Imputation of Missing Body Sizes in Parasites

Daniel Smith

17 April 2019

## Intro

## Missing Data Structure

Before we move further with imputation of missing values it’s important to understand the overall structure of data and the missing values present within that.

### Visualising Missing Data for SSA

Load the required packages.

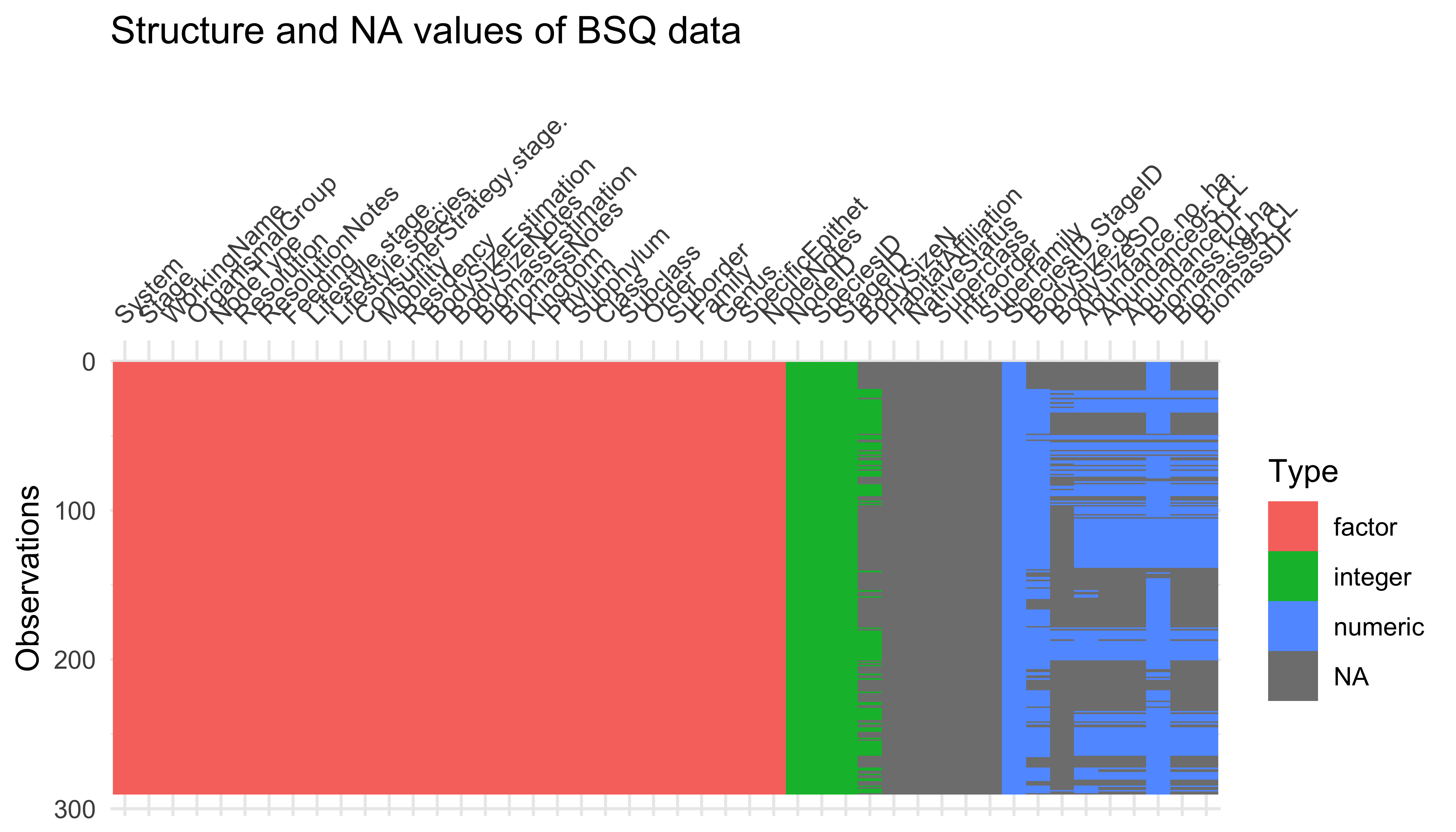
library(tidyverse) # general data wizadry and manipulation  
library(VIM) # visualise missing data patterns  
library(naniar) # data structure visualisation in a 'tidy' manner  
library(visdat) # accompanies the naniar package  
  
# set environment  
rm(list=ls()) # remove all objects  
set.seed(12) # set the random seed to 12 so results are reproducible

First we need to read in the required data from a GitHub repositary. For this methods demonstration we will just use the BSQ subportion of the Carpinteria foodweb as an example. The process is exactly the same for all foodwebs.

bsq <- read.csv("https://raw.githubusercontent.com/SmithD19/FoodWeb/master/data/interactionwebdb/Carpinteria/BSQweb\_Nodes.csv") # load data from GitHub repo - simple

A plot below using the visdat library allows us to visualise this further and gives more information about overall data structure and the type of data.

# Overall data structure and missing values using visdat library  
vis\_dat(bsq) +  
 labs(title = "Structure and NA values of BSQ data")



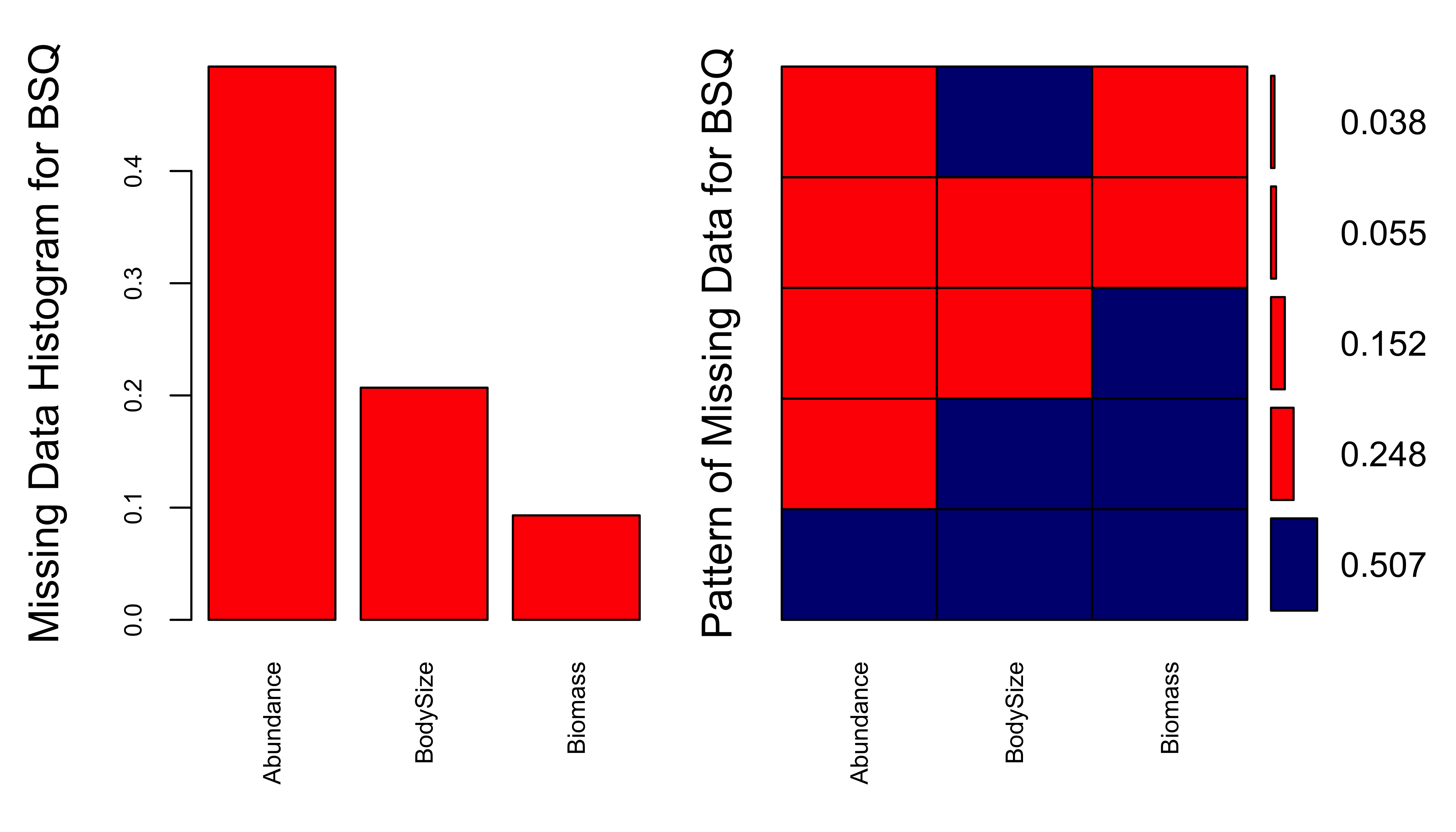
We can clearly see the NA values in the data frame here marked by the grey blocking. This also makes it easy for us to select only the values we care about for imputation, the trait values and give them better, easier to write names in a new and less cluttered data frame.

bsq\_wrk <-   
 bsq %>% # select and rename some variables  
 dplyr::select(FunctionalGroup = ConsumerStrategy.stage., BodySize = BodySize.g.,   
 Biomass = Biomass.kg.ha., Abundance = Abundance.no..ha.) %>%   
 mutate(BodySize = BodySize / 1000) # change BodySize from g to Kg so that units are equal  
  
# take a look at the new dataset  
glimpse(bsq\_wrk)

## Observations: 290  
## Variables: 4  
## $ FunctionalGroup <fct> detritus, detritus, detritus, autotroph, autotro…  
## $ BodySize <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  
## $ Biomass <dbl> 3.779020e-02, 1.080667e+02, 1.080667e+03, 7.3193…  
## $ Abundance <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …

Using the handy VIM package we can quickly visualise the missing data present in the original data set - in particular the trait values of the nodes are what we’re interested in for now.

aggr(bsq\_wrk[,2:4], col=c('navyblue','red'),   
 numbers=TRUE, sortVars=TRUE, labels=names(bsq\_wrk[, 2:4]),   
 cex.axis=.7, gap=3, ylab=c("Missing Data Histogram for BSQ","Pattern of Missing Data for BSQ"))



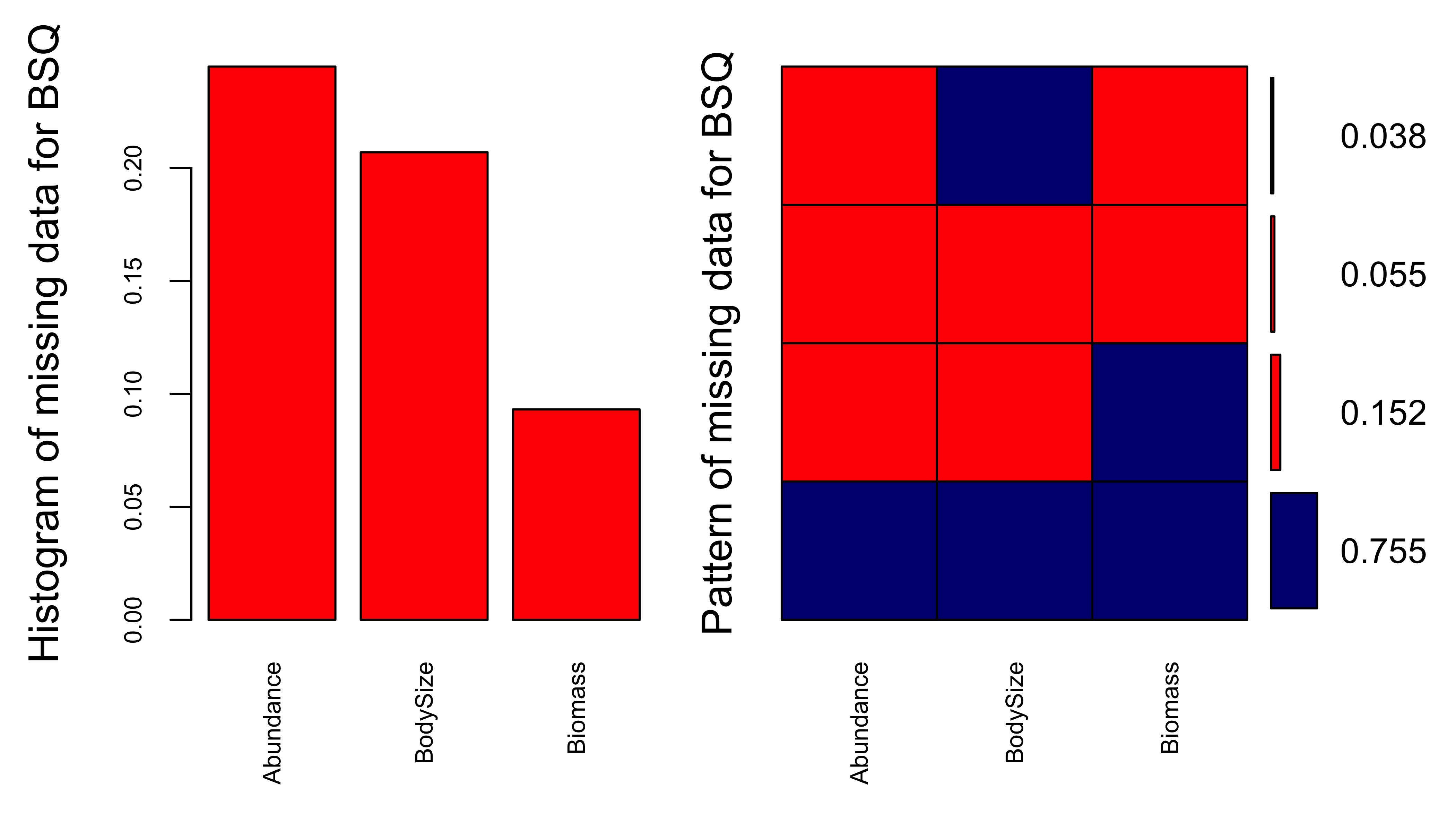
Using the equation we can conditionally fill in variables that are missing from data points using mutate() and case\_when(). Below is the workflow used for conditionally replacing this data based on the equation and reassembling the data back to it’s original structure but containing the new values inferred from the equation above.

bsq\_new <-   
 bsq\_wrk %>%   
 # if BodySize = NA & if Biomass & Abundance != NA then BodySize = Biomass/Abundance  
 mutate(BodySizeNew = case\_when(!is.na(Abundance & Biomass) ~ Biomass / Abundance),  
 # if Abundance = NA and BodySize & Biomass != NA then Abundance = Biomass/BodySize   
 AbundanceNew = case\_when(!is.na(BodySize & Biomass) ~ Biomass / BodySize),  
 # if Biomass = NA and BodySize & Abundance != NA then Biomass = Abundance \* BodySize  
 BiomassNew = case\_when(!is.na(Abundance & BodySize) ~ Abundance \* BodySize)) %>%   
 # join the columns of new and old together use coalesce -- Biomass + BiomassNew...  
 # this selects the original value first so only values that are missing from original dataset and then computed are selected   
 mutate(BodySize\_Work = coalesce(BodySize, BodySizeNew),  
 Biomass\_Work = coalesce(Biomass, BiomassNew),  
 Abundance\_Work = coalesce(Abundance, AbundanceNew)) %>%   
 # now dplyr::select and rename the new working colums to replace the old incomplete data set  
 dplyr::select(FunctionalGroup, BodySize = BodySize\_Work,  
 Biomass = Biomass\_Work, Abundance = Abundance\_Work)

Now with the extra values we added from the inferred relationship between BodySize, Biomass and Abundance we can look at how much data we have to work with when getting ready for the imputation process.

First lets take a look at another plot using the VIM aggr plotting function.

aggr(bsq\_new[,2:4], col=c('navyblue','red'),  
 numbers=TRUE, sortVars=TRUE, labels=names(bsq\_new[, 2:4]),  
 cex.axis=.7, gap=3, ylab=c("Histogram of missing data for BSQ","Pattern of missing data for BSQ"))



Now it looks like we have 75% of the data with no missing values for any of that traits we’re interested in. This is significantly better than the 50% we had earlier and gives us a better starting point for the imputation process using the mice package, which is shorthand for Multivariate Imputation using Chained Equations.

## Imputation

For imputation we have the choice of several packages in the CRAN repository. For now the most appropriate package is that of mice as shown by penone2014, without phylogenetic data this method provides the best imputation results, particularly for body size data and traits highly correlated to this.

library(mice) # R package for Multivariate Imputation using Chained Equations  
# Be aware that the MASS package attatched to mice masks the select() function by dplyr  
# It is therefore necessry to specifically call dplyr::select() to use this function

### Dealing with co-linearity

Lets create a new data frame for the imputation process. It’s important in the imputation process to log transform values that could be correlated with each otehr, this process reduces the co-linearity of the values and therefore makes imputation more accurate.

bsq\_impute <- bsq\_new %>% dplyr::select(BodySize, Biomass, Abundance) %>%   
 # log transform the variables you want to impute, addition of 1 solves problems   
 # with infinite results due to logging values below 0  
 mutate(BodySize = log(BodySize + 1), Biomass = log(Biomass + 1), Abundance = log(Abundance + 1))

### Setup of mice()

The code below is standard setup and can be used for any imputation in mice, just change the dataset name.

init = mice(bsq\_impute, maxit=0)   
init # An overview of the mice object you just created using the code above and your dataset

## Class: mids  
## Number of multiple imputations: 5   
## Imputation methods:  
## BodySize Biomass Abundance   
## "pmm" "pmm" "pmm"   
## PredictorMatrix:  
## BodySize Biomass Abundance  
## BodySize 0 1 1  
## Biomass 1 0 1  
## Abundance 1 1 0

Before we run the imputation we want to check which values mice is using for the imputation, by default this is all the variables you provide it with, this may not be the best approach as we only need BodySize and Abundance variables for our future analysis, so using the Biomass variable as a predictor that won’t be imputed but is still used in the imputation process is probably correct, but for this example we will continue with all variables being imputed and for imputation prediction but the example code of how to look at what mice is using will be shown below.

meth = init$method  
init$method # The method used for the imputation of each variable

## BodySize Biomass Abundance   
## "pmm" "pmm" "pmm"

predM = init$predictorMatrix  
init$predictorMatrix # The prediction matrix for your imputed dataset

## BodySize Biomass Abundance  
## BodySize 0 1 1  
## Biomass 1 0 1  
## Abundance 1 1 0

### mice() usage

Using the imputation function mice() we can now impute the missing data. The m value corresponds to the number of data sets imputed and for this example we will use 10 as it’s easier to show, but for real analysis more appropriate iterations of imputation are suggested (50+). The data will be imputed using the pmm method which is short for predictive mean matching. A list of methods for imputation using mice are listed below.

bsq\_imputed <- mice(bsq\_impute, m = 10,   
 method=meth, # using both the methods and predictor matrix we specified earlier  
 predictorMatrix=predM) # we didnt change any values in the predictor matrix so this isn't entirely necessary

# the number of imputation methods are available by calling this command  
methods(mice)

## [1] mice.impute.2l.bin mice.impute.2l.lmer   
## [3] mice.impute.2l.norm mice.impute.2l.pan   
## [5] mice.impute.2lonly.mean mice.impute.2lonly.norm   
## [7] mice.impute.2lonly.pmm mice.impute.cart   
## [9] mice.impute.jomoImpute mice.impute.lda   
## [11] mice.impute.logreg mice.impute.logreg.boot   
## [13] mice.impute.mean mice.impute.midastouch   
## [15] mice.impute.norm mice.impute.norm.boot   
## [17] mice.impute.norm.nob mice.impute.norm.predict  
## [19] mice.impute.panImpute mice.impute.passive   
## [21] mice.impute.pmm mice.impute.polr   
## [23] mice.impute.polyreg mice.impute.quadratic   
## [25] mice.impute.rf mice.impute.ri   
## [27] mice.impute.sample mice.mids   
## [29] mice.theme   
## see '?methods' for accessing help and source code

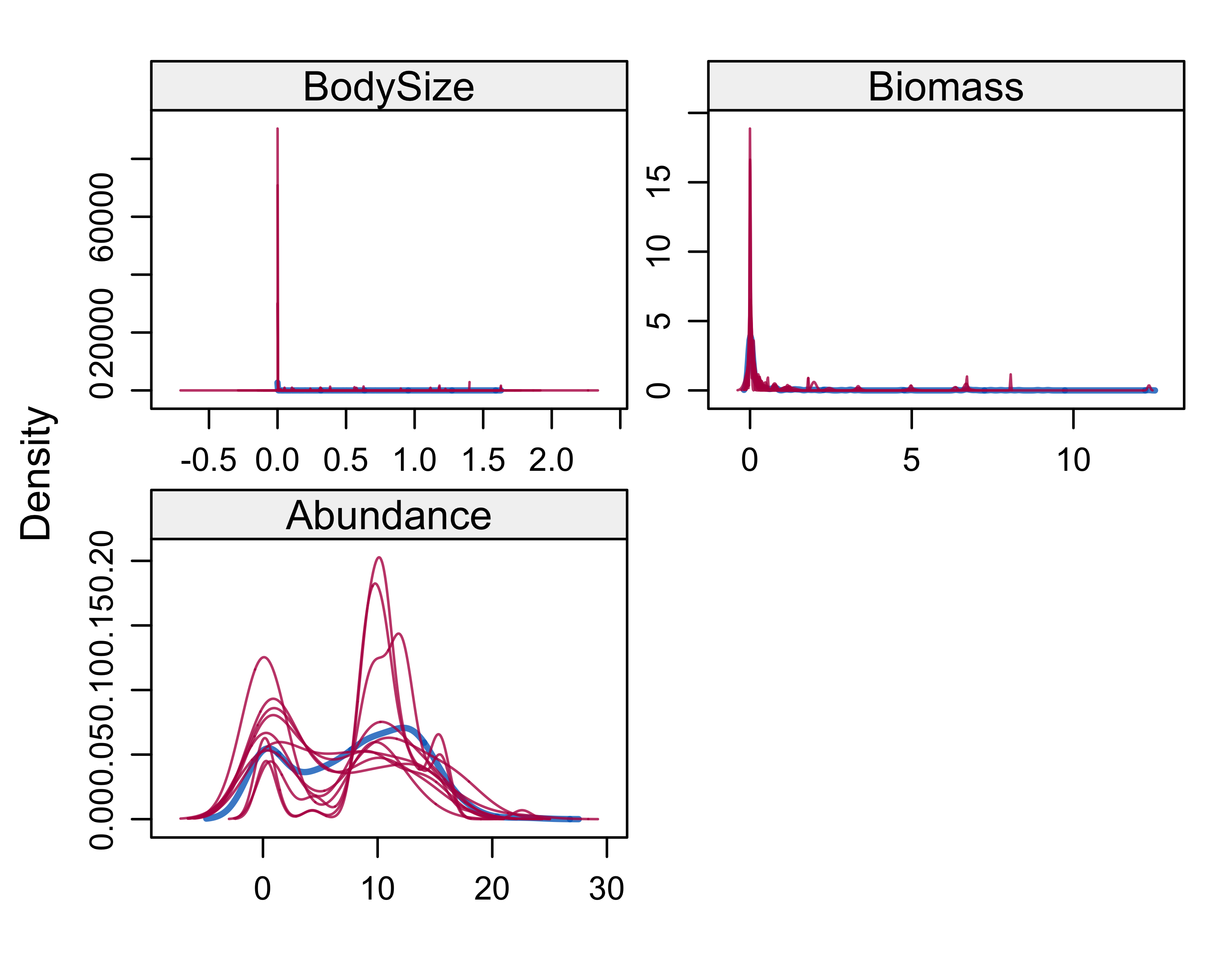
### Visual analysis of imputed data

We can look at some of the imputed values and compare them to our original data set using some convenient functions provided in the mice package.

#### Density plots

This density plot shows us the density of imputed values compared to our original input data (red and blue, respectively), assuming that BodySize is proportional to Abundance and Biomass.

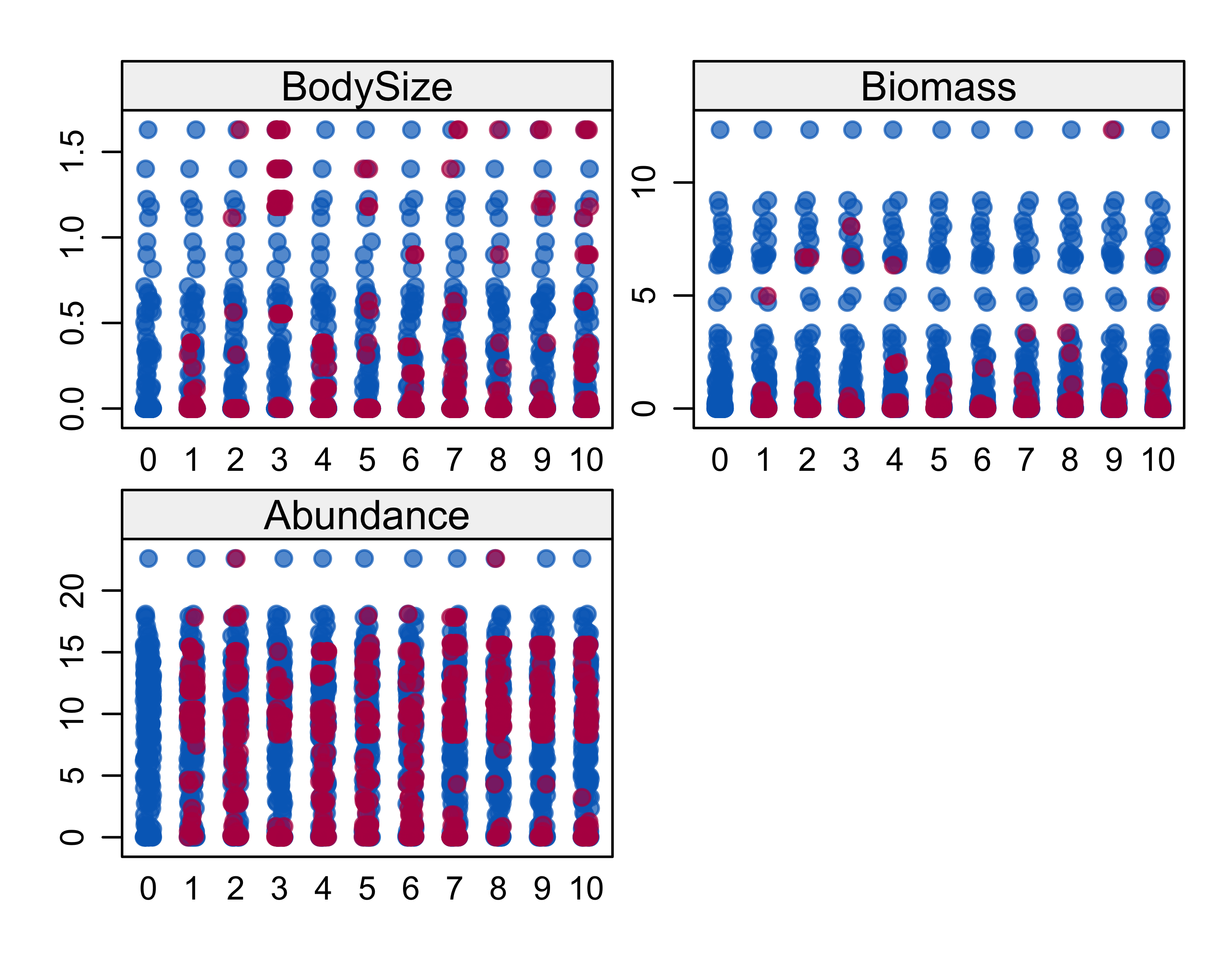
densityplot(bsq\_imputed)



#### Strip plots

A strip plot here shows the imputed values of each value compared tothe original data set for each imputed data set (red and blue, respectively). It’s much easier here to see the variance in each imputed data set. Dealing with this variance is important and we will be using some functions present in the mice package to coalesce all this data together, and then to test the significance of it compared to the original data.

stripplot(bsq\_imputed, pch = 20, cex = 1.2)



To complete the imputation process the complete() function does all the hard work for you, turning your mice object back into a data frame that contains all the imputed values inputted into it, giving you a complete data frame. There are various ways to complete this and even add multiple imputed data frames into one large data frame or a list.

bsq\_complete <- complete(bsq\_imputed, "all") # use the all the imputed data sets  
bsq\_complete <- complete(bsq\_imputed, 1) # use the first imputed data set  
glimpse(bsq\_complete)

## Observations: 290  
## Variables: 3  
## $ BodySize <dbl> 3.833558e-01, 6.565608e-06, 3.740000e-09, 3.370000e-08…  
## $ Biomass <dbl> 0.03709364, 4.69195935, 6.98625837, 8.89841114, 6.6498…  
## $ Abundance <dbl> 0.09531018, 10.33295268, 15.06015911, 8.34503026, 9.81…

summary(bsq\_complete)

## BodySize Biomass Abundance   
## Min. :0.0000000 Min. : 0.000000 Min. : 0.000   
## 1st Qu.:0.0000000 1st Qu.: 0.000283 1st Qu.: 4.331   
## Median :0.0000027 Median : 0.016625 Median : 9.347   
## Mean :0.0864769 Mean : 0.672503 Mean : 8.433   
## 3rd Qu.:0.0007833 3rd Qu.: 0.265289 3rd Qu.:12.256   
## Max. :1.6292405 Max. :12.333661 Max. :22.597

The last step in this process is to return the values back to their original states so we can compare the values to the original input values we looked at in real terms, seeing if they make sense or not

# Transform the data back into non-logarithmic form using the exponent function  
bsq\_transformed <-   
 bsq\_complete %>%   
 mutate(BodySizeTran = exp(BodySize - 1),  
 BiomassTran = exp(Biomass - 1),  
 AbundanceTran = exp(Abundance - 1))  
# the results of the mutation  
glimpse(bsq\_transformed)

## Observations: 290  
## Variables: 6  
## $ BodySize <dbl> 3.833558e-01, 6.565608e-06, 3.740000e-09, 3.370000…  
## $ Biomass <dbl> 0.03709364, 4.69195935, 6.98625837, 8.89841114, 6.…  
## $ Abundance <dbl> 0.09531018, 10.33295268, 15.06015911, 8.34503026, …  
## $ BodySizeTran <dbl> 0.5397527, 0.3678819, 0.3678794, 0.3678795, 0.3678…  
## $ BiomassTran <dbl> 3.817817e-01, 4.012339e+01, 3.979229e+02, 2.693000…  
## $ AbundanceTran <dbl> 4.046674e-01, 1.130446e+04, 1.277172e+06, 1.548482…

# summary of descriptive statistics  
summary(bsq\_transformed)

## BodySize Biomass Abundance BodySizeTran   
## Min. :0.0000000 Min. : 0.000000 Min. : 0.000 Min. :0.3679   
## 1st Qu.:0.0000000 1st Qu.: 0.000283 1st Qu.: 4.331 1st Qu.:0.3679   
## Median :0.0000027 Median : 0.016625 Median : 9.347 Median :0.3679   
## Mean :0.0864769 Mean : 0.672503 Mean : 8.433 Mean :0.4163   
## 3rd Qu.:0.0007833 3rd Qu.: 0.265289 3rd Qu.:12.256 3rd Qu.:0.3682   
## Max. :1.6292405 Max. :12.333661 Max. :22.597 Max. :1.8762   
## BiomassTran AbundanceTran   
## Min. : 0.37 Min. :0.000e+00   
## 1st Qu.: 0.37 1st Qu.:2.800e+01   
## Median : 0.37 Median :4.230e+03   
## Mean : 333.76 Mean :8.807e+06   
## 3rd Qu.: 0.48 3rd Qu.:7.736e+04   
## Max. :83588.45 Max. :2.396e+09

We can then add these new transformed and inputed values back into the original data set so we can compare side by side the new and old values, along with the additional data we were given such as species phylogeny and working names.

bsq\_compare <- # new data.frame to compare the values  
 bsq\_wrk %>% # mutate this into log values so we can compare to the imputed  
 mutate(BodySize = log(BodySize + 1),   
 Biomass = log(Biomass + 1),   
 Abundance = log(Abundance + 1))  
  
bsq\_compare <- # bind the new and old data together into one large dataset  
 bsq\_complete %>%   
 select(BodySizeImputed = BodySize,  
 AbundanceImputed = Abundance,   
 BiomassImputed = Biomass) %>%   
 bind\_cols(bsq\_compare) # binding function

We can compare the new and old values grouping by functional groups to test to see if the data is significantly different form one another, and visualise this using boxplots.

# does the imputed data set's mean differ from the original data?  
t.test(bsq\_compare$BodySizeImputed, bsq\_compare$BodySize)

##   
## Welch Two Sample t-test  
##   
## data: bsq\_compare$BodySizeImputed and bsq\_compare$BodySize  
## t = -0.51527, df = 469.56, p-value = 0.6066  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -0.05405726 0.03159707  
## sample estimates:  
## mean of x mean of y   
## 0.08647687 0.09770696

t.test(bsq\_compare$AbundanceImputed, bsq\_compare$Abundance)

##   
## Welch Two Sample t-test  
##   
## data: bsq\_compare$AbundanceImputed and bsq\_compare$Abundance  
## t = 1.5991, df = 283.16, p-value = 0.1109  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -0.198460 1.917464  
## sample estimates:  
## mean of x mean of y   
## 8.433489 7.573988

t.test(bsq\_compare$BiomassImputed, bsq\_compare$Biomass)

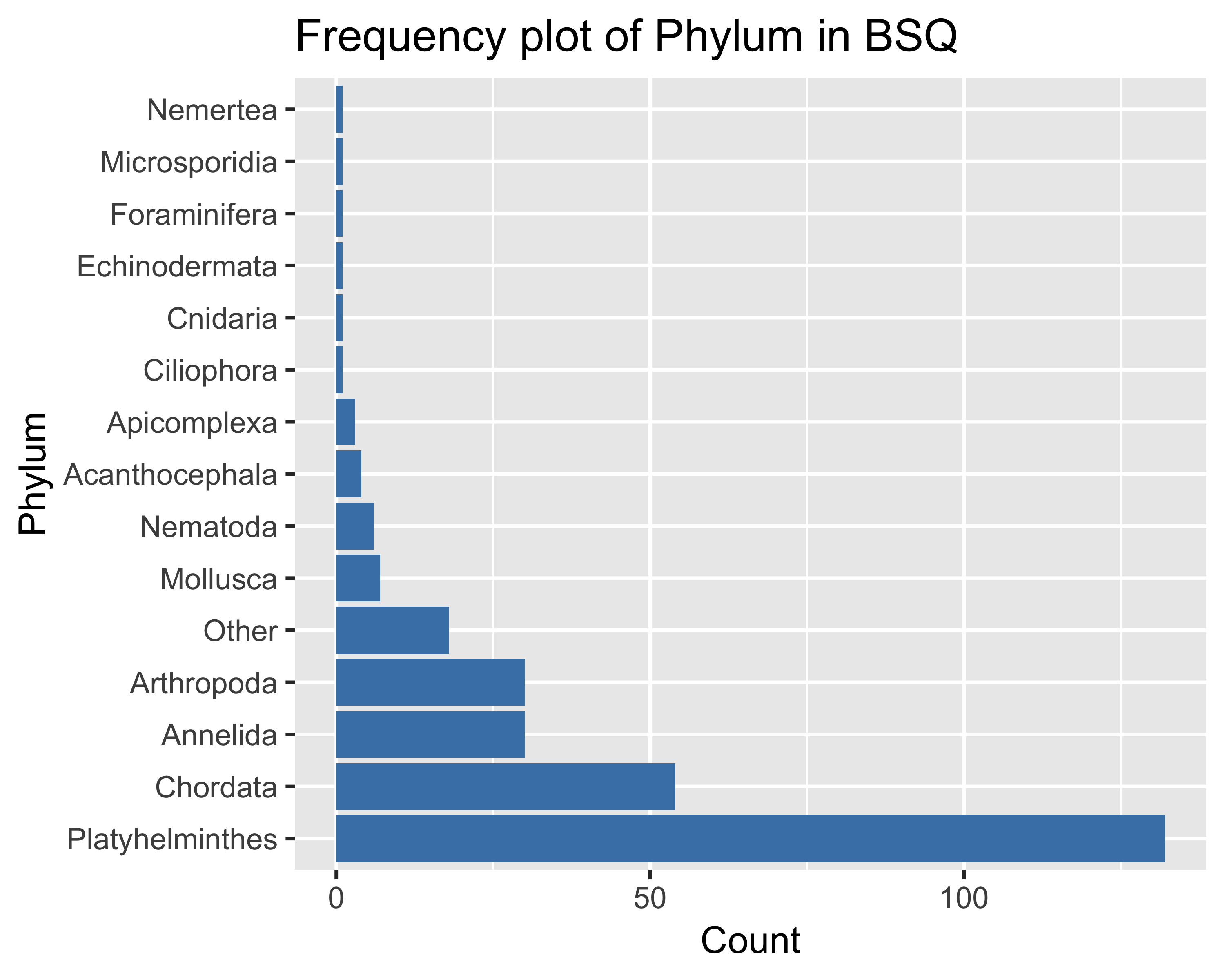
##   
## Welch Two Sample t-test  
##   
## data: bsq\_compare$BiomassImputed and bsq\_compare$Biomass  
## t = -0.25171, df = 541.59, p-value = 0.8014  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -0.3503823 0.2707877  
## sample estimates:  
## mean of x mean of y   
## 0.6725034 0.7123007

The datasets do not differ in means, statistically insignificantly anyway.

## Alternative approaches and accuracy

There may be more appropriate methods of imputation that take adavantage of evolutionary models or phylogeneteic distances that allow constraining of imputed values depending upon the values of their closest evolutionary relatives that may have data already present or imputed. Phylogenetic data is incomplete for my datasets but the majority of data points have taxa at least to the phylum level and below. With the most speciose phyla compromising the majority of data points present.

# phylogeny count plot  
plot.freq <-   
 bsq %>%  
 # count factor levels and frequency  
 count(Phylum) %>%   
 # rename blank factor varables to other  
 mutate(Phylum = fct\_recode(Phylum, "Other" = "")) %>%   
 # plot  
 ggplot(aes(x = reorder(Phylum, -n), y = n)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 ggtitle("Frequency plot of Phylum in BSQ") +  
 xlab("Phylum") +  
 ylab("Count") +  
 coord\_flip()  
plot.freq



We can also visualise what levels of data are missing in relation to phylogenetic grouping using the naniar package and the gg\_miss\_fct() function that comes with this. It’s also useful to see how data is missing with regards to the functional grouping of that taxa in the data set. Combining these two extra variables we can easily see that clades and phyla typically containing smaller animals are more likely to have missing data points.

# several useful ways to visualise this missing data by functional and taxanomic grouping  
plot.missing.group <-   
 gg\_miss\_fct(x = bsq\_new, fct = FunctionalGroup) +   
 labs(title = "% NA Values by Functional Group for BSQ")  
  
plot.missing.phylo <-   
 bsq %>%   
 dplyr::select(BodySize = BodySize.g., Biomass = Biomass.kg.ha.,   
 Abundance = Abundance.no..ha., Phylum) %>%   
 gg\_miss\_fct(fct = Phylum) +   
 labs(title = "% NA Values by Phylogeny Group for BSQ")

