreneur has created an online cryptography course and would want to advertise it on her blog. She currently targets audiences originating from various countries. In the past, she ran ads to advertise a related course on the same blog and collected data in the process. She would now like to employ your services as a Data Science Consultant to help her identify which individuals are most likely to click on her ads.

In order to work on the above problem, you need to do the following:

- Define the question, the metric for success, the context, experimental design taken and the appropriateness of the available data to answer the given question.
- Find and deal with outliers, anomalies, and missing data within the dataset.
- Perform univariate and bivariate analysis.
- From your insights provide a conclusion and recommendation.

Experimental design

1. Import the data to R
2. Perform data exploration
3. Define metrics for success
4. Perform Univariate and Bivariate data Analysis
5. Provide conclusion

Loading Dataset

```
ad <- read.csv("http://bit.ly/IPAdvertisingData")
```

```
head(ad)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 1                    68.95  35    61833.90               256.09
## 2                    80.23  31    68441.85               193.77
## 3                    69.47  26    59785.94               236.50
## 4                    74.15  29    54806.18               245.89
## 5                    68.37  35    73889.99               225.58
## 6                    59.99  23    59761.56               226.74
##                           Ad.Topic.Line           City Male    Country
## 1    Cloned 5thgeneration orchestration    Wrightburgh    0    Tunisia
## 2    Monitored national standardization      West Jodi    1      Nauru
## 3      Organic bottom-line service-desk       Davidton    0 San Marino
## 4 Triple-buffered reciprocal time-frame West Terrifurt    1      Italy
## 5         Robust logistical utilization   South Manuel    0    Iceland
## 6       Sharable client-driven software      Jamieberg    1     Norway
##             Timestamp Clicked.on.Ad
## 1 2016-03-27 00:53:11             0
## 2 2016-04-04 01:39:02             0
## 3 2016-03-13 20:35:42             0
## 4 2016-01-10 02:31:19             0
## 5 2016-06-03 03:36:18             0
## 6 2016-05-19 14:30:17             0
```

```
tail(ad)
```

```
##      Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage
## 995                     43.70  28    63126.96               173.01
## 996                     72.97  30    71384.57               208.58
## 997                     51.30  45    67782.17               134.42
## 998                     51.63  51    42415.72               120.37
## 999                     55.55  19    41920.79               187.95
## 1000                    45.01  26    29875.80               178.35
##                             Ad.Topic.Line          City Male
## 995         Front-line bifurcated ability  Nicholasland    0
## 996         Fundamental modular algorithm     Duffystad    1
## 997       Grass-roots cohesive monitoring   New Darlene    1
## 998          Expanded intangible solution South Jessica    1
```

```
## 999  Proactive bandwidth-monitored policy   West Steven    0
## 1000      Virtual 5thgeneration emulation   Ronniemouth    0
##                      Country           Timestamp Clicked.on.Ad
## 995                  Mayotte 2016-04-04 03:57:48             1
## 996                  Lebanon 2016-02-11 21:49:00             1
## 997  Bosnia and Herzegovina 2016-04-22 02:07:01             1
## 998                 Mongolia 2016-02-01 17:24:57             1
## 999                Guatemala 2016-03-24 02:35:54             0
## 1000                  Brazil 2016-06-03 21:43:21             1
```

Checking dataset.

```
# Finding the Shape of the dataset
dim(ad)
```

```
## [1] 1000   10
```

```
# Finding the datatypes of the dataset
str(ad)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Daily.Time.Spent.on.Site: num  69 80.2 69.5 74.2 68.4 ...
##  $ Age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ Area.Income             : num  61834 68442 59786 54806 73890 ...
##  $ Daily.Internet.Usage    : num  256 194 236 246 226 ...
##  $ Ad.Topic.Line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ Male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ Timestamp               : chr  "2016-03-27 00:53:11" "2016-04-04 01:39:02" "2016-03-13 20:35:42"
##  $ Clicked.on.Ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

Data cleaning

```
# checking for missing Data
colSums(is.na(ad))
```

```
## Daily.Time.Spent.on.Site                      Age             Area.Income
##                        0                        0                        0
##     Daily.Internet.Usage            Ad.Topic.Line                     City
##                        0                        0                        0
##                     Male                  Country                Timestamp
##                        0                        0                        0
##            Clicked.on.Ad
##                        0
```

```
# Check for duplicated data in the ad
ad1 <- ad[duplicated(ad),]
ad1
```

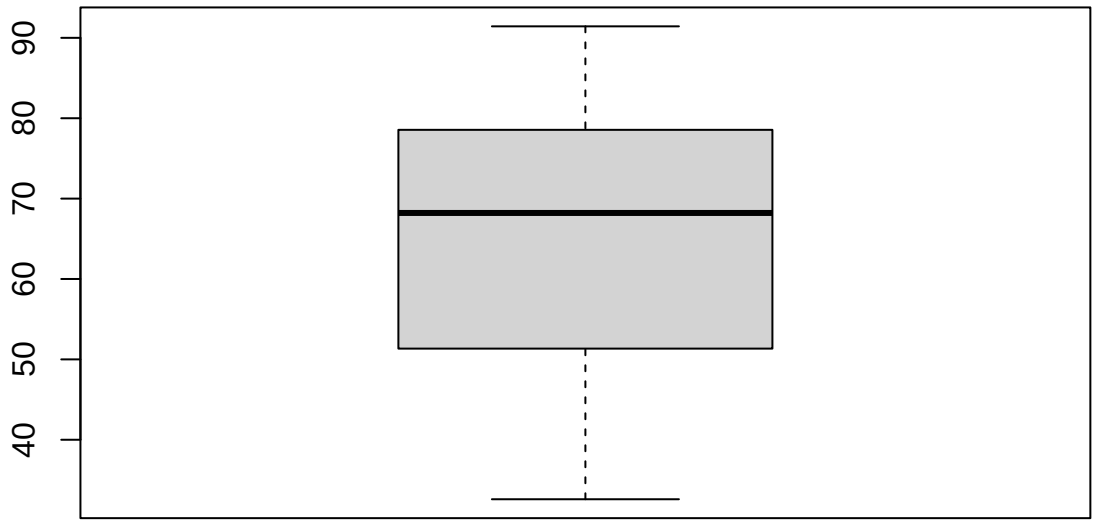
```
##  [1] Daily.Time.Spent.on.Site Age                      Area.Income
##  [4] Daily.Internet.Usage     Ad.Topic.Line            City
##  [7] Male                     Country                  Timestamp
## [10] Clicked.on.Ad
## <0 rows> (or 0-length row.names)
```

Univariate Analysis.

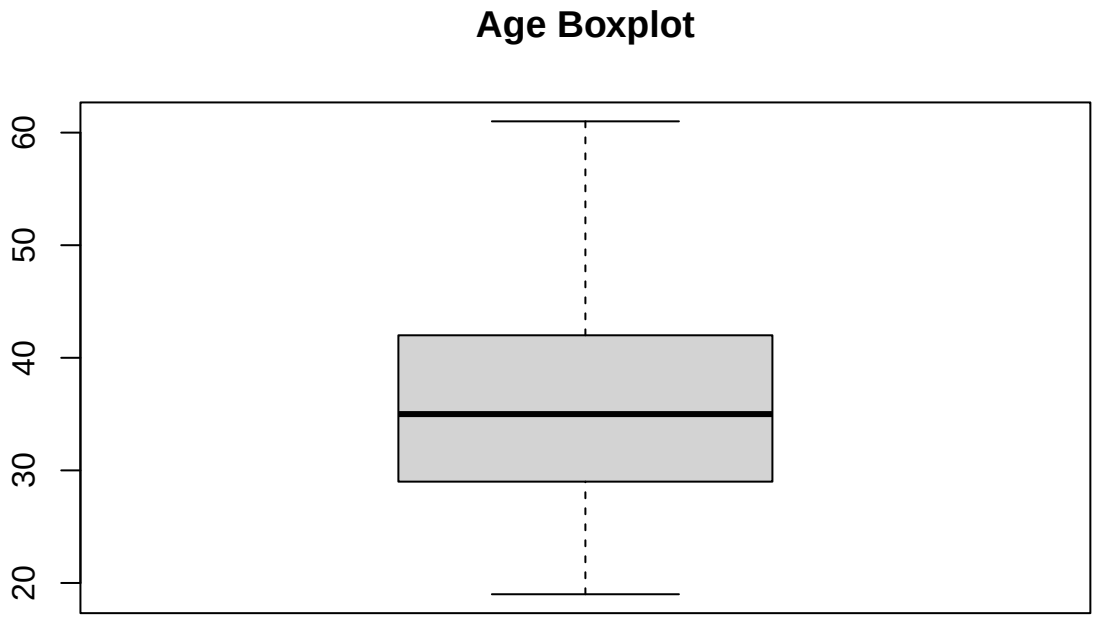
```
str(ad)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Daily.Time.Spent.on.Site: num  69 80.2 69.5 74.2 68.4 ...
##  $ Age                     : int  35 31 26 29 35 23 33 48 30 20 ...
##  $ Area.Income             : num  61834 68442 59786 54806 73890 ...
##  $ Daily.Internet.Usage    : num  256 194 236 246 226 ...
##  $ Ad.Topic.Line           : chr  "Cloned 5thgeneration orchestration" "Monitored national standardi
##  $ City                    : chr  "Wrightburgh" "West Jodi" "Davidton" "West Terrifurt" ...
##  $ Male                    : int  0 1 0 1 0 1 0 1 1 1 ...
##  $ Country                 : chr  "Tunisia" "Nauru" "San Marino" "Italy" ...
##  $ Timestamp               : chr  "2016-03-27 00:53:11" "2016-04-04 01:39:02" "2016-03-13 20:35:42"
##  $ Clicked.on.Ad           : int  0 0 0 0 0 0 0 1 0 0 ...
```

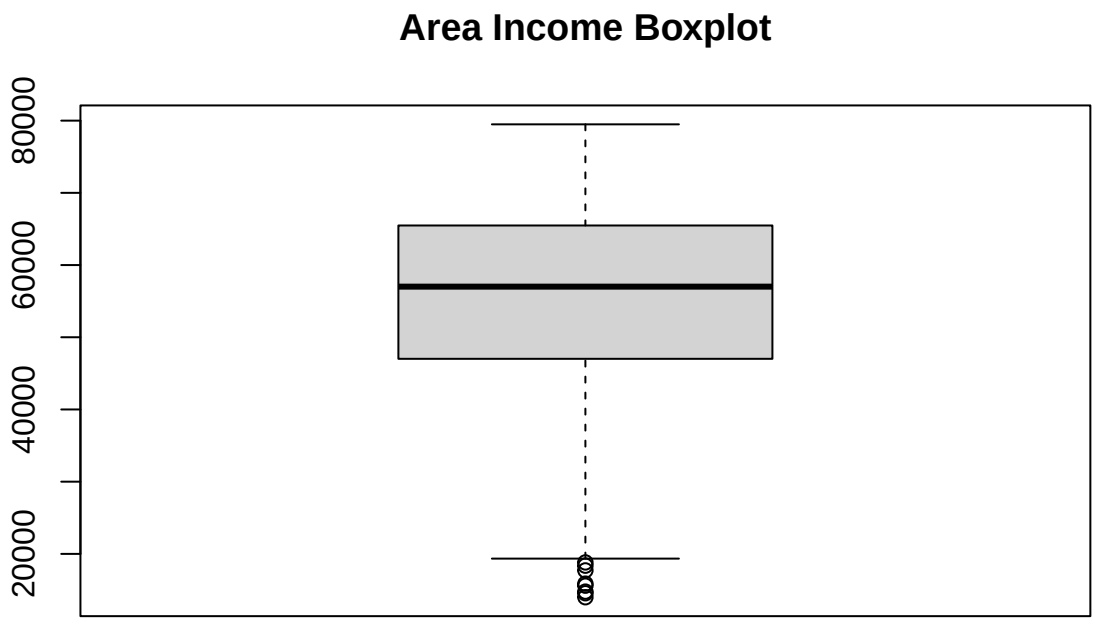
```
boxplot(ad$Daily.Time.Spent.on.Site, main = 'Daily Time Spent on-site')
```



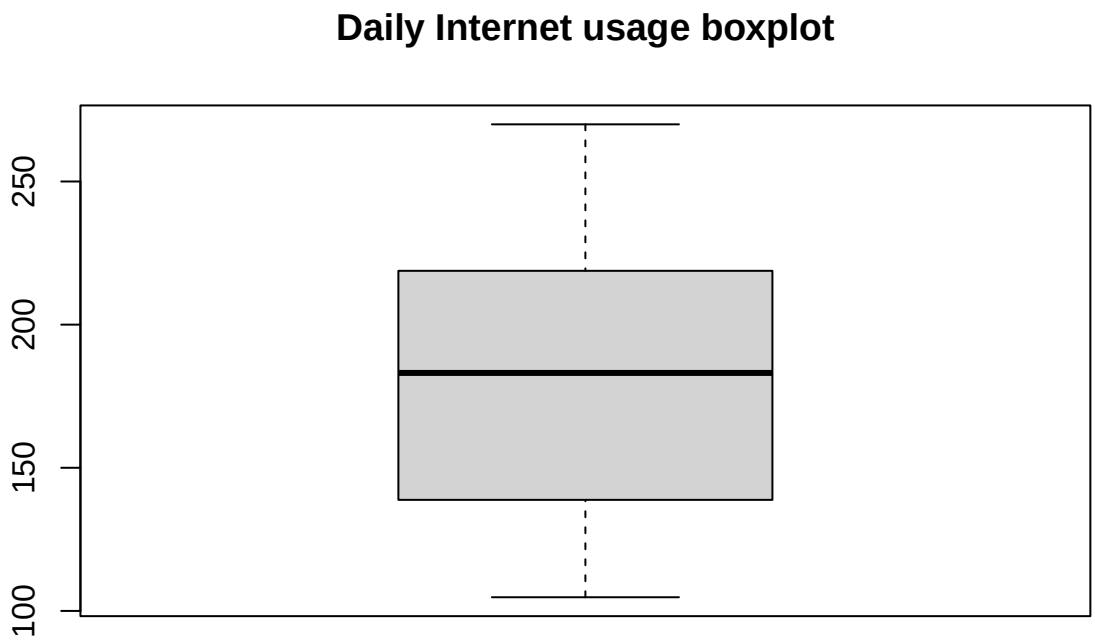

```
boxplot(ad$Age, main = 'Age Boxplot')
```




```
boxplot(ad$Area.Income, main = 'Area Income Boxplot')
```

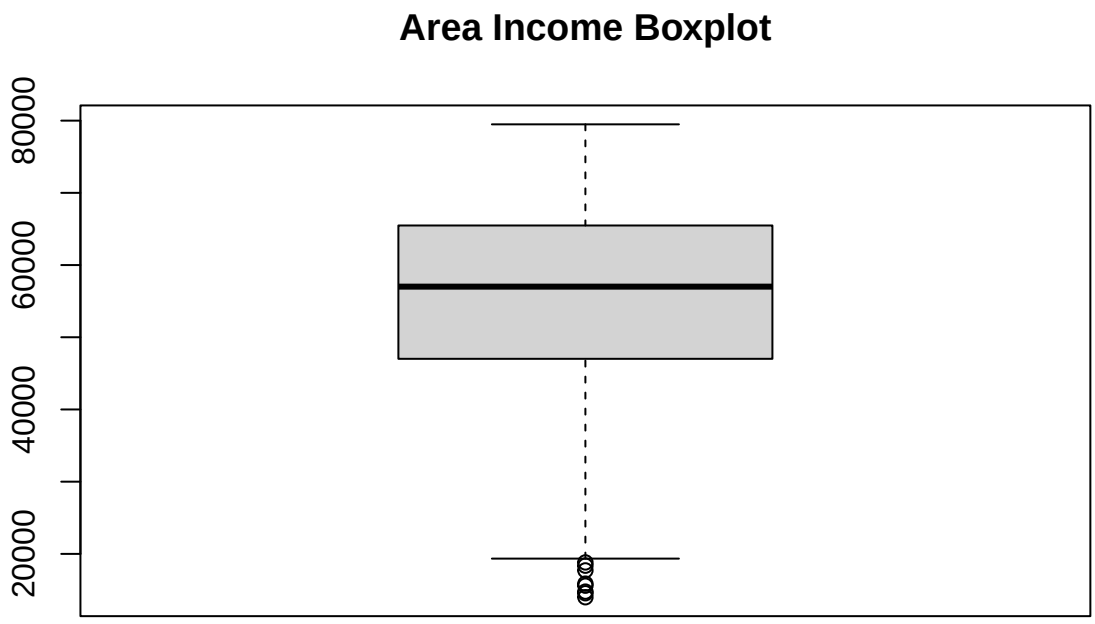



```
boxplot(ad$Daily.Internet.Usage, main = 'Daily Internet usage boxplot')
```



print out the outliers

```
boxplot(ad$Area.Income, main = 'Area Income Boxplot')$out
```




```
## [1] 17709.98 18819.34 15598.29 15879.10 14548.06 13996.50 14775.50 18368.57
```

```
ad[['Timestamp']] <- as.POSIXct(ad[['Timestamp']],
                                format = "%Y-%m-%d %H:%M:%S")
```

Numerical columns.

```
summary(ad)
```

```
##  Daily.Time.Spent.on.Site      Age          Area.Income    Daily.Internet.Usage
##  Min.   :32.60            Min.   :19.00   Min.   :13996   Min.   :104.8
##  1st Qu.:51.36            1st Qu.:29.00   1st Qu.:47032   1st Qu.:138.8
##  Median :68.22            Median :35.00   Median :57012   Median :183.1
##  Mean   :65.00            Mean   :36.01   Mean   :55000   Mean   :180.0
##  3rd Qu.:78.55            3rd Qu.:42.00   3rd Qu.:65471   3rd Qu.:218.8
##  Max.   :91.43            Max.   :61.00   Max.   :79485   Max.   :270.0
##  Ad.Topic.Line          City                Male         Country
##  Length:1000        Length:1000        Min.   :0.000   Length:1000
##  Class :character   Class :character   1st Qu.:0.000   Class :character
##  Mode  :character   Mode  :character   Median :0.000   Mode  :character
##                                        Mean   :0.481
##                                        3rd Qu.:1.000
##                                        Max.   :1.000
```

```
##    Timestamp                     Clicked.on.Ad
##  Min.   :2016-01-01 02:52:10   Min.   :0.0
##  1st Qu.:2016-02-18 02:55:42   1st Qu.:0.0
##  Median :2016-04-07 17:27:29   Median :0.5
##  Mean   :2016-04-10 10:34:06   Mean   :0.5
##  3rd Qu.:2016-05-31 03:18:14   3rd Qu.:1.0
##  Max.   :2016-07-24 00:22:16   Max.   :1.0
```

There are outliers that do not look like they are in the extreme. There are areas where poverty is prevelant in such areas the total income could be that small.

Mean.

```
mean.age <- mean(ad$Age)
mean.age
```

```
## [1] 36.009
```

Function to get the mode.

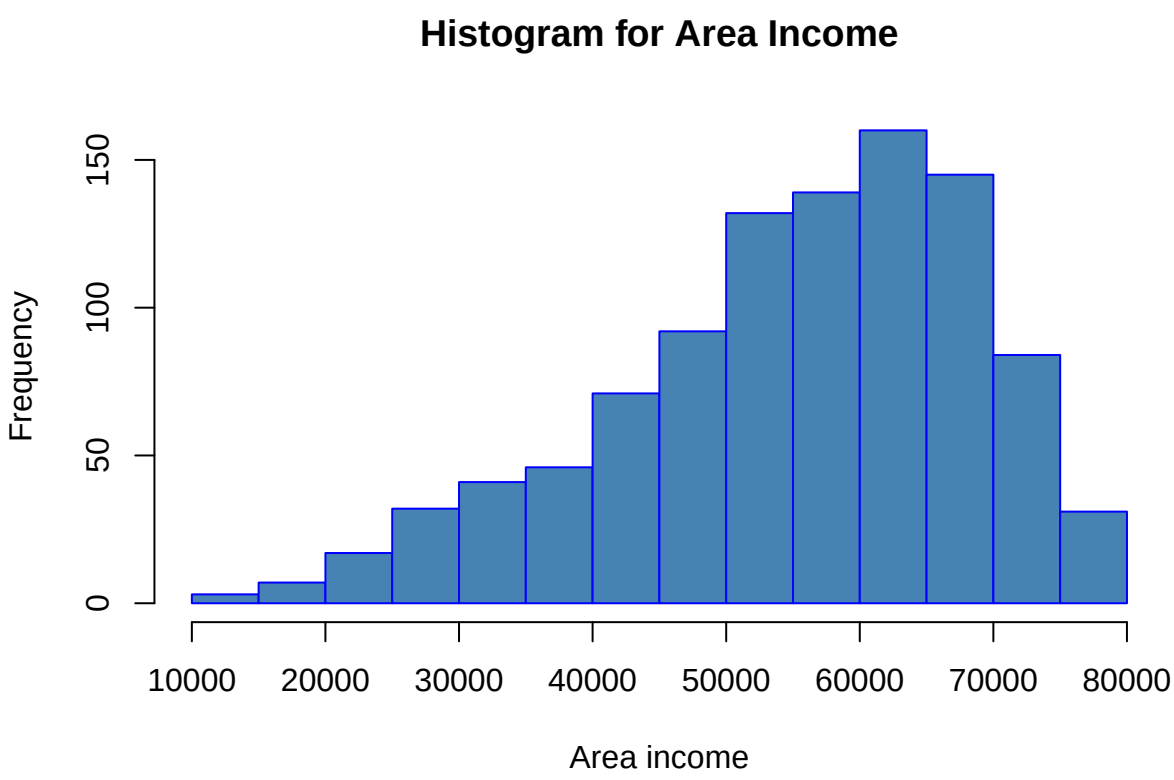
```
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

##Area income

```
mean.areaincome <- mean(ad$Area.Income)
mean.areaincome
```

```
## [1] 55000
```

```
hist(ad$Area.Income,
     main="Histogram for Area Income",
     xlab="Area income",
     border="blue",
     col="steelblue",)
```



Daily time spent on site

```
mean.dtsos <- mean(ad$Daily.Time.Spent.on.Site)
mean.dtsos
```

```
## [1] 65.0002
```

```
uniq_clickers <- unique(ad$Clicked.on.Ad,)
length(uniq_clickers)
```

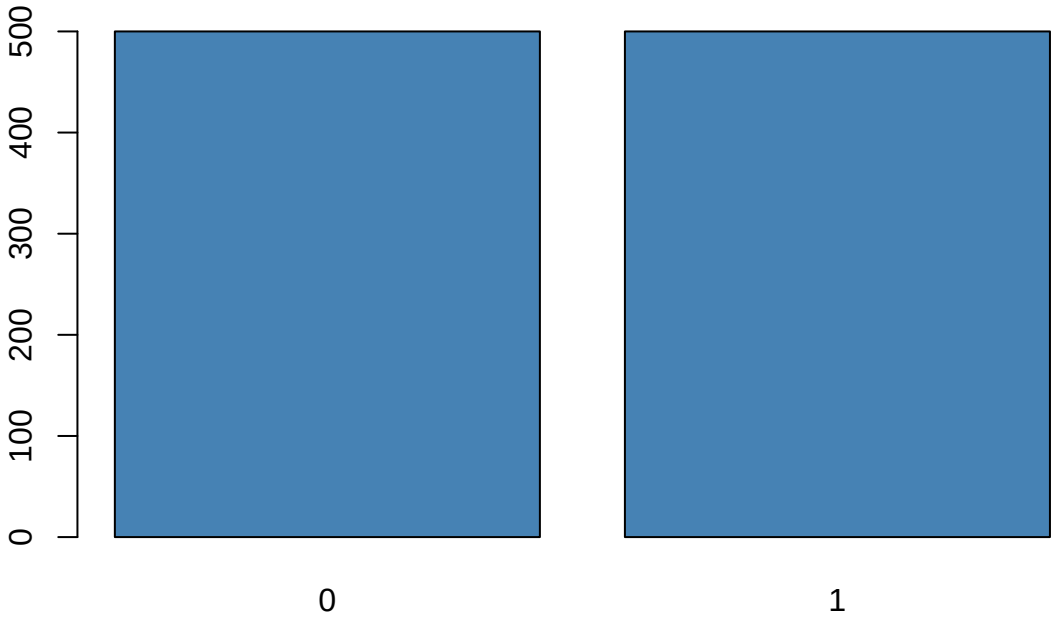
Clicked.on.Ad

```
## [1] 2
```

There are two categories of the people who clicked on ads

Let us plot the frequency of each

```
clickers <- ad$Clicked.on.Ad
clickers_frequency <- table (clickers)
barplot(clickers_frequency, col = "steelblue")
```



There are 500 people who clicked on ads and another 500 did not click on the ads.

Categorical Columns

####Ad.Topic.line

There are 1000 unique topic lines meaning it would be impossible to get a good visualization.

```
uniq_country <- unique(ad$Country)
length(uniq_country)
```

Country

```
## [1] 237
```

There are 237 unique countries.

```
library(sf)
```

```
## Linking to GEOS 3.9.1, GDAL 3.2.1, PROJ 7.2.1; sf_use_s2() is TRUE
```

```
library(raster)
```

```
## Loading required package: sp
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:raster':
##
##     intersect, select, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library
```

```
## function (package, help, pos = 2, lib.loc = NULL, character.only = FALSE,
##     logical.return = FALSE, warn.conflicts, quietly = FALSE,
##     verbose = getOption("verbose"), mask.ok, exclude, include.only,
##     attach.required = missing(include.only))
## {
##     conf.ctrl <- getOption("conflicts.policy")
##     if (is.character(conf.ctrl))
##         conf.ctrl <- switch(conf.ctrl, strict = list(error = TRUE,
##             warn = FALSE), depends.ok = list(error = TRUE, generics.ok = TRUE,
##             can.mask = c("base", "methods", "utils", "grDevices",
##                 "graphics", "stats"), depends.ok = TRUE), warning(gettextf("unknown conflict policy:
##             sQuote(conf.ctrl)), call. = FALSE, domain = NA))
##     if (!is.list(conf.ctrl))
##         conf.ctrl <- NULL
##     stopOnConflict <- isTRUE(conf.ctrl$error)
##     if (missing(warn.conflicts))
##         warn.conflicts <- if (isFALSE(conf.ctrl$warn))
##             FALSE
##         else TRUE
##     if ((!missing(include.only)) && (!missing(exclude)))
##         stop(gettext("only one of 'include.only' and 'exclude' can be used"),
##             call. = FALSE, domain = NA)
```

```
##     testRversion <- function(pkgInfo, pkgname, pkgpath) {
##         if (is.null(built <- pkgInfo$Built))
##             stop(gettextf("package %s has not been installed properly\n",
##                 sQuote(pkgname)), call. = FALSE, domain = NA)
##         R_version_built_under <- as.numeric_version(built$R)
##         if (R_version_built_under < "3.0.0")
##             stop(gettextf("package %s was built before R 3.0.0: please re-install it",
##                 sQuote(pkgname)), call. = FALSE, domain = NA)
##         current <- getRversion()
##         if (length(Rdeps <- pkgInfo$Rdepends2)) {
##             for (dep in Rdeps) if (length(dep) > 1L) {
##                 target <- dep$version
##                 res <- do.call(dep$op, if (is.character(target))
##                   list(as.numeric(R.version[["svn rev"]]), as.numeric(sub("^r",
##                     "", target)))
##                 else list(current, as.numeric_version(target)))
##                 if (!res)
##                   stop(gettextf("This is R %s, package %s needs %s %s",
##                     current, sQuote(pkgname), dep$op, target),
##                     call. = FALSE, domain = NA)
##             }
##         }
##         if (R_version_built_under > current)
##             warning(gettextf("package %s was built under R version %s",
##                 sQuote(pkgname), as.character(built$R)), call. = FALSE,
##                 domain = NA)
##         platform <- built$Platform
##         r_arch <- .Platform$r_arch
##         if (.Platform$OS.type == "unix") {
##         }
##         else {
##             if (nzchar(platform) && !grepl("mingw", platform))
##                 stop(gettextf("package %s was built for %s",
##                   sQuote(pkgname), platform), call. = FALSE,
##                   domain = NA)
##         }
##         if (nzchar(r_arch) && file.exists(file.path(pkgpath,
##             "libs")) && !file.exists(file.path(pkgpath, "libs",
##             r_arch)))
##             stop(gettextf("package %s is not installed for 'arch = %s'",
##                 sQuote(pkgname), r_arch), call. = FALSE, domain = NA)
##     }
##     checkNoGenerics <- function(env, pkg) {
##         nenv <- env
##         ns <- .getNamespace(as.name(pkg))
##         if (!is.null(ns))
##             nenv <- asNamespace(ns)
##         if (exists(".noGenerics", envir = nenv, inherits = FALSE))
##             TRUE
##         else {
##             !any(startsWith(names(env), ".__T"))
##         }
##     }
##     checkConflicts <- function(package, pkgname, pkgpath, nogenerics,
```

```
##         env) {
##         dont.mind <- c("last.dump", "last.warning", ".Last.value",
##             ".Random.seed", ".Last.lib", ".onDetach", ".packageName",
##             ".noGenerics", ".required", ".no_S3_generics", ".Depends",
##             ".requireCachedGenerics")
##         sp <- search()
##         lib.pos <- which(sp == pkgname)
##         ob <- names(as.environment(lib.pos))
##         if (!nogenerics) {
##             these <- ob[startsWith(ob, ".__T__")]
##             gen <- gsub(".__T__(.*):([^:]+)", "\\1", these)
##             from <- gsub(".__T__(.*):([^:]+)", "\\2", these)
##             gen <- gen[from != package]
##             ob <- ob[!(ob %in% gen)]
##         }
##         ipos <- seq_along(sp)[-c(lib.pos, match(c("Autoloads",
##             "CheckExEnv"), sp, 0L))]
##         cpos <- NULL
##         conflicts <- vector("list", 0)
##         for (i in ipos) {
##             obj.same <- match(names(as.environment(i)), ob, nomatch = 0L)
##             if (any(obj.same > 0L)) {
##                 same <- ob[obj.same]
##                 same <- same[!(same %in% dont.mind)]
##                 Classobjs <- which(startsWith(same, ".__"))
##                 if (length(Classobjs))
##                   same <- same[-Classobjs]
##                 same.isFn <- function(where) vapply(same, exists,
##                   NA, where = where, mode = "function", inherits = FALSE)
##                 same <- same[same.isFn(i) == same.isFn(lib.pos)]
##                 not.Ident <- function(ch, TRAFO = identity, ...) vapply(ch,
##                   function(.) !identical(TRAFO(get(., i)), TRAFO(get(.,
##                     lib.pos)), ...), NA)
##                 if (length(same))
##                   same <- same[not.Ident(same)]
##                 if (length(same) && identical(sp[i], "package:base"))
##                   same <- same[not.Ident(same, ignore.environment = TRUE)]
##                 if (length(same)) {
##                   conflicts[[sp[i]]] <- same
##                   cpos[sp[i]] <- i
##                 }
##             }
##         }
##         if (length(conflicts)) {
##             if (stopOnConflict) {
##                 emsg <- ""
##                 pkg <- names(conflicts)
##                 notOK <- vector("list", 0)
##                 for (i in seq_along(conflicts)) {
##                   pkgname <- sub("^package:", "", pkg[i])
##                   if (pkgname %in% canMaskEnv$canMask)
##                     next
##                   same <- conflicts[[i]]
##                   if (is.list(mask.ok))
```

```
##                     myMaskOK <- mask.ok[[pkgname]]
##                   else myMaskOK <- mask.ok
##                   if (isTRUE(myMaskOK))
##                     same <- NULL
##                   else if (is.character(myMaskOK))
##                     same <- setdiff(same, myMaskOK)
##                   if (length(same)) {
##                     notOK[[pkg[i]]] <- same
##                     msg <- .maskedMsg(sort(same), pkg = sQuote(pkg[i]),
##                       by = cpos[i] < lib.pos)
##                     emsg <- paste(emsg, msg, sep = "\n")
##                   }
##                 }
##                 if (length(notOK)) {
##                   msg <- gettextf("Conflicts attaching package %s:\n%s",
##                     sQuote(package), emsg)
##                   stop(errorCondition(msg, package = package,
##                     conflicts = conflicts, class = "packageConflictError"))
##                 }
##             }
##             if (warn.conflicts) {
##                 packageStartupMessage(gettextf("\nAttaching package: %s\n",
##                   sQuote(package)), domain = NA)
##                 pkg <- names(conflicts)
##                 for (i in seq_along(conflicts)) {
##                   msg <- .maskedMsg(sort(conflicts[[i]]), pkg = sQuote(pkg[i]),
##                     by = cpos[i] < lib.pos)
##                   packageStartupMessage(msg, domain = NA)
##                 }
##             }
##         }
##     }
##     if (verbose && quietly)
##         message("'verbose' and 'quietly' are both true; being verbose then ..")
##     if (!missing(package)) {
##         if (is.null(lib.loc))
##             lib.loc <- .libPaths()
##         lib.loc <- lib.loc[dir.exists(lib.loc)]
##         if (!character.only)
##             package <- as.character(substitute(package))
##         if (length(package) != 1L)
##             stop("'package' must be of length 1")
##         if (is.na(package) || (package == ""))
##             stop("invalid package name")
##         pkgname <- paste0("package:", package)
##         newpackage <- is.na(match(pkgname, search()))
##         if (newpackage) {
##             pkgpath <- find.package(package, lib.loc, quiet = TRUE,
##                 verbose = verbose)
##             if (length(pkgpath) == 0L) {
##                 if (length(lib.loc) && !logical.return)
##                   stop(packageNotFoundError(package, lib.loc,
##                     sys.call()))
##                 txt <- if (length(lib.loc))
```

```
##                   gettextf("there is no package called %s", sQuote(package))
##                 else gettext("no library trees found in 'lib.loc'")
##                 if (logical.return) {
##                   if (!quietly)
##                     warning(txt, domain = NA)
##                   return(FALSE)
##                 }
##                 else stop(txt, domain = NA)
##             }
##             which.lib.loc <- normalizePath(dirname(pkgpath),
##                 "/", TRUE)
##             pfile <- system.file("Meta", "package.rds", package = package,
##                 lib.loc = which.lib.loc)
##             if (!nzchar(pfile))
##                 stop(gettextf("%s is not a valid installed package",
##                   sQuote(package)), domain = NA)
##             pkgInfo <- readRDS(pfile)
##             testRversion(pkgInfo, package, pkgpath)
##             if (is.character(pos)) {
##                 npos <- match(pos, search())
##                 if (is.na(npos)) {
##                   warning(gettextf("%s not found on search path, using pos = 2",
##                     sQuote(pos)), domain = NA)
##                   pos <- 2
##                 }
##                 else pos <- npos
##             }
##             deps <- unique(names(pkgInfo$Depends))
##             depsOK <- isTRUE(conf.ctrl$depends.ok)
##             if (depsOK) {
##                 canMaskEnv <- dynGet("__library_can_mask__",
##                   NULL)
##                 if (is.null(canMaskEnv)) {
##                   canMaskEnv <- new.env()
##                   canMaskEnv$canMask <- union("base", conf.ctrl$can.mask)
##                   "__library_can_mask__" <- canMaskEnv
##                 }
##                 canMaskEnv$canMask <- unique(c(package, deps,
##                   canMaskEnv$canMask))
##             }
##             else canMaskEnv <- NULL
##             if (attach.required)
##                 .getRequiredPackages2(pkgInfo, quietly = quietly)
##             cr <- conflictRules(package)
##             if (missing(mask.ok))
##                 mask.ok <- cr$mask.ok
##             if (missing(exclude))
##                 exclude <- cr$exclude
##             if (packageHasNamespace(package, which.lib.loc)) {
##                 if (isNamespaceLoaded(package)) {
##                   newversion <- as.numeric_version(pkgInfo$DESCRIPTION["Version"])
##                   oldversion <- as.numeric_version(getNamespaceVersion(package))
##                   if (newversion != oldversion) {
##                     tryCatch(unloadNamespace(package), error = function(e) {
```

```
##                     P <- if (!is.null(cc <- conditionCall(e)))
##                       paste("Error in", deparse(cc)[1L], ": ")
##                     else "Error : "
##                     stop(gettextf("Package %s version %s cannot be unloaded:\n %s",
##                       sQuote(package), oldversion, paste0(P,
##                         conditionMessage(e), "\n")), domain = NA)
##                   })
##                 }
##               }
##               tt <- tryCatch({
##                 attr(package, "LibPath") <- which.lib.loc
##                 ns <- loadNamespace(package, lib.loc)
##                 env <- attachNamespace(ns, pos = pos, deps,
##                   exclude, include.only)
##               }, error = function(e) {
##                 P <- if (!is.null(cc <- conditionCall(e)))
##                   paste(" in", deparse(cc)[1L])
##                 else ""
##                 msg <- gettextf("package or namespace load failed for %s%s:\n %s",
##                   sQuote(package), P, conditionMessage(e))
##                 if (logical.return && !quietly)
##                   message(paste("Error:", msg), domain = NA)
##                 else stop(msg, call. = FALSE, domain = NA)
##               })
##               if (logical.return && is.null(tt))
##                 return(FALSE)
##               attr(package, "LibPath") <- NULL
##               {
##                 on.exit(detach(pos = pos))
##                 nogenerics <- !.isMethodsDispatchOn() || checkNoGenerics(env,
##                   package)
##                 if (isFALSE(conf.ctrl$generics.ok) || (stopOnConflict &&
##                   !isTRUE(conf.ctrl$generics.ok)))
##                   nogenerics <- TRUE
##                 if (stopOnConflict || (warn.conflicts && !exists(".conflicts.OK",
##                   envir = env, inherits = FALSE)))
##                   checkConflicts(package, pkgname, pkgpath,
##                     nogenerics, ns)
##                 on.exit()
##                 if (logical.return)
##                   return(TRUE)
##                 else return(invisible(.packages()))
##               }
##           }
##           else stop(gettextf("package %s does not have a namespace and should be re-installed",
##               sQuote(package)), domain = NA)
##       }
##       if (verbose && !newpackage)
##           warning(gettextf("package %s already present in search()",
##               sQuote(package)), domain = NA)
##   }
##   else if (!missing(help)) {
##       if (!character.only)
##           help <- as.character(substitute(help))
```

```
##         pkgName <- help[1L]
##         pkgPath <- find.package(pkgName, lib.loc, verbose = verbose)
##         docFiles <- c(file.path(pkgPath, "Meta", "package.rds"),
##             file.path(pkgPath, "INDEX"))
##         if (file.exists(vignetteIndexRDS <- file.path(pkgPath,
##             "Meta", "vignette.rds")))
##             docFiles <- c(docFiles, vignetteIndexRDS)
##         pkgInfo <- vector("list", 3L)
##         readDocFile <- function(f) {
##             if (basename(f) %in% "package.rds") {
##                 txt <- readRDS(f)$DESCRIPTION
##                 if ("Encoding" %in% names(txt)) {
##                   to <- if (Sys.getlocale("LC_CTYPE") == "C")
##                     "ASCII//TRANSLIT"
##                   else ""
##                   tmp <- try(iconv(txt, from = txt["Encoding"],
##                     to = to))
##                   if (!inherits(tmp, "try-error"))
##                     txt <- tmp
##                   else warning("'DESCRIPTION' has an 'Encoding' field and re-encoding is not possible
##                     call. = FALSE)
##                 }
##                 nm <- paste0(names(txt), ":")
##                 formatDL(nm, txt, indent = max(nchar(nm, "w")) +
##                   3L)
##             }
##             else if (basename(f) %in% "vignette.rds") {
##                 txt <- readRDS(f)
##                 if (is.data.frame(txt) && nrow(txt))
##                   cbind(basename(gsub("\\.[[:alpha:]]+$", "",
##                     txt$File)), paste(txt$Title, paste0(rep.int("(source",
##                     NROW(txt)), ifelse(nzchar(txt$PDF), ", pdf",
##                     ""), ")")))
##                 else NULL
##             }
##             else readLines(f)
##         }
##         for (i in which(file.exists(docFiles))) pkgInfo[[i]] <- readDocFile(docFiles[i])
##         y <- list(name = pkgName, path = pkgPath, info = pkgInfo)
##         class(y) <- "packageInfo"
##         return(y)
##     }
##     else {
##         if (is.null(lib.loc))
##             lib.loc <- .libPaths()
##         db <- matrix(character(), nrow = 0L, ncol = 3L)
##         nopkgs <- character()
##         for (lib in lib.loc) {
##             a <- .packages(all.available = TRUE, lib.loc = lib)
##             for (i in sort(a)) {
##                 file <- system.file("Meta", "package.rds", package = i,
##                   lib.loc = lib)
##                 title <- if (nzchar(file)) {
##                   txt <- readRDS(file)
```

```
##                   if (is.list(txt))
##                     txt <- txt$DESCRIPTION
##                   if ("Encoding" %in% names(txt)) {
##                     to <- if (Sys.getlocale("LC_CTYPE") == "C")
##                       "ASCII//TRANSLIT"
##                     else ""
##                     tmp <- try(iconv(txt, txt["Encoding"], to,
##                       "?"))
##                     if (!inherits(tmp, "try-error"))
##                       txt <- tmp
##                     else warning("'DESCRIPTION' has an 'Encoding' field and re-encoding is not possi
##                       call. = FALSE)
##                   }
##                   txt["Title"]
##                 }
##                 else NA
##                 if (is.na(title))
##                   title <- " ** No title available ** "
##                 db <- rbind(db, cbind(i, lib, title))
##             }
##             if (length(a) == 0L)
##                 nopkgs <- c(nopkgs, lib)
##         }
##         dimnames(db) <- list(NULL, c("Package", "LibPath", "Title"))
##         if (length(nopkgs) && !missing(lib.loc)) {
##             pkglist <- paste(sQuote(nopkgs), collapse = ", ")
##             msg <- sprintf(ngettext(length(nopkgs), "library %s contains no packages",
##                 "libraries %s contain no packages"), pkglist)
##             warning(msg, domain = NA)
##         }
##         y <- list(header = NULL, results = db, footer = NULL)
##         class(y) <- "libraryIQR"
##         return(y)
##     }
##     if (logical.return)
##         TRUE
##     else invisible(.packages())
## }
## <bytecode: 0x0000000012f25cf8>
## <environment: namespace:base>
```

```
#library(spDataLarge)
library(tmap)
library(leaflet)
library(ggplot2)
```

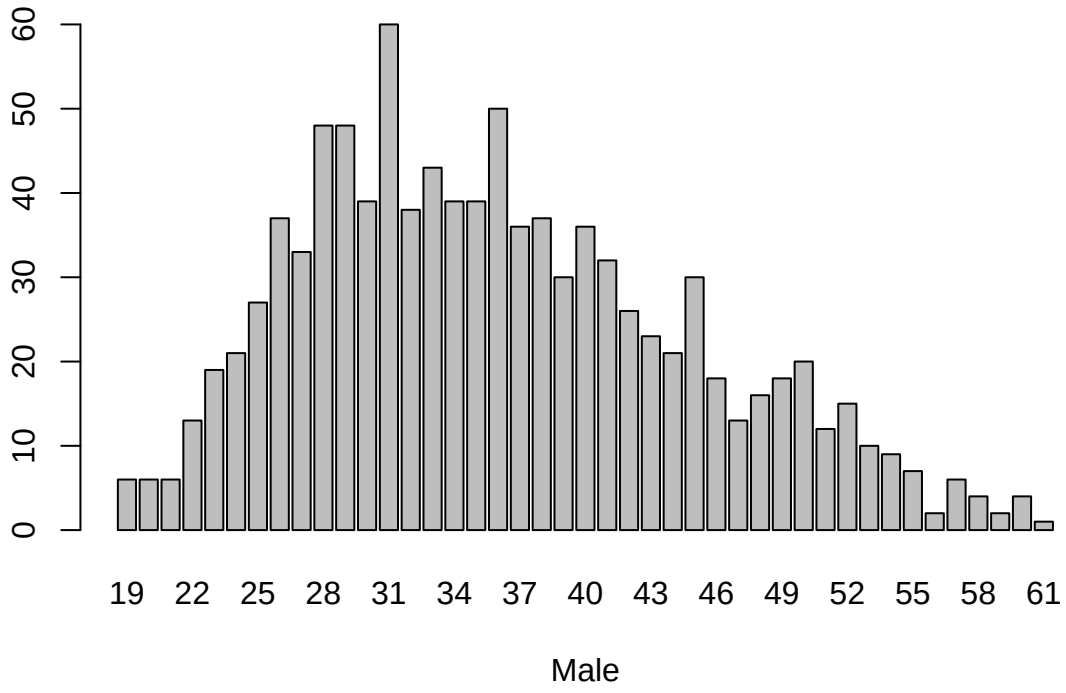
```
Country <- ad$Country
countyfreq <- table(Country)
```

```
hist(ad$`Age`, xlab = "Age")
```


```
#An even spread on time spent on site
```

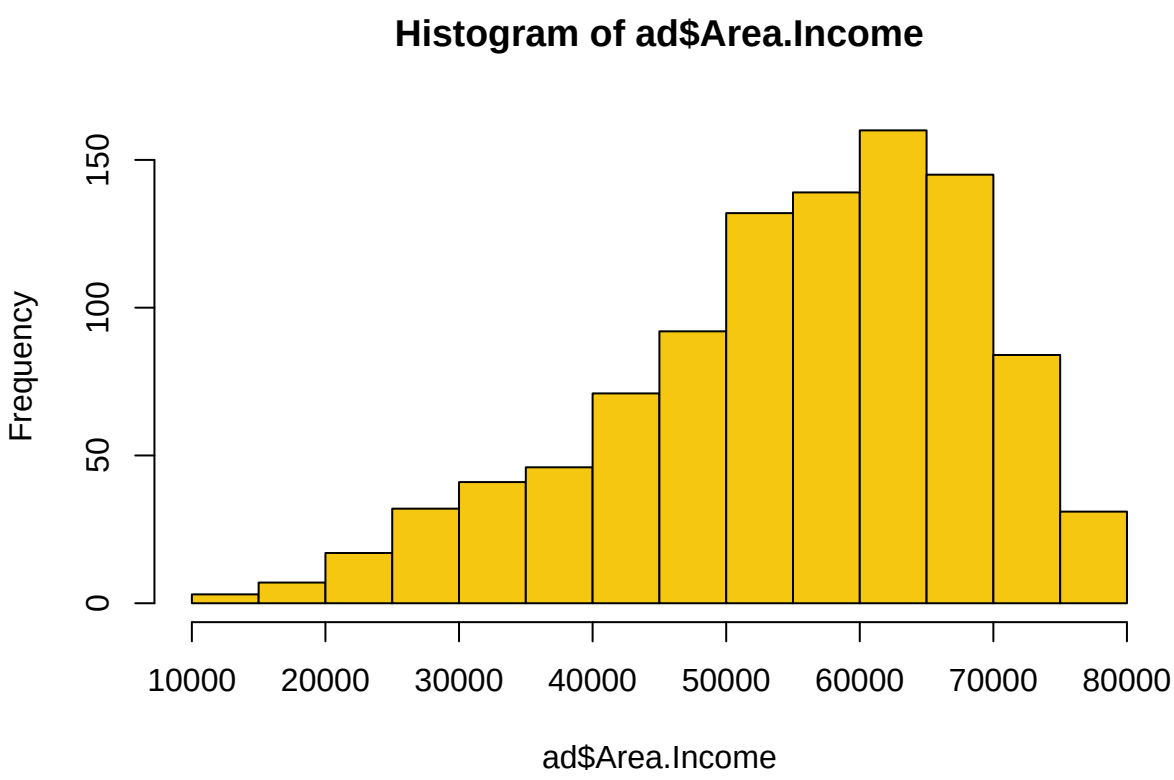
hist(ad$Male, xlab = “Male”)

```
age <- ad$Age
ages <- table(age)
barplot(ages, xlab = "Male")
```




```
#The age distribution
```

```
hist(ad$Area.Income, col = 7)
```




```
AGE  <- table(ad$Age)
AGE
```

```
##
## 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
##  6  6  6 13 19 21 27 37 33 48 48 39 60 38 43 39 39 50 36 37 30 36 32 26 23 21
## 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
## 30 18 13 16 18 20 12 15 10  9  7  2  6  4  2  4  1
```

```
barplot(AGE, main = "Simple bar plot", xlab = "Age ", ylab = "Frequency")
```

Bivariate Analysis

Lets find the covariance between a variety of the features

```
daily <- ad$`Daily Time Spent on Site`
age <- ad$Age
income <- ad$`Area Income`
sex <-ad$Male
use <- ad$`Daily Internet Usage`

click <- ad$`Clicked on Ad`
#cov(daily,click)
#That has a negative correlation
```

```
ad2 <- ad$Age
ad3 <- ad$Area.Income
cov(ad2, ad3)
```

```
## [1] -21520.93
```

```
cor(ad2, ad3)
```

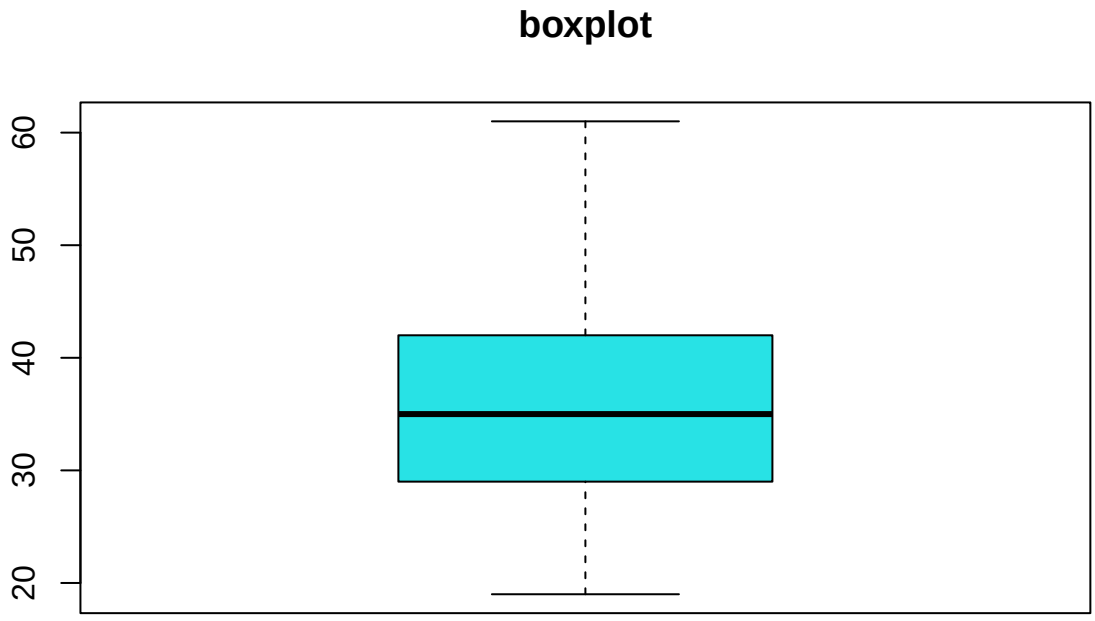
```
## [1] -0.182605
```

checking correlation matrix

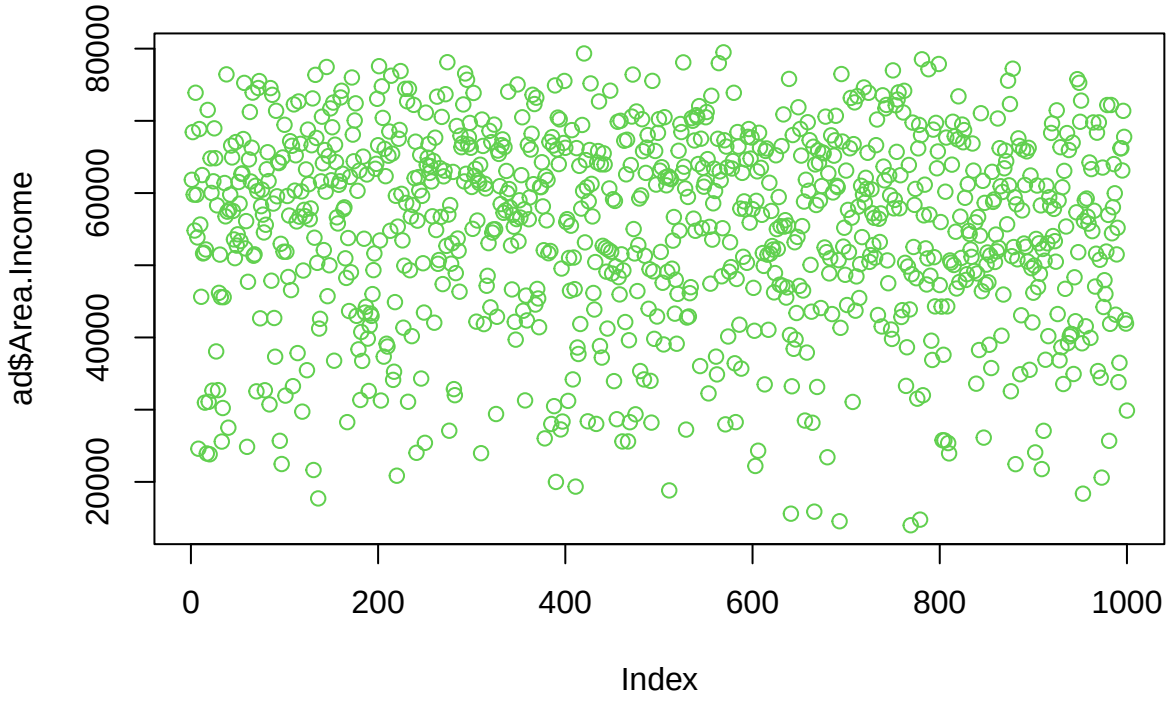
```
num_ads <- unlist(lapply(ad, is.numeric))
num_ad <- ad[ , num_ads]
cor(num_ad)
```

```
##                           Daily.Time.Spent.on.Site         Age  Area.Income
## Daily.Time.Spent.on.Site               1.00000000 -0.33151334  0.310954413
## Age                                   -0.33151334  1.00000000 -0.182604955
## Area.Income                            0.31095441 -0.18260496  1.000000000
## Daily.Internet.Usage                   0.51865848 -0.36720856  0.337495533
## Male                                  -0.01895085 -0.02104406  0.001322359
## Clicked.on.Ad                         -0.74811656  0.49253127 -0.476254628
##                           Daily.Internet.Usage         Male Clicked.on.Ad
## Daily.Time.Spent.on.Site           0.51865848 -0.018950855   -0.74811656
## Age                               -0.36720856 -0.021044064    0.49253127
## Area.Income                        0.33749553  0.001322359   -0.47625463
## Daily.Internet.Usage               1.00000000  0.028012326   -0.78653918
## Male                               0.02801233  1.000000000   -0.03802747
## Clicked.on.Ad                     -0.78653918 -0.038027466    1.00000000
```

```
boxplot(ad$Age, main='boxplot', xlab = 'area income', col = 5)
```




```
plot(ad$Area.Income, col = 3)
```



Correlation #creating with only interger columns

```
numerical_df = ad[c("Daily.Time.Spent.on.Site", "Age", "Area.Income","Daily.Internet.Usage" ,"Male", "C
head(numerical_df)
```

```
##   Daily.Time.Spent.on.Site Age Area.Income Daily.Internet.Usage Male
## 1                    68.95  35    61833.90               256.09    0
## 2                    80.23  31    68441.85               193.77    1
## 3                    69.47  26    59785.94               236.50    0
## 4                    74.15  29    54806.18               245.89    1
## 5                    68.37  35    73889.99               225.58    0
## 6                    59.99  23    59761.56               226.74    1
##   Clicked.on.Ad
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
```

```
correlation = cor(numerical_df)
correlation
```

```
##                          Daily.Time.Spent.on.Site         Age  Area.Income
```

```
## Daily.Time.Spent.on.Site               1.00000000 -0.33151334  0.310954413
## Age                                   -0.33151334  1.00000000 -0.182604955
## Area.Income                            0.31095441 -0.18260496  1.000000000
## Daily.Internet.Usage                   0.51865848 -0.36720856  0.337495533
## Male                                  -0.01895085 -0.02104406  0.001322359
## Clicked.on.Ad                         -0.74811656  0.49253127 -0.476254628
##                           Daily.Internet.Usage         Male Clicked.on.Ad
## Daily.Time.Spent.on.Site           0.51865848 -0.018950855   -0.74811656
## Age                               -0.36720856 -0.021044064    0.49253127
## Area.Income                        0.33749553  0.001322359   -0.47625463
## Daily.Internet.Usage               1.00000000  0.028012326   -0.78653918
## Male                               0.02801233  1.000000000   -0.03802747
## Clicked.on.Ad                     -0.78653918 -0.038027466    1.00000000
```

Conclusion

From the analysis we can get several deductions: - The mean Age of the population is 36, and as the age increased more people clicked on ads. - There is an inverse relationship between the daily time spent on site and the number of people who click the ads - The gender of the users had the least effect on the number of ads clicked and barely affected any other variables.