WLS_data_manipulation

Kim Dill-McFarland

April 10, 2017

Introduction

The Wisconsin Longitudinal Study (WLS) is a long-term study of 1957 high school graduates in Wisconsin (N=10317). The survey-based data includes physical and mental health as well as human-human and human-environment interactions from late adolescence (1957) through 2011. These include variables such as social background, aspirations, schooling, military service, labor market experiences, family characteristics and events, social participation, psychological characteristics and retirement.

> Herd, Pamela, Deborah Carr, and Carol Roan. 2014. "Cohort Profile: Wisconsin Longitudinal Study (WLS)." International Journal of Epidemiology 43:34-41 PMCID: PMC3937969

In 2015-6, a microbiota component was added to the WLS with fecal sample collection. The bacterial microbiota was determined by amplicon sequencing of the 16S rRNA gene, variable region 4 (V4). Graduates (g, N=179), their spouses (p, N=63), their siblings (s, N=134), and their sibling's spouses (e, N=32) were successfully sequenced.

This document describes manipulation of the WLS and microbiota data sets to be used in analysis. This includes

- determining beta-diversity for matched spouse and sibling pairs as well as unrelated persons
- pulling relevant variables out of the full WLS data (Pheno)
- pulling relevant variables out of the survey given at the time of fecal sampling (Pheno.sm)
- re-coding variables so that meaning is consistent across all samples.

More information and access to the data available at http://www.ssc.wisc.edu/wlsresearch/

Load data

Packages

```r
#The vegan package provides tools for descriptive community ecology. It has most basic functions of div
#In general, this package is used for Bray-Curtis and Jaccard analyses.
library(vegan)
```

```
## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.4-5
```

```r
#The phyloseq package seeks to address issues with multiple microbiome analysis packages by providing a
#In general, this package is used for UniFrac analyses.
library(phyloseq)
#Analyses of Phylogenetics and Evolution package. Required for tree calculations to be used with phylos
library(ape)
#Graphing package used in phyloseq. To edit the default setting of a plot, you need to use functions in
library(PMCMR)
#Reverse coding variables
```

```
library(psych)
#To read in mothur-formatted data files
library(phangorn)
```

```
##
## Attaching package: 'phangorn'

## The following objects are masked from 'package:vegan':
##
##     diversity, treedist
```

```
#To read in .dta formatted kinship data
library(foreign)
```

Functions

Pairwise adonis function. Modified from https://www.researchgate.net/post/How_can_I_do_PerMANOVA_pairwise_contrasts_in_R

```
source("Pairwise_adonis.r")
```

SIMPER related functions from https://github.com/asteinberger9/seq_scripts

```
source("R_krusk.r")
```

Data

Available through wls@ssc.wisc.edu

```
#.shared file generated using mothur. Samples normalizsed to 10,000 sequences per sample. 'label' and '
#Rows=samples Columns=OTUs
OTU = read.table("WLS.OTU.txt", header=TRUE, row.names=1, sep="\t")

#OTUs in .shared file summed into genus-level groups by taxonomy names
genera = as.data.frame(t(read.table("Genus.txt", header=TRUE, row.names=1, sep="\t")))

#Alpha-diversity metrics of normalized OTU table calculated with summary.single in mothur
Met = read.table("WLS.meta.txt", header=TRUE, row.names=1, sep="\t")

#.cons.taxonomy file generated in mothur. 'Size' column removed. Taxonomy split into separate columns f
Tax = read.table("WLS.taxonomy.txt", header=TRUE, row.names=1, sep="\t")

#Raw WLS survey dataset
Pheno = read.csv("Pheno.sub.csv", header=TRUE, sep=",")
    #Add g to the row names so sample IDs match the other data files
    row.names(Pheno) = paste(Pheno$idpriv, "g", sep = "")

#Survey at time of fecal sampling.
load("pheno_idpriv.rdata")
    Pheno.sm = pheno_idpriv
    #Rename and alter row names to include Link (g,p,s,e)
    row.names(Pheno.sm) = paste(Pheno.sm$idpriv, Pheno.sm$rtype, sep = "")
    #Remove samples that did not successfully sequence
    Pheno.sm = subset(Pheno.sm, row.names(Pheno.sm) %in% row.names(Met))
```

```
#Replace all negatives (missing data in WLS data) with blanks
Pheno = replace(Pheno, Pheno < 0, NA)
Pheno.sm = replace(Pheno.sm, Pheno.sm < 0, NA)

#Kinship scores
relatedness = read.dta("kinship_microbiome_saveold.dta")
```

Order all data by participant IDs

```
Met = Met[order(row.names(Met)),]
OTU = OTU[order(row.names(Met)),]
genera = genera[order(row.names(genera)),]
row.names(genera) = row.names(OTU)
Pheno = Pheno[order(row.names(Pheno)),]
Pheno.sm = Pheno.sm[order(row.names(Pheno.sm)),]
```

Beta-diversity

Phyloseq object creation (UniFrac)

For all samples

```
#Set random seed for reproducibility
set.seed(51234)

#Calcuate tree of OTUs
#Load in representative sequence distance matrix created in mothur. Then calculate a neighbor-joinging
#NJ.tree = bionj(import_mothur_dist("clean_repFasta.phylip.dist"))

#OR

#Load pre-calculated tree that was made using the above code
load("NJ.tree2.rdata")

#Designate OTU, metadata, and taxonomy tables + tree as such to phyloseq package
OTU.UF = otu_table(as.matrix(OTU), taxa_are_rows=FALSE)
Tax.UF = tax_table(as.matrix(Tax))
Met.UF = sample_data(Met)
NJ.tree.UF = phy_tree(NJ.tree)

#Create phyloseq object
physeq.tree = phyloseq(OTU.UF, Tax.UF, Met.UF, NJ.tree.UF)
```

For graduates only

```
#Subset the data
OTU.g = OTU[Met$Link == "g",]
Met.g = Met[Met$Link == "g",]
Pheno.sm.g = Pheno.sm[Pheno.sm$rtype == "g",]

#Designate objects
OTU.UF.g = otu_table(as.matrix(OTU.g), taxa_are_rows=FALSE)
Tax.UF.g = tax_table(as.matrix(Tax))
```

```r
#Add gender to Met
Met.g$gender = Pheno.sm.g$gender
Met.UF.g = sample_data(Met.g)
physeq.tree.g = phyloseq(OTU.UF.g, Tax.UF.g, Met.UF.g, NJ.tree.UF)
```

Calculate beta-diversity

```r
#For all data
BC.dist=as.matrix(vegdist(OTU, distance="bray"))
J.dist=(BC.dist*2)/(BC.dist+1)
wUF.dist = as.matrix(UniFrac(physeq.tree, weighted=TRUE, normalized=TRUE))
```

```
## Warning in UniFrac(physeq.tree, weighted = TRUE, normalized = TRUE):
## Randomly assigning root as -- Otu02123 -- in the phylogenetic tree in the
## data you provided.
```

```r
uwUF.dist = as.matrix(UniFrac(physeq.tree, weighted=FALSE, normalized=TRUE))
```

```
## Warning in UniFrac(physeq.tree, weighted = FALSE, normalized = TRUE):
## Randomly assigning root as -- Otu03197 -- in the phylogenetic tree in the
## data you provided.
```

```r
#For graduates only
BC.dist.g = as.matrix(vegdist(OTU.g, distance="bray"))
J.dist.g = (BC.dist.g*2)/(BC.dist.g+1)
wUF.dist.g = as.matrix(UniFrac(physeq.tree.g, weighted=TRUE, normalized=TRUE))
```

```
## Warning in UniFrac(physeq.tree.g, weighted = TRUE, normalized = TRUE):
## Randomly assigning root as -- Otu02631 -- in the phylogenetic tree in the
## data you provided.
```

```r
uwUF.dist.g = as.matrix(UniFrac(physeq.tree.g, weighted=FALSE, normalized=TRUE))
```

```
## Warning in UniFrac(physeq.tree.g, weighted = FALSE, normalized = TRUE):
## Randomly assigning root as -- Otu05558 -- in the phylogenetic tree in the
## data you provided.
```

Extract distances for spouse & sibling pairs

Create distance matrix from ID comparisons so that you can pull out gp, gs, se comparisons from the same family (ID distance = 0)

```r
ID.vec = as.vector(Met$ID)
ID.dist=as.matrix(dist(ID.vec, method="euclidean"))
```

Create matrix of link comparisons so that you can pull out gp (spouse), gs (sibling), and se (spouse) pairs

```r
#Create a vector of the link IDs (g,p,s,e)
Link.vec = as.vector(Met$Link)

#Using a loop, create a matrix of link ID comparison (i.e. gp, gs, ge, etc.)
Link.mat = matrix(numeric(0), length(ID.vec), length(ID.vec))
for(j in 1:dim(Link.mat)[1] ){
  for(k in 1:dim(Link.mat)[2]){
```

```
  Link.mat[j, k] = paste(Link.vec[j], Link.vec[k], sep = "")
 }}
```

Pull out pairs of interest

- Link.mat == “gp” or “se” or “gs”
- ID.dist == 0

Bray-Curtis

```
#Create a vector of the gp comparisons where the graduate IDs match
match.BC.gp = BC.dist[ID.dist == 0 & Link.mat == "gp"]
#Rename the vector values using the graduate ID
  names(match.BC.gp) = ID.vec[which(ID.dist == 0 & Link.mat == "gp", arr.ind = T)[, 1]]
#Add g to the names to differ from gp comparison names
  names(match.BC.gp) = paste(names(match.BC.gp), "g", sep = "")
#Create a vector of the se comparisons where the graduate IDs match
match.BC.se = BC.dist[ID.dist == 0 & Link.mat == "se"]
#Rename the vector values using the graduate ID
  names(match.BC.se) = ID.vec[which(ID.dist == 0 & Link.mat == "se", arr.ind = T)[, 1]]
#Add s to the names to differ from gp comparison names
  names(match.BC.se) = paste(names(match.BC.se), "s", sep = "")
#Combine gp and se vectors
match.BC.gp.se = append(match.BC.gp, match.BC.se)

  #Create a vector of the gs comparisons where the graduate IDs match
match.BC.gs = BC.dist[ID.dist == 0 & Link.mat == "gs"]
  names(match.BC.gs) = ID.vec[which(ID.dist == 0 & Link.mat == "gs", arr.ind = T)[, 1]]
  names(match.BC.gs) = paste(names(match.BC.gs), "g", sep = "")
```

Repeat for all beta-diversity metrics

```
#Jaccard
match.J.gp = J.dist[ID.dist == 0 & Link.mat == "gp"]
  names(match.J.gp) = ID.vec[which(ID.dist == 0 & Link.mat == "gp", arr.ind = T)[, 1]]
  names(match.J.gp) = paste(names(match.J.gp), "g", sep = "")
match.J.se = J.dist[ID.dist == 0 & Link.mat == "se"]
  names(match.J.se) = ID.vec[which(ID.dist == 0 & Link.mat == "se", arr.ind = T)[, 1]]
  names(match.J.se) = paste(names(match.J.se), "s", sep = "")
match.J.gp.se = append(match.J.gp, match.J.se)

match.J.gs = J.dist[ID.dist == 0 & Link.mat == "gs"]
  names(match.J.gs) = ID.vec[which(ID.dist == 0 & Link.mat == "gs", arr.ind = T)[, 1]]
  names(match.J.gs) = paste(names(match.J.gs), "g", sep = "")

#Weighted UniFrac
match.wUF.gp = wUF.dist[ID.dist == 0 & Link.mat == "gp"]
  names(match.wUF.gp) = ID.vec[which(ID.dist == 0 & Link.mat == "gp", arr.ind = T)[, 1]]
  names(match.wUF.gp) = paste(names(match.wUF.gp), "g", sep = "")
match.wUF.se = wUF.dist[ID.dist == 0 & Link.mat == "se"]
  names(match.wUF.se) = ID.vec[which(ID.dist == 0 & Link.mat == "se", arr.ind = T)[, 1]]
  names(match.wUF.se) = paste(names(match.wUF.se), "s", sep = "")
match.wUF.gp.se = append(match.wUF.gp, match.wUF.se)

match.wUF.gs = wUF.dist[ID.dist == 0 & Link.mat == "gs"]
  names(match.wUF.gs) = ID.vec[which(ID.dist == 0 & Link.mat == "gs", arr.ind = T)[, 1]]
```

```
    names(match.wUF.gs) = paste(names(match.wUF.gs), "g", sep = "")

#Unweighted UniFrac
match.uwUF.gp = uwUF.dist[ID.dist == 0 & Link.mat == "gp"]
    names(match.uwUF.gp) = ID.vec[which(ID.dist == 0 & Link.mat == "gp", arr.ind = T)[, 1]]
    names(match.uwUF.gp) = paste(names(match.uwUF.gp), "g", sep = "")
match.uwUF.se = uwUF.dist[ID.dist == 0 & Link.mat == "se"]
    names(match.uwUF.se) = ID.vec[which(ID.dist == 0 & Link.mat == "se", arr.ind = T)[, 1]]
    names(match.uwUF.se) = paste(names(match.uwUF.se), "s", sep = "")
match.uwUF.gp.se = append(match.uwUF.gp, match.uwUF.se)

match.uwUF.gs = uwUF.dist[ID.dist == 0 & Link.mat == "gs"]
    names(match.uwUF.gs) = ID.vec[which(ID.dist == 0 & Link.mat == "gs", arr.ind = T)[, 1]]
    names(match.uwUF.gs) = paste(names(match.uwUF.gs), "g", sep = "")
```

Extract distance for unrelated pairs

In order to make this group comparable to the sibling group, we will subset to similar ratios of same sex and opposite sex pairs.

Create matrix of sex comparisons

```
#Create a vector of sexes
Pheno.sm$gender = ifelse(Pheno.sm$gender == 1, "M", ifelse(Pheno.sm$gender == 2, "F", Pheno.sm$gender))
gender.vec = as.vector(Pheno.sm$gender)
#Using a loop, create a matrix of sex comparison (i.e. MF, FM, MM FF)
gender.mat = matrix(numeric(0), length(ID.vec), length(ID.vec))
for(j in 1:dim(gender.mat)[1] ){
  for(k in 1:dim(gender.mat)[2]){
    gender.mat[j, k] = paste(gender.vec[j], gender.vec[k], sep = "")
  }}

#Rename gender matrix with same and diff
gender.mat.rename = as.matrix(ifelse(gender.mat == "FF", "same",
                                     ifelse(gender.mat == "MM", "same", "diff")))
```

Pull out comparisons that are not from the same family (ID.dist =/= 0)

```
#Pick out different gender distances
match.BC.MF = BC.dist[ID.dist != 0 & gender.mat.rename == "diff"]
match.J.MF = J.dist[ID.dist != 0 & gender.mat.rename == "diff"]
match.wUF.MF = wUF.dist[ID.dist != 0 & gender.mat.rename == "diff"]
match.uwUF.MF = uwUF.dist[ID.dist != 0 & gender.mat.rename == "diff"]

#Pick out same gender distances
match.BC.MMFF = BC.dist[ID.dist != 0 & gender.mat.rename == "same"]
match.J.MMFF = J.dist[ID.dist != 0 & gender.mat.rename == "same"]
match.wUF.MMFF = wUF.dist[ID.dist != 0 & gender.mat.rename == "same"]
match.uwUF.MMFF = uwUF.dist[ID.dist != 0 & gender.mat.rename == "same"]
```

WLS data

Re-code variables

Gender

Re-code to categorical

- 1 = male = M
- 2 = female = F

```
Pheno.sm$gender = ifelse(Pheno.sm$gender == 1, "M", ifelse(Pheno.sm$gender == 2, "F", Pheno.sm$gender))
```

Cohabitation

Cohabiting YN: Re-code spouse co-cohabitation to yes or no

1 = currently married = Y 2 = separated = N *3 = divorced = N* 4 = widowed = N *5 = never married = N* 6 = Cohab = Y

```
Pheno.sm$cohab = ifelse(Pheno.sm$marcohab %in% c(1,6), "Y",
                           ifelse(Pheno.sm$marcohab %in% c(2,3,4,5), "N", Pheno.sm$marcohab))
```

Year cohabiting/married: Calculate years married from century month codes (CMC). For non-married cohabiting couples, use years cohabiting instead of years married.

- years = 2011-year of CMC
- year of CMC = 1900+int((CMC-1)/12)

```
Pheno$hc005re.yrs = 2011 - (1900+round((Pheno$hc005re-1)/12)) #marriage
Pheno$kc005re.yrs = 2011 - (1900+round((Pheno$kc005re-1)/12)) #marriage

Pheno$hc102re.yrs = 2011 - (1900+round((Pheno$hc102re-1)/12)) #cohab
Pheno$kc102re.yrs = 2011 - (1900+round((Pheno$kc102re-1)/12)) #cohab

#Merge yrs.cohab and yrs.marriage to get cohabitation years for all samples
##Two people do not have data for years married. These two people, and only these two people, have data
Pheno$cohab.marr.g = ifelse(is.na(Pheno$hc102re.yrs), Pheno$hc005re.yrs, Pheno$hc102re.yrs)
Pheno$cohab.marr.s = ifelse(is.na(Pheno$kc102re.yrs), Pheno$kc005re.yrs, Pheno$kc102re.yrs)
```

Children living with participants: re-code # of children living with participant

- 2 = No = 0
- NA = 0

```
Pheno$HD01701 = ifelse(is.na(Pheno$HD01701), 0,
                       ifelse(Pheno$HD01701 == 2, 0, Pheno$HD01701))
Pheno$HD01702 = ifelse(is.na(Pheno$HD01702), 0,
                       ifelse(Pheno$HD01702 == 2, 0, Pheno$HD01702))
Pheno$HD01703 = ifelse(is.na(Pheno$HD01703), 0,
                       ifelse(Pheno$HD01703 == 2, 0, Pheno$HD01703))
Pheno$HD01704 = ifelse(is.na(Pheno$HD01704), 0,
                       ifelse(Pheno$HD01704 == 2, 0, Pheno$HD01704))
Pheno$HD01705 = ifelse(is.na(Pheno$HD01705), 0,
                       ifelse(Pheno$HD01705 == 2, 0, Pheno$HD01705))
Pheno$HD01706 = ifelse(is.na(Pheno$HD01706), 0,
                       ifelse(Pheno$HD01706 == 2, 0, Pheno$HD01706))
```

```
Pheno$KD01701 = ifelse(is.na(Pheno$KD01701), 0,
                     ifelse(Pheno$KD01701 == 2, 0, Pheno$KD01701))
Pheno$KD01702 = ifelse(is.na(Pheno$KD01702), 0,
                     ifelse(Pheno$KD01702 == 2, 0, Pheno$KD01702))
Pheno$KD01703 = ifelse(is.na(Pheno$KD01703), 0,
                     ifelse(Pheno$KD01703 == 2, 0, Pheno$KD01703))
Pheno$KD01704 = ifelse(is.na(Pheno$KD01704), 0,
                     ifelse(Pheno$KD01704 == 2, 0, Pheno$KD01704))
Pheno$KD01705 = ifelse(is.na(Pheno$KD01705), 0,
                     ifelse(Pheno$KD01705 == 2, 0, Pheno$KD01705))
Pheno$KD01706 = ifelse(is.na(Pheno$KD01706), 0,
                     ifelse(Pheno$KD01706 == 2, 0, Pheno$KD01706))
```

Closeness

Re-code sibling closeness to be 4 = closer so is comparable to spousal closeness survey question

```
Pheno$sib.close.g = as.vector(reverse.code(-1, Pheno$HK062SSD, mini=1, maxi=4))
Pheno$sib.close.s = as.vector(reverse.code(-1, Pheno$KK062SSD, mini=1, maxi=4))
```

Pets

Replace all NAs in q20 (Y/N pet) and q21 (Y/N type of pet) with no = 2

```
Pheno.sm$petYN = replace(Pheno.sm$q20, is.na(Pheno.sm$q20), as.numeric(2))
Pheno.sm$dog = replace(Pheno.sm$q21_a, is.na(Pheno.sm$q21_a), as.numeric(2))
Pheno.sm$cat = replace(Pheno.sm$q21_b, is.na(Pheno.sm$q21_b), as.numeric(2))
Pheno.sm$bird = replace(Pheno.sm$q21_c, is.na(Pheno.sm$q21_c), as.numeric(2))
Pheno.sm$reptile = replace(Pheno.sm$q21_d, is.na(Pheno.sm$q21_d), as.numeric(2))
```

Combine bird, reptile and fish pets into "other" group

```
Pheno.sm$bird.rep = ifelse(Pheno.sm$bird == 1, 1,
                        ifelse(Pheno.sm$reptile == 1, 1, 2))

#Also alter the other category answer for individual you specified having a fish in the supplementary t
Pheno.sm$other = ifelse(is.na(Pheno.sm$q21_e), 2,
                          ifelse(Pheno.sm$q21_e == 1, Pheno.sm$OE_textq21_e, Pheno.sm$q21_e))
Pheno.sm$bird.rep.fish = ifelse(Pheno.sm$other == "5 fish in aquarium", 1, Pheno.sm$bird.rep)
Pheno.sm$other = Pheno.sm$bird.rep.fish
```

Alter cat answers for individuals who specified having a cat in the other category but did not denote having a cat in the cat category.

```
Pheno.sm$cat.fix = ifelse(Pheno.sm$other == "Live on farm and have daily contact with cats and cattle",
                     ifelse(Pheno.sm$other == "Horse - chicken - sheep - cas. 'In the barn'", 1,
                             Pheno.sm$cat))
```

Farm-urban

Re-code farm vs. urban variables to categorical

OCF357: Father's occupation

- 1 = Not farming
- 2 = Farming, on both April, 1957 Questionnaire and Tax Return
- 3 = Farming, on Tax Return but not ascertained or unskilled on April, 1957 Questionnaire
- 4 = Farming, on April, 1957 Questionnaire but not on Tax Return
- 5 = Farming, on Tax Return but skilled, white collar or professional on April, 1957 Questionnaire

```
Pheno$farm.father = ifelse(Pheno$OCF357 == 1, "urban",
                         ifelse(Pheno$OCF357 %in% c(2,3,4,5), "farm", Pheno$OCF357))
```

hf017j1e and kf017j1e: 1990 occupation code for graduates and siblings

- 1 = Professional, Technical & Specialty: Self-Employed & w/o Pay
- 2 = Professional, Technical & Specialty: Salaried & N.A.
- 3 = Executives, Administrators & Managers: Salaried & N.A.
- 4 = Executives, Administrators & Managers: Self-Employed & w/o Pay
- 5 = Sales: Not Retail Trade
- 6 = Sales: Retail Trade
- 7 = Administrator Support, Including Clerical
- 8 = Precision Production, Craftsmen, Repair: Manufacturing
- 9 = Precision Production, Craftsmen, Repair: Construction
- 10 = Precision Production, Craftsmen, Repair: All Other & N.A.
- 11 = Operators & Fabricators: Manufacturing
- 12 = Operators & Fabricators: All Other & N.A.
- 13 = Service Occupations
- 14 = Handlers, Equipment Cleaners, Helpers/Laborers: Manufacturing
- 15 = Handlers, Equipment Cleaners, Helpers/Laborers: All Other & N.A.
- 16 = Farm Operators & Managers
- 17 = Farm Workers & Related Occupations
- 18 = Military Occupations

```
Pheno$farm11.g = ifelse(Pheno$hf017j1e < 0, NA,
                     ifelse(Pheno$hf017j1e > 0 & Pheno$hf017j1e <= 15, "urban",
                     ifelse(Pheno$hf017j1e %in% c(16,17), "farm", Pheno$hf017j1e)))
Pheno$farm11.s = ifelse(Pheno$kf017j1e < 0, NA,
                     ifelse(Pheno$kf017j1e > 0 & Pheno$kf017j1e <= 15, "urban",
                     ifelse(Pheno$kf017j1e %in% c(16,17), "farm", Pheno$kf017j1e)))
```

AB044RE and BB044RE: WI high school in

- 1 = city
- 2 = town
- 3 = rural area

```
Pheno$RU.HS.p = ifelse(Pheno$AB044RE %in% c(1,2), "urban",
                         ifelse(Pheno$AB044RE == 3, "rural", Pheno$BB044RE))
Pheno$RU.HS.e = ifelse(Pheno$BB044RE %in% c(1,2), "urban",
                         ifelse(Pheno$BB044RE == 3, "rural", Pheno$BB044RE))
```

Diet

Fill in any protein NAs for vegetarians with 0 and change 8 (have not eaten in past year) to 0

```
Pheno.sm$meat = ifelse(Pheno.sm$q12 == 2 & is.na(Pheno.sm$q13), 0,
                ifelse(Pheno.sm$q13 == 8, 0, Pheno.sm$q13))
Pheno.sm$poul = ifelse(Pheno.sm$q12 == 2 & is.na(Pheno.sm$q14), 0,
                ifelse(Pheno.sm$q14 == 8, 0, Pheno.sm$q14))
```

```
Pheno.sm$pork = ifelse(Pheno.sm$q12 == 2 & is.na(Pheno.sm$q15), 0,
                ifelse(Pheno.sm$q15 == 8, 0, Pheno.sm$q15))
Pheno.sm$sea = ifelse(Pheno.sm$q12 == 2 & is.na(Pheno.sm$q16), 0,
                ifelse(Pheno.sm$q16 == 8, 0, Pheno.sm$q16))
```

Create summed variable for total animal protein per week

```
Pheno.sm$prot.sum = Pheno.sm$meat + Pheno.sm$poul + Pheno.sm$pork + Pheno.sm$se
```

Add together number of fruits or veggies checked as “yes, eaten regularly”.

```
Pheno.sm$fruit.sum = rowSums(subset(Pheno.sm, select=q18_1:q18_24), na.rm=TRUE)
Pheno.sm$veg.sum = rowSums(subset(Pheno.sm, select=q17_1:q17_76), na.rm=TRUE)
```

Create variables

Graduates

Subset the data

```
Pheno.sm.g = Pheno.sm[Pheno.sm$rtype == "g",]
#Pheno is already only g rows so subset to only those with microbiota data
Pheno.g = subset(Pheno, row.names(Pheno) %in% row.names(Met.g))
```

Create matrix of all variables of interest

```
vars.g = data.frame(
  gender = Pheno.sm.g$gender, #sex
  age = Pheno.sm.g$age, #age
  AB = Pheno.sm.g$q24, #antibiotics last 6mo, YN
  RU57 = Pheno.g$farm.father, #Rural v. urban 1957
  RU11 = Pheno.g$farm11.g, #Rural v. urban 2011
  yrs.edu = Pheno.g$hb103red, #Yrs education
  iq = Pheno.sm.g$iq, #IQ (Henmom-Nelson score)
  cohabYN = Pheno.sm.g$cohab,#Cohabitating with spouse, YN
  child.sum = Pheno.g$HD01701 +Pheno.g$HD01702 +Pheno.g$HD01703 +Pheno.g$HD01704 +Pheno.g$HD01705 +Phen
  groom = Pheno.g$ha103re, #Personal grooming score (0-10)
  social.sum = Pheno.g$jz023rer +Pheno.g$jz024rer, #social interactions last 4 wks
  dog = Pheno.sm.g$dog, #Dog
  cat = Pheno.sm.g$cat.fix, #Cat
  pet.other = Pheno.sm.g$other, #Other pet (bird, reptile, fish)
  clean.house = Pheno.g$ha114re, #Residence cleanliness score (1-10)
  bmi = Pheno.sm.g$bmi, #BMI
  srh11 = Pheno.sm.g$srh11, #Self-reported health score (1-5)
  walk.ave = (Pheno.g$HX472RE+Pheno.g$HX473RE)/2, #Average walking speed (sec)
  hbs11 = Pheno.sm.g$hbs11, #High-blood sugar, YN
  hbp11 = Pheno.sm.g$hbp11, #High-blood pressure, YN
  heart11 = Pheno.sm.g$heart11, #Heart disease, YN
  arth11 = Pheno.sm.g$arth11, #Arthritis, YN
  cancer11 = Pheno.sm.g$cancer11, #Cancer, YN
  stroke11 = Pheno.sm.g$stroke11, #Stroke, YN
  IBS = Pheno.g$jx148rer, #IBS
  smokeYN = Pheno.g$jx013rec, #Smoker, YN
  prot.sum = Pheno.sm.g$prot.sum, #Times consume protein / wk
  fruit.sum =  Pheno.sm.g$fruit.sum, #different fruits consume / week
  veg.sum = Pheno.sm.g$veg.sum, #different vegetables consume  / week
  meat = Pheno.sm.g$meat, #Times consume (red) meat / week
  poul =  Pheno.sm.g$poul, #Times consume poultry / week
  pork =  Pheno.sm.g$pork, #Times consume pork / week
  sea =  Pheno.sm.g$sea, #Times consume seafood / week
  row.names = row.names(Pheno.sm.g)
  )
```

Spouses

Create a list of all IDs for participants with a corresponding spouse in the data set

```
#Create a list of IDs repeated x2 for gp and se groups
ID.list.gp.se = as.list(rep(times=2, Pheno$idpriv))
#Create a list of g and s to append to ID numbers to separate gp and se groups
link.list.gp.se = as.list(c(rep(times=length(Pheno$idpriv), "g"), rep(times=length(Pheno$idpriv), "s")))
#Merge ID numbers and links so that gp comparisons are labeled with graduate ID + g and se by graduate
names.gp.se = paste(ID.list.gp.se, link.list.gp.se, sep="")
```

Create matrix of all variables of interest. First, from the large WLS data set (Pheno)

```
vars.gp.se = data.frame(
  RU.diff = paste(c(Pheno$farm.father, Pheno$farm.father), c(Pheno$RU.HS.p, Pheno$RU.HS.e), sep="."), #
  #OCF357 only has data for grads and sibs. Use ab/bb044re as alternative for spouses
  yrs.cohab.marr = c(Pheno$cohab.marr.g, Pheno$cohab.marr.s), #Years cohabitating with spouse
  closeness = c(Pheno$hc040re, Pheno$kc040re), #Relationship closeness score (1-4)
  spouse = c(Pheno$hc034sp, Pheno$kc034sp), #Do spouses live together (All do)
  sib = c(Pheno$HK074SS, Pheno$HK074SS), #Do g-s live together (None do)
  hbs05.p.e = c(Pheno$AX346RE, Pheno$BX346RE),
  heart05.p.e = c(Pheno$AX351RE, Pheno$BX351RE),
      #Health questions not answered by spouses in 2011 (Pheno.sm dataset). Need to use 2003-2005 answe
  row.names=names.gp.se)

#Subsample to include only graduates and siblings with microbiota comparisons
vars.gp.se = subset(vars.gp.se, row.names(vars.gp.se) %in% names(match.BC.gp.se))
```

Then from small data set taken at time of fecal sampling (Pheno.sm)

```
#Subset Pheno.sm by ID
Pheno.sm.gs = Pheno.sm[Pheno.sm$rtype %in% c("g", "s"),]
Pheno.sm.pe = Pheno.sm[Pheno.sm$rtype %in% c("p", "e"),]

#Edit rtype so that merging matches by idpriv+rtype
Pheno.sm.pe$rtype.gs = ifelse(Pheno.sm.pe$rtype == "p", "g", "s")
Pheno.sm.pe$idpriv.gs = paste(Pheno.sm.pe$idpriv, Pheno.sm.pe$rtype.gs, sep="")
Pheno.sm.gs$idpriv.gs = paste(Pheno.sm.gs$idpriv, Pheno.sm.gs$rtype, sep="")

#Merge g.s and p.e by grad ID
Pheno.sm.gs.pe = merge(Pheno.sm.gs, Pheno.sm.pe, by="idpriv.gs", all=TRUE)
row.names(Pheno.sm.gs.pe) = Pheno.sm.gs.pe$idpriv.gs

#Select variables of interest
Pheno.sm.gp.se.vars = data.frame(
  age.diff = abs(Pheno.sm.gs.pe$age.x-Pheno.sm.gs.pe$age.y), #Age difference
  AB.diff = Pheno.sm.gs.pe$q24.x+Pheno.sm.gs.pe$q24.y, #Antibiotics last 6 mo YN same v. different
  meat.diff = abs(Pheno.sm.gs.pe$meat.x-Pheno.sm.gs.pe$meat.y), #Times consume (red) meat / wk differen
  poul.diff = abs(Pheno.sm.gs.pe$poul.x-Pheno.sm.gs.pe$poul.y), #Times consume poultry / wk difference
  heart11.g.s = Pheno.sm.gs.pe$heart11.x, #Heart disease YN same v. different
  hbs11.g.s = Pheno.sm.gs.pe$hbs11.x, #High-blood sugar YN same v. different
  ID = Pheno.sm.gs.pe$idpriv.gs,
  link = Pheno.sm.gs.pe$rtype.gs,
  row.names=row.names(Pheno.sm.gs.pe))
```

Append Pheno and Pheno.sm derived variables into one data set

```
vars.gp.se.all = merge(vars.gp.se, Pheno.sm.gp.se.vars, by="row.names", all=TRUE)
row.names(vars.gp.se.all) = vars.gp.se.all$Row.names
```

Re-code variables

Rural vs. urban

- Make any answer with at least one NA = NA
- farm.rural = both rural = RR
- urban.urban = both urban = UU
- urban.rural = different between spouses = RU

```
#Re-code RU diff
vars.gp.se.all$RU.diff = ifelse(vars.gp.se.all$RU.diff %in% c("urban.NA", "farm.NA"), NA,
                                    ifelse(vars.gp.se.all$RU.diff == "farm.rural", "RR",
                                           ifelse(vars.gp.se.all$RU.diff == "urban.urban", "UU",
                                                  ifelse(vars.gp.se.all$RU.diff %in% c("urban.rural", "
                                                         vars.gp.se.all$RU.diff))))
vars.gp.se.all$RU.diff.cat = ifelse(vars.gp.se.all$RU.diff== "RR", "same",
                                    ifelse(vars.gp.se.all$RU.diff == "UU", "same",
                                           ifelse(vars.gp.se.all$RU.diff == "RU", "diff", vars.gp.se.al
```

Antibiotics

- 1 = Y and 2 = N so,
- 1+1 = 2 = both Y = same
- 2+2 = 4 = both N = same
- 1+2 = 3 = different = diff

```
vars.gp.se.all$AB.diff.cat = ifelse(vars.gp.se.all$AB.diff %in% c(2,4), "same",
                                    ifelse(vars.gp.se.all$AB.diff == 3, "diff", vars.gp.se.all$AB.diff))
```

Health: heart disease and high blood sugar were answered in different surveys for grads and sibs (Pheno.sm) vs. spouses (Pheno). Calculate hbs and heart difference variables (one spouse from Pheno, one from Pheno.sm)

```
vars.gp.se.all$hbs.diff = vars.gp.se.all$hbs11.g.s + vars.gp.se.all$hbs05.p.e
vars.gp.se.all$heart.diff = vars.gp.se.all$heart11.g.s + vars.gp.se.all$heart05.p.e

#Re-codesimilar to antibiotics
vars.gp.se.all$hbs.diff.cat = ifelse(vars.gp.se.all$hbs.diff %in% c(2,4), "same",
                                    ifelse(vars.gp.se.all$hbs.diff == 3, "diff", vars.gp.se.all$hbs.dif
vars.gp.se.all$heart.diff.cat = ifelse(vars.gp.se.all$heart.diff %in% c(2,4), "same",
                                    ifelse(vars.gp.se.all$heart.diff == 3, "diff", vars.gp.se.all$heart
```

Sub-sample FINAL DATA to those with microbiota samples and order

```
#SUBSAMPLE TO MICROBIOTA SAMPLES
vars.gp.se.all = subset(vars.gp.se.all, row.names(vars.gp.se.all) %in% names(match.BC.gp.se))
#ORDER DATA TO MATCH DISTANCE DATA
vars.gp.se.all = vars.gp.se.all[match(names(match.BC.gp.se), row.names(vars.gp.se.all)),]
```

Siblings

Create a list of IDs for gs groups

```
names.gs = paste(Pheno$idpriv, "g", sep="")
```

Create matrix of all variables of interest. First, from the large WLS data set (Pheno)

```
vars.gs = data.frame(
  closeness.g = Pheno$sib.close.g, #closeness
  closeness.s = Pheno$sib.close.s, #closeness sib answer
  social.sum.g = Pheno$jz023rer+Pheno$jz024rer, #social interactions last 4 wks grad
  social.sum.s = Pheno$pz023rer+Pheno$pz024rer, #social interactions last 4 wks sib
  social.sum.diff = abs((Pheno$jz023rer+Pheno$jz024rer) - (Pheno$pz023rer+Pheno$pz024rer)), #social int
  RU11.g = Pheno$farm11.g, #Rural v. urban 2011 grad
  RU11.s = Pheno$farm11.s, #Rural v. urban 2011 sib
  RU11.diff = paste(Pheno$farm11.g, Pheno$farm11.s, sep="."), #Rural v. urban 2011 same v. different
  row.names=names.gs)

#Subset vars.gs to only include graduates in gs pairs
vars.gs = subset(vars.gs, row.names(vars.gs) %in% names(match.BC.gs))
```

Then from small data set taken at time of fecal sampling (Pheno.sm)

```
#Subset Pheno.sm by ID
Pheno.sm.g = Pheno.sm[Pheno.sm$rtype == "g",]
Pheno.sm.s = Pheno.sm[Pheno.sm$rtype == "s",]

#Edit rtype so that merging matches by idpriv+rtype
Pheno.sm.s$rtype.g = ifelse(Pheno.sm.s$rtype == "s", "g", NA)
Pheno.sm.s$idpriv.g = paste(Pheno.sm.s$idpriv, Pheno.sm.s$rtype.g, sep="")
Pheno.sm.g$idpriv.g = paste(Pheno.sm.g$idpriv, Pheno.sm.g$rtype, sep="")

#Merge g.s and p.e by grad ID
Pheno.sm.g.s = merge(Pheno.sm.g, Pheno.sm.s, by="idpriv.g", all=TRUE)
row.names(Pheno.sm.g.s) = Pheno.sm.g.s$idpriv.g

#Select variables of interest
Pheno.sm.gs.vars = data.frame(
  gender.diff = paste(Pheno.sm.g.s$gender.x, Pheno.sm.g.s$gender.y, sep=""), #Sex same v. different
  age.diff = abs(Pheno.sm.g.s$age.x-Pheno.sm.g.s$age.y), #Age difference
  AB.diff =  Pheno.sm.g.s$q24.x+Pheno.sm.g.s$q24.y, #Antibiotics last 6 mo YN same v. different
  cohab.diff = paste(Pheno.sm.g.s$cohab.x, Pheno.sm.g.s$cohab.y, sep=""), #Cohabitating with spouse YN
  heart.diff = Pheno.sm.g.s$heart11.x + Pheno.sm.g.s$heart11.y,#Heart disease YN same v. different
  hbs.diff = Pheno.sm.g.s$hbs11.x + Pheno.sm.g.s$hbs11.y, #High-blood sugar YN same v. different
  meat.diff = abs(Pheno.sm.g.s$meat.x-Pheno.sm.g.s$meat.y), #Times consume (red) meat / wk difference
  poul.diff = abs(Pheno.sm.g.s$poul.x-Pheno.sm.g.s$poul.y), #Times consume poultry / wk difference
  ID = paste(Pheno.sm.g.s$idpriv.g, "s", sep=""),
  link = paste(Pheno.sm.g.s$rtype.g, "s", sep=""),
  row.names=row.names(Pheno.sm.g.s))
```

Append Pheno and Pheno.sm derived variables into one data set

```
vars.gs.all = merge(vars.gs, Pheno.sm.gs.vars, by="row.names", all=TRUE)
row.names(vars.gs.all) = vars.gs.all$Row.names
```

Re-code variables

Sex

- Any answers with 1 NA become NA overall
- male-female = female-males = diff
- male-male and female-female = same

```
#Recode gender diff
vars.gs.all$gender.diff = ifelse(vars.gs.all$gender.diff %in% c("MNA", "FNA", "NAM", "NAF"), NA,
                                  ifelse(vars.gs.all$gender.diff == "MM", "MM",
                                         ifelse(vars.gs.all$gender.diff == "FF", "FF",
                                                ifelse(vars.gs.all$gender.diff %in% c("FM", "MF"), "MF"
                                                       vars.gs.all$gender.diff))))
vars.gs.all$gender.diff.cat = ifelse(vars.gs.all$gender.diff %in% c("MM", "FF"), "same",
                                  ifelse(vars.gs.all$gender.diff == "MF", "diff", vars.gs.all$gender.di
```

Rural vs. urban

- Make any answer with at least one NA = NA
- farm.rural = both rural = RR
- urban.urban = both urban = UU
- urban.rural = different between spouses = RU

```
vars.gs.all$RU11.diff =  ifelse(vars.gs.all$RU11.diff %in% c("urban.NA", "NA.urban", "NA.NA", "farm.NA"
                                ifelse(vars.gs.all$RU11.diff == "urban.urban", "UU",
                                       ifelse(vars.gs.all$RU11.diff == "farm.farm", "RR",
                                              ifelse(vars.gs.all$RU11.diff %in% c("farm.urban", "urban.
                                                     vars.gs.all$RU11.diff))))
vars.gs.all$RU11.diff.cat = ifelse(vars.gs.all$RU11.diff %in% c("UU", "RR"), "same",
                                   ifelse(vars.gs.all$RU11.diff == "UR", "diff", vars.gs.all$RU11.diff)
```

Antibiotics

- 1 = Y and 2 = N so,
- 1+1 = 2 = both Y = same
- 2+2 = 4 = both N = same
- 1+2 = 3 = different = diff

```
vars.gs.all$AB.diff.cat = ifelse(vars.gs.all$AB.diff %in% c(2,4), "same",
                                    ifelse(vars.gs.all$AB.diff == 3, "diff", vars.gs.all$AB.diff))
```

Health: heart disease and high blood sugar were answered in different surveys for grads and sibs (Pheno.sm) vs. spouses (Pheno). Calculate hbs and heart difference variables (one spouse from Pheno, one from Pheno.sm)

```
vars.gs.all$hbs.diff.cat = ifelse(vars.gs.all$hbs.diff %in% c(2,4), "same",
                                    ifelse(vars.gs.all$hbs.diff == 3, "diff", vars.gs.all$hbs.diff))
vars.gs.all$heart.diff.cat = ifelse(vars.gs.all$heart.diff %in% c(2,4), "same",
                                    ifelse(vars.gs.all$heart.diff == 3, "diff", vars.gs.all$heart.diff)
```

Cohabitation

- any NAs = NA

- yes-no = no-yes = diff
- yes-yes and no-no = same

```
vars.gs.all$cohab.diff = ifelse(vars.gs.all$cohab.diff %in% c("NNA", "YNA", "NAY", "NAN", "NANA"), NA,
                                    ifelse(vars.gs.all$cohab.diff %in% c("NY", "YN"), "YN",
                                              ifelse(vars.gs.all$cohab.diff == "YY", "YY",
                                                     ifelse(vars.gs.all$cohab.diff == "NN", "NN", vars.g
vars.gs.all$cohab.diff.cat = ifelse(vars.gs.all$cohab.diff %in% c("YY", "NN"), "same",
                                    ifelse(vars.gs.all$cohab.diff == "YN", "diff", vars.gs.all$cohab.di
```

Closeness: Create average variables from grad and sibling responses. Replace NAs with either g or s value if have one

```
vars.gs.all$closeness.ave = ifelse(is.na(vars.gs.all$closeness.g), vars.gs.all$closeness.s,
                           ifelse(is.na(vars.gs.all$closeness.s), vars.gs.all$closeness.g,
                                 ((vars.gs.all$closeness.g + vars.gs.all$closeness.s)/2))) #Relationshi
```

Sub-sample FINAL DATA to those with microbiota samples and order

```
#SUBSAMPLE TO MICROBIOTA SAMPLES
vars.gs.all = subset(vars.gs.all, row.names(vars.gs.all) %in% names(match.BC.gs))
#ORDER DATA TO MATCH DISTANCE DATA
vars.gs.all = vars.gs.all[match(names(match.BC.gs), row.names(vars.gs.all)),]
```

Combine spouses and siblings

Combine gp.se and gs data sets for variables that occur in both

```
#Rename gs rows with gs
vars.gs.all.gs = vars.gs.all
row.names(vars.gs.all.gs) = paste(row.names(vars.gs.all), "s", sep="")

#Create yrs.edu.diff variable

#Combine variables into new dataframe
vars.gp.se.gs = data.frame(
  age.diff = c(vars.gp.se.all$age.diff, vars.gs.all.gs$age.diff),
  AB.diff = c(vars.gp.se.all$AB.diff, vars.gs.all.gs$AB.diff),
  AB.diff.cat = c(vars.gp.se.all$AB.diff.cat, vars.gs.all.gs$AB.diff.cat),
  closeness = c(vars.gp.se.all$closeness, vars.gs.all.gs$closeness.ave),
  closeness.g = c(vars.gp.se.all$closeness, vars.gs.all.gs$closeness.g),
  meat.diff = c(vars.gp.se.all$meat.diff, vars.gs.all.gs$meat.diff),
  poul.diff = c(vars.gp.se.all$poul.diff, vars.gs.all.gs$poul.diff),
  heart.diff = c(vars.gp.se.all$heart.diff, vars.gs.all.gs$heart.diff),
  hbs.diff = c(vars.gp.se.all$hbs.diff, vars.gs.all.gs$hbs.diff),
  heart.diff.cat = c(vars.gp.se.all$heart.diff.cat, vars.gs.all.gs$heart.diff.cat),
  hbs.diff.cat = c(vars.gp.se.all$hbs.diff.cat, vars.gs.all.gs$hbs.diff.cat),
  ID = c(as.character(vars.gp.se.all$ID), as.character(vars.gs.all.gs$ID)),
  link = c(as.character(vars.gp.se.all$link), as.character(vars.gs.all.gs$link)),
  row.names = c(row.names(vars.gp.se.all), row.names(vars.gs.all.gs)))

#Add sib vs spouse group variable
vars.gp.se.gs$group = ifelse(vars.gp.se.gs$link %in% c("g", "s"), "spouse",
                             ifelse(vars.gp.se.gs$link == "gs", "sib", vars.gp.se.gs$link))
```

Merge distance lists for spouses and siblings

```
match.BC.gp.se.gs = append(match.BC.gp.se, match.BC.gs)
  names(match.BC.gp.se.gs) = row.names(vars.gp.se.gs)
match.J.gp.se.gs = append(match.J.gp.se, match.J.gs)
  names(match.J.gp.se.gs) = row.names(vars.gp.se.gs)
match.wUF.gp.se.gs = append(match.wUF.gp.se, match.wUF.gs)
  names(match.wUF.gp.se.gs) = row.names(vars.gp.se.gs)
match.uwUF.gp.se.gs = append(match.uwUF.gp.se, match.uwUF.gs)
  names(match.uwUF.gp.se.gs) = row.names(vars.gp.se.gs)
```

Diet

Create data frame of just dietary components

```
diet = subset(Pheno.sm, select = c(idpriv, rtype, meat:sea, q17_1:q17_76, q18_1:q18_24), na.rm=TRUE)
#Replace blanks with 0 for no in fruit and veg data
diet[,7:length(diet)][is.na(diet[,7:length(diet)])] = 0
#Remove columns with all nos (sum = 0)
diet = diet[,colSums(diet[,7:length(diet)])> 0]
```

Create variable for differences in protein consumption between spousal and sibling pairs

```
#Compute differences between variables of matched pairs.
diet.gp.diff = aggregate(.~idpriv, data=diet[diet$rtype %in% c("g","p"),], FUN=diff, na.action=na.omit)
diet.se.diff = aggregate(.~idpriv, data=diet[diet$rtype %in% c("s","e"),], FUN=diff, na.action=na.omit)
diet.gs.diff = aggregate(.~idpriv, data=diet[diet$rtype %in% c("g","s"),], FUN=diff, na.action=na.omit)
#Rename rows by ID plus g or s modifier
row.names(diet.gp.diff) = paste(diet.gp.diff[,1], "g", sep = "")
row.names(diet.se.diff) = paste(diet.se.diff[,1], "s", sep = "")
row.names(diet.gs.diff) = paste(diet.gs.diff[,1], "g", sep = "")
#Merge gp and se
diet.gp.se.diff = rbind(diet.gp.diff, diet.se.diff)

#Subset diet diffs to only include pairs in microbiota data
diet.gp.se.diff = subset(diet.gp.se.diff, row.names(diet.gp.se.diff) %in% names(match.BC.gp.se))
diet.gs.diff = subset(diet.gs.diff, row.names(diet.gs.diff) %in% names(match.BC.gs))
#Remove integer(0) rows. These result from an NA in one of the pair to NA
diet.gp.se.diff = diet.gp.se.diff[diet.gp.se.diff$rtype == "1",]
diet.gs.diff = diet.gs.diff[diet.gs.diff$rtype == "1",]
#Alter values to absolute values
diet.gp.se.diff = data.frame(abs(data.matrix(diet.gp.se.diff)))
diet.gs.diff = data.frame(abs(data.matrix(diet.gs.diff)))
```

Create variable that adds group diffs together

```
diet.gp.se.diff$prot.diff = rowSums(subset(diet.gp.se.diff, select=meat:sea), na.rm=TRUE)
diet.gp.se.diff$fruit.diff = rowSums(subset(diet.gp.se.diff, select=q18_1:q18_24), na.rm=TRUE)
diet.gp.se.diff$veg.diff = rowSums(subset(diet.gp.se.diff, select=q17_1:q17_76), na.rm=TRUE)

diet.gs.diff$prot.diff = rowSums(subset(diet.gs.diff, select=meat:sea), na.rm=TRUE)
diet.gs.diff$fruit.diff = rowSums(subset(diet.gs.diff, select=q18_1:q18_24), na.rm=TRUE)
diet.gs.diff$veg.diff = rowSums(subset(diet.gs.diff, select=q17_1:q17_76), na.rm=TRUE)
```

Calculate distances based on diet

```r
##Protein
diet.for.dist.prot = subset(diet, select=meat:sea)
diet.dist.prot = as.matrix(dist(diet.for.dist.prot, method="euclidean"))

#Pull out distances for sib/spouse pairs
match.prot.gp = diet.dist.prot[ID.dist == 0 & Link.mat == "gp"]
  names(match.prot.gp) = ID.vec[which(ID.dist == 0 & Link.mat == "gp", arr.ind = T)[, 1]]
  names(match.prot.gp) = paste(names(match.prot.gp), "g", sep = "")
match.prot.se = diet.dist.prot[ID.dist == 0 & Link.mat == "se"]
  names(match.prot.se) = ID.vec[which(ID.dist == 0 & Link.mat == "se", arr.ind = T)[, 1]]
  names(match.prot.se) = paste(names(match.prot.se), "s", sep = "")
match.prot.gp.se = append(match.prot.gp, match.prot.se)
match.prot.gs = diet.dist.prot[ID.dist == 0 & Link.mat == "gs"]
  names(match.prot.gs) = ID.vec[which(ID.dist == 0 & Link.mat == "gs", arr.ind = T)[, 1]]
  names(match.prot.gs) = paste(names(match.prot.gs), "gs", sep = "")
match.prot.gp.se.gs =  append(match.prot.gp.se, match.prot.gs)

#Subset distances to match complete diet data
##Spouses
match.BC.gp.se.diet = match.BC.gp.se[names(match.BC.gp.se) %in% row.names(diet.gp.se.diff)]
match.J.gp.se.diet = match.J.gp.se[names(match.J.gp.se) %in% row.names(diet.gp.se.diff)]
match.wUF.gp.se.diet = match.wUF.gp.se[names(match.wUF.gp.se) %in% row.names(diet.gp.se.diff)]
match.uwUF.gp.se.diet = match.uwUF.gp.se[names(match.uwUF.gp.se) %in% row.names(diet.gp.se.diff)]
##Siblings
match.BC.gs.diet = match.BC.gs[names(match.BC.gs) %in% row.names(diet.gs.diff)]
match.J.gs.diet = match.J.gs[names(match.J.gs) %in% row.names(diet.gs.diff)]
match.wUF.gs.diet = match.wUF.gs[names(match.wUF.gs) %in% row.names(diet.gs.diff)]
match.uwUF.gs.diet = match.uwUF.gs[names(match.uwUF.gs) %in% row.names(diet.gs.diff)]
```

Kinship

Subset kinship dataset to just related pairs

```r
relatedness.gs = relatedness[relatedness$idpriv_id1 == relatedness$idpriv_id2,]
```

Rename row names to match other data

```r
row.names(relatedness.gs) = paste(relatedness.gs$idpriv_id1, "g", sep="")
```

Subset to sibling pairs

```r
relatedness.gs.match = relatedness.gs[row.names(relatedness.gs) %in% names(match.BC.gs),]

relatedness.gs.match = subset(relatedness.gs.match, row.names(relatedness.gs.match) %in% names(match.BC
```

Save environment

```r
#save.image("F:/Box Sync/Kim/WLS/R/Paper 1/WLS_environment_final.RData")
```