Jordan Lally

BIO 539 Final Assignment

Statement of Problem:

Earlier this year, Dr. Sarah Reed from the University of Connecticut provided Dr. Hoffman (my graduate advisor) and I with a large dataset in an excel file to organize and manipulate. To date, Dr Reed's lab group had been manually organizing these files and removing unwanted information. However, the files were too large and cumbersome to continue doing this. Dr. Hoffman knew that we could help provide support for this issue while allowing myself to gain experience with these datasets.

Objectives:

1) Dr. Reed requests each separate excel sheets be compiled into one large data frame
2) Dr. Reed requests the data frame is transformed so that proteins are column labels and the animal ID's are row labels.
3) Dr. Reed requests all NA data and associated rows/columns are removed.
4) Add a new column that includes the experimental test group names based on the Animal's unique ID.
5) Separate, create, and organize new data frames based on the test groups.
6) Accurately prepare data frames for plotting.
7) Create box plots to visualize and compare the relative protein abundance for the three treatment groups sampled at different times.
8) Gain experience organizing and analyzing large data sets using R studio

Dr. Reed's email, with the attached dataset, requested that the dataset was organized and transformed for further analysis, and closely resembled two of the example coding ideas provided in the assignment guidelines:

Objectives Figure 1:

Some example ideas:

- Code and automate a series of analysis steps you have been doing manually (e.g. in Excel)
- Filter your data in different ways to analyze different subsets

Methods:

Step 1: With clear objectives in mind, I needed to view the excel file to see what type of data I was working with. Upon first glance, I discovered that the excel file contained multiple different sheets, that represented the different "runs" that needed to be "compiled" into one sheet. I figured I could do this by reading each of the 12 excel sheets into their own data frame in R studio and then merging them from there. I also noticed that a lot of the columns in each of the original excel sheets was not necessarily important for this assignment, so I needed to remove these in order to trim down on the size of the data set and provide better visualization of the important data.

Methods Figure 1:

```
#Read input from the user. Store inputed file name into the filename variable.
filename <- readline(prompt = "Please enter the file name with appropriate extension (Revised All Data.xlsx): ")

#read and store sheet 1 (group 1) of excel file
dataSheet1 <- read_excel(filename, sheet = "Group 1")
#Remove unessecary columns so that only the accession and animal ID's are present
#if you wanted to keep all columns, remove the followig line that appears after each dataframe read
dataSheet1 <- dataSheet1[-c(2:7)]

#read and store sheet 2 (group 2) of excel file
dataSheet2 <- read_excel(filename, sheet = "Group 2")
#Remove unessecary columns
dataSheet2 <- dataSheet2[-c(2:7)]
```

Step 2: Once each of the 12 sheets were read into their individual 12 data frames without the inclusion of the unnecessary columns, the next step was compiling them all into a singular large dataframe. I decided to use the full_join function, that is part of the dplyr package, which is similar to the left_join function we performed in class on the australia fire data, except full_join includes every row and column from each dataset and inputs NA wherever the data frames don't match up. This was important since, as stated in Dr. Reeds email, not all proteins were represented in each run so there was some missing data.

Methods Figure 2:

```
#Merge all the data sheets together and order them by their Accession key
#Use full_join in order to include all rows and coulumns, including ones that don't match
#Unmatched columns will input NA as a default. store in total_merge dataframe
merged_data_sheets <- full_join(dataSheet1, dataSheet2, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet3, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet4, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet5, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet6, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet7, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet8, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet9, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet10, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet11, by = "Accession")
merged_data_sheets <- full_join(merged_data_sheets, dataSheet12, by = "Accession")
```

Step 3: Now that I had a fully and accurately merged data frame containing all 12 of the sheets from the original excel file, I decided to make a new data frame that removed all rows and columns that were missing data by using the na.omit() function. With missing data removed, the final challenge was to make it so the animal ID's were the row labels and the proteins were the column labels. I accomplished this by simply using R's built in transpose function and storing the data frame that is outputted from that function into a new data frame. The final result was a fully compiled data frame, without any missing data, that was transposed to the specifications asked by Dr. Reed.

Methods Figure 3:

```
#remove any rows with NA values and store in a new dataframe
final_merge <- na.omit(merged_data_sheets)

#transpose the data frame so that animal ID's are the row names and proteins are the column names
final_merge2 <- as.data.frame(t(final_merge))
```

Step 4: At this point in time, I received additional information on the different test groups this study was done on and most of the animal ID's that made up each group. With this list of animal ID's and their matching test groups, I was able to pipeline the mutate function to add a new column called Test_Groups and also the case.when function in order to fill in that column with the test group labels based on the different ID's. In order to use the case.when function for this comparison however, I first had to use the rownames_to_column function in order to make the list of animal ID's its own column.

Methods Figure 4:

```
#Use the row_names_to_column function to move the row labels of the dataframe into it's own column
#We can now use this new Animal_ID column for comparisons in the following mutate function
final_merge2 <- rownames_to_column(final_merge2, var = "Animal_ID")

#Use mutate to add a new column named Test_Groups. Use the case_when function to label each Animal_ID
#when it matches any of the animal ID's (seperated by group) listed in the code below.
final_mutate_groups <- final_merge2 %>%
  mutate(Test_Groups = case_when(
    #Adding labels in new column for control group day 90 for animals matching following ID's
    Animal_ID == "38F2" | Animal_ID == "38F1" | Animal_ID == "119F1" | Animal_ID == "119F2" |
      Animal_ID == "124F1" | Animal_ID == "87F1" | Animal_ID == "87F2" | Animal_ID == "184F1"
        ~ "Control_D90",

    #Adding labels in new column for overfed group day 90 for animals matching following ID's
    Animal_ID == "120F1" | Animal_ID == "120F2" | Animal_ID == "26F1" | Animal_ID == "26F2" |
      Animal_ID == "117F1" | Animal_ID == "117F2" | Animal_ID == "109F1" | Animal_ID == "12F1"
        ~ "Over_D90",

    #Adding labels in new column for res group day 90 for animals matching following ID's
    Animal_ID == "19F1" | Animal_ID == "19F2" | Animal_ID == "20F1" | Animal_ID == "20F2" |
      Animal_ID == "43F1" | Animal_ID == "43F2" | Animal_ID == "39F1" | Animal_ID == "127F3"
        ~"Res_D90",

    #Adding labels in new column for control group day 135 for animals matching following ID's
    Animal_ID == "41F1" | Animal_ID == "79F1" | Animal_ID == "79F2" | Animal_ID == "81F1" |
      Animal_ID == "81F2" | Animal_ID == "75F2" | Animal_ID == "75F3" | Animal_ID == "107F1" |
      Animal_ID == "107F2" ~ "Control_D135",

    #Adding labels in new column for res group day 135 for animals matching following ID's
    Animal_ID == "94F1" | Animal_ID == "94F2" | Animal_ID == "101F1" | Animal_ID == "118F1" |
      Animal_ID == "118F2" | Animal_ID == "3F2" | Animal_ID == "3F3" | Animal_ID == "69F1" |
      Animal_ID == "69F2" ~"Res_D135",

    #Adding labels in new column for overfed group day 135 for animals matching following ID's
    Animal_ID == "63F1" | Animal_ID == "63F2" | Animal_ID == "89F1" | Animal_ID == "89F2" |
      Animal_ID == "93F1" | Animal_ID == "93F2" | Animal_ID == "56F1" | Animal_ID == "56F2" |
      Animal_ID == "21F1" ~ "Over_D135",

    #Add NA to new column for any unmatched animal ID
    TRUE ~ "NA"
  ))
```

Step 5: A new column titled “Test groups” was now added to the end of the data frame, which ended up being the last column out of 986 total columns. In order to better visualize the data in the dataframe, I decided to use the select function to create a new dataframe that placed the test groups column in front of the rest of the columns. I also arranged the data by the test groups using the arrange function at the end of the pipeline.

Methods Figure 5:

```
#move the Test_Group column from the last column to the first column so that it appears near the Animal ID's
arranged_final_mutate_groups <- final_mutate_groups %>%
  select(Test_Groups, everything()) %>%
  arrange(Test_Groups)
```

Step 6: With the addition of the test groups and arrangement of the data by said test groups, I separated the test groups that were sampled at day 90 from the test groups that were sampled at day 135, by storing them in their own data frames. This was done using a pipeline to filter the data into a new dataframe if the str_detect found either “D90” for the day 90 data or “D135” for the day 135 data in the Test_Groups column.

Methods Figure 6:

```
#Filter out all day 90 animals in all test groups into their own data frame.
#Arrage them so the control, over, and res groups all appear together on the data frame.
day90_group <- arranged_final_mutate_groups %>%
  filter(str_detect(Test_Groups, "D90")) %>%
  arrange(Test_Groups)

#Filter out all day 135 animals in all test groups into their own data frame
#Arrage them so the control, over, and res groups all appear together on the data frame.
day135_group <- arranged_final_mutate_groups %>%
  filter(str_detect(Test_Groups, "D135")) %>%
  arrange(Test_Groups)
```

Step 7: In order to create accurate box plots, the final step in preparing the data for plotting involved converting all numbers in the data frame from the factor class type, to the numerical class type. I first created a vector containing all column indexes that contained actual numbers instead of words. After that, I used the apply function to apply the as.numeric function to all of the columns indicated in the columns_to_convert vector for both data frames.

Methods Figure 7:

```
#vector to store all column numbers that contain numbers as factors
columns_to_convert <- c(3:986)

#Use the apply function to apply the as.numeric function to every column containing numbers as factors (using the vector created above)
#These next two functions will convert all factors in the data frame to the numerical class for plotting purposes.
day90_group[ ,columns_to_convert] <- apply(day90_group[ , columns_to_convert], 2,
                                           function(x) as.numeric(as.character(x)))

day135_group[ ,columns_to_convert] <- apply(day135_group[ , columns_to_convert], 2,
                                            function(x) as.numeric(as.character(x)))
```

Step 8: With the data frames now fully prepared for accurate plotting, it was time to create the two box plots using the ggplot function from the ggplot2 package. The first step was to create a variable that would contain the protein name the user wanted to plot and compare between the two data frames. I did this by using the readline function to prompt and receive input from the user via the terminal to store in the created protein variable. Finally, I used the ggplot package and its included functions to create a box plot that accurately represented the data and was visually appealing for comparisons between the two data frames.

Methods Figure 8:

```
#Box plot comparing the protein expression between the three day 90 treatment groups. The protein to be plotted is
#entered in by the user. (For example: A2SW69)
protein <- readline(prompt = "Please enter any protein from the dataframe (Ex = A2SW69): ")


#Box plot that shows the Relative Abundance of the user inputted protein for the 3 test groups sampled at day 90
day90_boxplot <- ggplot(day90_group, aes_string(x = "Test_Groups", y = protein, group = "Test_Groups", fill = "Test_Groups")) +
  geom_boxplot() +
  ggtitle(paste("Day 90 Samples: Relative Protein Abundance for Protein ", protein)) +
  scale_y_continuous(name = "Relative Protein Abundance", breaks = seq(0.0, 170.0, 5.0)) +
  scale_x_discrete(name = "Test Groups", breaks = c("Control_D90", "OverFed_D90", "Restricted_D90"),
                   labels = c("Control Diet", "Over Fed Diet", "Restricted Diet"))+
  scale_fill_discrete(name = "Test Groups", labels = c("Control", "Over Fed Diet", "Restricted Diet"))

#View the day 90 box plot
day90_boxplot

#Box plot that shows the Relative Abundance of the user inputted protein for the 3 test groups sampled at day 135
day135_boxplot <- ggplot(day135_group, aes_string(x = "Test_Groups", y = protein, group = "Test_Groups", fill = "Test_Groups")) +
  geom_boxplot() +
  ggtitle(paste("Day 135 Samples: Relative Protein Abundance for Protein ", protein)) +
  scale_y_continuous(name = "Relative Protein Abundance", breaks = seq(0.0, 170.0, 5.0)) +
  scale_x_discrete(name = "Test Groups", breaks = c("Control_D135", "OverFed_D135", "Restricted_D135"),
                   labels = c("Control Diet", "Over Fed Diet", "Restricted Diet"))+
  scale_fill_discrete(name = "Test Groups", labels = c("Control", "Over Fed Diet", "Restricted Diet"))

#View the day 135 box plot
day135_boxplot
```

Results:

Results Table 1*:*

	Accession	Description	Exp. q-valu	Sum PEP S	Pfam IDs	Entrez Ger	Ensembl G	107F2	79F1	10F1	56F1	118F1	69F1
2	A2SW69	Annexin A2 OS=Ovis aries OX	0	159.024	Pf00191	10004899	ENSOARG(	95.4	94.5	99.9	101.7	104.2	104.3
3	A6YRY8	40S ribosomal protein SA OS	0	72.959	Pf00318	10012562		99.2	97.3	96.7	88.4	104.5	113.9
4	A7UHZ2	Proteasome 26S non-ATPase	0	39.002	Pf02809, P	10010123	ENSOARG(	98.1	103.5	97.4	105.8	93.5	101.8
5	A8D8X1	60S ribosomal protein L10 O	0	4.616	Pf00252	10560184	ENSOARG(						
6	A8DR93	Heat shock protein 90 alpha	0	443.637	Pf00183, P	10012720	ENSOARG(	100.5	105.1	92	105.9	94.8	101.7
7	A9YUY8	Adipocyte fatty acid-binding	0	66.083	Pf00061, P	10013706	ENSOARG(	39.2	42.9	59.3	53.8	362.4	42.3
8	B0FYY4	Integrin beta-1 OS=Ovis aries	0	16.924	Pf00362, P	443141		92.9	118.4	89.8	93.1	100.7	105
9	B0FZL9	Pre-mRNA splicing factor SRI	0	6.382	Pf00076, P	10013544	ENSOARG(	87	97.3	97.9	103.9	99.3	114.6
10	B2LYK6	RAB7A, member RAS oncoge	0	30.086	Pf00025, P	10014588	ENSOARG(	101.1	112.2	95.4	97.2	87	107
11	B2LYL4	SNPRA OS=Ovis aries OX=994	0	13.552	Pf00076, P	10014586	ENSOARG(	98	115.9	88.3	88.4	100.9	108.5
12	B3VSB7	Hippocalcin-like protein 1 OS	0	11.358	Pf00036, P	10018754		112.2	101	87.3	87.7	102.6	109.2
13	B6UV59	Hydroxyacyl-CoA dehydroge	0	99.019	Pf00378, P	10019231	ENSOARG(	102.7	88.9	77.6	86.4	157.7	86.7
14	B7TJ15	Mitogen-activated protein ki	0	42.637	Pf00069, P	10021741	ENSOARG(	89.9	107.7	105.6	107	88.5	101.3
15	B9VH02	Eukaryotic translation initiat	0	24.117	Pf01176	10027071	ENSOARG(	90.2	88.3	99.3	96.3	104.6	121.3
16	C3V6M4	Calpastatin isoform II OS=Ov	0	112.904	Pf00748	443364	ENSOARG(	102.2	95.2	104.9	98.8	91.6	107.2
17	C5IJ99	RHOA OS=Ovis aries OX=994	0	63.147	Pf00025, P	10030207	ENSOARG(	91	104.9	94.2	110.8	100.1	99
18	C5IJA0	GTP-binding nuclear protein	0	43.493	Pf00025, P	10030207	ENSOARG(	101.6	106.6	94.8	105.2	95.9	95.9
19	C5ISA2	Tubulin alpha chain OS=Ovis	0	367.747	Pf00091, P	10030235	ENSOARG(	96	108.6	102	96.3	86.7	110.5
20	C5ISA4	COP9 constitutive photomor	0	14.936	Pf01399	10030233	ENSOARG(	109.5	95.6	94.8	113.3	78.3	108.5
21	C5ISB1	Replication protein A subunit	0	17.769	Pf01336, P	10030210	ENSOARG(	93.5	116.7	86.8	89.1	100.1	113.9
22	C5IWT0	ADP-ribosylation factor 4 OS	0	49.717	Pf00025, P	10030230	ENSOARG(	94.6	116.4	94	102	94.2	98.8
23	C5IWT7	Thioredoxin domain containi	0	11.499	Pf06110, P	10030235	ENSOARG(	96.9	111.5	104.4	92.1	96.6	98.5
24	C5IWU0	ADP-ribosylation factor 1 OS	0	82.478	Pf00025, P	10030230	ENSOARG(	95.1	107.8	90.1	100.7	105.7	100.7
25	C5IWU4	ADP-ribosylation factor-like	0	28.756	Pf00009, P	10030231	ENSOARG(	100	108.8	93.3	102.4	93.7	101.9
26	C5IWV1	Fumarate hydratase OS=Ovis	0	245.089	Pf00206, P	10030210	ENSOARG(	100.1	97.3	89	101.4	141.4	70.8
27	C8BKC5	Peroxiredoxin 2 OS=Ovis arie	0	124.15	Pf00578, P	10030704	ENSOARG(	100.8	103.8	95.4	107.9	101.5	90.6
28	C8BKE1	Signal transducer and activat	0	17.594	Pf00017, P	10030704	ENSOARG(	105.1	97.8	94.6	105	101.6	96
29	D8X187	Leukocyte elastase inhibitor	0	78.862	Pf00079	10111071	ENSOARG(	90.2	87.9	105.8	91.8	106	118.3
30	E5FXR5	Myozenin 2 OS=Ovis aries OX	0.091	0.979	Pf05556	10052681	ENSOARG(						
31	E7ECV8	ADP-ribose pyrophosphatase	0	16.419	Pf00293	10052915	ENSOARG(	101.8	105.2	91.4	98.9	109.3	93.4

Group 1 | Group 2 | Group 3 | Group 4 | Group 5 | Group 6 | Group 7 | Group 8 | Group 9 | Group 10 | Group 11 | Group 12

Results Table 2:

	A2SW69	A6YRY8	A7UHZ2	A8DR93	B0FYY4	B0FZL9	B2LYL4	B6UV59	B7TJ15	B9VH02	C3V6M4	C5IJ99	C5IJA0
107F2	95.4	99.2	98.1	100.5	92.9	87.0	98.0	102.7	89.9	90.2	102.2	91.0	101.6
79F1	94.5	97.3	103.5	105.1	118.4	97.3	115.9	88.9	107.7	88.3	95.2	104.9	106.6
10F1	99.9	96.7	97.4	92.0	89.8	97.9	88.3	77.6	105.6	99.3	104.9	94.2	94.8
56F1	101.7	88.4	105.8	105.9	93.1	103.9	88.4	86.4	107.0	96.3	98.8	110.8	105.2
118F1	104.2	104.5	93.5	94.8	100.7	99.3	100.9	157.7	88.5	104.6	91.6	100.1	95.9
69F1	104.3	113.9	101.8	101.7	105.0	114.6	108.5	86.7	101.3	121.3	107.2	99.0	95.9
75F2	96.4	81.7	92.5	94.2	92.1	93.2	115.3	89.2	99.6	79.9	96.3	120.0	94.9
107F1	92.5	110.2	97.3	100.0	99.9	96.1	95.2	102.8	101.9	97.8	98.8	86.3	97.8
93F1	88.7	85.5	104.1	97.7	105.9	104.7	99.9	108.6	99.9	80.8	98.5	89.7	99.4
21F1	106.1	99.4	114.0	114.6	104.6	114.4	104.3	104.4	101.5	93.2	109.1	90.3	109.9
3F2	111.5	104.9	93.5	91.2	83.1	65.2	76.4	88.1	92.1	123.9	92.3	104.7	102.8
94F2	104.8	118.3	98.6	102.3	114.4	126.3	108.9	107.0	105.0	124.4	105.0	109.0	95.3
75F3	92.9	85.4	107.0	89.4	107.9	92.1	91.2	96.1	97.0	72.2	92.0	102.0	98.1
81F1	93.3	88.1	96.9	96.5	112.1	114.4	114.6	115.6	84.3	82.6	85.6	100.4	92.3
89F1	100.0	111.9	92.1	111.7	99.0	83.5	105.2	108.5	103.5	126.7	99.3	100.3	101.1
93F2	92.2	93.2	107.6	107.8	109.6	93.4	96.0	97.7	100.7	76.4	99.0	102.2	97.0
69F2	108.6	106.2	96.5	96.3	79.5	116.8	102.4	88.8	102.2	119.6	111.0	99.9	107.6

Showing 1 to 20 of 72 entries, 984 total columns

Table 1 shows the original Revised All Data.xlsx excel file provided to me. **Table 2** shows the data frame produced after completing objectives 1, 2, and 3 as detailed in the methods section, steps 1-3.

Results Table 3:

	Test_Groups	Animal_ID	A2SW69	A6YRY8	A7UHZ2	A8DR93	B0FYY4	B0FZL9	B2LYL4	B6UV59	B7TJ15	B9VH02	C3V6N
1	Control_D135	107F2	95.4	99.2	98.1	100.5	92.9	87.0	98.0	102.7	89.9	90.2	102.2
2	Control_D135	79F1	94.5	97.3	103.5	105.1	118.4	97.3	115.9	88.9	107.7	88.3	95.2
3	Control_D135	75F2	96.4	81.7	92.5	94.2	92.1	93.2	115.3	89.2	99.6	79.9	96.3
4	Control_D135	107F1	92.5	110.2	97.3	100.0	99.9	96.1	95.2	102.8	101.9	97.8	98.8
5	Control_D135	75F3	92.9	85.4	107.0	89.4	107.9	92.1	91.2	96.1	97.0	72.2	92.0
6	Control_D135	81F1	93.3	88.1	96.9	96.5	112.1	114.4	114.6	115.6	84.3	82.6	85.6
7	Control_D135	41F1	117.4	96.8	96.0	92.9	99.2	94.9	95.8	91.9	95.3	88.2	98.8
8	Control_D90	119F2	62.7	81.5	112.2	109.2	94.9	90.0	84.8	106.6	109.7	106.4	112.4
9	Control_D90	87F1	160.7	103.1	87.2	86.1	117.1	135.2	124.8	100.3	80.3	97.9	90.0
10	Control_D90	38F2	84.8	105.9	98.9	100.9	91.3	83.0	85.8	97.9	107.4	99.2	92.8
11	Control_D90	87F2	83.7	101.9	99.8	105.2	88.2	115.5	105.6	77.5	104.7	109.4	102.1
12	Control_D90	124F1	95.6	108.6	100.3	98.1	97.0	103.9	100.1	96.0	100.8	99.5	91.3
13	Control_D90	119F1	81.0	87.9	107.2	98.1	109.7	93.7	98.5	108.8	106.1	98.8	101.4
14	Control_D90	38F1	95.7	114.0	101.0	100.3	90.4	105.3	90.5	99.5	106.4	97.6	92.6
15	NA	10F1	99.9	96.7	97.4	92.0	89.8	97.9	88.3	77.6	105.6	99.3	104.9
16	NA	75F1	100.7	103.4	106.0	94.1	105.2	87.7	89.0	91.4	99.8	98.3	112.8
17	NA	66F1	101.1	97.1	94.6	111.7	100.8	97.4	105.7	98.8	98.4	96.0	99.8

Showing 1 to 20 of 72 entries, 986 total columns

Table 3 shows the data frame produced after completing objectives 4, detailed in the methods section, steps 4 and 5. The test group rows with NA represent the Animal ID's, and affiliated test groups, that were not provided to me and therefore were not included in the final box plot analysis.

Results Table 4:

	Test_Groups	Animal_ID	A2SW69	A6YRY8	A7UHZ2	A8DR93	B0FYY4
1	Control_D90	119F2	62.7	81.5	112.2	109.2	94.9
2	Control_D90	87F1	160.7	103.1	87.2	86.1	117.1
3	Control_D90	38F2	84.8	105.9	98.9	100.9	91.3
4	Control_D90	87F2	83.7	101.9	99.8	105.2	88.2
5	Control_D90	124F1	95.6	108.6	100.3	98.1	97.0
6	Control_D90	119F1	81.0	87.9	107.2	98.1	109.7
7	Control_D90	38F1	95.7	114.0	101.0	100.3	90.4
8	OverFed_D90	120F1	68.4	104.0	94.9	96.1	81.2
9	OverFed_D90	117F1	99.4	86.0	102.8	100.9	101.2
10	OverFed_D90	120F2	88.2	79.4	103.3	100.3	112.9
11	OverFed_D90	26F1	155.9	107.5	93.2	91.3	102.3
12	OverFed_D90	12F1	93.8	101.9	94.5	94.3	98.7
13	OverFed_D90	26F2	89.1	100.5	97.8	99.0	91.7
14	Restricted_D90	19F1	85.2	106.3	107.6	107.9	100.4
15	Restricted_D90	43F2	66.2	104.2	106.4	107.5	96.9
16	Restricted_D90	20F1	93.9	103.0	103.9	98.8	102.7
17	Restricted_D90	20F2	127.4	107.6	97.2	95.0	103.7
18	Restricted_D90	43F1	113.0	88.1	102.3	113.2	105.8

Showing 1 to 18 of 18 entries, 986 total columns

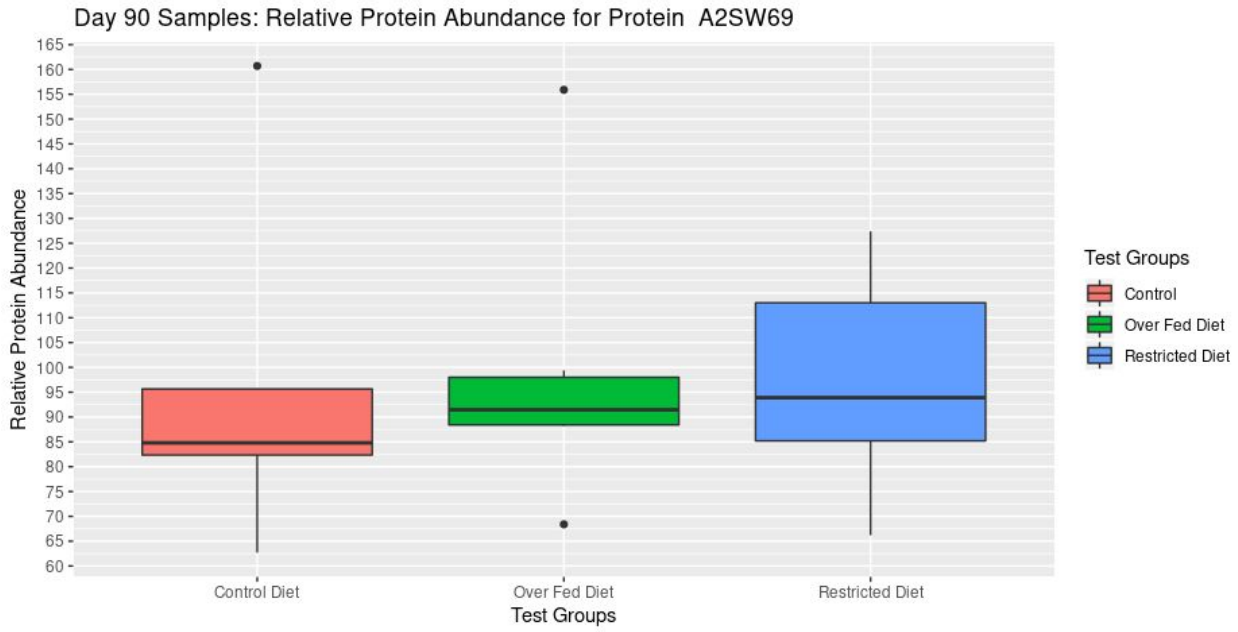
Results Table 5:

	Test_Groups	Animal_ID	A2SW69	A6YRY8	A7UHZ2	A8DR93	B0FYY4
1	Control_D135	107F2	95.4	99.2	98.1	100.5	92.9
2	Control_D135	79F1	94.5	97.3	103.5	105.1	118.4
3	Control_D135	75F2	96.4	81.7	92.5	94.2	92.1
4	Control_D135	107F1	92.5	110.2	97.3	100.0	99.9
5	Control_D135	75F3	92.9	85.4	107.0	89.4	107.9
6	Control_D135	81F1	93.3	88.1	96.9	96.5	112.1
7	Control_D135	41F1	117.4	96.8	96.0	92.9	99.2
8	OverFed_D135	56F1	101.7	88.4	105.8	105.9	93.1
9	OverFed_D135	93F1	88.7	85.5	104.1	97.7	105.9
10	OverFed_D135	21F1	106.1	99.4	114.0	114.6	104.6
11	OverFed_D135	89F1	100.0	111.9	92.1	111.7	99.0
12	OverFed_D135	93F2	92.2	93.2	107.6	107.8	109.6
13	OverFed_D135	63F1	82.7	102.6	97.8	100.3	89.3
14	Restricted_D135	118F1	104.2	104.5	93.5	94.8	100.7
15	Restricted_D135	69F1	104.3	113.9	101.8	101.7	105.0
16	Restricted_D135	3F2	111.5	104.9	93.5	91.2	83.1
17	Restricted_D135	94F2	104.8	118.3	98.6	102.3	114.4
18	Restricted_D135	69F2	108.6	106.2	96.5	96.3	79.5

Showing 1 to 21 of 21 entries, 986 total columns

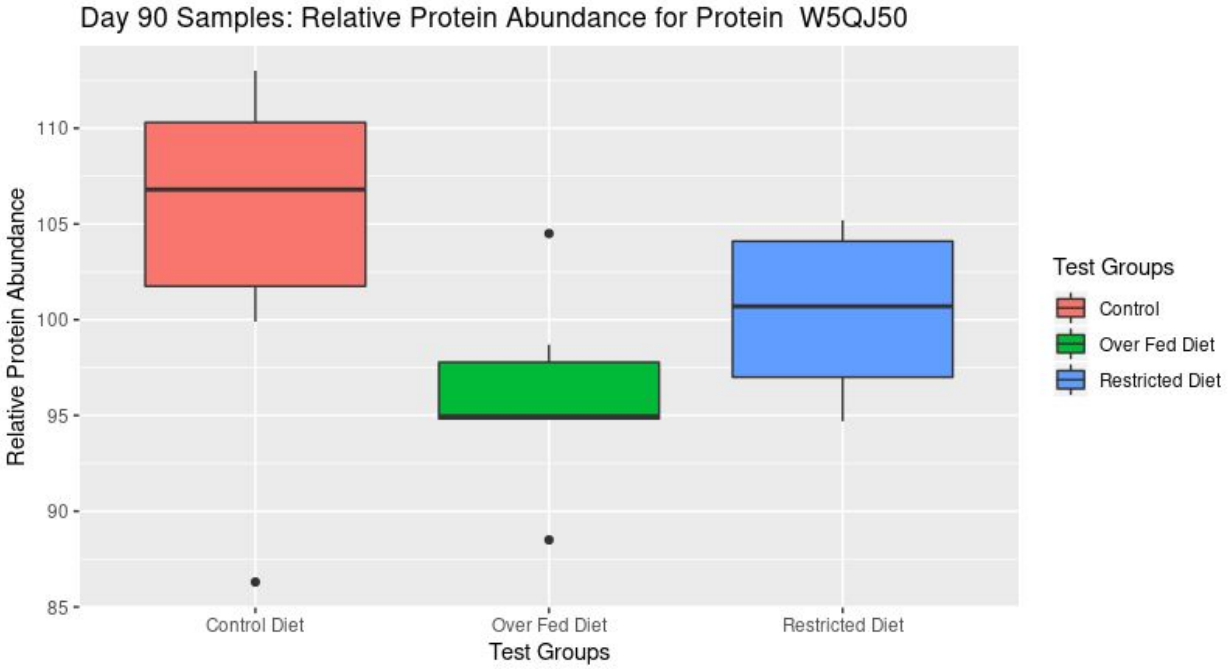
Table 4 and **Table 5** show the data frames produced after completing objectives 5 and 6, detailed in the methods section, steps 6 and 7. These are the data frames I used for plotting, since I removed all rows with undetermined test groups and converted the numbers class type from factor to numerical.

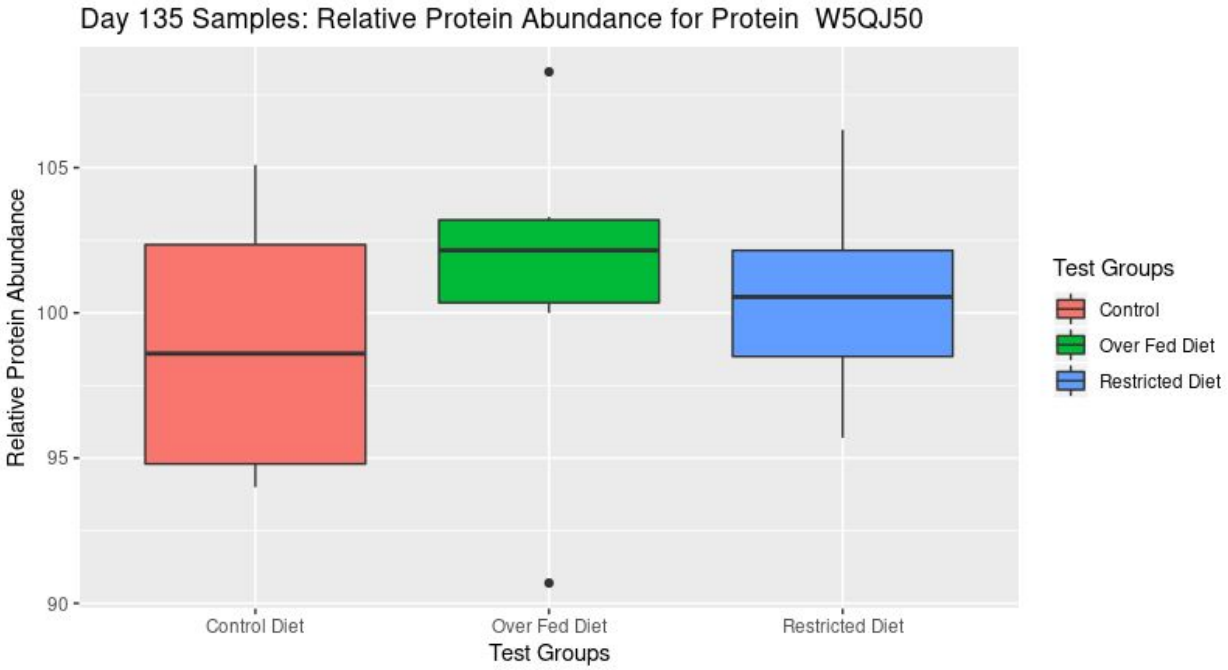
Results Box Plots 1 and 2 (Protein A2SW69):



Box Plot 1 and 2 represent the three treatment group's relative abundance of protein A2SW69 in the fetal muscle tissue collected at day 90 and day 135 of gestation. The code used to generate these box plots is detailed in the methods section, step 8.

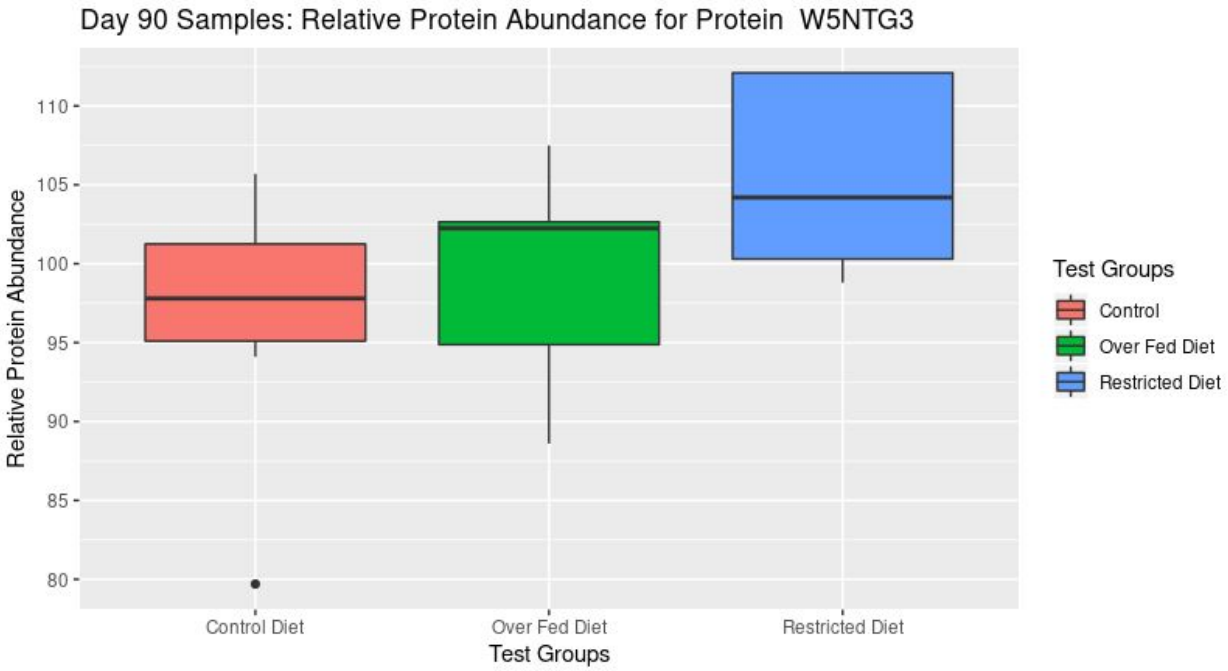
Results Box Plots 3 and 4 (Protein W5QJ50):

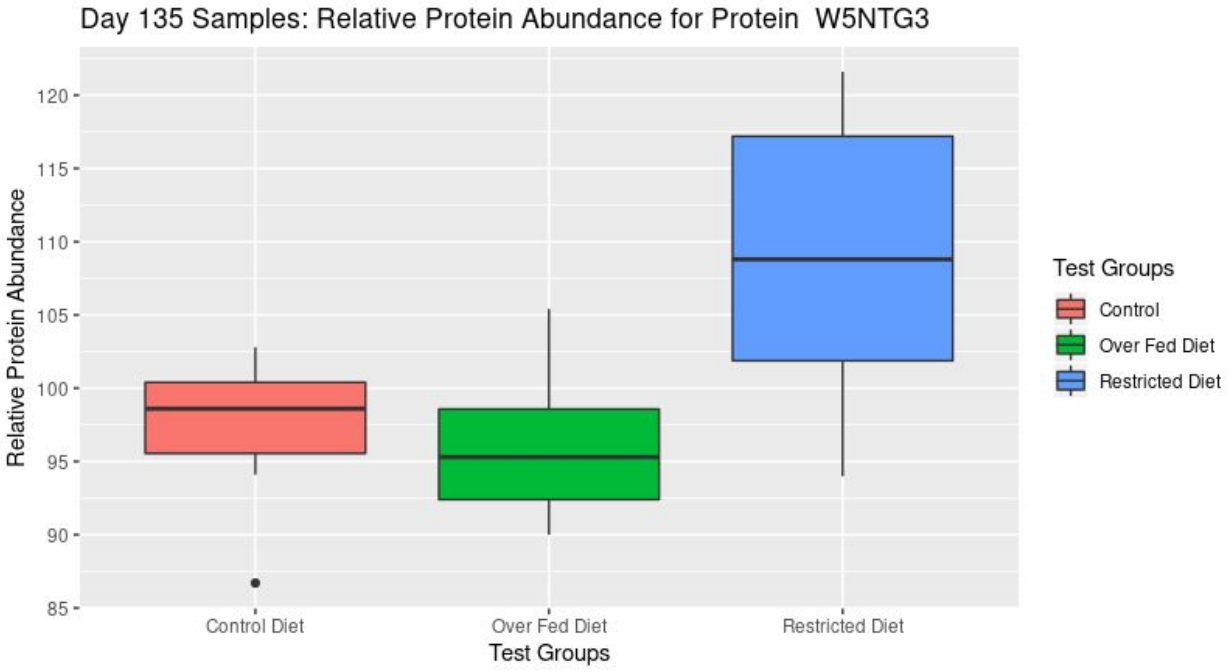




Box Plot 3 and 4 represent the three treatment group's relative abundance of protein W5QJ50 in the fetal muscle tissue collected at day 90 and day 135 of gestation. The code used to generate these box plots is detailed in the methods section, step 8.

Results Box Plots 5 and 6 (Protein W5NTG3):





Box Plot 5 and 6 represent the three treatment group's relative abundance of protein W5NTG3 in the fetal muscle tissue collected at day 90 and day 135 of gestation. The code used to generate these box plots is detailed in the methods section, step 8.

Conclusion:

As of right now, Dr. Reed has yet to get back to me with guidance on how to properly analyze the dataset provided, and without in depth background knowledge on protemic genetic analysis, it is hard for me to compare and draw conclusions on the box plots generated from my R script. The main focus I had for this assignment was to organize the data as requested by Dr. Reed and furthermore generate box plots that could later be interpreted by Dr. Reed, or even myself once taught. However, I do know that this study was conducted on ewes, pregnant with twins, and involved starting the ewes on a feeding trial at day 30 of their gestation. Currently the NRC, or National Research Council, is responsible for dictating the proper nutritional requirements for sheep to thrive. Based on the NRC requirements for a ewe pregnant with twins, the ewes were separated into three test groups, each of which was fed a treatment diet that met some percentage of the NRC requirement. The control group was fed a diet that met 100% of the NRC requirement, while the overfed and restricted diet groups met 140% and 60% of the nutritional requirements respectively. At day 90 and day 135 of gestation, the ewes were euthanized and fetal muscle tissue was collected from each of the fetuses. This tissue was then analyzed via mass spectroscopy, and the data found was then manually inputted into the excel sheet by an undergraduate in Dr. Reeds lab. Due to the large size of this data set, manual input was no longer practical, therefore the development of this R script was needed to effectively and efficiently organize the data for future in depth analysis.