1. Welcome to the world of data science

Throughout the world of data science, there are many languages and tools that can be used to complete a given task. While you are often able to use whichever tool you prefer, it is often important for analysts to work with similar platforms so that they can share their code with one another. Learning what professionals in the data science industry use while at work can help you gain a better understanding of things that you may be asked to do in the future.

In this project, we are going to find out what tools and languages professionals use in their day-to-day work. Our data comes from the Kaggle Data Science Survey (https://www.kaggle.com/kaggle/kaggle-survey-2017?utm_medium=partner&utm_source=datacamp.com&utm_campaign=ml+survey+case+study) which includes responses from over 10,000 people that write code to analyze data in their daily work.

In [2]:
```
# Loading necessary packages
library(tidyverse)

# Loading the data
responses <- read_csv("datasets/kagglesurvey.csv")

# Printing the first 10 rows
head(responses,10)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.2.0
--
<U+221A> ggplot2 3.0.0     <U+221A> purrr   0.2.5
<U+221A> tibble  1.4.2     <U+221A> dplyr   0.7.6
<U+221A> tidyr   0.8.1     <U+221A> stringr 1.3.1
<U+221A> readr   1.1.1     <U+221A> forcats 0.3.0
-- Conflicts ------------------------------------------ tidyverse_conflicts()
--
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
Parsed with column specification:
cols(
  Respondent = col_integer(),
  WorkToolsSelect = col_character(),
  LanguageRecommendationSelect = col_character(),
  EmployerIndustry = col_character(),
  WorkAlgorithmsSelect = col_character()
)
```

Respondent	WorkToolsSelect
1	Amazon Web services,Oracle Data Mining/ Oracle R Enterprise,Perl
2	Amazon Machine Learning,Amazon Web services,Cloudera,Hadoop/Hive/Pig,Impala,Java,Mathematica,MATLAB/Octave,Mi Excel Data Mining,Microsoft SQL Server Data Mining,NoSQL,Python,R,SAS Base, JMP,SQL,Tableau
3	C/C++,Jupyter notebooks,MATLAB/Octave,Python,R,TensorFlow
4	Jupyter notebooks,Python,SQL,TensorFlow
5	C/C++,Cloudera,Hadoop/Hive/Pig,Java,NoSQL,R,Unix shell / awk
6	SQL
7	Jupyter notebooks,NoSQL,Python,R,SQL,Unix shell / awk
8	Python,Spark / MLlib,Tableau,TensorFlow,Other
9	Jupyter notebooks,MATLAB/Octave,Python,SAS Base,SQL
10	C/C++,IBM Cognos,MATLAB/Octave,Microsoft Excel Data Mining,Microsoft R Serv (Formerly Revolution Analytics),Microsoft SQL Server Data Mining,Perl,Python,R,SQL,Unix shell / awk

In [3]:
```
library("testthat")
library('IRkernel.testthat')

run_tests({
    test_that("Read in data correctly.", {
        expect_is(responses, "tbl_df",
            info = 'You should use read_csv (with an underscore) to read "data
sets/kagglesurvey.csv" into responses')
    })

    test_that("Read in data correctly.", {
        responses_test <- read_csv('datasets/kagglesurvey.csv')
        expect_equivalent(responses, responses_test,
            info = 'responses should contain the data in "datasets/kagglesurve
y.csv"')
    })

})
```

```
Attaching package: 'testthat'

The following object is masked from 'package:dplyr':

    matches

The following object is masked from 'package:purrr':

    is_null


<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 1.637 0.084 35.587 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

2. Using multiple tools

Now that we've loaded in the survey results, we want to focus on the tools and languages that the survey respondents use at work.

In [4]:
```
# Printing the first respondents' tools and languages
print(responses$WorkToolsSelect[1])
print(responses$LanguageRecommendationSelect[1])

# Creating a new data frame called tools
tools <- data.frame(responses$WorkToolsSelect)

# Adding a new column to tools which splits the WorkToolsSelect column at the
 commas and unnests the new column
tools <- tools  %>%
  mutate(work_tools = str_split(responses$WorkToolsSelect,',')) %>%
  unnest(work_tools)

# Viewing the first 6 rows of tools
head(tools)
```

```
[1] "Amazon Web services,Oracle Data Mining/ Oracle R Enterprise,Perl"
[1] "F#"
```

responses.WorkToolsSelect	**work_**
Amazon Web services,Oracle Data Mining/ Oracle R Enterprise,Perl	Amaz Web servic
Amazon Web services,Oracle Data Mining/ Oracle R Enterprise,Perl	Oracle Data Mining Oracle Enterp
Amazon Web services,Oracle Data Mining/ Oracle R Enterprise,Perl	Perl
Amazon Machine Learning,Amazon Web services,Cloudera,Hadoop/Hive/Pig,Impala,Java,Mathematica,MATLAB/Octave,Microsoft Excel Data Mining,Microsoft SQL Server Data Mining,NoSQL,Python,R,SAS Base,SAS JMP,SQL,Tableau	Amaz Machi Learni
Amazon Machine Learning,Amazon Web services,Cloudera,Hadoop/Hive/Pig,Impala,Java,Mathematica,MATLAB/Octave,Microsoft Excel Data Mining,Microsoft SQL Server Data Mining,NoSQL,Python,R,SAS Base,SAS JMP,SQL,Tableau	Amaz Web servic
Amazon Machine Learning,Amazon Web services,Cloudera,Hadoop/Hive/Pig,Impala,Java,Mathematica,MATLAB/Octave,Microsoft Excel Data Mining,Microsoft SQL Server Data Mining,NoSQL,Python,R,SAS Base,SAS JMP,SQL,Tableau	Cloud

In [5]:
```
run_tests({
    test_that("Tools and Languages were Split and Unnested", {
        expect_true(nrow(tools) == 47409,
            info = 'Make sure that you split the tools at the commas and unnes
ted them')
    })

    test_that("Tools and Languages were Unnested", {
        expect_is(tools$work_tools, "character",
            info = 'The work_tools column should be of class "character". Make
 sure that you unnested the results of strsplit()')
    })

})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.004 0.088 35.957 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

3. Counting users of each tool

Now that we've split apart all of the tools used by each respondent, we can figure out which tools are the most popular.

```
In [6]: # Creating a new data frame
        tool_count <- data.frame(tools)

        # Grouping the data by work_tools, calculate the number of responses in each g
        roup
        tool_count <- tool_count  %>%
          group_by(work_tools) %>%
          summarize(count = n()) %>%


        # Sorting tool_count so that the most popular tools are at the top
        arrange(desc(count))

        # Printing the first 6 results
        head(tool_count)
```

work_tools	**count**
Python	6073
R	4708
SQL	4261
Jupyter notebooks	3206
TensorFlow	2256
NA	2198

In [7]:
```
run_tests({
    test_that("Tools were Grouped and Summarised", {
        expect_true(nrow(tool_count) == 50,
            info = 'Make sure that you grouped by tools and then summarised')
    })

    test_that("Values were sorted correctly", {
        expect_true(tool_count[1, 2] == 6073,
            info = 'Do not forget to sort your tool counts from largest to sma
llest')
    })

})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.07 0.096 36.031 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

4. Plotting the most popular tools

Let's see how your favorite tools stack up against the rest.

In [8]:
```
# Creating a bar chart of the work_tools column.
# Arranging the bars so that the tallest are on the far right
ggplot(tool_count, aes(x=work_tools, y= count)) +

# Arranging the bars so that the tallest are on the far right
    geom_bar() +

# Rotating the bar labels 90 degrees
    theme(axis.text.x=element_text(angle=90, hjust=1))
```

```
ERROR while rich displaying an object: Error: stat_count() must not be used w
ith a y aesthetic.

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     rpr <- mime2repr[[mime]](obj)
 .     if (is.null(rpr)) 
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     rpr <- mime2repr[[mime]](obj)
 .     if (is.null(rpr)) 
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. mime2repr[[mime]](obj)
8. repr_text.default(obj)
9. paste(capture.output(print(obj)), collapse = "\n")
10. capture.output(print(obj))
11. evalVis(expr)
12. withVisible(eval(expr, pf))
13. eval(expr, pf)
14. eval(expr, pf)
15. print(obj)
16. print.ggplot(obj)
17. ggplot_build(x)
18. ggplot_build.ggplot(x)
19. by_layer(function(l, d) l$compute_statistic(d, layout))
20. f(l = layers[[i]], d = data[[i]])
21. l$compute_statistic(d, layout)
22. f(..., self = self)
23. self$stat$setup_params(data, self$stat_params)
24. f(...)
25. stop("stat_count() must not be used with a y aesthetic.", call. = FALSE)
```

In [9]:
```
run_tests({
   test_that("Plot is a bar chart",{
      p <- last_plot()
      q <- p$layers[[1]]
      expect_is(q$geom, "GeomBar",
                info = "You should plot a bar chart with ggplot")
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.21 0.096 36.17 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

5. The R vs Python debate

Within the field of data science, there is a lot of debate among professionals about whether R or Python should reign supreme. You can see from our last figure that R and Python are the two most commonly used languages, but it's possible that many respondents use both R and Python. Let's take a look at how many people use R, Python, and both tools.

In [10]:
```
debate_tools <- data.frame(responses)

# Creating a new column called language preference, based on the conditions sp
ecified in the Instructions
debate_tools <- debate_tools  %>%
   mutate(language_preference = case_when(
     str_detect(WorkToolsSelect,"R")      & !str_detect(WorkToolsSelect,"Pytho
n")  ~ "R",
     str_detect(WorkToolsSelect,"Python") & !str_detect(WorkToolsSelect,"R")
     ~ "Python",
     str_detect(WorkToolsSelect,"R")      & str_detect(WorkToolsSelect,"Pytho
n")   ~ "both",
    !str_detect(WorkToolsSelect,"R")      & !str_detect(WorkToolsSelect,"Pytho
n")  ~ "neither"))

# Printing the first 6 rows
head(debate_tools)
```

Respondent	WorkToolsSelect
1	Amazon Web services,Oracle Data Mining/ Oracle R Enterprise,Perl
2	Amazon Machine Learning,Amazon Web services,Cloudera,Hadoop/Hive/Pig,Impala,Java,Mathematica,MATLAB/Octave,Mi Excel Data Mining,Microsoft SQL Server Data Mining,NoSQL,Python,R,SAS Base, JMP,SQL,Tableau
3	C/C++,Jupyter notebooks,MATLAB/Octave,Python,R,TensorFlow
4	Jupyter notebooks,Python,SQL,TensorFlow
5	C/C++,Cloudera,Hadoop/Hive/Pig,Java,NoSQL,R,Unix shell / awk
6	SQL

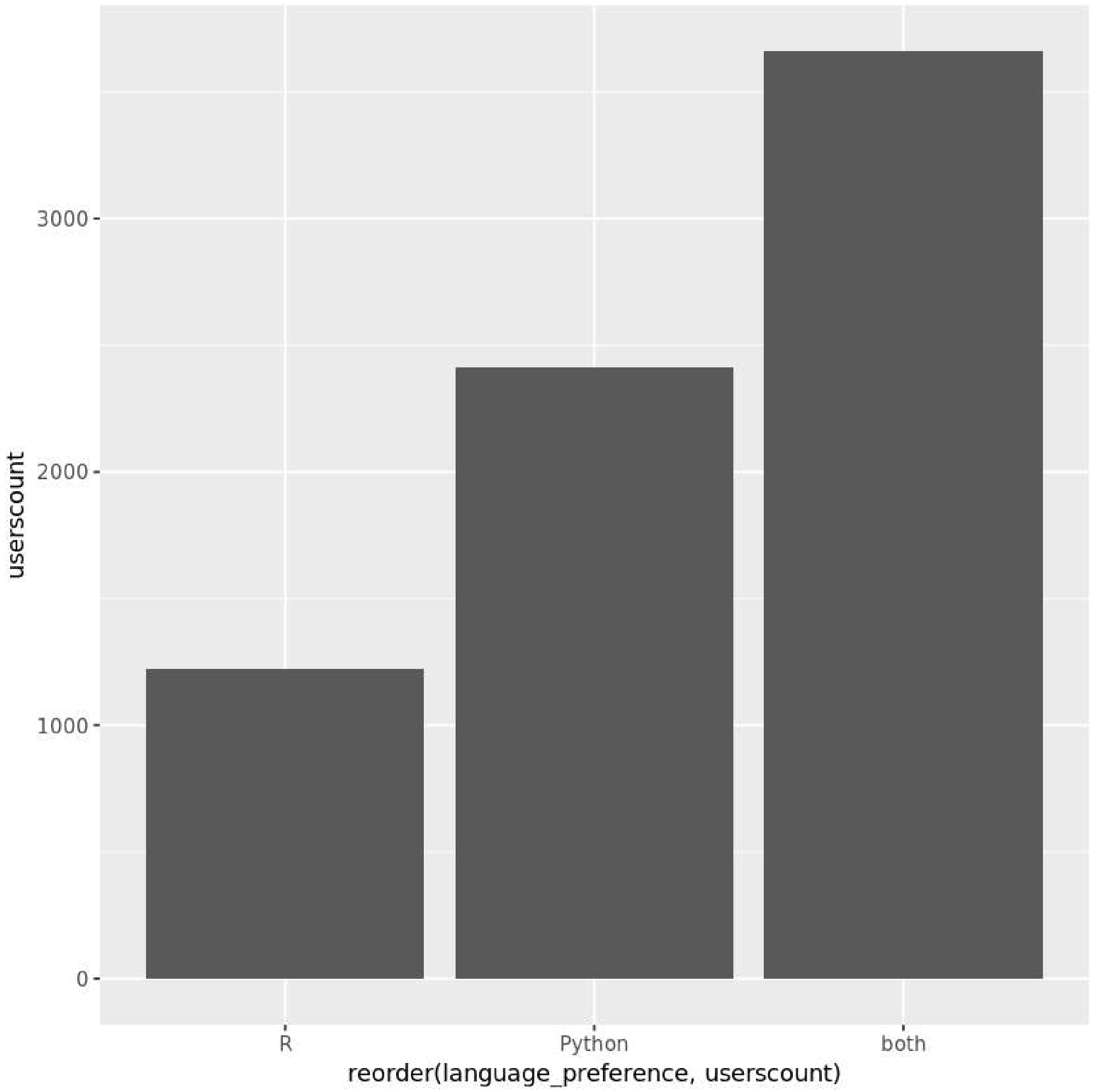
In [11]:
```
run_tests({
    test_that("New column was created", {
        expect_is(debate_tools$language_preference, "character",
            info = 'The language_preference column should be of class "charact
er". Make sure that you filled this new column correctly')
    })

})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.28 0.096 36.24 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

6. Plotting R vs Python users

Now we just need to take a closer look at how many respondents use R, Python, and both!

In [12]:
```
# Creating a new data frame
debate_plot <- data.frame(debate_tools)

# Grouping by language preference and calculate number of responses
debate_plot <- debate_plot  %>%
   group_by(language_preference)  %>%
   summarize(userscount = n()) %>%

# Removing the row for users of "neither"
    filter(!language_preference == "neither")

# Creating a bar chart
 ggplot(debate_plot, aes(x= reorder(language_preference ,userscount) , y = use
rscount)) + geom_bar(stat = "identity")
```



In [13]:
```
run_tests({
   test_that("Plot is a bar chart",{
      p <- last_plot()
      q <- p$layers[[1]]
      expect_is(q$geom, "GeomBar",
               info = "You should plot a bar chart with ggplot")
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.952 0.096 36.939 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

7. Language recommendations

It looks like the largest group of professionals program in both Python and R. But what happens when they are asked which language they recommend to new learners? Do R lovers always recommend R?

In [14]:
```
recommendations <- data.frame(debate_tools)

 # Grouping by language_preference and then LanguageRecommendationSelect
 recommendations <- recommendations  %>%
   group_by(language_preference, LanguageRecommendationSelect) %>%
   summarize(langRecomm = n()) %>%


 # Removing empty responses and include the top recommendations
 filter(!is.na(LanguageRecommendationSelect ) , !is.na(language_preference ) )
  %>%
   mutate( rowNum = row_number(langRecomm)) %>%
   filter(rowNum %in% 1:4)
```

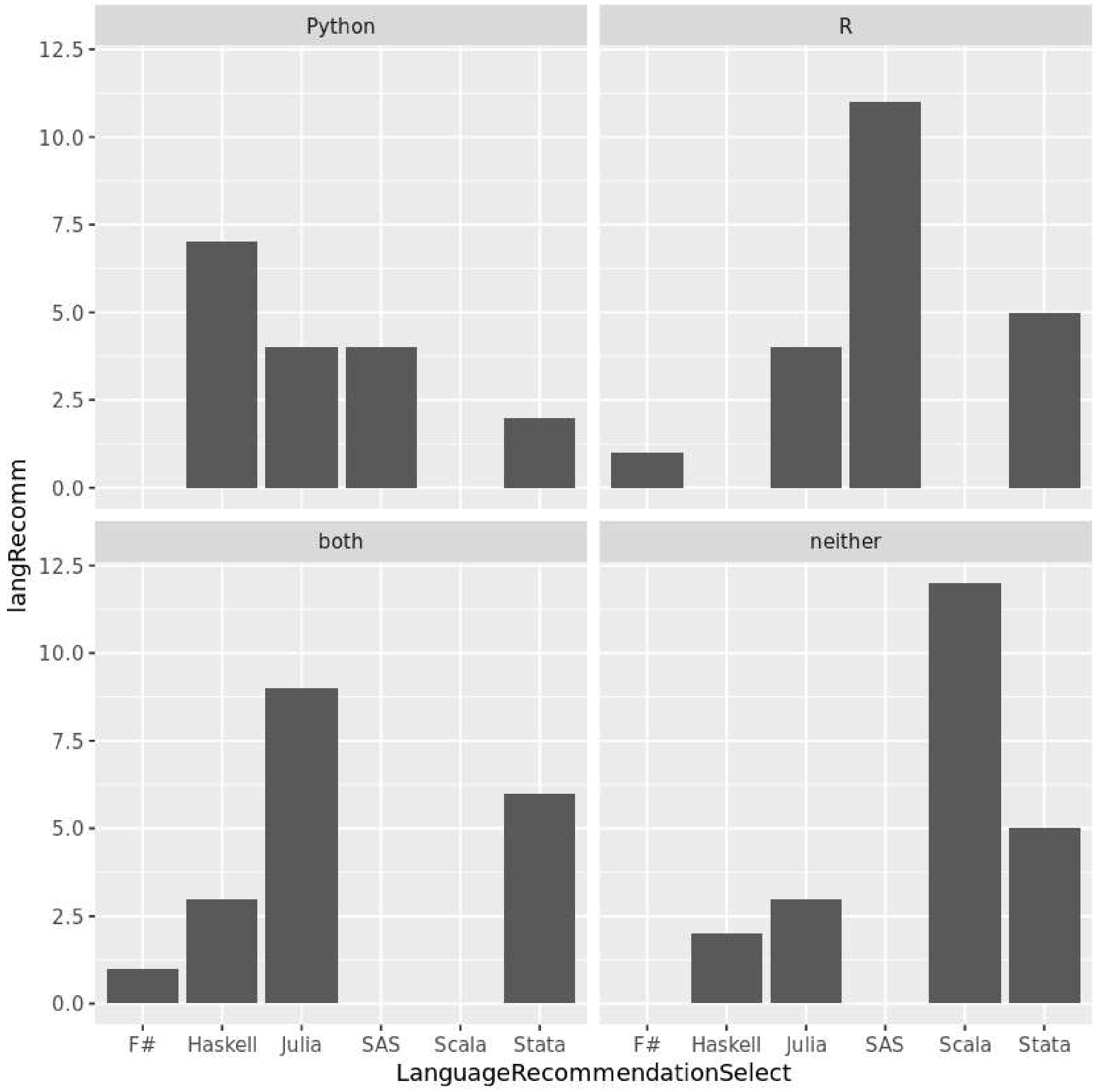
In [15]:
```
run_tests({
    test_that("Tools have been summarised", {
        expect_true(nrow(recommendations) == 16,
            info = 'Make sure that you are only keeping the top 4 responses fo
r each language used')
    })

})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.99 0.096 36.977 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

8. The most recommended language by the language used

Just one thing left. Let's graphically determine which languages are most recommended based on the language that a person uses.

```
In [16]: # Creating a faceted bar plot
         ggplot(recommendations,aes(x=LanguageRecommendationSelect,y=langRecomm)) +
            geom_bar(stat = "identity") + facet_wrap(~language_preference)
```



In [17]:
```
run_tests({
   test_that("Plot is a bar chart",{
      p <- last_plot()
      q <- p$layers[[1]]
      expect_is(q$geom, "GeomBar")
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 3.911 0.096 37.898 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

9. The moral of the story

So we've made it to the end. We've found that Python is the most popular language used among Kaggle data scientists, but R users aren't far behind. And while Python users may highly recommend that new learners learn Python, would R users find the following statement `TRUE` or `FALSE`?

In [18]:
```
# Would R users find this statement TRUE or FALSE?
R_is_number_one = TRUE
```

In [19]:
```
run_tests({
    test_that("The question has been answered", {
        expect_true(R_is_number_one,
            info = 'Try again! Should R_is_number_one be set to TRUE or FALS
E?')
    })

})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 3.941 0.096 37.928 0.004 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```