# Assignment 2: Code Review and Workflow Refinement using Git and R

**Student:** Farah Sadoon, 1302190
**Group Members:** Eva Innocente, Fangyi Li, Farah Sadoon, Lishita Rowjee
**Primary author of the script I edited:** Eva Innocente
**Github Links:** Eva Innocente, Fangyi Li, Farah Sadoon, Lishita Rowjee

## Section 1: Contributions to Eva's Code

Change #1

The first major edit I made was to create a more dynamic, reproducible workflow for joining Eva's two data frames together and mutating a new column. The first data frame, originating from BOLD, was filtered down to include species, BIN URI, and specimen count. The second data frame from IUCN included species and the corresponding Red List category. The original code combined the two datasets for downstream plotting and statistical analysis. The original approach used base R's merge() function; however, using the merge() function makes it difficult to address differences between two datasets. To address this challenge, Eva used dplyr::slice() and hard-coded the row number to exclude in the resulting data frame. I suggested using janitor::clean_names(), then using dplyr::left_join() to join the datasets together on the common key, species. Using a left join is more dynamic because it keeps all rows in the left data frame (i.e., the one containing BOLD data), without hard-coding rows to exclude from the IUCN data frame. In addition to the left join, I introduced a new function, dendrometry::spNmReduce(), to prevent hard-coding the abbreviated species names. Since Eva wanted to include specimens with no BIN URI, I used case_when() to describe how the new column "spec_abbrev" should be defined from the previous species column. The spNmReduce() function takes full scientific names and abbreviates them exactly as before. With the proposed changes, the species names in the data frame could change, but the mutation would still work.

*PR change log*

```
109 125   Merging both dataframes for plotting and further analyses. Adding a column of abbreviated species name for graph labelling purposes.
110 126
111 127   ```{r}
112     - df_merged <- merge(df_phoc_bincount, df_iucn_simple, all.x = T, all.y = T) # the dataframes have different numbers of rows, I deal with this below
    128 + #### fs_edit: I would use dplyr left_join() instead of base R merge() here. Using a left join would retain all the values in df_phoc_bincount while retaining all the
        + matching values you want, using species names as the join key. You would not have to manually address the records that do not appear in the BOLD data. It also prevents
        + errors because you would not have to re-adjust these values later. You can use the clean_names() function in the janitor package to make sure the column names in both
        + data frames are the same before doing the join. Joining with dplyr is generally more efficient. I would also use the spNmreduce() function in the dendrometry package to
        + handle species names so that you don't have to do that manually in a list. This will avoid any errors that arise from hard-coding names.
    129 +
    130 + # make sure column names are formatted correctly before joining - this is important for the join where species is the join key
    131 + clean_names(df_phoc_bincount)
    132 + clean_names(df_iucn_simple)
    133 +
    134 + # join data frames
    135 + df_merged <- left_join(df_phoc_bincount, df_iucn_simple, by = "species")
113 136
114 137   df_merged <- df_merged %>%
115     -   mutate(spec_abbrev = c("C. cristata", "E. barbarus", "H. grypus", "H. fasciata", "H. leptonyx", "L. weddellii", "L. carcinophaga", "M. angustirostris", "M. leonina",
        + "M. monachus", "N. schauinslandi", "N. tropicalis", "No BIN", "O. rossii", "P. groenlandicus", "P. largha", "P. vitulina", "P. caspica", "P. hispida", "P. sibirica")) %>%
116     -   mutate(redlistCategory = replace_na(redlistCategory, "no_bin")) %>% # recoding NAs to say no_bin
117     -   dplyr::slice(-12) # removing the extinct species N tropicalis because it is not in BOLD
    138 +   mutate(redlistCategory = replace_na(redlistCategory, "no_bin"),
    139 +          spec_abbrev = case_when( # use case_when() to address when there are scientific names and when there is "no_bin"
    140 +            redlistCategory == "no_bin" ~ "No BIN", # assign "No BIN" to those with no_bin category
    141 +            TRUE ~ spNmReduce(species) # abbreviate scientific names
    142 +            )
    143 +          )
    144 +
    145 + #### fs_edit: inspect data frame after joining
    146 + head(df_merged)
    147 + tail(df_merged)
    148 + dim(df_merged)
    149 +
    150 + #### fs_edit: remove df_phoc_bincount
    151 + rm(df_phoc_bincount)
```

*Before*

```r
df_merged <- merge(df_phoc_bincount, df_iucn_simple, all.x = T, all.y = T)
# the dataframes have different numbers of rows, I deal with this below

df_merged <- df_merged %>%
  mutate(spec_abbrev = c("C. cristata", "E. barbarus", "H. grypus", "H.
fasciata", "H. leptonyx", "L. weddellii", "L. carcinophaga", "M.
angustirostris", "M. leonina", "M. monachus", "N. schauinslandi", "N.
tropicalis", "No BIN", "O. rossii", "P. groenlandicus", "P. largha", "P.
vitulina", "P. caspica", "P. hispida", "P. sibirica")) %>%
  mutate(redlistCategory = replace_na(redlistCategory, "no_bin")) %>% #
recoding NAs to say no_bin
  dplyr::slice(-12) # removing the extinct species N tropicalis because it
is not in BOLD
```

*After*

```r
#### fs_edit: I would use dplyr left_join() instead of base R merge() here.
Using a left join would retain all the values in df_phoc_bincount while
retaining all the matching values you want, using species names as the join
key. You would not have to manually address the records that do not appear
in the BOLD data. It also prevents errors because you would not have to
re-adjust these values later. You can use the clean_names() function in the
janitor package to make sure the column names in both data frames are the
same before doing the join. Joining with dplyr is generally more efficient.
```

```
I would also use the spNmreduce() function in the dendrometry package to
handle species names so that you don't have to do that manually in a list.
This will avoid any errors that arise from hard-coding names.

# make sure column names are formatted correctly before joining - this is
important for the join where species is the join key
clean_names(df_phoc_bincount)
clean_names(df_iucn_simple)

# join data frames
df_merged <- left_join(df_phoc_bincount, df_iucn_simple, by = "species")

df_merged <- df_merged %>%
  mutate(redlistCategory = replace_na(redlistCategory, "no_bin"),
         spec_abbrev = case_when( # use case_when() to address when there
are scientific names and when there is "no_bin"
           redlistCategory == "no_bin" ~ "No BIN", # assign "No BIN" to
those with no_bin category
           TRUE ~ spNmReduce(species) # abbreviate scientific names
           )
         )

#### fs_edit: inspect data frame after joining
head(df_merged)
tail(df_merged)
dim(df_merged)

#### fs_edit: remove df_phoc_bincount
rm(df_phoc_bincount)
```

Change #2

The second major edit I made improves the reproducibility and flexibility of the bar chart creation process. In the original code, a bar chart was made to display the number of specimens in each BIN from the BOLD data frame. In ggplot(), to assign different colours to individual bars, the colour variable must be mapped within the aesthetics; however, this becomes tedious if only one specific bar needs to be highlighted. In the original approach, Eva created a vector specifying each bar colour, which corresponded to the total number of bars. If the data frame changes in the future, the resulting chart may not highlight the same features. For example, if the species names were rearranged or if new rows were added, the vector for the colour mapping would need to change. To address this issue, I suggested temporarily introducing a column to the data frame that maps each BIN to a colour. I used conditional logic and ifelse() to

display records in navy blue when species are “no_bin” and all other records in sky blue. Since the colour mapping column is not needed after this point, I suggested using select(-) to remove it. With the proposed changes, the resulting chart looks the same, but if the values in the data frame change, the code does not need to be updated.

*PR change log*

```
120  154   Plotting how many specimens are in each BIN and saving the figure as a .png.
121  155
122  156   ```{r}
123       - plot_count_bin <- ggplot(df_merged, aes(x = spec_abbrev, y = count, fill = spec_abbrev)) +
     157  + #### fs_edit: rather than hard-coding all the bar colours, you could add a column to your data frame specifying the bar colour you want, then remove the column later.
            This might make your code more dynamic. Lishita R pointed out the hard-coded bar colours and suggested looking for ways to make this dynamic.
     158  +
     159  + # add column for colour mapping in figure
     160  + df_merged <- df_merged %>%
     161  +   mutate(bar_colour = ifelse(spec_abbrev == "No BIN", "navyblue", "skyblue3"))
     162  +
     163  + plot_count_bin <- ggplot(df_merged, aes(x = spec_abbrev, y = count, fill = bar_colour)) +
124  164     geom_col(stat = "identity") +
125  165     labs(x = "Species associated with BIN", y = "Number of specimens", title = "Number of specimens in each BIN in the Phocidae family") +
     166  +   scale_fill_identity("navyblue", "skyblue3") + # tell ggplot() to use these colours for the bars
126  167     theme_grey() +
127       -   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
128       -   scale_fill_manual(values = c("skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3",
            "navyblue", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3")) +
129       -   guides(fill = F)
     168  +   theme(axis.text.x = element_text(angle = 45, hjust = 1))
     169  +
     170  + plot_count_bin
130  171
131  172   ggsave("../figs/count_bin_plot.png", plot = plot_count_bin, device = png)
     173  +
     174  + # remove bar colour column as it is not needed after this
     175  + df_merged <- df_merged %>%
     176  +   select(-"bar_colour")
132  177   ```
```

*Before*

```
plot_count_bin <- ggplot(df_merged, aes(x = spec_abbrev, y = count, fill =
spec_abbrev)) +
  geom_col(stat = "identity") +
  labs(x = "Species associated with BIN", y = "Number of specimens", title
= "Number of specimens in each BIN in the Phocidae family") +
  theme_grey() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("skyblue3", "skyblue3", "skyblue3",
"skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3",
"skyblue3", "skyblue3", "navyblue", "skyblue3", "skyblue3", "skyblue3",
"skyblue3", "skyblue3", "skyblue3", "skyblue3", "skyblue3")) +
  guides(fill = F)

ggsave("../figs/count_bin_plot.png", plot = plot_count_bin, device = png)
```

*After*

```
#### fs_edit: rather than hard-coding all the bar colours, you could add a
column to your data frame specifying the bar colour you want, then remove
the column later. This might make your code more dynamic. Lishita R pointed
out the hard-coded bar colours and suggested looking for ways to make this
```

```
dynamic.

# add column for colour mapping in figure
df_merged <- df_merged %>%
  mutate(bar_colour = ifelse(spec_abbrev == "No BIN", "navyblue",
"skyblue3"))

plot_count_bin <- ggplot(df_merged, aes(x = spec_abbrev, y = count, fill =
bar_colour)) +
  geom_col(stat = "identity") +
  labs(x = "Species associated with BIN", y = "Number of specimens", title
= "Number of specimens in each BIN in the Phocidae family") +
  scale_fill_identity("navyblue", "skyblue3") + # tell ggplot() to use
these colours for the bars
  theme_grey() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

plot_count_bin

ggsave("../figs/count_bin_plot.png", plot = plot_count_bin, device = png)

# remove bar colour column as it is not needed after this
df_merged <- df_merged %>%
  select(-"bar_colour")
```

Change #3

The final major contribution I made to the code involved evaluating and refining the statistical test used to assess differences in specimen counts among IUCN Red List categories. Eva chose to use an analysis of variance, ANOVA, for this test. Although ANOVA is used to detect differences in means of three or more groups, some assumptions should be met for the test to be accurate. My feedback in this section focused on testing these assumptions and implementing appropriate follow-up steps if they were violated. I first noted that the assumption of independence between records could not be challenged in this scenario because there was no reliable way to verify. However, since BOLD is an open-source database, it is likely that this is not the case. Additionally, I noted that it was good practice for Eva to have included a box plot to show the distribution of data for each group. This is primarily useful for identifying outliers. The first assumption tested was the normality of residuals. As a parametric test, ANOVA assumes that samples originate from continuous, normally distributed populations. As Eva's data is discrete, this assumption may not hold. I suggested using a Shapiro-Wilk (SW) test and a Q-Q plot to assess the normality of residuals. The p-values for the SW test were < 0.05, indicating a significant deviation from a normal distribution. The Q-Q plot confirmed this as the

values drift from the expected line towards the tail end of the plot. Next, I suggested testing the homogeneity of variances between groups. Levene's test addresses this assumption by determining whether each group differs from the mean by equal amounts. The results of this test show that the variances do not differ from what would be expected if variances were equal. Seeing as the normality assumption was violated but the homogeneity of variances was not, I suggested using the non-parametric Kruskal-Wallis test to compare specimen abundance between Red List categories. Seeing as the Kruskal-Wallis test can be used with non-normal, discrete data, it is an appropriate alternative. Although the Kruskal-Wallis test does not compare means between groups as ANOVA does, it does look at whether the mean ranks are the same. Overall, it is an appropriate statistical test for detecting differences in specimen abundance between Red List Categories: Least Concern, Vulnerable, and Endangered.

*PR change log*

```
Running a one-way analysis of variance (ANOVA), to test for differences in count of specimens between Red List Categories Least Concern, Endangered, and Vulnerable. First I quickly visualised differences between groups with a boxplot. I saved the model output to the output folder.

```{r}
+
+ #### fs_edit: I think the ANOVA works here, however it can be tricky because it usually assumes continuous variable. I would check your assumptions to make sure the counts don't impact the other assumptions. We will assume that the observations are independent because you got them from BOLD, and we cannot adjust the independence of observations as that has to do with study design. If you knew the samples were not independent, you could use a test that works for repeated-measures, or remove the values that you know were repeated. No need to test for sphericity because we are assuming no repeated measures. Usually you want to remove outliers from your data before running your statistical test. You plotted a boxplot showing the distribution for each group in the analysis, which is a good way to visually assess that. Reference(s): https://mgimond.github.io/Stats-in-R/ANOVA.html, https://numiqo.com/tutorial/kruskal-wallis-test
+
+ # test assumption: no outliers
  ggplot(df_merged_nobin, aes(redlistCategory, count)) +
-   geom_boxplot()
+   geom_boxplot() # there are some points that look far outside the range of the data - it may be worth investigating or removing those points

+ # run the anova
  anova <- aov(count ~ redlistCategory, data = df_merged_nobin)
-
  report(anova)

- summarybin <- summary(anova)
+ # test assumption: normality of residuals
+ residuals_anova <- residuals(anova)
+ shapiro.test(residuals_anova) # the residuals deviate significantly from what you would expect if they were normally distributed.

+ # visualize normality of residuals with a Q-Q plot
+ ggplot(data.frame(residuals = residuals_anova), aes(sample = residuals)) +
+   stat_qq() +
+   stat_qq_line(color = "red") +
+   theme_minimal() +
+   labs(title = "Q-Q Plot of ANOVA Residuals")
+
+ # test assumption: homogeneity of variances (are the variances between your groups the same?)
+ leveneTest(count ~ redlistCategory, data = df_merged_nobin) # seems that the variance does not deviate significantly from what would be expected if there was no difference
+
+ # since normality of residuals is violated, use kruskal-wallis test instead of anova
+ kruskal_test <- kruskal.test(count ~ redlistCategory, data = df_merged_nobin)
+ kruskal_test
+
+ summarybin <- summary(anova)
  summarybin

  capture.output(summarybin, file = "../output/anova_output.txt")
+ capture.output(kruskal_test, file = "../output/kruskal_test.txt")
  ```
```

*Before*

```
ggplot(df_merged_nobin, aes(redlistCategory, count)) +
  geom_boxplot()

anova <- aov(count ~ redlistCategory, data = df_merged_nobin)

report(anova)

summarybin <- summary(anova)
```

```
summarybin

capture.output(summarybin, file = "../output/anova_output.txt")
```

*After*

```
#### fs_edit: I think the ANOVA works here, however it can be tricky
because it usually assumes continuous variables. I would check to make sure
the counts don't impact the other assumptions. We will assume that the
observations are independent because you got them from BOLD, and we cannot
adjust the independence of observations as that has to do with study
design. If you knew the samples were not independent, you could use a test
that works for repeated-measures, or remove the values that you know were
repeated. No need to test for sphericity because we are assuming no
repeated measures. Usually you want to remove outliers from your data
before running your statistical test. You plotted a boxplot showing the
distribution for each group in the analysis, which is a good way to
visually assess that. Reference(s):
https://mgimond.github.io/Stats-in-R/ANOVA.html,
https://numiqo.com/tutorial/kruskal-wallis-test

# test assumption: no outliers
ggplot(df_merged_nobin, aes(redlistCategory, count)) +
  geom_boxplot() # there are some points that look far outside the range of
the data - it may be worth investigating or removing those points

# run the anova
anova <- aov(count ~ redlistCategory, data = df_merged_nobin)
report(anova)

# test assumption: normality of residuals
residuals_anova <- residuals(anova)
shapiro.test(residuals_anova) # the residuals deviate significantly from
what you would expect if they were normally distributed.

# visualize normality of residuals with a Q-Q plot
ggplot(data.frame(residuals = residuals_anova), aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line(color = "red") +
  theme_minimal() +
  labs(title = "Q-Q Plot of ANOVA Residuals")
```

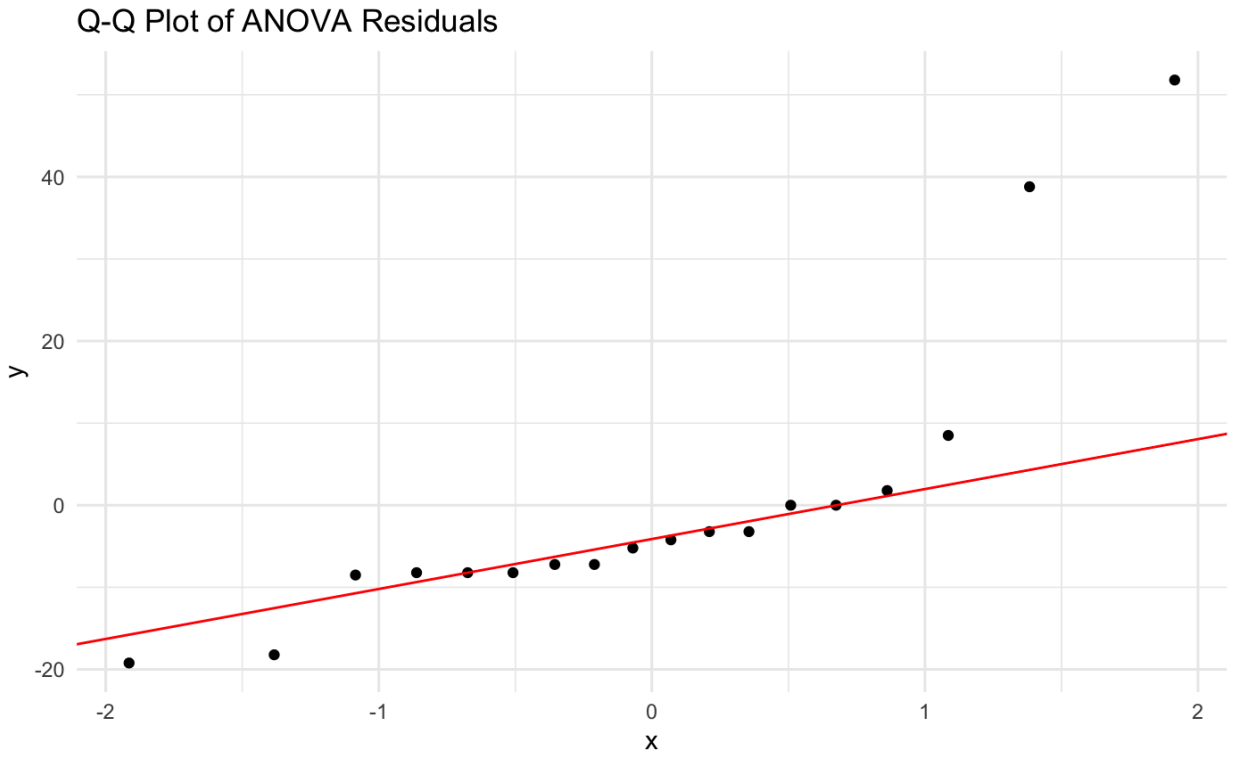
```
# test assumption: homogeneity of variances (are the variances between your groups the same?)
leveneTest(count ~ redlistCategory, data = df_merged_nobin) # seems that the variance does not deviate significantly from what would be expected if there was no difference

# since normality of residuals is violated, use kruskal-wallis test instead of anova
kruskal_test <- kruskal.test(count ~ redlistCategory, data = df_merged_nobin)
kruskal_test

summarybin <- summary(anova)
summarybin

capture.output(summarybin, file = "../output/anova_output.txt")
capture.output(kruskal_test, file = "../output/kruskal_test.txt")
```

*Additional figure*




**Section 2: Eva's Contributions to My Code**

The major edits that Eva made to my code were (1) use the bin_lat() function from the palaeoverse package to create latitude bins, (2) explore the number of specimens in each latitude band with a bar chart for direct comparison, and (3) remove the redundant information from the map by overlaying latitude bands on the map to plot specimens. I greatly appreciate the edits made to my script, as I have been introduced to new functions and packages that will be useful for future analyses. For example, Eva used the bin_lat() function from the palaeoverse package, which replaces the need to hard-code latitude bands. Additionally, she pointed out some important improvements to my workflow, including making more explicit comparisons when exploring my data. One suggestion was to use a bar chart to compare the number of specimens within each latitude bin. Implementing this change helped me to realize that a simpler figure can sometimes communicate information more effectively than a more elaborate one. While my map displayed the geographic spread of specimens, Eva's bar chart provided valuable context by summarizing the data and highlighting patterns that were less immediately apparent in the map. Finally, Eva made further edits to my work by improving the map figure I created. These edits included removing the legend, changing the y-axis to represent latitude bands rather than continuous latitude, and changing point size and transparency to represent density; all of which worked more harmoniously to present different pieces of information, rather than having multiple representations of the same thing. This approach has reinforced my understanding of how to create visuals and ensure that every element of the figure is necessary for conveying some information.

**Section 3: Reflecting on Teamwork**

Throughout this assignment, it has been a pleasure working with my group members Eva Innocente, Fangyi Li, and Lishita Rowjee. Although Eva and I were the primary contributors to each other's code, our group made an effort to meet regularly and review each other's work, contributing across all projects. In particular, Fangyi suggested an edit to a section where Eva assessed whether a transformation she made was successful. She argued that while head() is an appropriate way to assess the data overall, it does not explicitly address changes to missing values. Due to her assessment, I was able to directly compare the results of the data transformation by looking at the sum of blank values before and after Eva changed them. Furthermore, Lishita pointed out that the code for Eva's bar chart included several hard-coded values, which should be addressed for flexibility. From this observation, I was able to make a major change to Eva's code and improve the workflow overall. In turn, I contributed to others' work by suggesting improvements to script organization. Additionally, I advised on using Git and GitHub effectively for collaboration and version control. Throughout this process, I not only gained technical depth but also an understanding of how collaboration is applied in a technical setting. This assignment was an exercise in thinking critically about the goals of an analysis, where I learned that there are many ways to answer a question, and it is important to be thoughtful about the choices we make in the process.