Informedb Software Quality Metrics

Liam Whitenack

2021-06-11

Setup

Tidy Data

```
# record the row number of each row

#this while loop initializes the array and sets its first value
n <- 0
counter <- 0
while(n < 1){
  counter <- counter + 1
  if(Jira$`Issue Type`[counter] == "Bug" || Jira$`Issue Type`[counter]
     == "Pre-Release Bug"){
    bug_index <- counter
    n = 1
    counter = counter + 1
  }
}
# this for loop checks to see if the issues have the Issue Type "Bug".
# If they do, mark their row number for later
for(i in counter:length(Jira$`Issue Type`)){
  if(Jira$`Issue Type`[i] == "Bug" || Jira$`Issue Type`[i] == "Pre-Release Bug"){
    bug_index <- c(bug_index, i)
  }
}
```

```
# make a new tibble with only the rows recorded in bug_index
Jira_bugs <- Jira[c(bug_index), ]
```

```
# This code block updates the "Resolved" column.
# The resolved column only contains a value if the row's value in the Status
# column is "Resolved".
# If the logical statement is TRUE then the Resolved column's value is the same
```

```
# as the Updated column.
for(i in 1:length(Jira_bugs$Status)){
  Jira_bugs$Resolved[i]<-NA
}
for(i in 1:length(Jira_bugs$Status)){
        if(Jira_bugs$Status[i] == "Resolved"){
          Jira_bugs$Resolved[i]<-Jira_bugs$Updated[i]
        }
}
```

```
Jira_bugs <- Jira_bugs %>%
    mutate(
      Priority = recode(
       Priority,
          `1 - Trivial` = "R4",
          `Trivial` = "R4",
          `2 - Minor` = "R4",
          `Minor` = "R4",
          `3 - Major` = "R3",
          `Major` = "R3",
          `4 - Critical` = "R2",
          `Critical` = "R2",
          `5 - Blocker` = "R1",
          `Blocker` = "R1",
      ),
      # Rename each value to R1, R2, R3, and R4,
      # the military's preferred terms.
      completed = recode(
        Status,
        Resolved = TRUE,
        Done = FALSE,
        Closed = FALSE,
        `In Progress` = FALSE,
        `Ready For Review` = FALSE,
        `Ready For Test` = FALSE,
        Reopened = FALSE,
        .default = FALSE
      ), # set the value of each resolved issue to 1 for computational purposes
      open = recode(
        Status,
        Resolved = FALSE,
        Done = TRUE,
        Closed = FALSE,
        `In Progress` = TRUE,
        `Ready For Review` = TRUE,
        `Ready For Test` = TRUE,
        Reopened = TRUE,
```

```
        Todo = TRUE,
        .default = TRUE
      ), # set the value of each resolved issue to 1 for computational purposes
      failed_fix = recode(
        Status,
        Resolved = FALSE,
        Done = FALSE,
        Closed = TRUE,
        `In Progress` = FALSE,
        `Ready For Review` = FALSE,
        `Ready For Test` = FALSE,
        Reopened = FALSE,
        .default = FALSE
      ), # set the value of each resolved issue to 1 for computational purposes
      Parent = `Parent id`,
      Issue = `Issue id`,
      Type = `Issue Type`,
      date_created = as.Date(strptime(Created, format = "%m/%d/%Y %H:%M")),
      date_updated = as.Date(strptime(Updated, format = "%m/%d/%Y %H:%M")),
      date_resolved = as.Date(strptime(Resolved, format = "%m/%d/%Y %H:%M")),
      # read columns as dates instead of characters
      time_spent = date_updated - date_created,
      # create the time_spent parameter, which is computed by finding
      # the difference between the date of creation and last day updated
      repair_time = date_resolved - date_created
      # create the repair time parameter, which is computed by finding
      # the difference between the date of creation and day resolved
    )
```

```
# Select only the rows I need
Jira_reduced <- Jira_bugs %>%
  select(
    Priority, # The importance of fixing each bug R1 is the highest importance,
    # R4 is the lowest
    Status, # the status of each progress i.e ""In progress",
    # "todo", or "resolved"
    Assignee, # The person assigned to the issue.
    # Most issues do not typically have an assignee
    Reporter, #Typically the same as the creator
    Creator, # The person creaing the issue
    date_created, # The day each issue was first created
    date_updated, # The last time the issue was edited
    date_resolved, # The last time the issue was edited
    # IF the status is "resolved"
    time_spent, # date updated - date created
    repair_time, # date updated - date created IF the status is "resolved"
    completed, # TRUE if completed, FALSE if not
```

```
    open, # TRUE if open, FALSE if not
    failed_fix, # TRUE if "closed", FALSE if not
  )
```

```
# This code block will test to see if the date_created was in the last 365 days
# and create an array that contains all
# of the row numbers we want

# Initialize the array and set the first value
    # Note: there is probably a much more efficient way of initializing an array.
    # I do not know what the method is, so this will have to do.
n <- 0
counter <- 0
while(n < 1){ # this while statement will set the first value of the array
  counter <- counter + 1
  if(Jira_reduced$date_created[counter] > (max(Jira_reduced$date_created)-365)){
    index_365 <- counter
    n = 1
    counter = counter + 1
  }
}
# if the date created is greater than the last day anything was created - 365,
# record that row number in an array
for(i in counter:length(Jira_reduced$date_created)){
  if(Jira_reduced$date_created[i] > (max(Jira_reduced$date_created)-365)){
    index_365 <- c(index_365, i)
  }
}
```

```
# This code block will do the same thing as before,
# but uses the last 28 days instead of 365.
# Because this is being used for summary statistics and not a graph,
# this block will record a "TRUE" or "FALSE" for each row
# to be later added onto the tibble
n <- 0
counter <- 0
while(n < 1){
  counter <- counter + 1
  if(Jira_reduced$date_created[counter] > (max(Jira_reduced$date_created)-28)){
    index_28 <- TRUE
    n = 1
    counter = counter + 1
  }
  else{
    index_28 <- FALSE
    n = 1
    counter = counter + 1
```

```
  }
}
for(i in counter:length(Jira_reduced$date_created)){
  if(Jira_reduced$date_created[i] > (max(Jira_reduced$date_created)-28)){
    index_28 <- c(index_28, TRUE)
  }
  else{
    index_28 <- c(index_28, FALSE)
  }
}
```

```
# make a new tibble with only the rows selected from the earlier code block.
# The array contains the row number of every issue listed over the last week
Jira_reduced_year <- Jira_reduced[c(index_365), ]
```

```
# add a new row to Jira_reduced that has a value of TRUE
# if the issue was created in the last 28 days,
# FALSE if the issue was created before
Jira_reduced <- Jira_reduced %>%
  mutate(
    last28 = index_28
  )
```

Summary Statistics

```
# make a plot that shows the distribution of issues over the last year
Defect_Trends <- Jira_reduced_year %>%
  ggplot() +
  geom_histogram(aes(x = date_created, fill = Priority), bins = 12) +
  labs(title = "Defect Trends Histogram") +
  xlab("Issue Date")
```

```
# Calculate the summary statistics by their Priority
Summary_Statistics <- Jira_reduced %>%
  group_by(Priority) %>%
  summarise(
    `Defects in the last four weeks` = sum(last28),
    # count the number of rows that were created in the last 28 days
    MTTR = (mean((repair_time), na.rm = TRUE)),
    # Repair time average
    FFA = sum(failed_fix),
    # the number of "closed" issues i.e. issues that were opened
    # and then closed because they couldn't be fixed
    DRE = sum(completed) / n(),
    # count the number of rows marked as completed
```

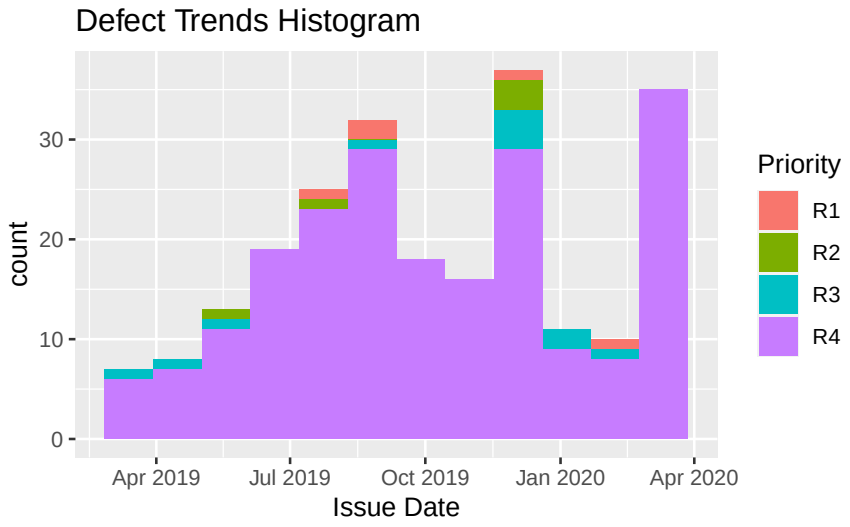
```
    Open = sum(open)
    # count the number of rows that aren't closed or resolved
  )
```

Output

```
write_csv(Summary_Statistics, "Summary_Statistics.csv")
# write my software quality metrics to their own file
```

```
write_csv(Jira_reduced, "Jira_reduced.csv")
# save the organized data to a new and improved file
```

```
Defect_Trends
```




```
Summary_Statistics
```

Priority	Defects in the last four weeks	MTTR	FFA	DRE	Open
R1	0	24.71429 days	0	0.8750000	2
R2	0	45.46154 days	0	0.6190476	8
R3	0	106.00000 days	1	0.4827586	14
R4	32	62.88976 days	1	0.3956386	387