2019 Stack Overflow Developer Survey Results

Introduction

This data analysis project uses the 2019 Stack Overflow Developer Survey Results dataset which can be found here (https://insights.stackoverflow.com/survey).

The survey had a total of 88,883 respondents, with 85 questions, ranging from general questions such as age, gender and academic background to professional questions such as what are the technologies worked with, employment status and more. The survey also included some questions regarding the respondent's interactions with Stack Overflow, but those were not a part of the objective of this analysis.

The dataset has not been modified further than what is done in this Notebook. The dataset is already very clean and structured, which means most of the modifications were for the sake of making the data easier to visualize.

The Plotly library was chosen for the visualizations due the interactivity of the resulting visualizations. Apart from that, common libraries were used for the data transformations.

Structure

The analysis has three main sections:

- General Data: includes general data about the respondents, including the gender distribution, academic background, employment status and country;
- Tech Stack: all the technology-related data, including programming languages, database environments, operating systems, etc.;
- Professional Life: data related to the professional lives of the respondents, such as work week hours, work location, most important job factors, etc.

The Conclusion at the very end aggregates all the "Key Takeaways" extracted from the enumerated sections. It has nothing new, only a TL;DR with the key takeaways of this data analysis.

Objectives

The following are the guiding questions and/or objectives of this analysis. They are answered throughout the outlined sections, in the "Key Takeaways" and Conclusion:

What are the most used technologies (programming languages, database environments, operating systems, etc.); Where are respondents from and what are their work conditions and employment status; How's the professional life of respondents?

Load Libraries

```
library(readr)
library(dplyr)
```

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

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

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

```
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(tidyr)
library(stringr)
library(docstring)
```

```
##
## Attaching package: 'docstring'
```

```
## The following object is masked from 'package:utils':
##
##     ?
```

Global Variables

```
general_data_color <- "#60BD68"
tech_stack_color <- "#5DA5DA"
prof_life_color <- "#DECF3F"
```

Helper Functions

Data Transformations

```
rename_genders <- function(x) {
  #' Used in combination with an *apply function to rename the options of the genders
  if(is.na(x)) {x}
  else if (x == "Man") {x}
  else if (x == "Woman") {x}
  else {"Other"}
}
```

```
rename_ed_level <- function(x) {
  #' Used in combination with an *apply function to rename the options of the education levels
  if(is.na(x)) {x}
  else if (x == "I never completed any formal education") {"No formal education"}
  else if (x == "Primary/elementary school") {"Primary School"}
  else if (x == "Secondary school (e.g. American high school, German Realschule or Gymnasium, et
c.)") {"Secondary School"}
  else if (x == "Some college/university study without earning a degree") {"Higher ed. study w/o
Degree"}
  else if (x == "Associate degree") {"Associate Degree"}
  else if (x == "Bachelor’s degree (BA, BS, B.Eng., etc.)") {"Bachelor’s Degree"}
  else if (x == "Master’s degree (MA, MS, M.Eng., MBA, etc.)") {"Master’s Degree"}
  else if (x == "Professional degree (JD, MD, etc.)") {"Professional Degree"}
  else {"Doctoral Degree"}
}
```

```
rename_underg_majors <- function(x) {
  #' Used in combination with an *apply function to rename the options of the undergrad majors
  if(is.na(x)) {x}
  else if (x == "Computer science, computer engineering, or software engineering") {"CompSci/Sof
tEng"}
  else if (x == "Web development or web design") {"Web Development/Design"}
  else if (x == "Information systems, information technology, or system administration") {"InfoT
ech/SysAdmin"}
  else if (x == "Mathematics or statistics") {"Mathematics/Statistics"}
  else if (x == "Another engineering discipline (ex. civil, electrical, mechanical)") {"Another
 Eng. Discipline"}
  else if (x == "A business discipline (ex. accounting, finance, marketing)") {"Business"}
  else if (x == "A health science (ex. nursing, pharmacy, radiology)") {"Health Science"}
  else if (x == "A humanities discipline (ex. literature, history, philosophy)") {"Humanities"}
  else if (x == "A natural science (ex. biology, chemistry, physics)") {"Natural Science"}
  else if (x == "A social science (ex. anthropology, psychology, political science)") {"Social S
cience"}
  else if (x == "Fine arts or performing arts (ex. graphic design, music, studio art)") {"Fine A
rts/Performing Arts"}
  else if (x == "I never declared a major") {"None"}
  else {"Other"}
}
```

```
rename_employment <- function(x) {
  #' Used in combination with an *apply function to rename the options of the employment status
  if(is.na(x)) {x}
  else if (x == "Not employed, and not looking for work") {"Not looking for work"}
  else if (x == "Not employed, but looking for work") {"Looking for work"}
  else if (x == "Employed full-time") {"Full-time"}
  else if (x == "Employed part-time") {"Part-time"}
  else if (x == "Independent contractor, freelancer, or self-employed") {"Freelancer/Self-employ
ed"}
  else if (x == "Retired") {"Retired"}
  else {"Other"}
}
```

```
rename_prog_usage <- function(x) {
  #' Used in combination with an *apply function to rename the options of the professional progr
amming usage
  if(is.na(x)) {x}
  else if (x == "Man") {x}
  else if (x == "I am a student who is learning to code") {"Student"}
  else if (x == "I am not primarily a developer, but I write code sometimes as part of my work")
{"Code as part of job"}
  else if (x == "I code primarily as a hobby") {"Hobbyist"}
  else if (x == "I am a developer by profession") {"Professional developer"}
  else if (x == "I used to be a developer by profession, but no longer am") {"Used to work as de
veloper"}
  else {"Other"}
}
```

```
rename_work_loc <- function(x) {
  #' Used in combination with an *apply function to rename the options of the work location
  if(is.na(x)) {x}
  else if (x == "Office") {"Office"}
  else if (x == "Home") {"Remote"}
  else if (x == "Other place, such as a coworking space or cafe") {"Remote"}
  else {"Other"}
}
```

```
rename_job_factors <- function(x) {
  #' Used in combination with an *apply function to rename the options of the job factors
  if(is.na(x)) {x}
  else if (x == "Languages, frameworks, and other technologies I'd be working with") {"Technical
compatibility"}
  else if (x == "Office environment or company culture") {"Culture compatibility"}
  else if (x == "Flex time or a flexible schedule") {"Flexible schedule"}
  else if (x == "Opportunities for professional development") {"Professional development"}
  else if (x == "Remote work options") {"Remote work options"}
  else if (x == "How widely used or impactful my work output would be") {"Impact of my work"}
  else if (x == "Industry that I'd be working in") {"Industry compatibility"}
  else if (x == "Financial performance or funding status of the company or organization") {"Comp
any's financial performance"}
  else if (x == "Specific department or team I'd be working on") {"Department/Team compatibilit
y"}
  else if (x == "Diversity of the company or organization") {"Diversity of the company"}
  else {"Other"}
}
```

```
rename_resume_update <- function(x) {
  #' Used in combination with an *apply function to rename the options of the reason for last re
sume update
  if(is.na(x)) {x}
  else if (x == "Something else changed (education, award, media, etc.)") {"New achievement"}
  else if (x == "I was preparing for a job search") {"Job search"}
  else if (x == "I heard about a job opportunity (from a recruiter, online job posting, etc.)")
 {"Job opportunity"}
  else if (x == "My job status changed (promotion, new job, etc.)") {"Job status change"}
  else if (x == "I had a negative experience or interaction at work") {"Negative experience at w
ork"}
  else if (x == "Re-entry into the workforce") {"Re-entry into workforce"}
  else {"Other"}
}
```

```
rename_options <- function(df, target_col, rename_func) {
  #' Renames the categorical values of a data column, given the function responsible for the con
ditional renaming
  #' @param df data.frame. The dataframe with the target column
  #' @param target_col character. Name of the column whose categories will be renamed
  #' @param rename_func function. The function responsible for the conditional renaming
  #' @return A new one-column dataframe with the renamed categories (same column name as input)
  updated_df <- data.frame(sapply(df[[target_col]], rename_func))

  colnames(updated_df) <- c(target_col)

  return(updated_df)
}
```

```
count_options <- function(df, target_col, desc_order = FALSE) {
  #' Counts the frequencies of each value in the target column
  #' @param wip_df data.frame. The dataframe with the target column
  #' @param target_col character. Name of the column whose categories will be renamed
  #' @param desc_order logical. Whether the returned dataframe should be sorted in descending or
der (defaults to FALSE, that is, ascending order)
  #' @return A new two-column dataframe that contains the unique values (same name as the input
 dataframe) and their respective frequency ("Frequency")
  df_counts <- df %>%
    group_by(df[target_col]) %>%
    summarize(Frequency = n())

  df_counts <-
    df_counts[order(df_counts["Frequency"], decreasing = desc_order),]

  return(df_counts)
}
```

```
delimeted_to_df <- function(og_df, target_col, delimiter) {
  #' Given a dataframe with a char column whose values are multiple unique categories separated
 by `delimiter`, split them into different rows (that is, return a longer, one-column dataframe)
  #' @param og_df data.frame. The dataframe with the delimited data column
  #' @param target_col character. Name of the column with delimited data
  #' @param delimiter character. Delimiter of the unique values
  #' @return A new dataframe with the delimited data split and unpivoted into a single column (n
ote that any other columns in the input dataframe are kept)

  # After splitting values, `max_split_cols` columns will be needed to make sure all the individ
ual values of the row with most values are kept
  # In summary, this is the maximum number of delimiters found in a single row (so that row has
 that number+1 values)
  max_split_cols <-
    max(mapply(str_count, og_df[[target_col]], delimiter), na.rm = TRUE) + 1

  # Sequence to name the temporary columns (generate a sequence from 1 to `max_split_cols` so th
at each column is named "Col" plus a number)
  sep_into_cols <-
    unname(mapply(paste, "Col", 1:max_split_cols, sep = ""))

  # For each row, separate its unique values into multiple columns
  separated_data <- og_df %>%
    separate(target_col, sep_into_cols, sep = delimiter, fill = "right")

  # Now unpivot all the columns with individual values (`sep_into_cols`) into a column called `t
arget_col`
  # The names of the columns are put in "TempCols" and the individual values in `target_col`
  unpivot_wide_data <- separated_data %>%
    pivot_longer(
      sep_into_cols,
      names_to = "TempCols",
      values_to = target_col,
      values_drop_na = TRUE
    )
  # Remove the column that has the name of the temporary columns (created during the unpivot)
  unpivot_wide_data <- unpivot_wide_data %>%
    select(-TempCols)

  return(unpivot_wide_data)
}
```

Plotting

```
plot_bar_chart <-
  function(plot_df,
           x_col,
           y_col,
           order_col,
           plot_title,
           axes_titles,
           fill_color) {
    #' Plot a bar chart with Plotly
    fig <- plot_ly() %>%
      add_bars(
        data = plot_df,
        x =  ~ plot_df[[x_col]],
        y = plot_df[[y_col]],
        marker = list(color = fill_color)
      ) %>%
      layout(
        title = plot_title,
        xaxis = list(title = axes_titles[1]),
        yaxis = list(
          title = axes_titles[2],
          categoryorder = "array",
          categoryarray = plot_df[, order_col]
        )
      )
    fig
  }
```

```
plot_column_chart <-
  function(plot_df,
           x_col,
           y_col,
           order_col,
           plot_title,
           axes_titles,
           fill_color) {
    #' Plot a column chart with Plotly
    fig <- plot_ly() %>%
      add_bars(
        data = plot_df,
        x =  ~ plot_df[[x_col]],
        y = plot_df[[y_col]],
        marker = list(color = fill_color)
      ) %>%
      layout(
        title = plot_title,
        xaxis = list(
          title = axes_titles[1],
          categoryorder = "array",
          categoryarray = plot_df[[order_col]]
        ),
        yaxis = list(title = axes_titles[2])
      )
    fig
  }
```

```
plot_histogram <-
  function(plot_df,
           bins,
           plot_title,
           axes_titles,
           fill_color) {
    #' Plot an histogram with Plotly
    fig <- plot_ly() %>%
      add_histogram(x = plot_df,
                    xbins = bins,
                    marker = list(color = fill_color)) %>%
      layout(
        title = plot_title,
        xaxis = list(title = axes_titles[1]),
        yaxis = list(title = axes_titles[2])
      )
    fig
  }
```

```
plot_pie <- function(plot_df,
                     data_labels,
                     data_values,
                     plot_title,
                     layout_marker) {
  #' Plot a pie chart with Plotly
  fig <- plot_ly() %>%
    add_pie(
      data = plot_df,
      labels = plot_df[[data_labels]],
      values = plot_df[[data_values]],
      marker = layout_marker
    ) %>%
    layout(title = plot_title)
  fig
}
```

```
plot_histogram_groups <- function(plot_df,
                                  x_data,
                                  n_bins,
                                  group_data,
                                  plot_title,
                                  axes_titles,
                                  fill_color) {
  #' Plot an histogram broken down by a second variable with Plotly
  fig <- plot_ly() %>%
    add_histogram(
      data = plot_df,
      x = plot_df[[x_data]],
      nbinsx = n_bins,
      color = plot_df[[group_data]],
      colors = fill_color,
      xbins = bins
    ) %>%
    layout(
      title = plot_title,
      xaxis = list(title = axes_titles[1]),
      yaxis = list(title = axes_titles[2])
    )
  fig
}
```

Load Data

```
data <- read_csv("..\\data\\survey_results_public.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Respondent = col_double(),
##   CompTotal = col_double(),
##   ConvertedComp = col_double(),
##   WorkWeekHrs = col_double(),
##   CodeRevHrs = col_double(),
##   Age = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
# Remove unused columns
data <- data %>%
  select (
    'Respondent',
    'Age',
    'ConvertedComp',
    'Country',
    'DatabaseWorkedWith',
    'DevEnviron',
    'EdLevel',
    'Employment',
    'Gender',
    'JobFactors',
    'LanguageWorkedWith',
    'MainBranch',
    'OpSys',
    'PlatformWorkedWith',
    'ResumeUpdate',
    'UndergradMajor',
    'WebFrameWorkedWith',
    'WorkLoc',
    'WorkWeekHrs',
    'WorkWeekHrs'
  )
```

```
num_responses <- dim(data)[1]
print(sprintf("Number of responses: %s", num_responses))
```

```
## [1] "Number of responses: 88883"
```

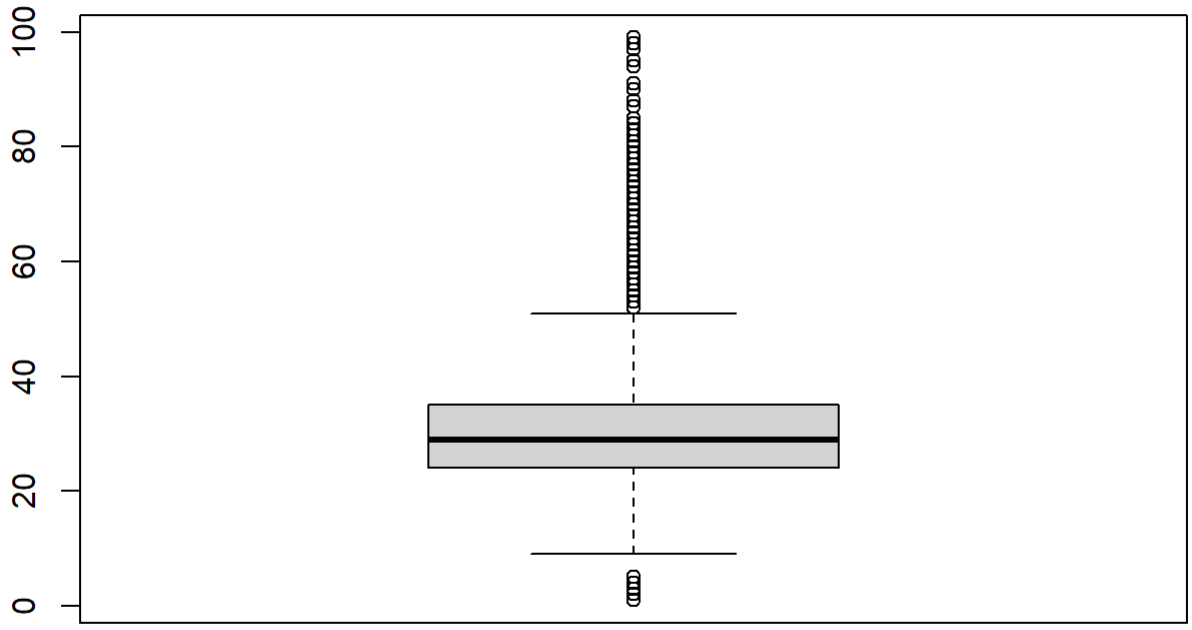
```
num_columns <- dim(data)[2]
print(sprintf("Number of variables kept: %s", num_columns))
```

```
## [1] "Number of variables kept: 19"
```

General Data Analysis

Ages

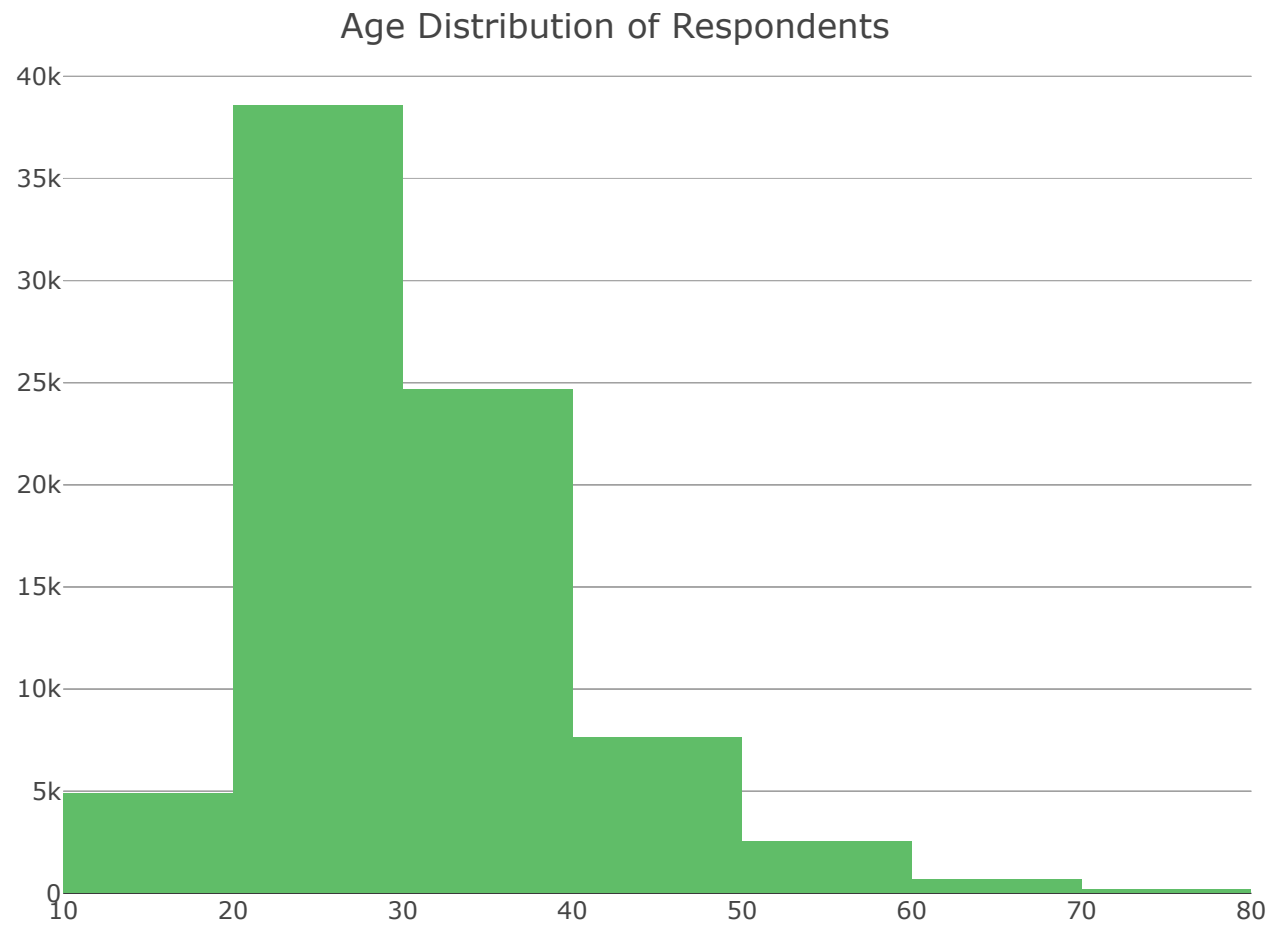
```
# Explore the data distribution
age <- as.integer(data$Age)
temp <- table(age)
boxplot(age)
```




```
# Keep only a certain interval of ages
age <- age[age >= 10 & age <= 80]
age_bins <- list(start = 0, end = 80, size = 10)
plot_histogram(age,
               age_bins,
               "Age Distribution of Respondents",
               c("", ""),
               general_data_color)
```

```
## Warning: Ignoring 9673 observations
```

```
## Warning: `arrange_()` is deprecated as of dplyr 0.7.0.
## Please use `arrange()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

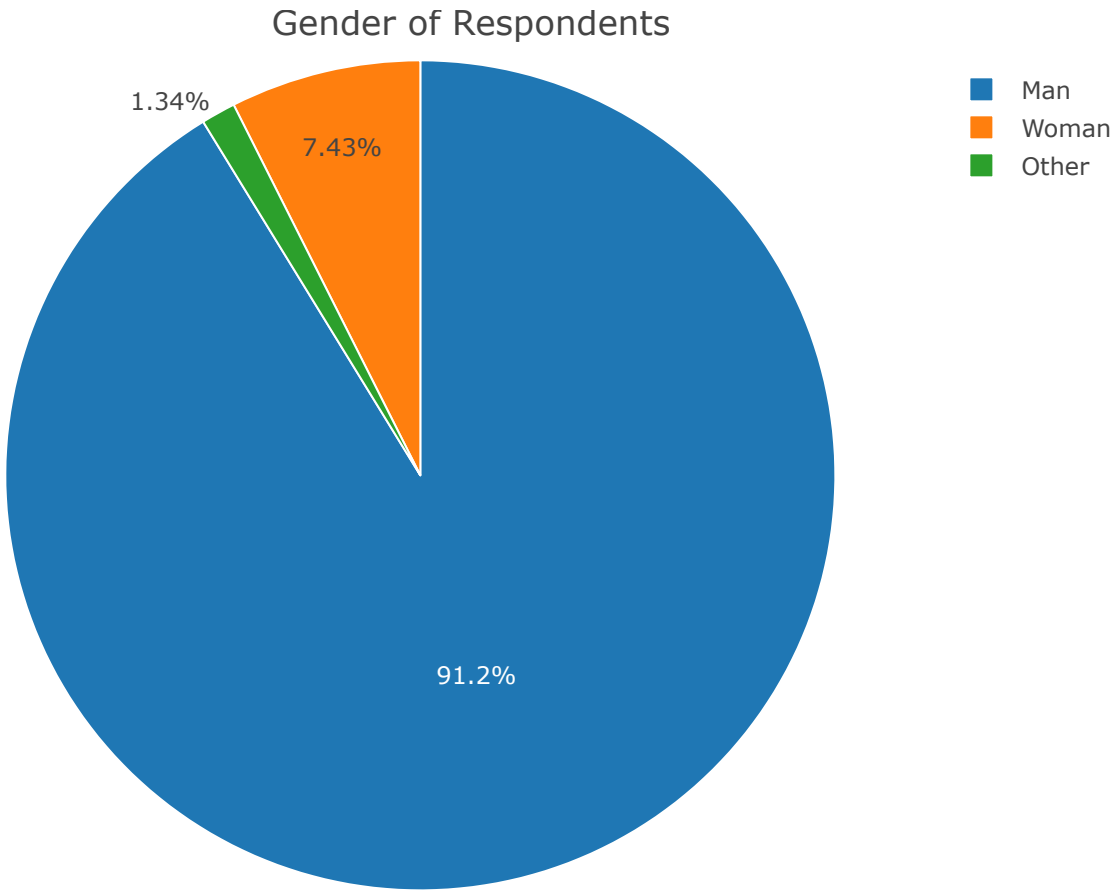


Gender

```
# Rename the gender categories and count their frequencies
gender <- data %>%
  rename_options("Gender", rename_genders) %>%
  count_options("Gender")
```

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

```
# Remove missing values
gender <- gender[!is.na(gender$Gender),]

marker_options <- list(color = "Reds",
                       line = list(color = "#FFFFFF", width = 1))

plot_pie(gender,
         "Gender",
         "Frequency",
         "Gender of Respondents",
         marker_options)
```



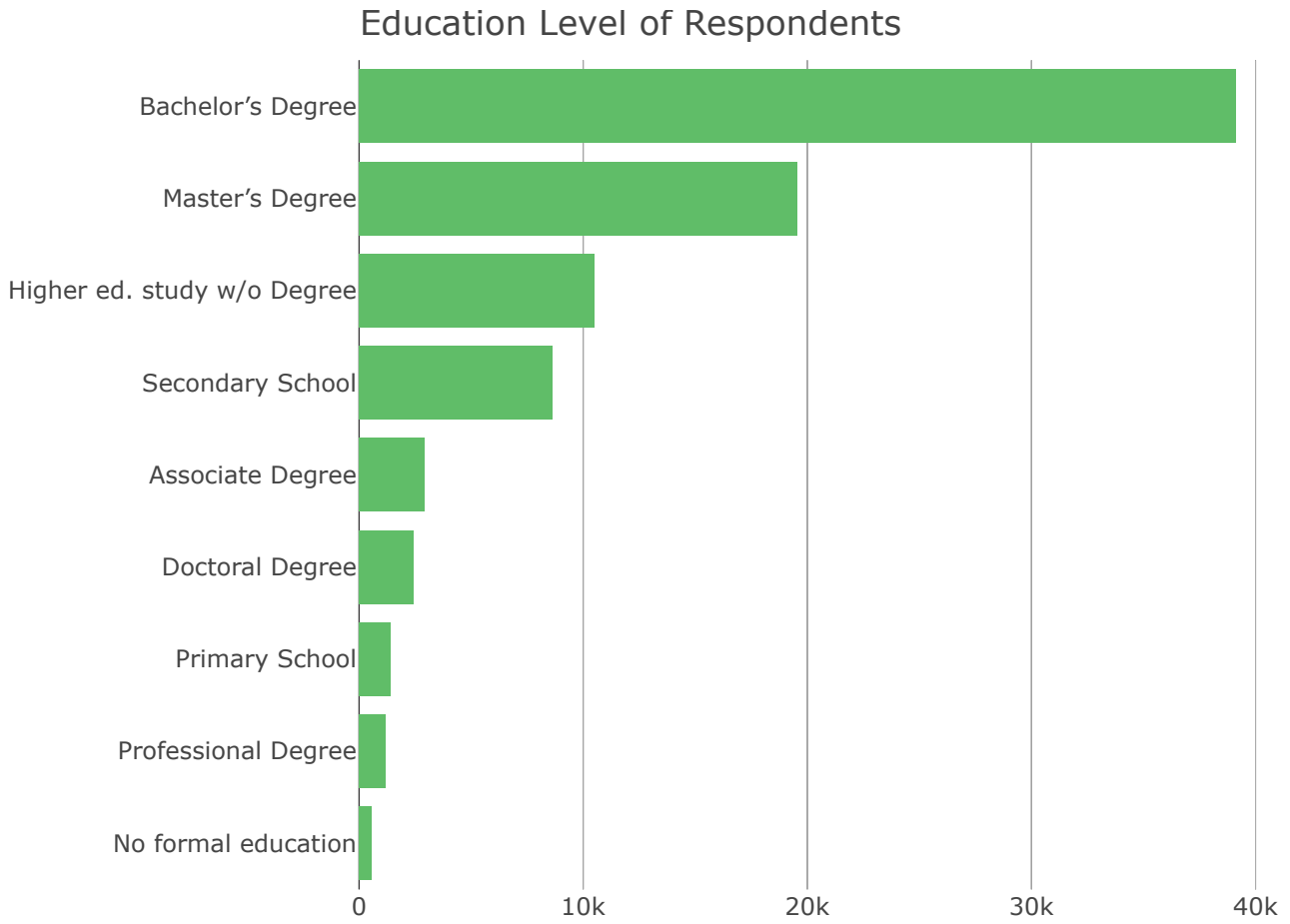
Education Level

```
# Rename the education level categories and count their frequencies
ed_level <- data %>%
  rename_options("EdLevel", rename_ed_level) %>%
  count_options("EdLevel")
```

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

```
plot_bar_chart(
  ed_level,
  "Frequency",
  "EdLevel",
  "Frequency",
  "Education Level of Respondents",
  c("", ""),
  general_data_color
)
```

```
## Warning: Ignoring 1 observations
```



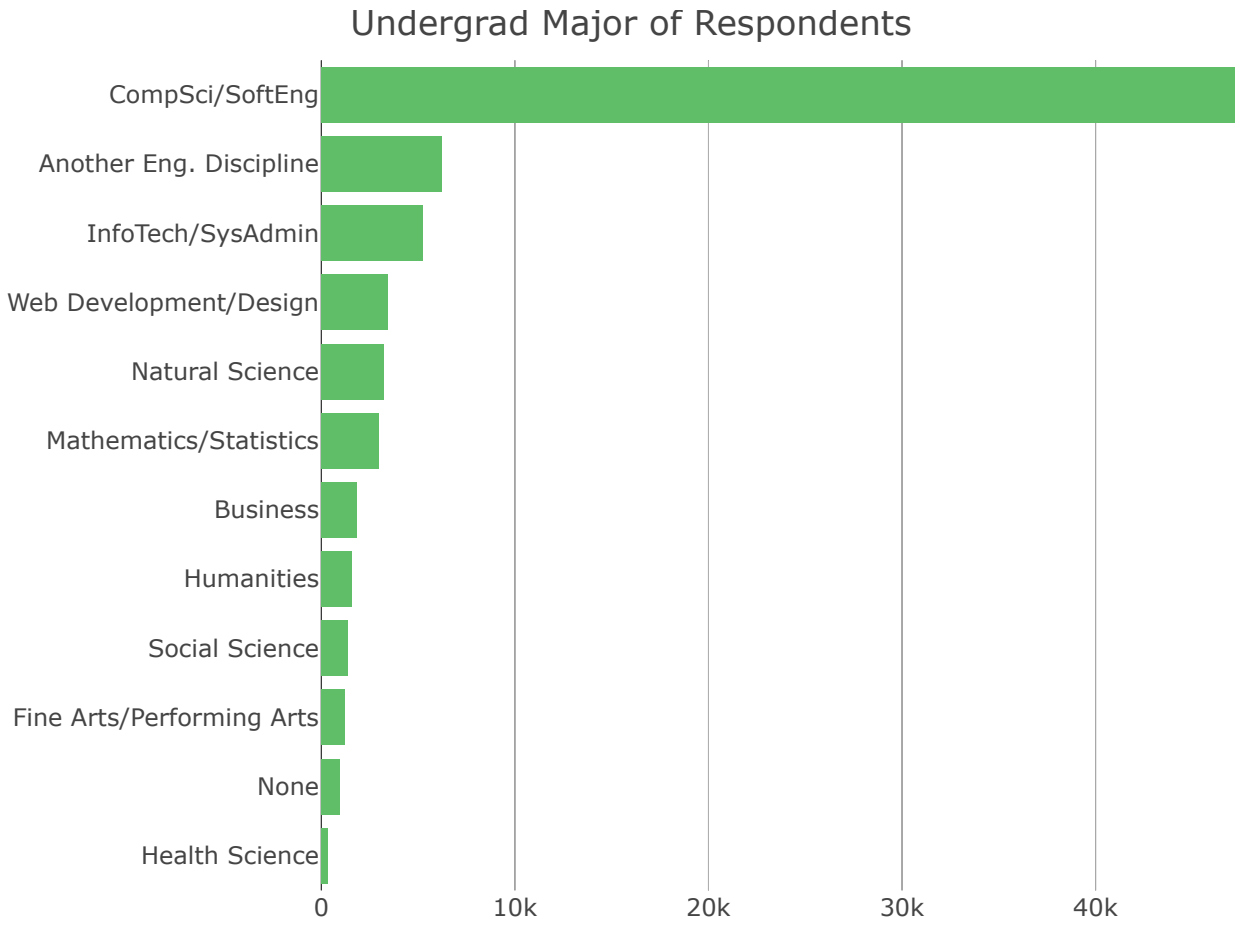
Undergrad Major

```
# Rename the undergrad major categories and count their frequencies
underg_major <- data %>%
  rename_options("UndergradMajor", rename_underg_majors) %>%
  count_options("UndergradMajor")
```

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

```
plot_bar_chart(
  underg_major,
  "Frequency",
  "UndergradMajor",
  "Frequency",
  "Undergrad Major of Respondents",
  c("", ""),
  general_data_color
)
```

```
## Warning: Ignoring 1 observations
```



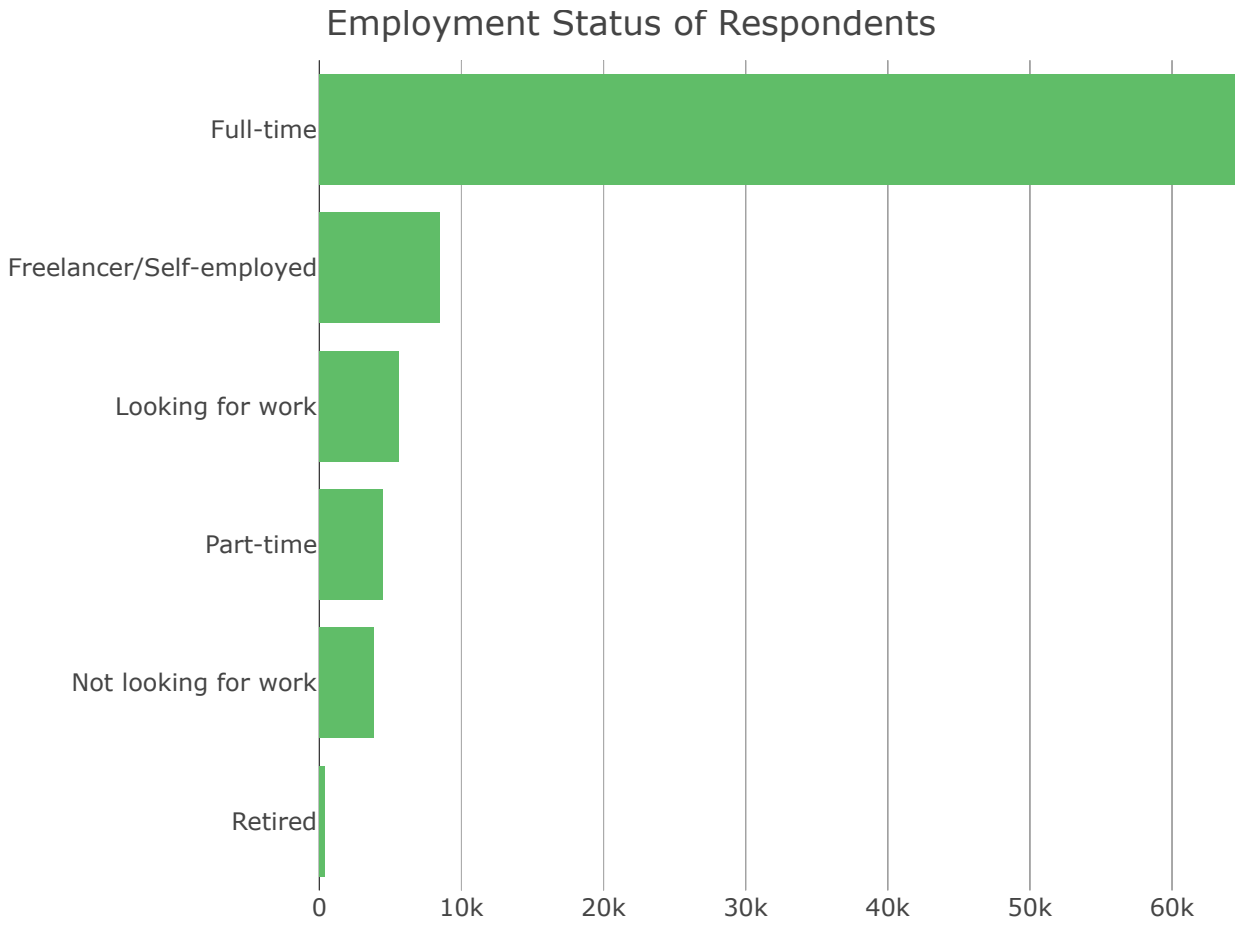
Employment Status

```
# Rename the employment status categories and count their frequencies
employment_status <- data %>%
  rename_options("Employment", rename_employment) %>%
  count_options("Employment")
```

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

```
plot_bar_chart(
  employment_status,
  "Frequency",
  "Employment",
  "Frequency",
  "Employment Status of Respondents",
  c("", ""),
  general_data_color
)
```

```
## Warning: Ignoring 1 observations
```

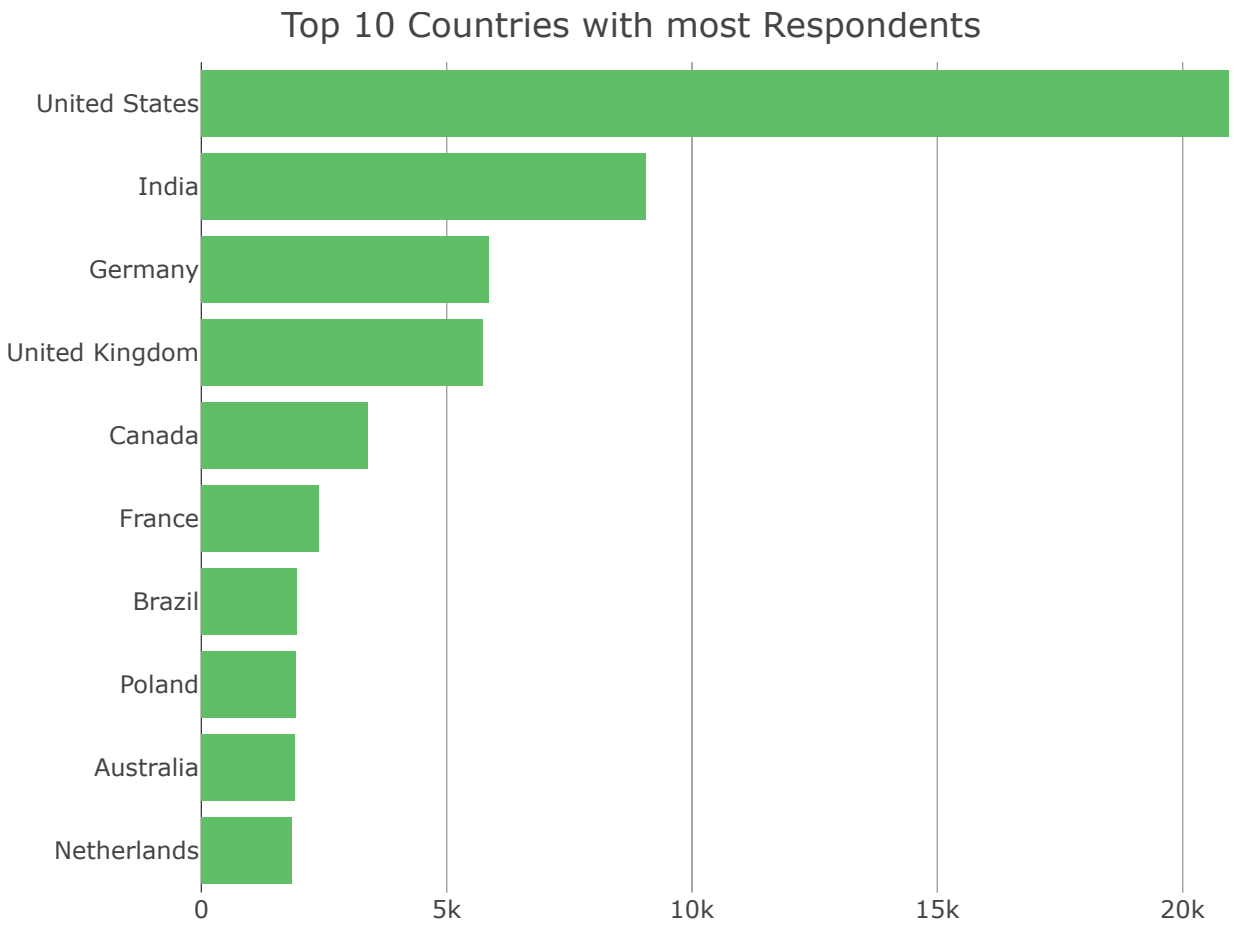


Country of Origin

```
# Count the frequency of each country
country <- data[, "Country"] %>%
  count_options("Country")
```

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

```
plot_bar_chart(
  tail(country, n = 10),
  "Frequency",
  "Country",
  "Frequency",
  "Top 10 Countries with most Respondents",
  c("", ""),
  general_data_color
)
```



Key Takeways

- Over 40% of the respondents were between the ages of 20 and 30 and almost 70% were between 20 and 40 years-old;
- Over 90% of the inquired were male, with less than 8% being of the female gender. The remaining 2% identified as another gender;
- About 43% of the respondents completed a Bachelor's Degree, but only half of those have gone on to complete a Master's Degree;
- More than half of respondents come from a Computer Science or Software Engineering academic background. The remaining inquired come primarily from other Engineering disciplines, Information Technologies, System Administration or Web Design and/or Development;

- 70% of the respondents have a full-time job, with freelance and/or self-employment being the second most common employment status;
- The five countries with most respondents are, from most to least, the United States of America, India, Germany, the United Kingdom and Canada.

Tech Stack

Programming Languages

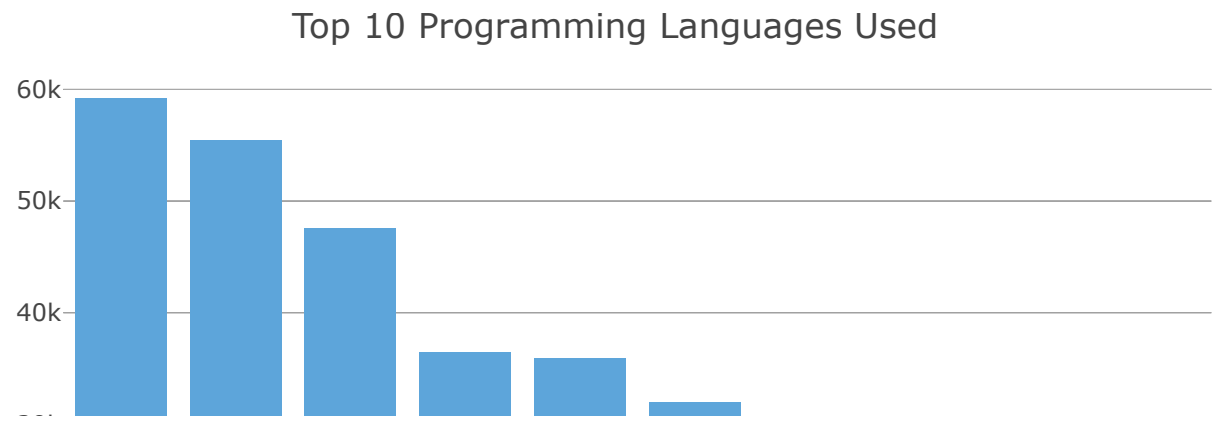
```
languages <- data %>%
  select("Respondent", "LanguageWorkedWith")

# Split and unpivot the individual programming languages
languages_undelimited <-
  delimeted_to_df(languages, "LanguageWorkedWith", ";")
```

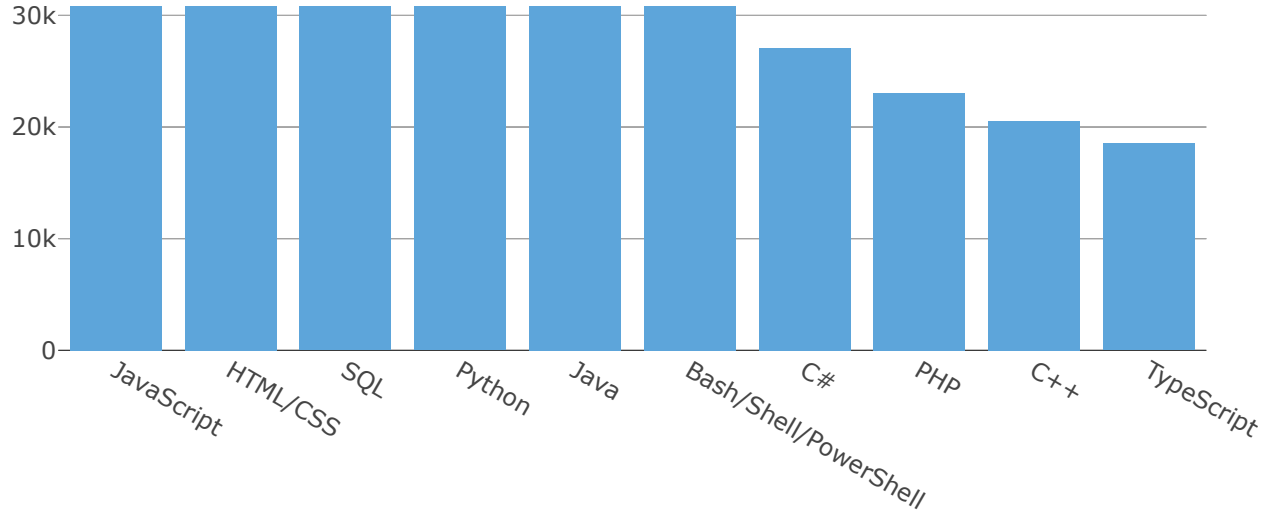
```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(sep_into_cols)` instead of `sep_into_cols` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
# Count the frequency of each language
language_counts <- languages_undelimited %>%
  count_options("LanguageWorkedWith", desc_order = TRUE)
```

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

```
plot_column_chart(
  head(language_counts, n = 10),
  "LanguageWorkedWith",
  "Frequency",
  "LanguageWorkedWith",
  "Top 10 Programming Languages Used",
  c("", ""),
  tech_stack_color
)
```



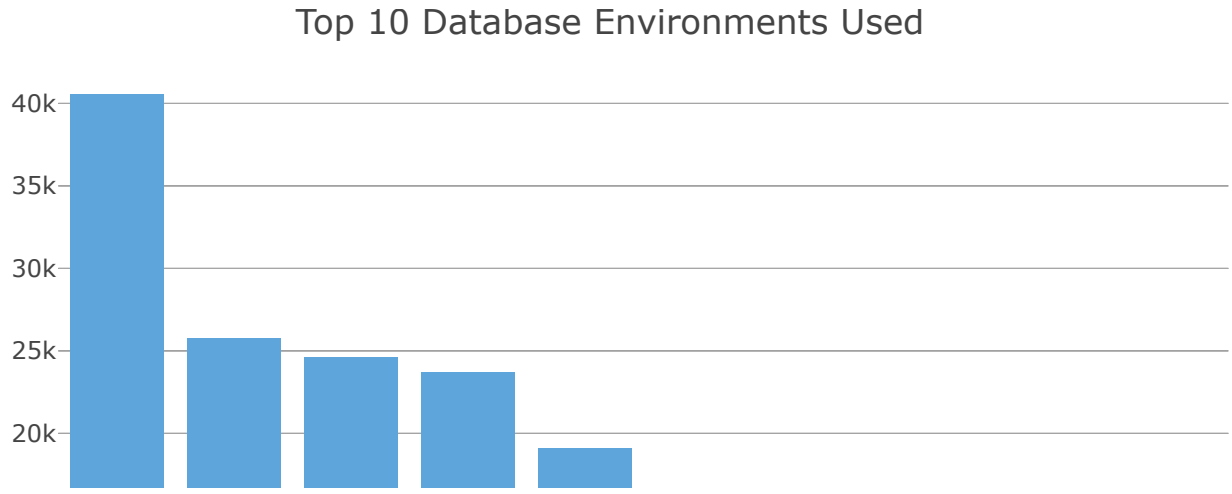


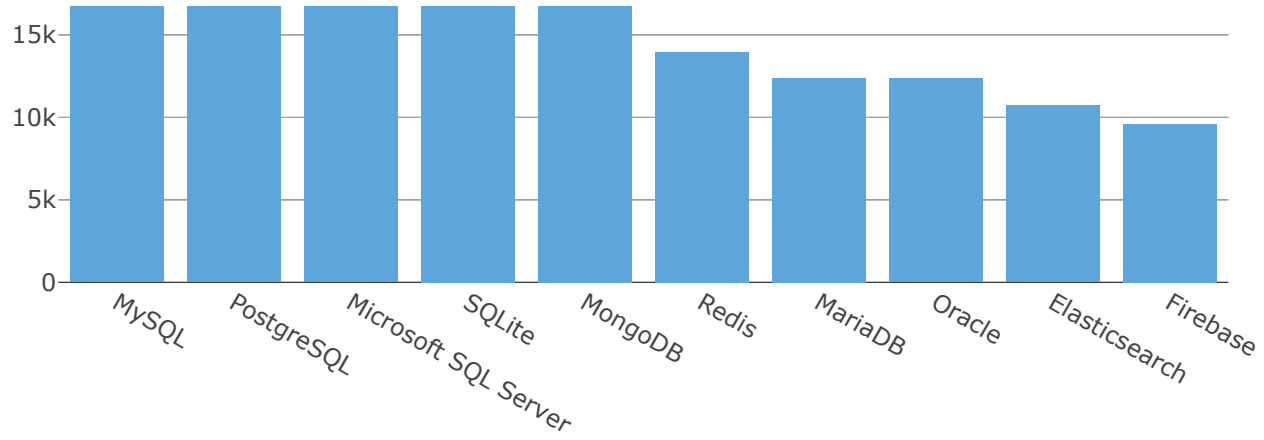
Database Environments

```
databases <- data %>%
  select("Respondent", "DatabaseWorkedWith")
# Split and unpivot the individual databases
databases_undelimited <- delimeted_to_df(databases, "DatabaseWorkedWith", ";")
# Count the frequency of each database
databases_counts <- databases_undelimited %>%
  count_options("DatabaseWorkedWith", desc_order=TRUE)
```

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

```
plot_column_chart(
  head(databases_counts, n=10),
  "DatabaseWorkedWith",
  "Frequency",
  "DatabaseWorkedWith",
  "Top 10 Database Environments Used",
  c("", ""),
  tech_stack_color
  )
```



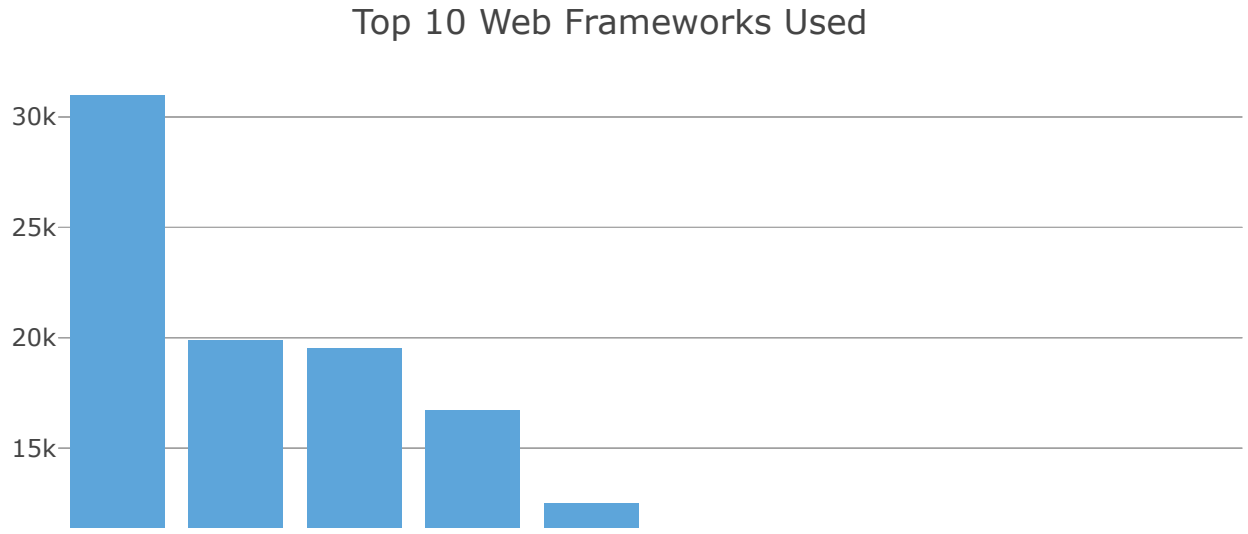


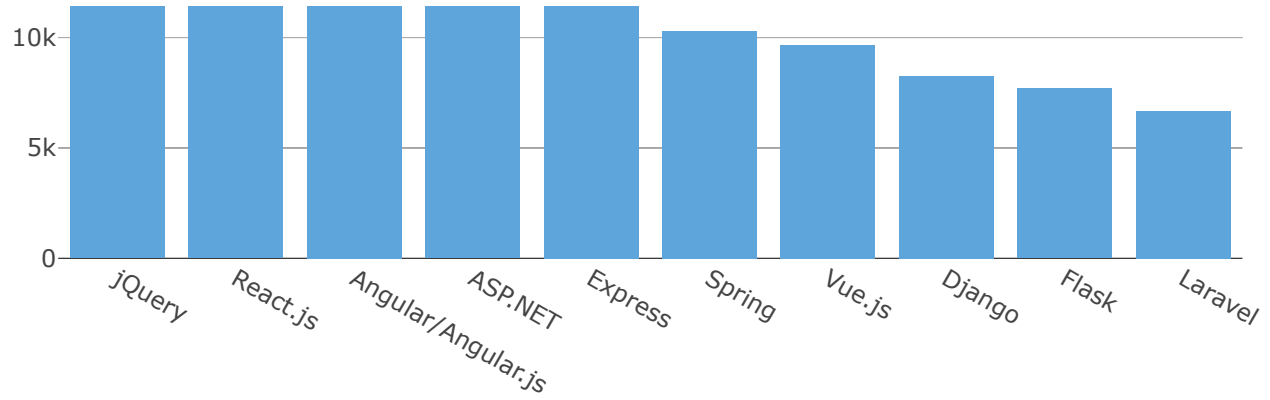
Web Frameworks

```
web_frameworks<- data %>%
  select("Respondent", "WebFrameWorkedWith")

# Split and unpivot the individual web frameworks
web_fw_undelimited <- delimeted_to_df(web_frameworks, "WebFrameWorkedWith", ";")
# Count the frequency of each web framework
web_fw_counts <- web_fw_undelimited %>%
  count_options("WebFrameWorkedWith", desc_order=TRUE)
```

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

```
plot_column_chart(
  head(web_fw_counts, n=10),
  "WebFrameWorkedWith",
  "Frequency",
  "WebFrameWorkedWith",
  "Top 10 Web Frameworks Used",
  c("", ""),
  tech_stack_color
  )
```



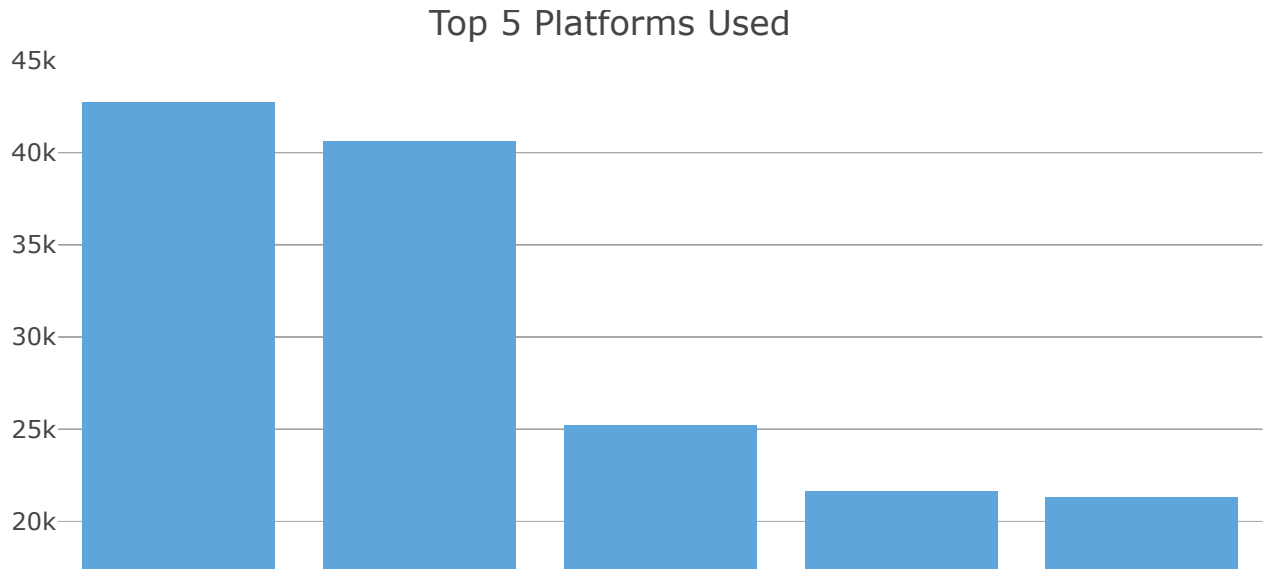


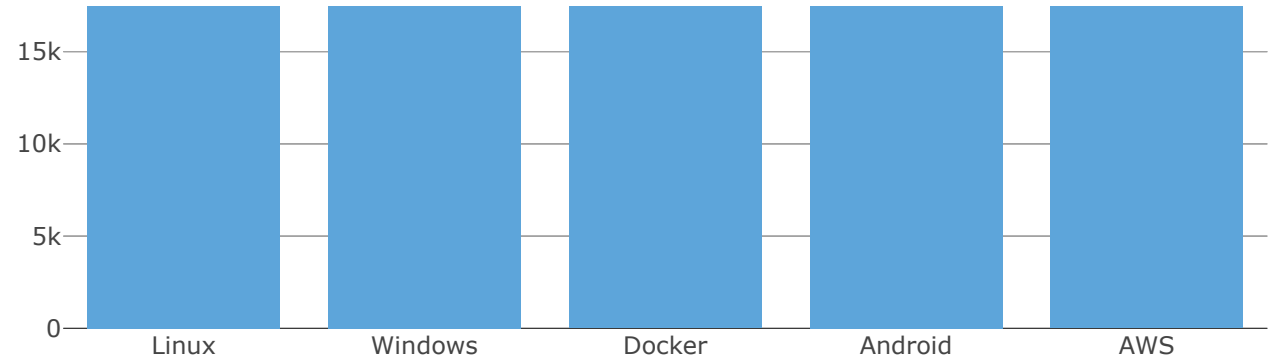
Platforms

```
platforms <- data %>%
  select("Respondent", "PlatformWorkedWith")

# Split and unpivot the individual platforms
platforms_undelimited <- delimeted_to_df(platforms, "PlatformWorkedWith", ";")
# Count the frequency of each platform
platforms_counts <- platforms_undelimited %>%
  count_options("PlatformWorkedWith", desc_order=TRUE)
```

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

```
plot_column_chart(
  head(platforms_counts, n=5),
  "PlatformWorkedWith",
  "Frequency",
  "PlatformWorkedWith",
  "Top 5 Platforms Used",
  c("", ""),
  tech_stack_color
  )
```



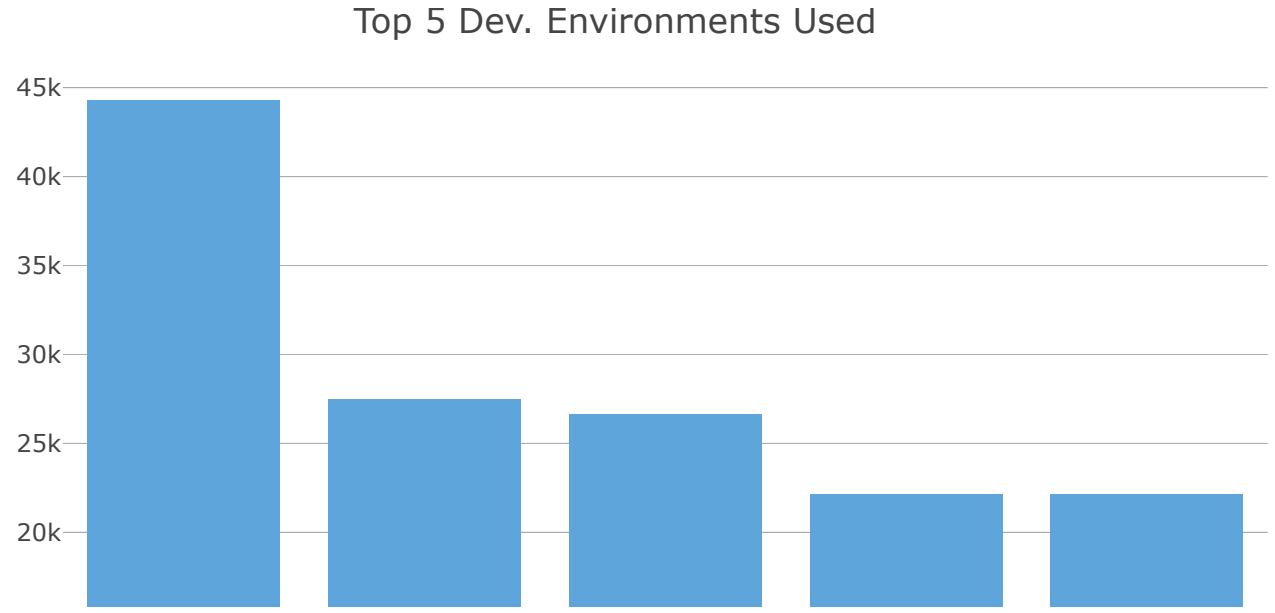


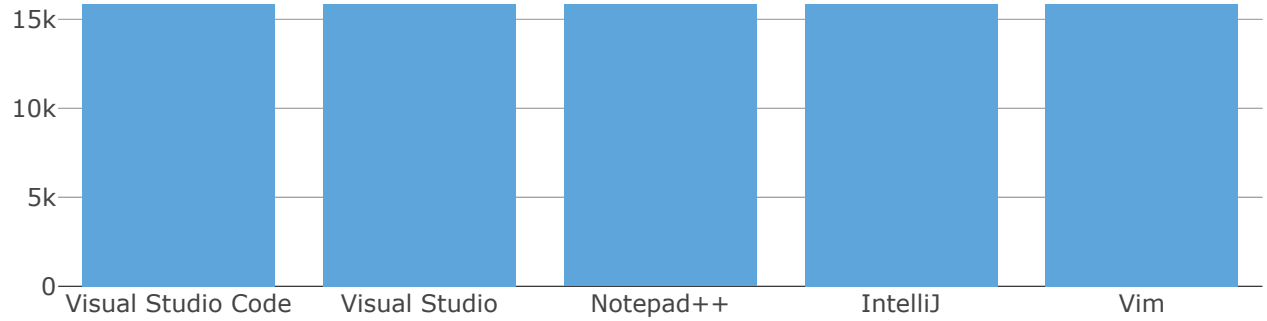
Development Environments

```
dev_envs <- data %>%
  select("Respondent", "DevEnviron")

# Split and unpivot the individual development environments
dev_envs_undelimited <- delimeted_to_df(dev_envs, "DevEnviron", ";")
# Count the frequency of each dev environment
dev_envs_counts <- dev_envs_undelimited %>%
  count_options("DevEnviron", desc_order=TRUE)
```

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

```
plot_column_chart(
  head(dev_envs_counts, n=5),
  "DevEnviron",
  "Frequency",
  "DevEnviron",
  "Top 5 Dev. Environments Used",
  c("", ""),
  tech_stack_color
  )
```



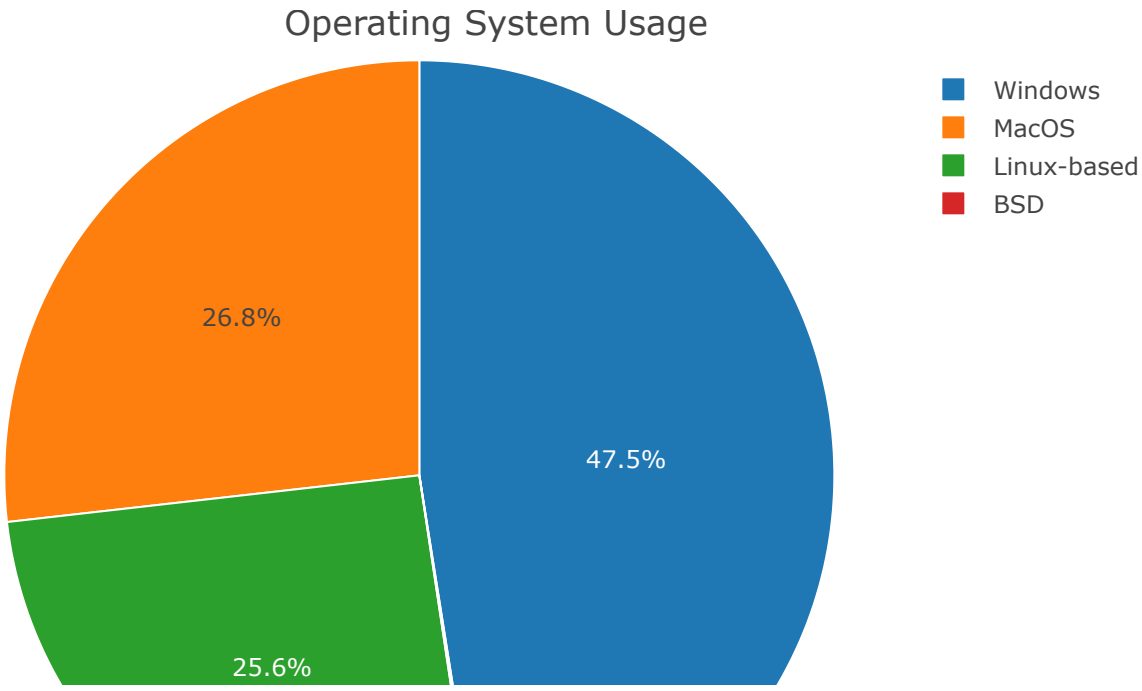


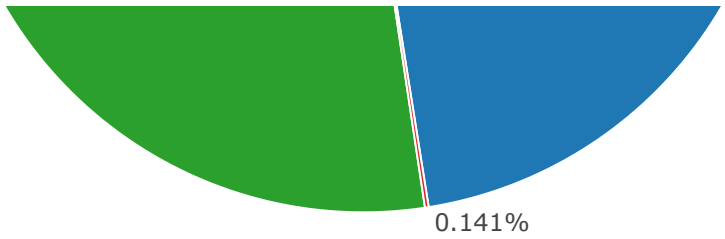
Operating Systems

```
# Count the frequency of each operating system
op_sys <- data[,"OpSys"] %>%
  count_options("OpSys")
```

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

```
# Remove missing values
op_sys <- op_sys[!is.na(op_sys$OpSys),]

marker_options <- list(
  color="Blues",
  line = list(color = "#FFFFFF", width = 1))

plot_pie(
  op_sys,
  "OpSys",
  "Frequency",
  "Operating System Usage",
  marker_options)
```





Key Takeaways

- Over 60% of the respondents use JavaScript and/or HTML/CSS, over 50% use SQL and 40% use Python and/or Java;
- Over 46% of the respondents use MySQL for database environments. On the other hand, only about 23% use PostgreSQL and/or Microsoft SQL Server, the second and third most popular environments, respectively; jQuery is still the most used Web Framework, used by almost a third of the inquired. React.js comes in second place (22%) and Angular(.js) in a very close third place;
- In a group of 10 developers, about 5 of them reported using Visual Studio Code as one of their IDEs/Development Environments. About 3 people in the group use Visual Studio and/or Notepad++ as well;
- Windows is the operating system of choice of almost half of the respondents. The remaining half is fairly evenly divided between MacOS and Linux-based systems, but the former takes second place.

Professional Life

Professional Usage of Programming

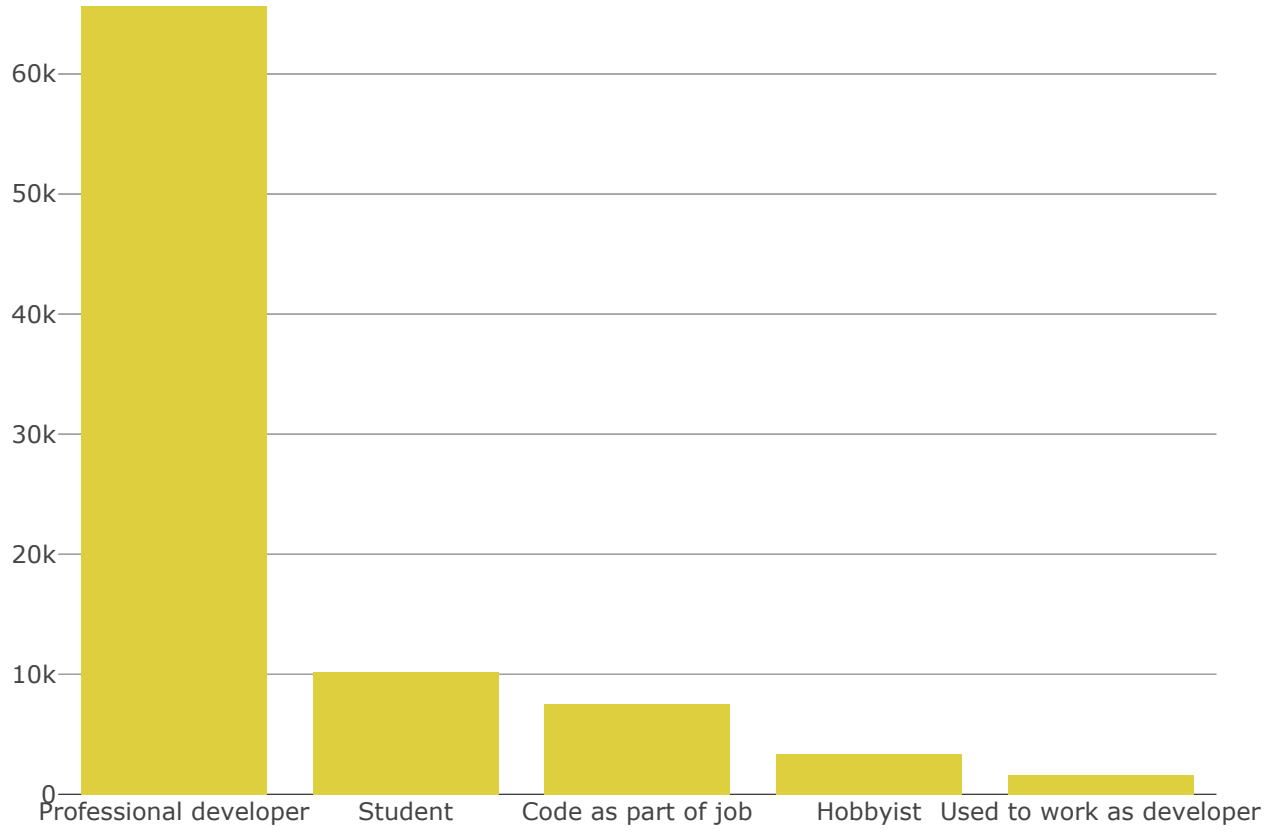
```
# Rename the usage of programming categories and count their frequencies
prog_usage <- data %>%
  rename_options("MainBranch", rename_prog_usage) %>%
  count_options("MainBranch", desc_order = TRUE)
```

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

```
plot_column_chart(
  prog_usage,
  "MainBranch",
  "Frequency",
  "MainBranch",
  "Professional Usage of Programming",
  c("", ""),
  prof_life_color
)
```

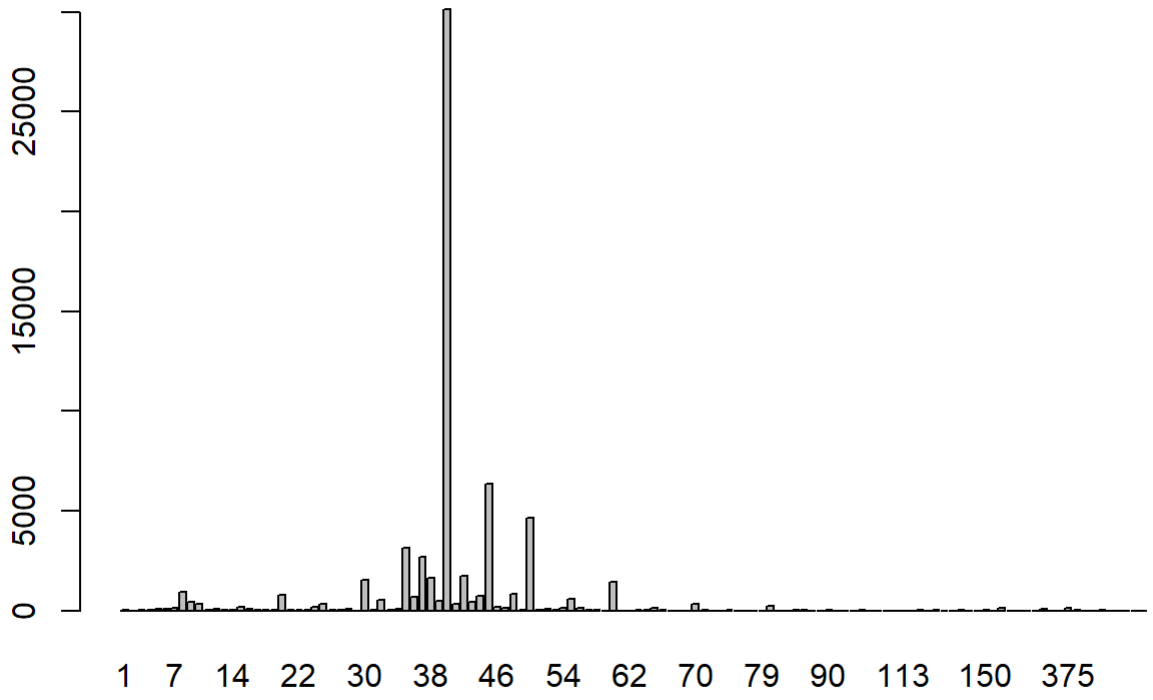
```
## Warning: Ignoring 1 observations
```

Professional Usage of Programming



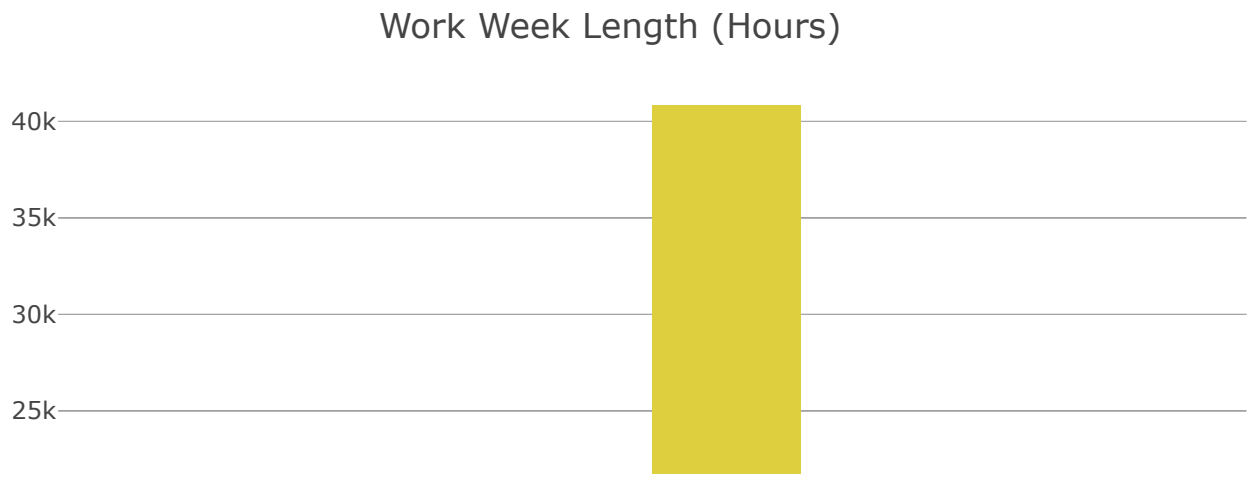
Work Week Hours

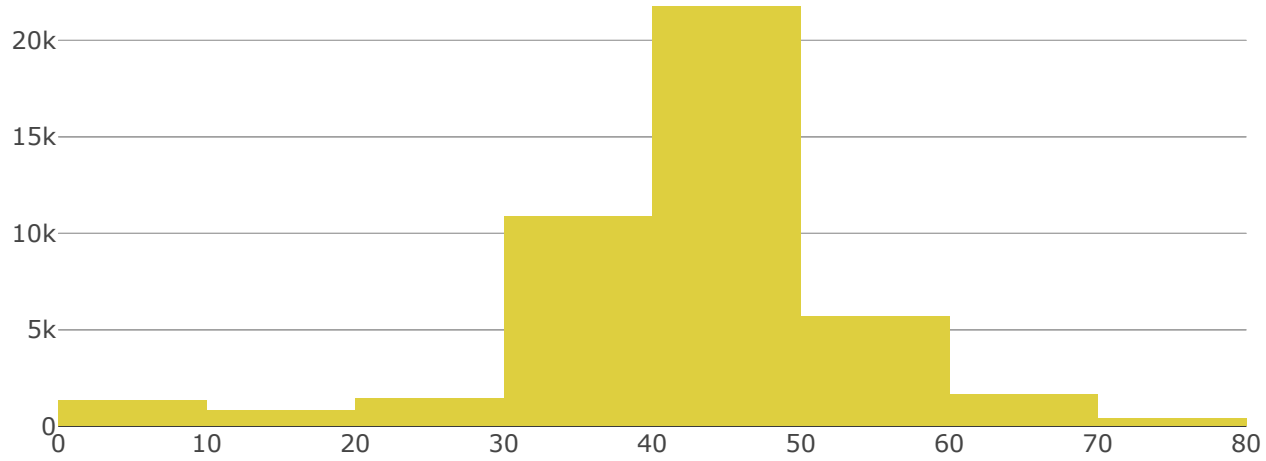
```
# Explore the data distribution
work_week_hrs <- as.integer(data$WorkWeekHrs)
temp <- table(work_week_hrs)
barplot(temp)
```




```
# Keep only answers between 8 and 80 work week hours
work_week_hrs <-work_week_hrs[
  work_week_hrs >= 8 &
  work_week_hrs < 80]
work_hours_bins <- list(start = 0, end = 80, size = 10)
plot_histogram(work_week_hrs,
               work_hours_bins,
               "Work Week Length (Hours)",
               c("", ""),
               prof_life_color)
```

```
## Warning: Ignoring 24380 observations
```



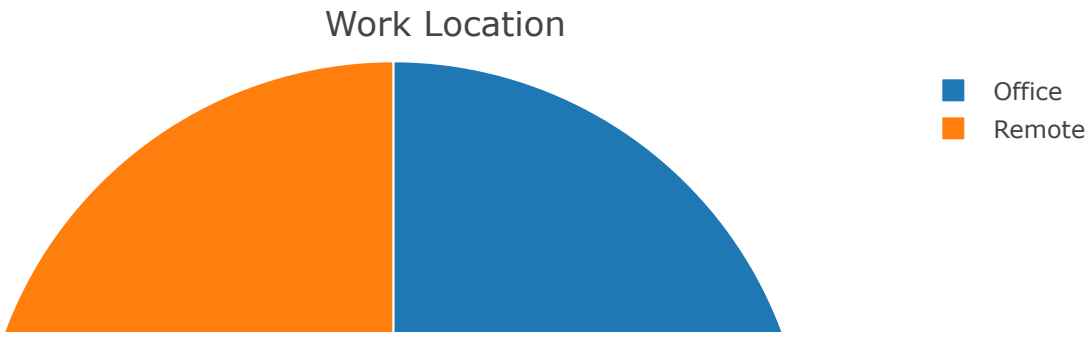


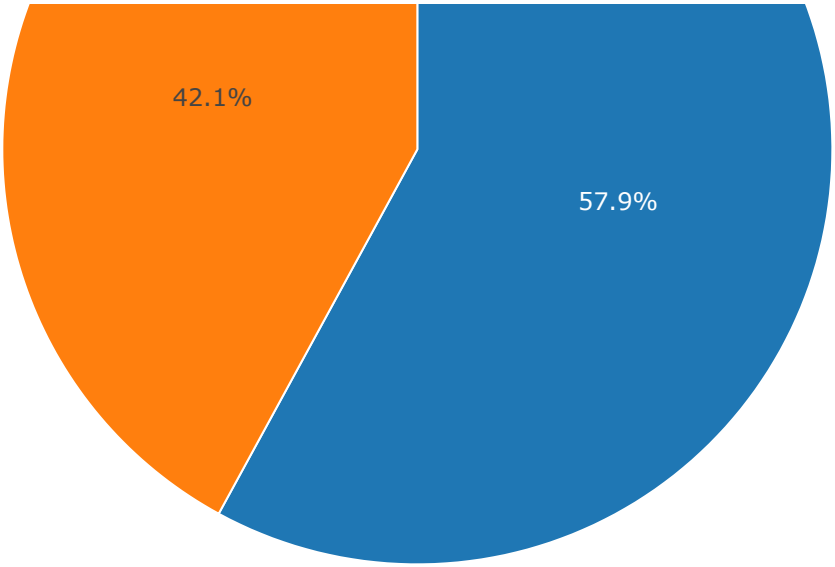
Work Location

```
work_loc <- data %>%
  select("Respondent", "WorkLoc")

# Rename the work location categories and replace the column in the dataframe
work_loc_renamed <- work_loc %>%
  rename_options("WorkLoc", rename_work_loc)
work_loc$WorkLoc <- work_loc_renamed$WorkLoc
work_loc <- work_loc[!is.na(work_loc$WorkLoc),]

# Count the work location categories' frequencies
work_loc_plot <- work_loc %>%
  count_options("WorkLoc")
```

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

```
marker_options <- list(
  color="Yellows",
  line = list(color = "#FFFFFF", width = 1))

plot_pie(
  work_loc_plot,
  "WorkLoc",
  "Frequency",
  "Work Location",
  marker_options)
```



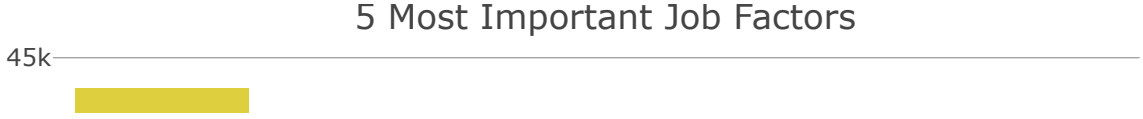


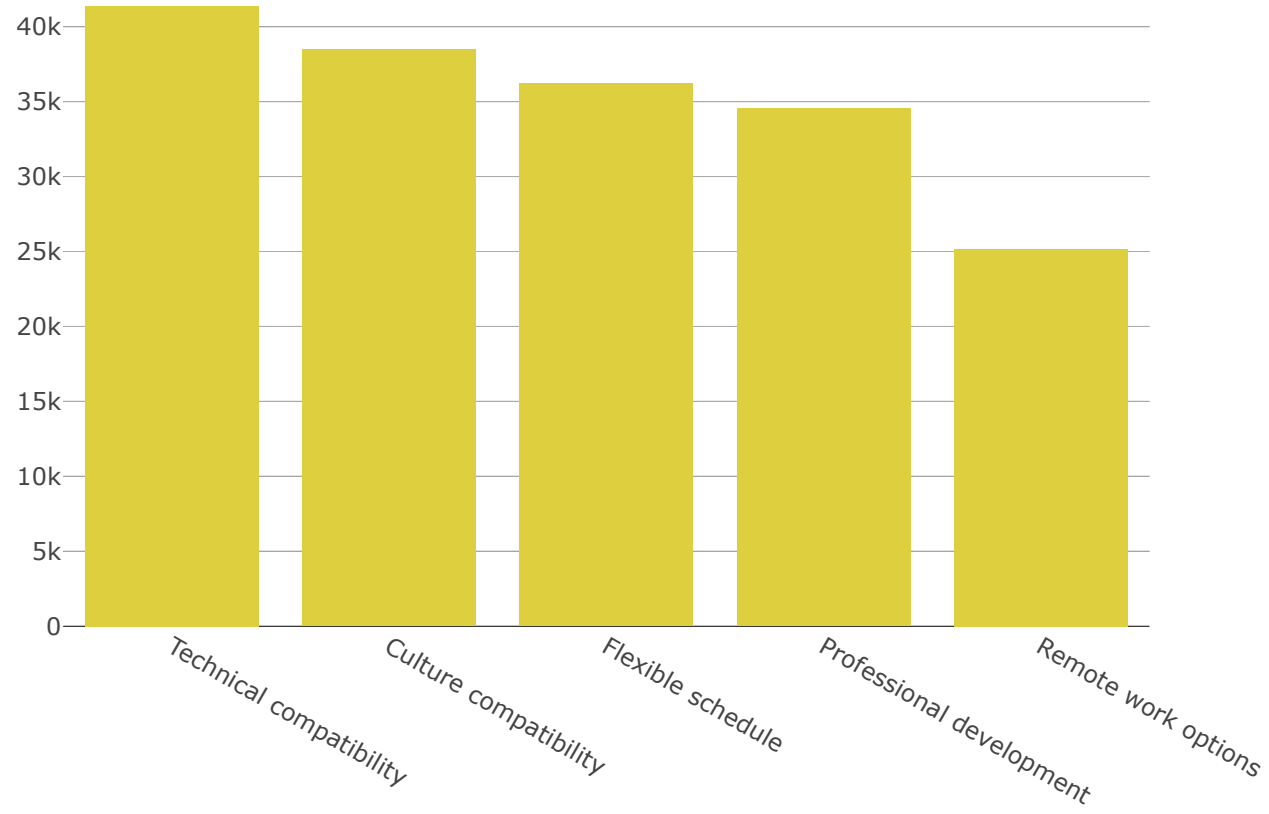
Most Important Job Factors

```
job_factors <- data %>%
  select("Respondent", "JobFactors")

# Split and unpivot the individual job factors
job_factors_undelimited <- delimeted_to_df(job_factors, "JobFactors", ";")

# Rename the categories
job_factors_renamed <- job_factors_undelimited %>%
  rename_options("JobFactors", rename_job_factors)
# Replace column of original categories with the renamed categories
job_factors_undelimited[,"JobFactors"] <- job_factors_renamed

# Count frequencies
job_factors_counts <- job_factors_undelimited %>%
  count_options("JobFactors", desc_order=TRUE)
```

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

```
plot_column_chart(
  head(job_factors_counts, n=5),
  "JobFactors",
  "Frequency",
  "JobFactors",
  "5 Most Important Job Factors",
  c("", ""),
  prof_life_color
  )
```





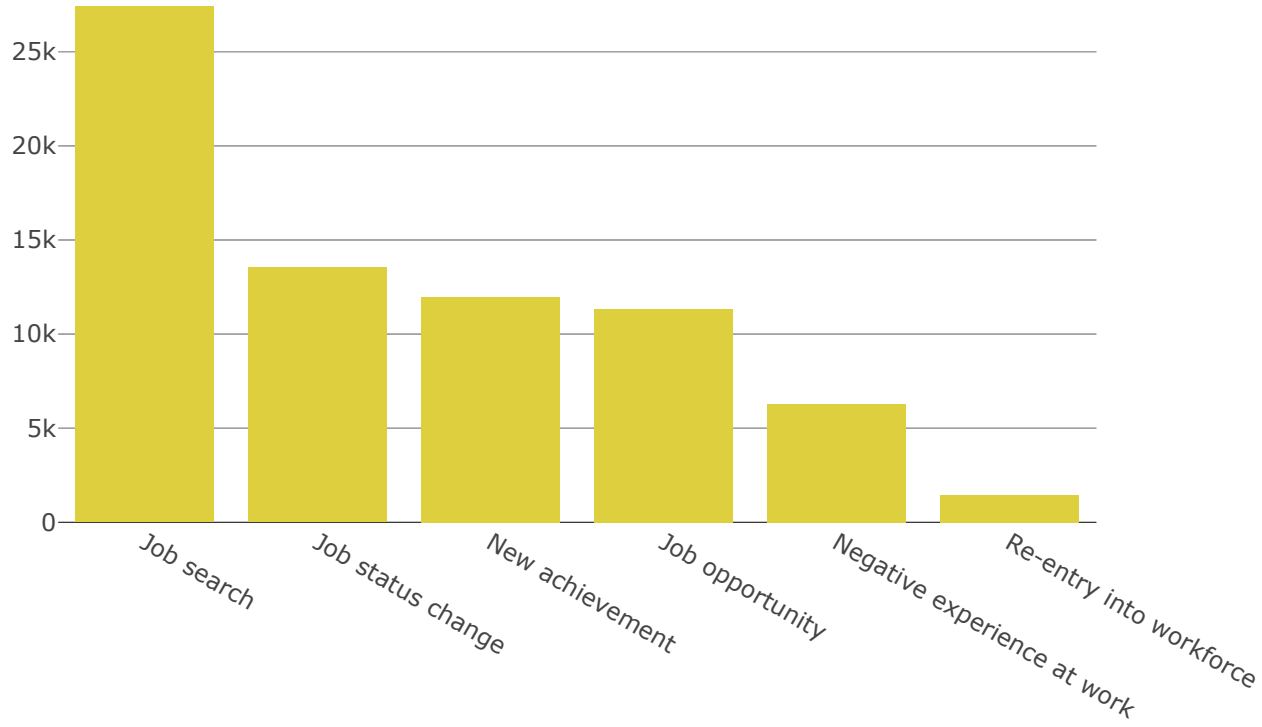
Reason for latest Resume Update

```
# Rename the resume update reason categories and count their frequencies
resume_update <- data %>%
  rename_options("ResumeUpdate", rename_resume_update) %>%
  count_options("ResumeUpdate", desc_order = TRUE)
```

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

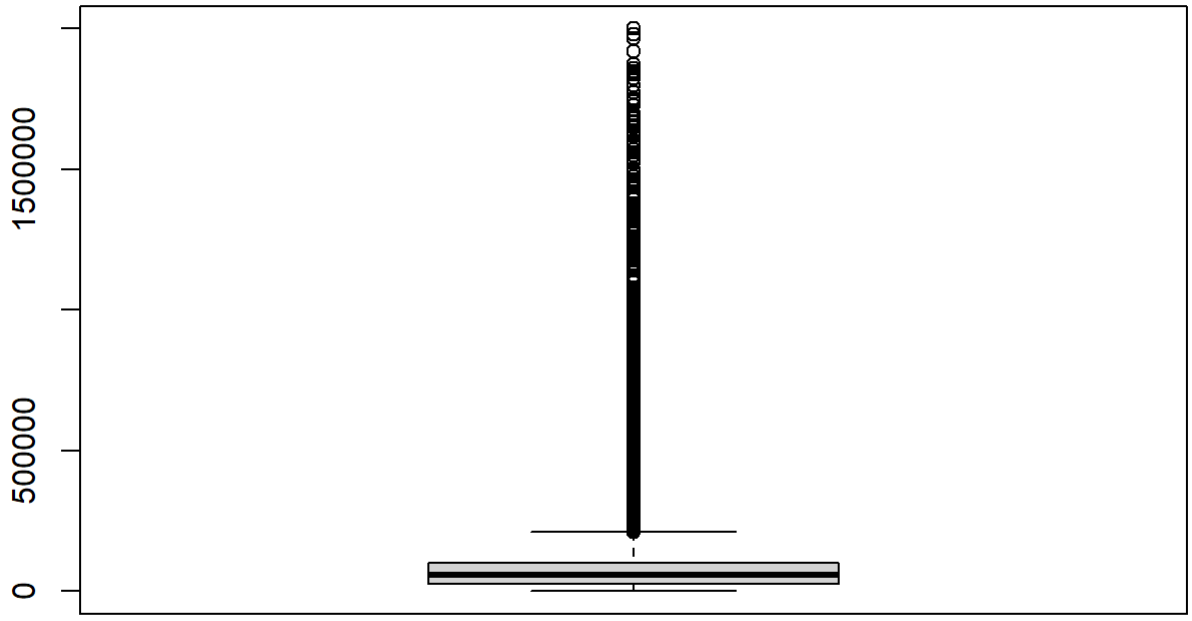
```
plot_column_chart(
  resume_update,
  "ResumeUpdate",
  "Frequency",
  "ResumeUpdate",
  "Reason for Last Resume Update",
  c("", ""),
  prof_life_color
)
```

```
## Warning: Ignoring 1 observations
```



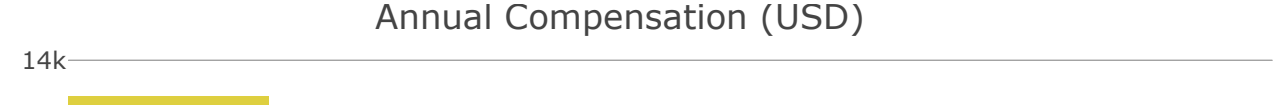
Annual Compensation (USD)

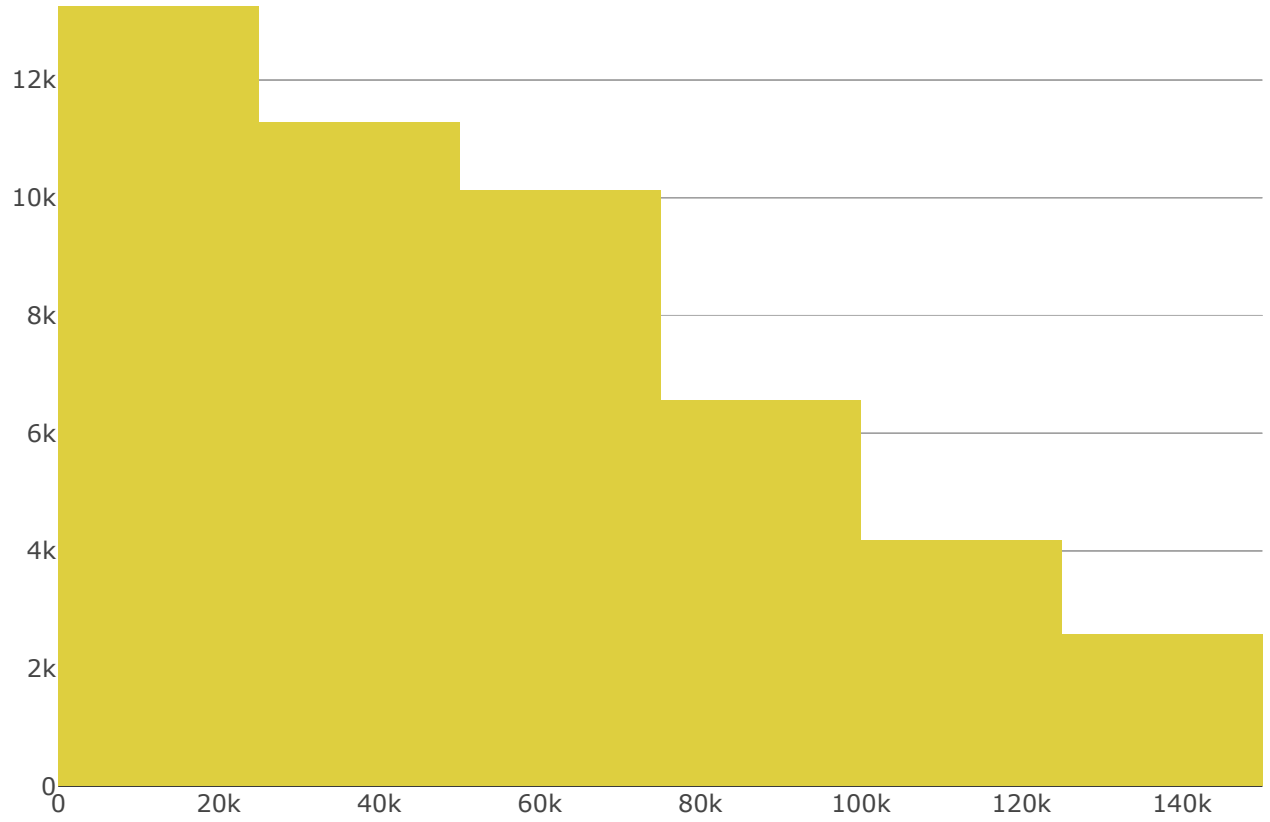
```
# Explore the data distribution
compensation <- as.integer(data$ConvertedComp)
boxplot(compensation)
```




```
annual_compensation <- data %>%
  select("Respondent", "ConvertedComp")

# Cast compensation to integer and keep only respondents with compensation between 0 and 150k US
D
annual_compensation$ConvertedComp <- as.integer(
  annual_compensation$ConvertedComp)
annual_compensation <- annual_compensation[
  annual_compensation$ConvertedComp > 0 &
  annual_compensation$ConvertedComp <= 150000
  ,]

comp_bins <- list(start=0, end=150000, size=25000)

plot_histogram(
  annual_compensation$ConvertedComp,
  comp_bins,
  "Annual Compensation (USD)",
  c("", ""),
  prof_life_color)
```

```
## Warning: Ignoring 33060 observations
```





Key Takeways

- Almost three quarters of the inquired are developers by profession. Over 10% of the respondents are students and 8% need to code as part of their job;
- 44% of the respondents work between 40 to 50 hours per week. 11% work between 30 to 40 hours and 6% between 50 to 60 hours;
- In a group of 10 people, close to 6 of them work at an office and 4 work remotely;
- Technical compatibility is a deal-breaker for almost half of the inquired when looking for a job. However, culture compatibility, flexible schedule and opportunity for professional development are equally important for about 40% of the respondents;
- 37% of the respondents' most recent resume update was to search for a job. A change in job status came in second, reported by 15% of the inquired;
- 16% of the inquired reported an annual compensation of 25,000$ or less. 13% reported an annual compensation between 25,000$ and 50,000$ and 11% between 50,000$ and 75,000$. On the other hand, only 16% reported to receive more than 75,000$.

Conclusion

General Data

- Over 40% of the respondents were between the ages of 20 and 30 and almost 70% were between 20 and 40 years-old;

- Over 90% of the inquired were male, with less than 8% being of the female gender. The remaining 2% identified as another gender;
- About 43% of the respondents completed a Bachelor's Degree, but only half of those have gone on to complete a Master's Degree;
- More than half of respondents come from a Computer Science or Software Engineering academic background. The remaining inquired come primarily from other Engineering disciplines, Information Technologies, System Administration or Web Design and/or Development;
- 70% of the respondents have a full-time job, with freelance and/or self-employment being the second most common employment status;
- The five countries with most respondents are, from most to least, the United States of America, India, Germany, the United Kingdom and Canada.

Tech Stack

- Over 60% of the respondents use JavaScript and/or HTML/CSS, over 50% use SQL and 40% use Python and/or Java;
- Over 46% of the respondents use MySQL for database environments. On the other hand, only about 23% use PostgreSQL and/or Microsoft SQL Server, the second and third most popular environments, respectively; jQuery is still the most used Web Framework, used by almost a third of the inquired. React.js comes in second place (22%) and Angular(.js) in a very close third place;
- In a group of 10 developers, about 5 of them reported using Visual Studio Code as one of their IDEs/Development Environments. About 3 people in the group use Visual Studio and/or Notepad++ as well;
- Windows is the operating system of choice of almost half of the respondents. The remaining half is fairly evenly divided between MacOS and Linux-based systems, but the former takes second place.

Professional Life

- Almost three quarters of the inquired are developers by profession. Over 10% of the respondents are students and 8% need to code as part of their job;
- 44% of the respondents work between 40 to 50 hours per week. 11% work between 30 to 40 hours and 6% between 50 to 60 hours;
- In a group of 10 people, close to 6 of them work at an office and 4 work remotely;
- Technical compatibility is a deal-breaker for almost half of the inquired when looking for a job. However, culture compatibility, flexible schedule and opportunity for professional development are equally important for about 40% of the respondents;
- 37% of the respondents' most recent resume update was to search for a job. A change in job status came in second, reported by 15% of the inquired;
- 16% of the inquired reported an annual compensation of 25,000$ or less. 13% reported an annual compensation between 25,000$ and 50,000$ and 11% between 50,000$ and 75,000$. On the other hand, only 16% reported to receive more than 75,000$.

```
# Clear environment variables
rm(list = ls())
```