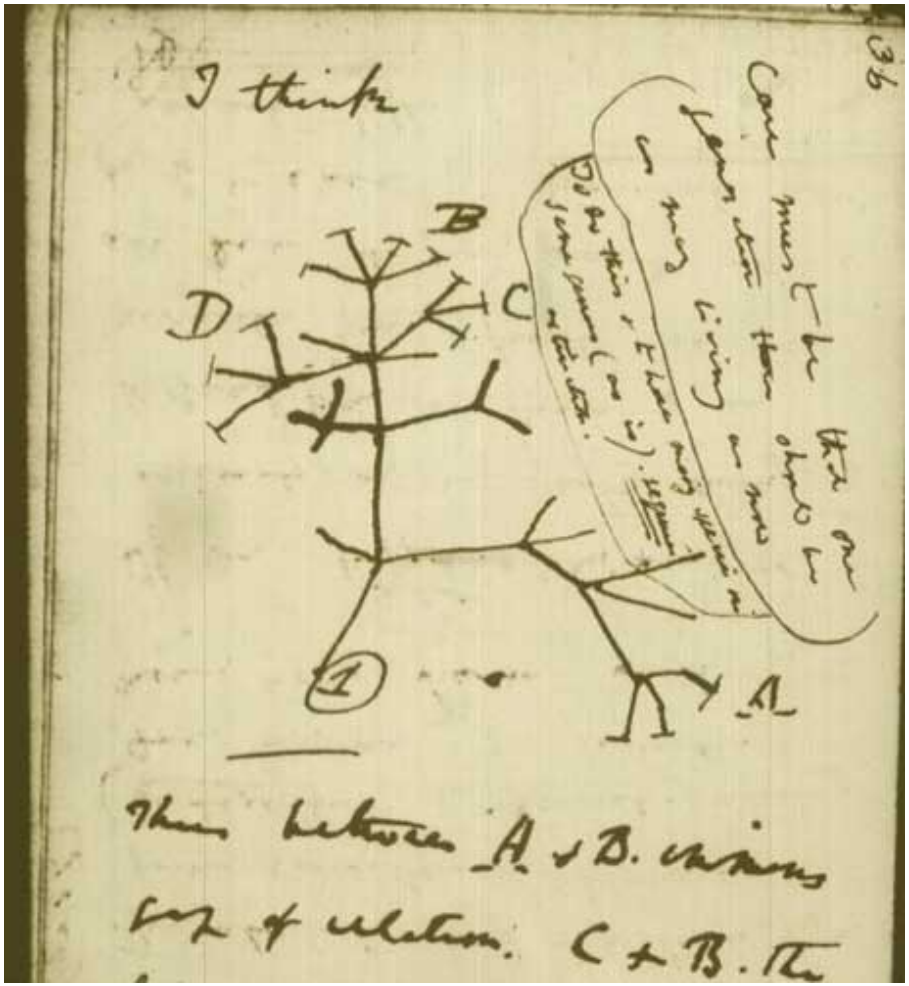
Robust testing and graphs

Susan Holmes



Robust, nonparametric methods

Kostic study of colorectal carcinoma

Genomic analysis identifies association of Fusobacterium with colorectal carcinoma. Kostic, A. D., Gevers, D., Pedamallu, C. S., Michaud, M., Duke, F., Earl, A. M., et al. (2012). Genome research, 22(2), 292-298.

```
library(phyloseq)
filepath = system.file("extdata", "study_1457_split_library_seqs_and_mapping.zip", package="phyloseq")
kostic = microbio_me_qiime(filepath)
```

```
## Found biom-format file, now parsing it...
## Done parsing biom...
## Importing Sample Metdadata from mapping file...
## Merging the imported objects...
## Successfully merged, phyloseq-class created.
##  Returning...
```

```
kostic
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 2505 taxa and 190 samples ]
## sample_data() Sample Data:       [ 190 samples by 71 sample variables ]
## tax_table()   Taxonomy Table:    [ 2505 taxa by 7 taxonomic ranks ]
```

```
head(sample_data(kostic)$DIAGNOSIS, 10)
```

```
##  [1] Healthy Tumor   Tumor   Healthy Healthy Healthy Tumor   Healthy
##  [9] Healthy Healthy
## Levels: Healthy None Tumor
```

A little preprocessing

First remove the 5 samples that had no DIAGNOSIS attribute assigned. These introduce a spurious third design class that is actually a rare artifact in the dataset. Also remove samples with less than 500 reads (counts). Note that this kind of data cleanup is useful, necessary, and should be well-documented because it can also be dangerous to alter or omit data without clear documentation. In this case I actually explored the data first, and am omitting some of the details (and explanatory plots) here for clarity.

```
kostic <- subset_samples(kostic, DIAGNOSIS != "None")
kostic <- prune_samples(sample_sums(kostic) > 500, kostic)
kostic
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 2505 taxa and 177 samples ]
## sample_data() Sample Data:       [ 177 samples by 71 sample variables ]
## tax_table()   Taxonomy Table:    [ 2505 taxa by 7 taxonomic ranks ]
```

```
sample_sums(kostic)
```

```
##  C0333.N.518126  C0333.T.518046 38U4VAHB.518100 XZ33PN7O.518030
##            5651            1286            6546            4572
## GQ6LSNI6.518106  C0270.N.518041 HZIMMAM6.518095 LRQ9WN6C.518048
##            8714            8018            9327            5833
## A8A34NCW.518023  C0332.N.518027    C10.S.517995 MQE9ONGV.518038
##            6971            1833            4536            2666
##  C0230.N.517992  C0256.N.518170 HZIMMNL5.518062 GQ6LSABJ.518010
##            1018            8641            6394            6250
##  C0282.N.518138 TV28INUO.518004  C0315.N.518021  C0235.N.517993
##            1672            4250            7439            1513
## 5TA9VN6K.518161 A8A34AP9.518086 UEL2GAPI.518173 O436FAO8.518097
##            8934            7214            8341            7389
##  C0378.N.518158 41E1KNBP.517985  C0378.T.518104 R8J9ZNOW.518040
##            7613            4798            9584            7524
```

```
## R8J9ZAGC.518059 5TA9VAT9.518088 OTGGZAZO.518108  C0269.N.518130
##           10973            6269            7846            7771
##  C0271.T.518078  C0318.N.518105  C0159.T.518087  C0318.T.518140
##            1872            5341            4311            7853
## O436FNP3.518007  C0209.N.517984  C0240.N.518020  C0306.N.518003
##            1041            5664            6017            6253
## MQETMNL4.518037  C0209.T.518128  C0112.T.518024  C0355.T.518157
##            4103            3075             617            3874
##  C0322.N.518165 6G2KBNS6.518045  C0240.T.518052  C0268.T.518114
##            2410            6963            3952            1388
##  C0285.T.518044 41E1KAMA.517991 QFHRSAG2.518005 I7ROLN9P.518079
##            4169            7016            5150            5630
## QFHRSNIH.518116  C0271.N.518022 KIXFRARY.518160  C0133.N.518051
##            6937            6316            5692            3594
##  C0268.N.518089 OTGGZN5Y.518063  C0134.N.518072  C0395.N.518054
##            2778            5324            5903            3874
## UZ65XAS5.518119  C0311.T.518131 5EKFOAO4.518164  C0294.T.518134
##            7817            8921            5864            2594
##  C0388.N.518136  C0186.N.518143 38U4VNBW.518090  C0334.T.518147
##            7084            3523            4320            4759
## JIDZEN4J.518073 JIDZEAPD.518149 LRQ9WAHA.518036 UZ65XN27.518028
##            3016            6192            3868            6081
##  C0159.N.518042  C0306.T.518122  C0394.N.518031  C0342.N.518068
##            5583            1451            3371            8459
## 82S3MAZ4.518145     C4.T.518118  C0398.N.518152  C0388.T.518015
##            6000            5191            4906            3504
##  C0285.N.518084  C0308.N.518154  C0355.N.518077  C0308.T.518162
##            8164            4774            2306            1965
##  C0211.T.518096  C0206.N.518065  C0149.N.518039     C4.S.518169
##            4140            6837            6303            5118
##  C0374.N.518099  C0362.N.518069  C0211.N.518117 G3UBQNDP.518121
##            8580            1911            6653            5902
##  C0341.N.518101     C1.T.518120  C0399.T.518153  C0294.N.517987
##            4005            1361            2115            2454
##  C0186.T.518019 G2OD5N55.518146 YOTV6ASH.518142  C0258.T.517997
##            4415            5018             936           11187
##  C0230.T.518144  C0315.T.518034  C0349.N.517989  C0212.N.518167
##            2032            1909            6014            5222
## YOTV6NC8.518156  C0112.N.518001  C0335.N.518107  C0344.N.518103
##            5711            2374            8019            7234
## 82S3MNBY.518050 5EKFONB3.518125  C0225.N.518047     C6.S.518082
##            6896            7131            3957            2034
##  C0241.N.518043     C1.S.518137 XBS5MNEH.518074  C0195.N.518110
##            6053            2556            3135            7847
```

```
##  C0322.T.518018  C0342.T.518093  C0198.T.518166  C0395.T.518075
##             868             865            2147            2581
## G3UBQAJU.518060  C0335.T.518115  C0349.T.518111  C0149.T.518132
##            6430            3228            5434            4504
##  C0311.N.518124  C0154.N.518002  C0258.N.518014  C0371.N.518009
##            4507            6152            5458            7256
##  C0330.N.518148  C0344.T.518029  C0206.T.518053  C0371.T.518139
##            1207            2330            2717            6757
##  C0394.T.518064  C0332.T.518017 G2OD5ABL.518057     C6.T.518061
##            1440            2322            6628            2077
##  C0256.T.518080  C0095.T.518141  C0252.T.518006  C0366.T.518172
##            1953            1992            2354            7770
##  C0241.T.518067  C0198.N.518081  C0330.T.518035  C0325.N.518109
##            2738            3532            1925            3966
##  C0225.T.518033  C0154.T.518091  C0366.N.518094  C0270.T.518163
##            5478            1477            5503            3560
## MQETMAZC.518171  C0095.N.518123  C0341.T.518113  C0252.N.518016
##            2074            2642            4233            5628
##  C0325.T.518083  C0212.T.518155  C0399.N.518011  C0334.N.518066
##            4722             524            9734            2285
## XZ33PA3O.518135 KIXFRNL2.518129  C0374.T.518127  C0275.N.518076
##            3206            6973             926            5875
## 59S9WAIH.518013  C0314.T.518025 6G2KBASQ.517999  C0133.T.518133
##            3420            3099            3098            2694
## UEL2GN92.518058 59S9WNC6.518055  C0235.T.518049  C0362.T.518026
##            4917            4724            4566            4063
## TV28IANZ.518070    C10.T.518056  C0275.T.518032 MQE9OAS7.518008
##            2606             812            6317            4263
## 32I9UAPQ.518112 UTWNWANU.518168 UTWNWN3P.518012 BFJMKAKB.518159
##            7359            6669            7122            2283
## 32I9UNA9.518098
##            3207
```

```
summary(sample_sums(kostic))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     524    2606    4722    4726    6430   11187
```

A tidy trick (using the pipe operator %>%)

```
library(dplyr)
sample_sums(kostic) %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     524    2606    4722    4726    6430   11187
```

Question 1a: How many nonzero elements in the matrix ?

```
sum(otu_table(kostic)>0)
```

```
## [1] 34776
```

Question 1b: How many cells have more than 3 reads in the matrix ?

```
sum(otu_table(kostic)>3)
```

```
## [1] 14773
```

Question 2a: How to make a presence / absence matrix such that presence is defined as at least 3 reads?

```
PA = data.frame(otu_table(kostic))
PA[PA<3] = 0
PA[PA>2] = 1
apply(PA,2,sum)
```

```
##   CO333.N.518126   CO333.T.518046 X38U4VAHB.518100  XZ33PN7O.518030
##              100               70              119               97
##  GQ6LSNI6.518106   CO270.N.518041  HZIMMAM6.518095  LRQ9WN6C.518048
```

```
##              143              139              115              102
##  A8A34NCW.518023   C0332.N.518027     C10.S.517995  MQE9ONGV.518038
##              108              100              111               68
##   C0230.N.517992   C0256.N.518170  HZIMMNL5.518062  GQ6LSABJ.518010
##               83              124              103              130
##   C0282.N.518138  TV28INUO.518004   C0315.N.518021   C0235.N.517993
##               92               77              113               70
## X5TA9VN6K.518161  A8A34AP9.518086  UEL2GAPI.518173  O436FAO8.518097
##              128              105              124               25
##   C0378.N.518158 X41E1KNBP.517985   C0378.T.518104  R8J9ZNOW.518040
##              210               75              106               98
##  R8J9ZAGC.518059 X5TA9VAT9.518088  OTGGZAZO.518108   C0269.N.518130
##               99               90              107              194
##   C0271.T.518078   C0318.N.518105   C0159.T.518087   C0318.T.518140
##              124              148               88               68
##  O436FNP3.518007   C0209.N.517984   C0240.N.518020   C0306.N.518003
##               31               89              138              147
##  MQETMNL4.518037   C0209.T.518128   C0112.T.518024   C0355.T.518157
##               85               74               41               89
##   C0322.N.518165 X6G2KBNS6.518045   C0240.T.518052   C0268.T.518114
##              111              160              134               68
##   C0285.T.518044 X41E1KAMA.517991  QFHRSAG2.518005  I7ROLN9P.518079
##              106               64              129               59
##  QFHRSNIH.518116   C0271.N.518022  KIXFRARY.518160   C0133.N.518051
##              141              194              201              115
##   C0268.N.518089  OTGGZN5Y.518063   C0134.N.518072   C0395.N.518054
##              134              148              196              104
##  UZ65XAS5.518119   C0311.T.518131 X5EKFOAO4.518164   C0294.T.518134
##              123               84              111              116
##   C0388.N.518136   C0186.N.518143 X38U4VNBW.518090   C0334.T.518147
##              141              112               96              108
##  JIDZEN4J.518073  JIDZEAPD.518149  LRQ9WAHA.518036  UZ65XN27.518028
##              130              121              177              189
##   C0159.N.518042   C0306.T.518122   C0394.N.518031   C0342.N.518068
##              117               63              114              197
## X82S3MAZ4.518145      C4.T.518118   C0398.N.518152   C0388.T.518015
##               90               97              135              111
##   C0285.N.518084   C0308.N.518154   C0355.N.518077   C0308.T.518162
##              120              163              103              100
##   C0211.T.518096   C0206.N.518065   C0149.N.518039      C4.S.518169
##              129               81              114              150
##   C0374.N.518099   C0362.N.518069   C0211.N.518117  G3UBQNDP.518121
##              158               96              152               96
##   C0341.N.518101      C1.T.518120   C0399.T.518153   C0294.N.517987
```

```
##             148              51             105             147
##   C0186.T.518019  G2OD5N55.518146  YOTV6ASH.518142   C0258.T.517997
##             105              79              39             166
##   C0230.T.518144   C0315.T.518034   C0349.N.517989   C0212.N.518167
##              92              52             126             137
##  YOTV6NC8.518156   C0112.N.518001   C0335.N.518107   C0344.N.518103
##              90              97             143             123
## X82S3MNBY.518050 X5EKFONB3.518125   C0225.N.518047      C6.S.518082
##             108             117              79              69
##   C0241.N.518043      C1.S.518137  XBS5MNEH.518074   C0195.N.518110
##             112              75              16             146
##   C0322.T.518018   C0342.T.518093   C0198.T.518166   C0395.T.518075
##              57              58              64              73
##  G3UBQAJU.518060   C0335.T.518115   C0349.T.518111   C0149.T.518132
##              60             102              94             122
##   C0311.N.518124   C0154.N.518002   C0258.N.518014   C0371.N.518009
##              77             120             134             117
##   C0330.N.518148   C0344.T.518029   C0206.T.518053   C0371.T.518139
##              62              74              34             117
##   C0394.T.518064   C0332.T.518017  G2OD5ABL.518057      C6.T.518061
##              42              52             102              64
##   C0256.T.518080   C0095.T.518141   C0252.T.518006   C0366.T.518172
##              60              80              58             162
##   C0241.T.518067   C0198.N.518081   C0330.T.518035   C0325.N.518109
##              79              96              86             106
##   C0225.T.518033   C0154.T.518091   C0366.N.518094   C0270.T.518163
##              65              51             144              71
##  MQETMAZC.518171   C0095.N.518123   C0341.T.518113   C0252.N.518016
##              51              77              41             193
##   C0325.T.518083   C0212.T.518155   C0399.N.518011   C0334.N.518066
##              84              47             180              97
##  XZ33PA3O.518135  KIXFRNL2.518129   C0374.T.518127   C0275.N.518076
##              62              78              53             103
## X59S9WAIH.518013   C0314.T.518025 X6G2KBASQ.517999   C0133.T.518133
##              63              28             117              64
##  UEL2GN92.518058 X59S9WNC6.518055   C0235.T.518049   C0362.T.518026
##              95              70              37              34
##  TV28IANZ.518070     C10.T.518056   C0275.T.518032  MQE9OAS7.518008
##              62              41              55              43
## X32I9UAPQ.518112  UTWNWANU.518168  UTWNWN3P.518012  BFJMKAKB.518159
##              41              47              35              25
## X32I9UNA9.518098
##              40
```

```
apply(PA,2,sum) %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   68.00   99.00   99.28  123.00  210.00
```

If we actually want to have a phyloseq object and have a new otu_table with only presence absence, we do the following.

```
# joey version
kMat <- kostic %>% otu_table %>% as("matrix")
PA <- ifelse(kMat > 2, 1, 0) %>% otu_table(taxa_are_rows = TRUE)
```

Use rankings instead of values.

What is the highest rank in the following vector?

```
tt = sample(100,19)
tt
```

```
##  [1] 29 19 86 11 40 55 77 17  7 87 25 32 53 20 18 97 76 72 64
```

```
rank(tt)
```

```
##  [1]  8  5 17  2 10 12 16  3  1 18  7  9 11  6  4 19 15 14 13
```

```
abund <- otu_table(kostic)
abund[1:5,1:8]
```

```
## OTU Table:          [5 taxa and 8 samples]
##                      taxa are rows
##        C0333.N.518126 C0333.T.518046 38U4VAHB.518100 XZ33PN7O.518030
## 304309             40              4               1               2
## 469478              0              0               0               0
```

```
## 208196              0              0              0              0
## 358030              0              0              0              0
## 16076             271             28            110              7
##        GQ6LSNI6.518106 C0270.N.518041 HZIMMAM6.518095 LRQ9WN6C.518048
## 304309              30              1              4              8
## 469478               0              0              0              0
## 208196               0              0              0              0
## 358030               0              0              0              0
## 16076               34              0              0              0
```

```
abund_ranks <- apply(abund, 2, rank)
abund_ranks[1:5,1:8]
```

```
##        C0333.N.518126 C0333.T.518046 38U4VAHB.518100 XZ33PN7O.518030
## 304309         2477.0         2452.0            2326          2398.5
## 469478         1146.5         1183.5            1145          1164.5
## 208196         1146.5         1183.5            1145          1164.5
## 358030         1146.5         1183.5            1145          1164.5
## 16076          2499.0         2494.0            2482          2444.5
##        GQ6LSNI6.518106 C0270.N.518041 HZIMMAM6.518095 LRQ9WN6C.518048
## 304309          2459.5          2272.5          2408.5            2449
## 469478          1093.5          1110.0          1134.5            1155
## 208196          1093.5          1110.0          1134.5            1155
## 358030          1093.5          1110.0          1134.5            1155
## 16076           2467.0          1110.0          1134.5            1155
```

```
abund_ranks <-abund_ranks - 2000
abund_ranks[abund_ranks < 1] <- 1
abund_ranks[1:5,1:8]
```

```
##        C0333.N.518126 C0333.T.518046 38U4VAHB.518100 XZ33PN7O.518030
## 304309            477            452             326           398.5
## 469478              1              1               1             1.0
## 208196              1              1               1             1.0
## 358030              1              1               1             1.0
## 16076             499            494             482           444.5
##        GQ6LSNI6.518106 C0270.N.518041 HZIMMAM6.518095 LRQ9WN6C.518048
## 304309           459.5           272.5           408.5             449
## 469478             1.0             1.0             1.0               1
## 208196             1.0             1.0             1.0               1
```

```
## 358030            1.0           1.0           1.0             1
## 16076           467.0           1.0           1.0             1
```

Plug the rankings to replace the original abundances in the phyloseq object:

```
kostica <-kostic
otu_table(kostica) <- otu_table(abund_ranks,taxa_are_rows=TRUE)
```

Perform a simple Wilcox test on some of the taxa-rows

```
tab2 <- abund_ranks[c(2008,2315,1886, 733, 816, 1481),]
diagnosis=as.factor(sample_data(kostic)$DIAGNOSIS)
```

```
x <- tab2[1,]
wilcox.test(x ~ diagnosis,
            alternative ="two.sided")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by diagnosis
## W = 2625.5, p-value = 0.0001506
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(tab2[4,] ~ diagnosis,
            alternative ="two.sided")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  tab2[4, ] by diagnosis
## W = 3956, p-value = 0.3366
## alternative hypothesis: true location shift is not equal to 0
```

Multiple testing correction can be done using the `multtest` Bioconductor package:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("multtest")
```

Hierarchical multiple testing

Hypothesis testing can identify individual bacteria whose abundance relates to sample variables of interest.

A standard approach is very similar to the approach we already visited in lecture 7 .

To integrate this information, we proposed a hierarchical testing procedure; where taxonomic groups are only tested if higher levels are found to be be associated.

In the case where many related species have a slight signal, this pooling of information can increase power.

We apply this method to test the association between microbial abundance and age.

Start by using the normalization protocols we discussed in lecture 7
following DESeq2 for RNA-seq data and the Waste Not, Want Not paper for 16S rRNA generated count data and available in the DESeq2 package:

```
library("DESeq2")
library("phyloseq")
ps1 = readRDS("/Users/susan/Books/CUBook/data/ps1.rds")
ps_dds = phyloseq_to_deseq2(ps1, ~ ageBin + family_relationship)
gm_mean = function(x, na.rm = TRUE){
  exp(sum(log(x[x > 0]), na.rm = na.rm) / length(x))
}
geoMeans = apply(counts(ps_dds), 1, gm_mean)
```

```
ps_dds = estimateSizeFactors(ps_dds, geoMeans = geoMeans)
ps_dds = estimateDispersions(ps_dds)
abund = getVarianceStabilizedData(ps_dds)
```

We use the structSSI package to perform the hierarchical testing (Sankaran and Holmes 2014).

Hierarchical testing

Procedure needs univariate tests for each higher-level taxonomic group, not just every taxa.

```
library("structSSI")
el = phy_tree(ps1)$edge
el0 = el
el0 = el0[rev(seq_len(nrow(el))), ]
el_names = c(rownames(abund), seq_len(phy_tree(ps1)$Nnode))
el[, 1] = el_names[el0[, 1]]
el[, 2] = el_names[as.numeric(el0[, 2])]
unadj_p = treePValues(el, abund, sample_data(ps1)$ageBin)
```

We can now do our FDR calculations using the hierarchical testing procedure. The test results are guaranteed to control several variants of FDR, but at different levels; we defer details to (Benjamini and Yekutieli 2003; Benjamini and Bogomolov 2014; Sankaran and Holmes 2014).

Interactive plotting command that will open a browser window:

```
hfdr_res = hFDR.adjust(unadj_p, el, 0.75)
summary(hfdr_res)
#plot(hfdr_res, height = 5000) # opens in a browser
```

```
tax = tax_table(ps1)[, c("Family", "Genus")] %>% data.frame()
tax$seq = rownames(abund)
hfdr_res@p.vals$seq = rownames(hfdr_res@p.vals)
tax %>%  left_join(hfdr_res@p.vals[,-3]) %>%
  arrange(adjp) %>% head(10)
```

```
##                 Family            Genus       seq       unadjp
## 1      Lachnospiraceae             <NA>  GCAAG.96 1.673295e-82
## 2  Erysipelotrichaceae             <NA>  GCGAG.46 1.134371e-79
## 3      Lachnospiraceae        Roseburia  GCAAG.71 3.078334e-75
## 4      Lachnospiraceae Clostridium_XlVa GCAAG.190 8.173451e-59
## 5      Lachnospiraceae             <NA> GCAAG.254 3.227107e-50
## 6      Lachnospiraceae Clostridium_XlVa GCAAG.150 1.056944e-49
## 7      Lachnospiraceae             <NA>  GCAAG.30 9.057568e-49
## 8  Erysipelotrichaceae     Turicibacter   GCAAG.4 2.896917e-48
## 9      Ruminococcaceae     Ruminococcus  GCGAG.78 1.978950e-46
## 10     Lachnospiraceae Clostridium_XlVa GCAAG.170 6.107952e-43
##             adjp
## 1  3.346591e-82
## 2  2.268741e-79
## 3  6.156668e-75
## 4  1.634690e-58
## 5  6.454215e-50
## 6  2.113888e-49
## 7  1.811514e-48
## 8  5.793834e-48
## 9  3.957900e-46
## 10 1.221590e-42
```

It seems that the most strongly associated bacteria all belong to family *Lachnospiraceae*.

Graph based testing: the Friedman–Rafsky test

Graph-based two-sample tests were introduced by Friedman and Rafsky (Friedman and Rafsky 1979) as a generalization of the Wald-Wolfowitz runs test.

Graph vertices associated with covariates.

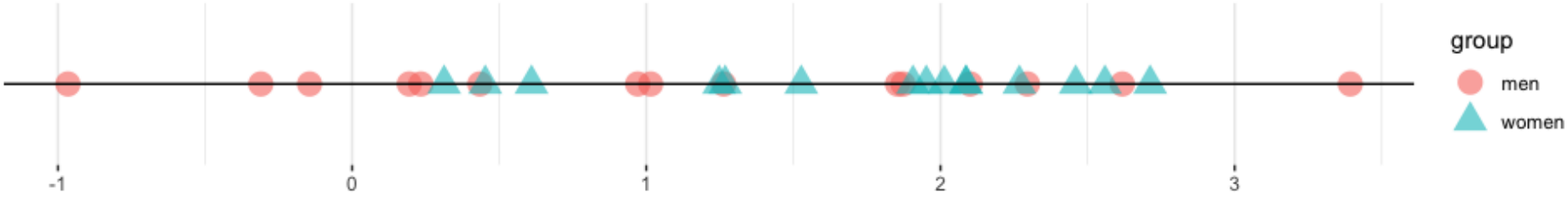
Test whether the covariate is significantly **associated** to the graph structure.

The **Friedman-Rafsky** tests for two/multiple sample segregation on a minimum spanning tree.

It was conceived as a generalization of the univariate Wald-Wolfowitz runs test. If we are comparing two samples, say men and women, whose coordinates represent a measurement of interest.

We color the two groups blue and red as below.

The Wald-Wolfowitz test looks for long runs of the samecolor that would indicate that the two groups have different means.



Instead of looking for consecutive values of one type ('runs'), we count the number of connected nodes of the same type.

Once the minimum spanning tree has been constructed, the vertices are assigned 'colors' according to the different levels of a categorical variable. We call **pure** edges those whose two nodes have the same level of the factor variable. We use S_O, the number of **pure** edges as our test statistic. To evaluate whether our observed value could have occurred by chance when the groups have the same distributions, we

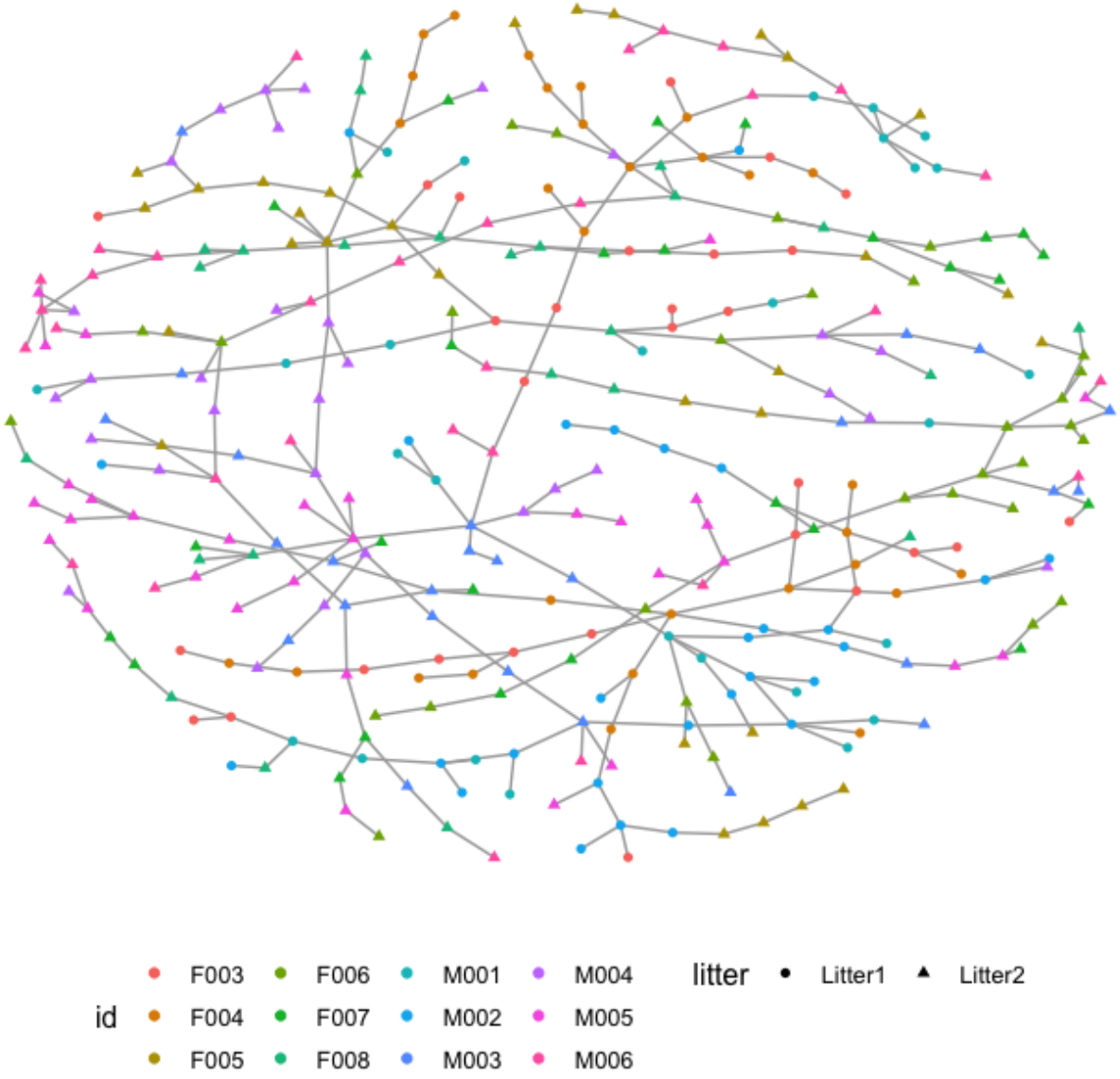
permute the vertix labels (colors) randomly and recount how many pure edges there are. This label swapping is repeated many times, creating our null distribution for S.

Example: Bacteria sharing between mice

```
library(phyloseq)
library(igraph)
ps1  = readRDS("/Users/susan/Books/CUBook/data/ps1.rds")
sampledata = data.frame(sample_data(ps1))
d1 = as.matrix(phyloseq::distance(ps1, method="jaccard"))
gr = graph.adjacency(d1, mode = "undirected", weighted = TRUE)
net = igraph::mst(gr)
V(net)$id = sampledata[names(V(net)), "host_subject_id"]
V(net)$litter = sampledata[names(V(net)), "family_relationship"]
```

We make a ggnetwork object from the resulting igraph generated minimum spanning tree and then plot it.

```
gnet=ggnetwork(net)
ggplot(gnet, aes(x = x, y = y, xend = xend, yend = yend))+
  geom_edges(color = "darkgray") +
  geom_nodes(aes(color = id, shape = litter)) + theme_blank()+
  theme(legend.position="bottom")
```


id
F003
F004
F005
F006
F007
F008
M001
M002
M003
M004
M005
M006
litter
Litter1
Litter2

MST plot

Distance based graphs (especially the Jaccard distance)

Distances

- Euclidean Distances SQRT(Sums of squares of coordinates).
- Mahalanobis distance (unequal weight per direction).
- Weighted euclidean distances, χ^2,...
- Manhattan/Hamming/City Block.
- Measurements of co-occurrence, ecological/ sociological data for instance **Jaccard**. When what really counts is how often certain species are found together then if the observations are just sequences of 0's and 1's, presence of 1's in the same spots does not present the same importance as that of the 0's: Jaccard distance=1-JC.
- Distances between `purified ' observations (we transform the data first).

There are almost a hundred different distances available combining outside information (distance on a graph, geodesic distance along a path, distance on a tree), combining categorical data and continuous data (Gower's distance) and using many different weighting schemes.

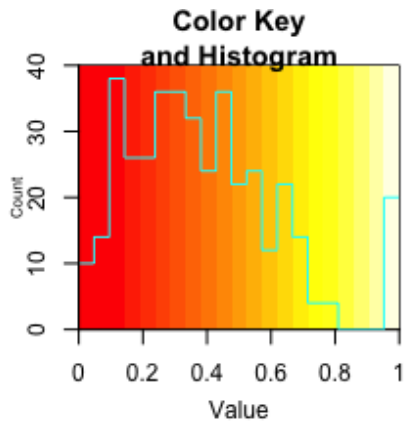
If you know what is the relevant notion of `closeness' or similarity for your data, you have (almost) solved the problem.

Distance functions available

phyloseq distances

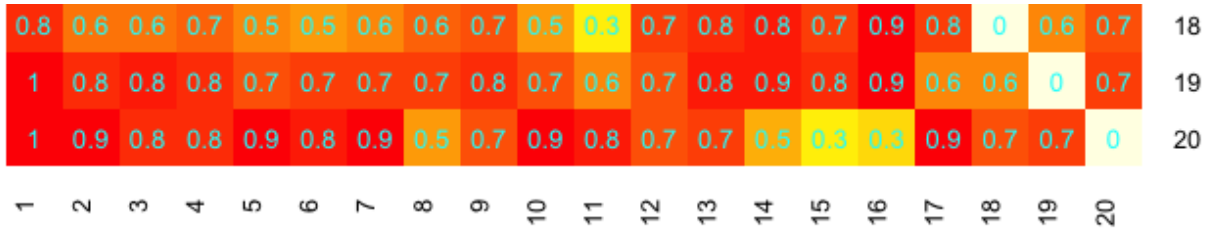
```
require(vegan)
data(dune)
dist.dune=vegdist(dune)
symnum(as.matrix(dist.dune))
```

```
##    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## 1
## 2  .
## 3  . .
## 4  . .
## 5  , . . .
## 6  , . . ,
## 7  . . . .
## 8  , . . . , . .
## 9  . . . . . . . .
## 10 .   . . . .   . .
## 11 . . . . , . . . . .
## 12 * , . . , , , . . ,  ,
## 13 + . . . , , , . . ,  ,  .
## 14 1 , , , + + + . , ,  +  ,  ,
## 15 1 * , , + + + . , +  ,  ,  ,  .
## 16 * + , , + + + . , +  +  .  ,  .  .
## 17 + + + + , , , + + ,  ,  *  +  +  +  1
## 18 , . , , . . . , , .  .  ,  ,  +  ,  +  ,
## 19 1 + + , , , , , , ,  .  ,  +  +  ,  *  .  .
## 20 1 * , , + + + . , +  +  ,  ,  .     .  *  ,  ,
## attr(,"legend")
## [1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

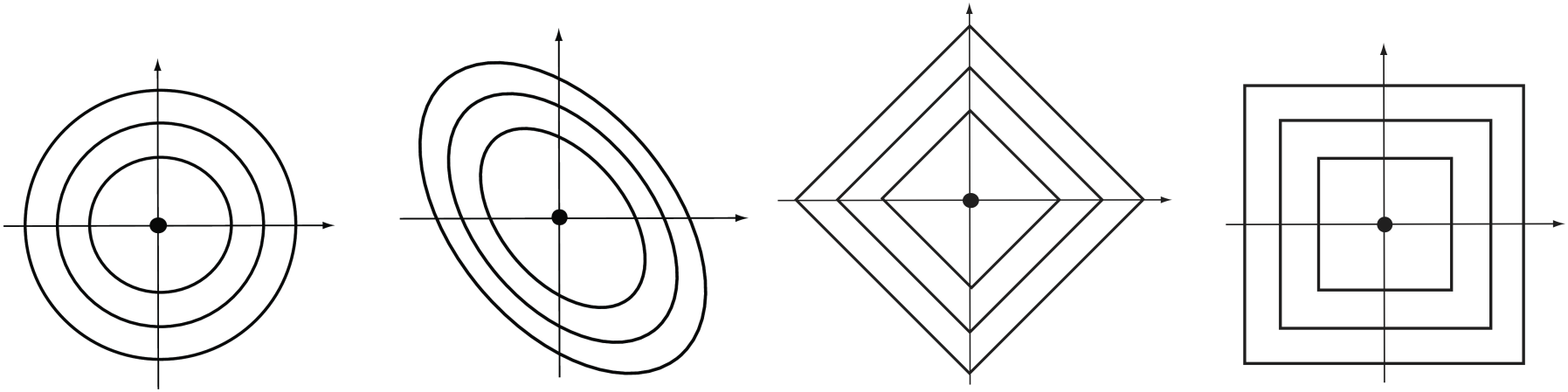


0	0.5	0.4	0.5	0.6	0.6	0.6	0.7	0.6	0.6	0.6	0.9	0.8	1	1	0.9	0.9	0.8	1	1
0.5	0	0.3	0.4	0.4	0.5	0.4	0.5	0.5	0.3	0.5	0.7	0.6	0.8	0.9	0.9	0.8	0.6	0.8	0.9
0.4	0.3	0	0.3	0.5	0.6	0.5	0.3	0.3	0.5	0.6	0.4	0.4	0.8	0.7	0.7	0.9	0.6	0.8	0.8
0.5	0.4	0.3	0	0.5	0.6	0.5	0.4	0.4	0.5	0.6	0.5	0.5	0.8	0.7	0.7	0.9	0.7	0.8	0.8
0.6	0.4	0.5	0.5	0	0.3	0.2	0.6	0.5	0.3	0.6	0.7	0.7	0.9	0.8	0.9	0.6	0.5	0.7	0.9
0.6	0.5	0.6	0.6	0.3	0	0.2	0.6	0.6	0.3	0.4	0.6	0.8	0.8	0.8	0.9	0.7	0.5	0.7	0.8
0.6	0.4	0.5	0.5	0.2	0.2	0	0.5	0.5	0.3	0.4	0.6	0.6	0.9	0.8	0.9	0.7	0.6	0.7	0.9
0.7	0.5	0.3	0.4	0.6	0.6	0.5	0	0.3	0.5	0.5	0.4	0.4	0.6	0.4	0.4	0.9	0.6	0.7	0.5
0.6	0.5	0.3	0.4	0.5	0.6	0.5	0.3	0	0.6	0.6	0.4	0.4	0.8	0.7	0.7	0.9	0.7	0.8	0.7
0.6	0.3	0.5	0.5	0.3	0.3	0.3	0.5	0.6	0	0.4	0.7	0.7	0.8	0.8	0.9	0.6	0.5	0.7	0.9
0.6	0.5	0.6	0.6	0.6	0.4	0.4	0.5	0.6	0.4	0	0.7	0.8	0.8	0.7	0.9	0.7	0.3	0.6	0.8
0.9	0.7	0.4	0.5	0.7	0.6	0.6	0.4	0.4	0.7	0.7	0	0.4	0.7	0.6	0.6	0.9	0.7	0.7	0.7
0.8	0.6	0.4	0.5	0.7	0.8	0.6	0.4	0.4	0.7	0.8	0.4	0	0.6	0.7	0.6	0.9	0.8	0.8	0.7
1	0.8	0.8	0.8	0.9	0.8	0.9	0.6	0.8	0.8	0.8	0.7	0.6	0	0.4	0.5	0.9	0.8	0.9	0.5
1	0.9	0.7	0.7	0.8	0.8	0.8	0.4	0.7	0.8	0.7	0.6	0.7	0.4	0	0.4	0.9	0.7	0.8	0.3
0.9	0.9	0.7	0.7	0.9	0.9	0.9	0.4	0.7	0.9	0.9	0.6	0.6	0.5	0.4	0	1	0.9	0.9	0.3
0.9	0.8	0.9	0.9	0.6	0.7	0.7	0.9	0.9	0.6	0.7	0.9	0.9	0.9	0.9	1	0	0.8	0.6	0.9

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17


0.8 0.6 0.6 0.7 0.5 0.5 0.6 0.6 0.7 0.5 0.3 0.7 0.8 0.8 0.7 0.9 0.8 0 0.6 0.7 18
1 0.8 0.8 0.8 0.7 0.7 0.7 0.7 0.8 0.7 0.6 0.7 0.8 0.9 0.8 0.9 0.6 0.6 0 0.7 19
1 0.9 0.8 0.8 0.9 0.8 0.9 0.5 0.7 0.9 0.8 0.7 0.7 0.5 0.3 0.3 0.9 0.7 0.7 0 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

How do we measure similarity?



Equal-distance contour plots according to four different distances: points on any one curve are all the same distance from the center point.

Euclidean The Euclidean distance between two points $A = (a_1, \ldots, a_p)$ and $B = (b_1, \ldots, b_p)$ in a p-dimensional space (for the p features) is the square root of the sum of squares of the differences in all p coordinate directions:

$$d(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_p - b_p)^2}.$$

Manhattan The Manhattan, City Block, Taxicab or L_1 distance takes the sum of the absolute differences in all coordinates.

$$d(A, B) = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_p - b_p|.$$

Maximum The maximum of the absolute differences between coordinates is also called the L_∞ distance:

$$d_\infty(A, B) = \max_i |a_i - b_i|.$$

Weighted Euclidean distance is a generalization of the ordinary Euclidean distance, by giving different directions in feature space different weights.

(the χ^2 distance)

The *Mahalanobis* distance is another weighted Euclidean distance.

Minkowski Allowing the exponent to be m instead of 2, as in the Euclidean distance, gives the Minkowski distance

$$d(A,B) = \left((a_1 - b_1)^m + (a_2 - b_2)^m + \ldots + (a_p - b_p)^m\right)^{\frac{1}{m}}.$$

Binary When the two vectors have binary bits as coordinates, we can think of the non-zero elements as 'on' and the zero elements as 'off'. The binary distance is the proportion of features having only one bit on amongst those features that have at least one bit on.

Jaccard Distance Occurrence of traits or features in ecological or mutation data can be translated into presence and absence and encoded as 1's and 0's.

In such situations, co-occurence is often more informative than co-absence.

For instance, when comparing mutation patterns in HIV, the co-existence in two different strains of a mutation tends to be a more important observation than its co-absence. For this reason, biologists use the **Jaccard index**.

Let's call our two observation vectors S and T, f_{11} the number of times a feature co-occurs in S and T, f_{10} (and f_{01}) the number of times a feature occurs in S but not in T (and vice versa), and f_{00} the number of times a feature is co-absent. The Jaccard index is

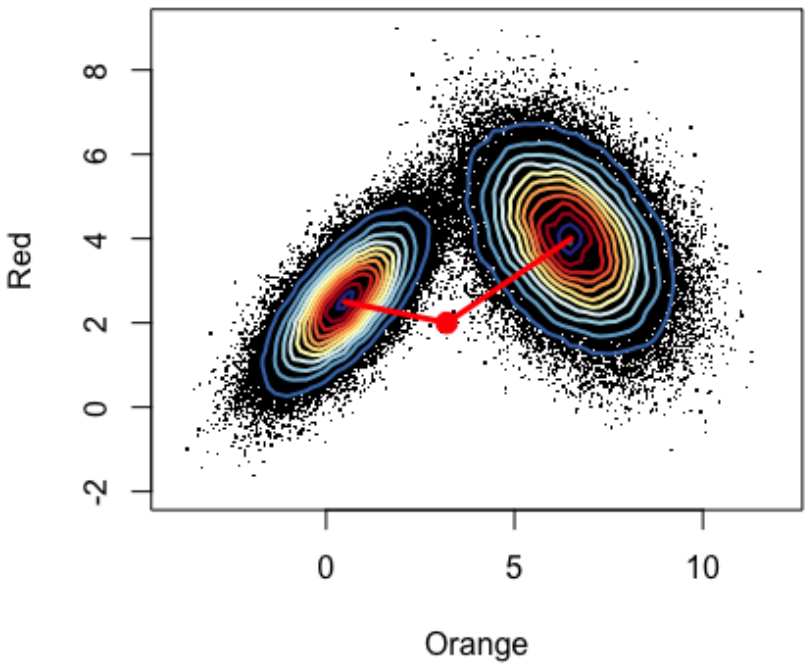
$$J(S,T) = \frac{f_{11}}{f_{01} + f_{10} + f_{11}},$$

(i.e., it ignores f_{00}), and the **Jaccard dissimilarity** is

$$d_J(S,T) = 1 - J(S,T) = \frac{f_{01} + f_{10}}{f_{01} + f_{10} + f_{11}}.$$

Correlation based distance

$$d(A, B) = \sqrt{2(1 - \text{cor}(A, B))}.$$



An example for the use of Mahalanobis distances to measure the distance of a new data point (red) from two cluster centers.

Which of the two cluster centers in the Figure is the red point closest to?

Computations related to distances in R

The `dist` function in R is optimized to use less space than the full n^2 positions a complete $n \times n$ distance matrix between n objects would require. The function computes one of six choices of distance (euclidean, maximum, manhattan, canberra, binary, minkowski) and outputs a vector of values sufficient to reconstruct the complete distance matrix. The function returns a special object of class `dist` that encodes the relevant vector of size $n \times (n-1)/2$. Here is the output for a 3 by 3 matrix:

```
mx  = c(0, 0, 0, 1, 1, 1)
my  = c(1, 0, 1, 1, 0, 1)
mz  = c(1, 1, 1, 0, 1, 1)
mat = rbind(mx, my, mz)
dist(mat)
```

```
##           mx       my
## my 1.732051
## mz 2.000000 1.732051
```

```
dist(mat, method = "binary")
```

```
##           mx        my
## my 0.6000000
## mz 0.6666667 0.5000000
```

In order to access a particular distance (for example the distance between observations 1 and 2), one has to turn the `dist` class object back into a matrix.

```
load(url("http://bios221.stanford.edu/data/Morder.RData"))
sqrt(sum((Morder[1, ] - Morder[2, ])^2))
```

```
## [1] 5.593667
```

```
as.matrix(dist(Morder))[2, 1]
```

```
## [1] 5.593667
```

Let's look at how we would compute the Jaccard distance we defined above between HIV strains.

```
mut = read.csv("http://bios221.stanford.edu/data/HIVmutations.csv")
mut[1:3, 10:16]
```

```
##   p32I p33F p34Q p35G p43T p46I p46L
## 1    0    1    0    0    0    0    0
## 2    0    1    0    0    0    1    0
## 3    0    1    0    0    0    0    0
```

Compare the Jaccard distance (available as the function `vegdist` in the R package **vegan**) between mutations in the HIV data `mut` to the correlation based distance.

```
library("vegan")
mutJ = vegdist(mut, "jaccard")
mutC = sqrt(2 * (1 - cor(t(mut))))
mutJ
```

```
##       1     2     3     4
## 2 0.800
## 3 0.750 0.889
## 4 0.900 0.778 0.846
## 5 1.000 0.800 0.889 0.900
```

```
as.dist(mutC)
```

```
##      1    2    3    4
## 2 1.19
## 3 1.10 1.30
## 4 1.32 1.13 1.30
## 5 1.45 1.19 1.30 1.32
```

The Jaccard index between graphs can be computed by looking at two graphs built on the same nodes and counting the number of co-occurring edges.

This is implemented in the function `similarity` in the **igraph** package.

Distances and dissimilarities are also used to compare images, sounds, maps and documents.

Asking yourself what is the *relevant* notion of “closeness” or similarity for your data can provide useful ways of representing them.

Testing relation between graph and factors

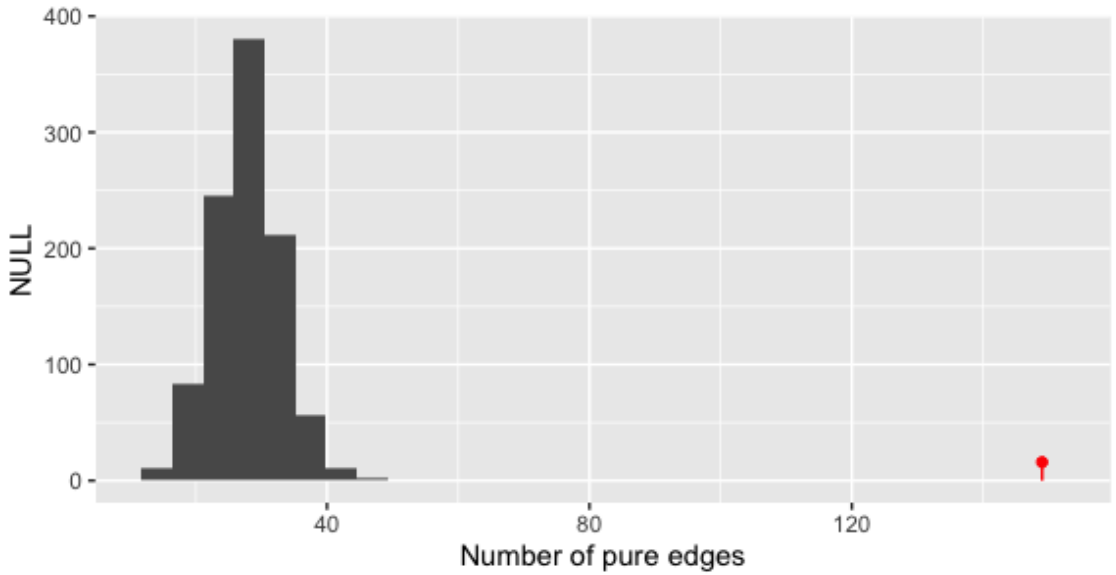
Compute the null distribution and p-value for the test, this is implemented in the `phyloseqGraphTest` package:

```
library("phyloseqGraphTest")
gt = graph_perm_test(ps1, "host_subject_id", distance="jaccard",
                     type="mst",  nperm=1000)
gt$pval
```

```
## [1] 0.000999001
```

Histogram of the null distribution generatedby permutation using:

```
plot_permutations(gt)
```



Different choices for the skeleton graph

It is not necessary to use an MST for the skeleton graph that defines the edges.

Graphs made by linking nearest neighbors (Schilling 1986)

Distance thresholding work also works well (sometimes called geometric graphs).

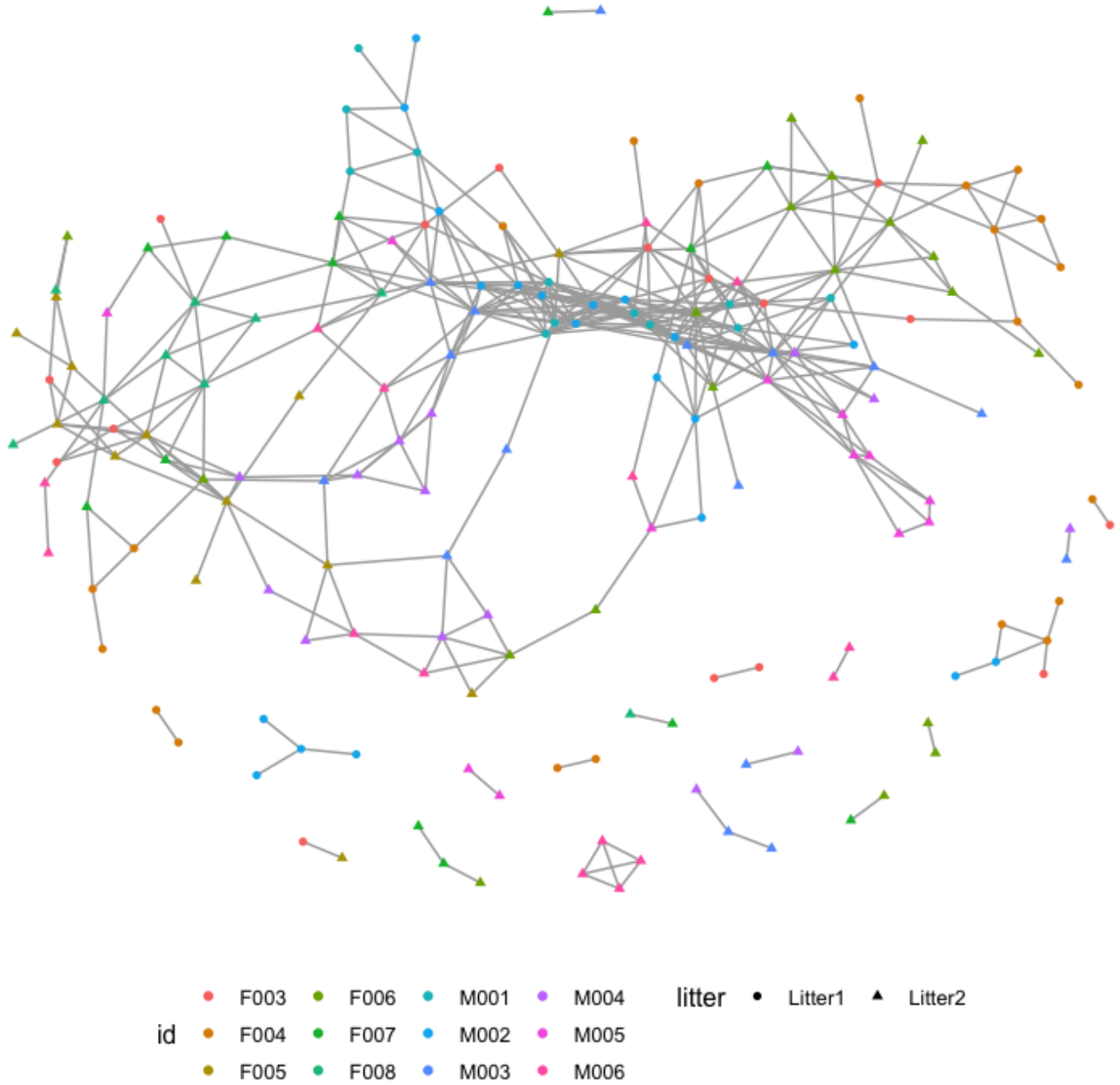
`phyloseq` has functionality for creating graphs based on thresholding a distance matrix through the function `make_network`.

Create a network by creating a edge between samples whose Jaccard dissimilarity is less than a threshold, which we set below via the parameter `max.dist` .

This is a co-occurrence network.

```
net = make_network(ps1, max.dist = 0.35)
sampledata = data.frame(sample_data(ps1))
V(net)$id = sampledata[names(V(net)), "host_subject_id"]
V(net)$litter = sampledata[names(V(net)), "family_relationship"]
netg = ggnetwork(net)
```

```r
ggplot(netg, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(color = "darkgray") +
  geom_nodes(aes(color = id, shape = litter)) + theme_blank()+
    theme(legend.position="bottom")
```


id
F003
F004
F005
F006
F007
F008
M001
M002
M003
M004
M005
M006
litter
Litter1
Litter2

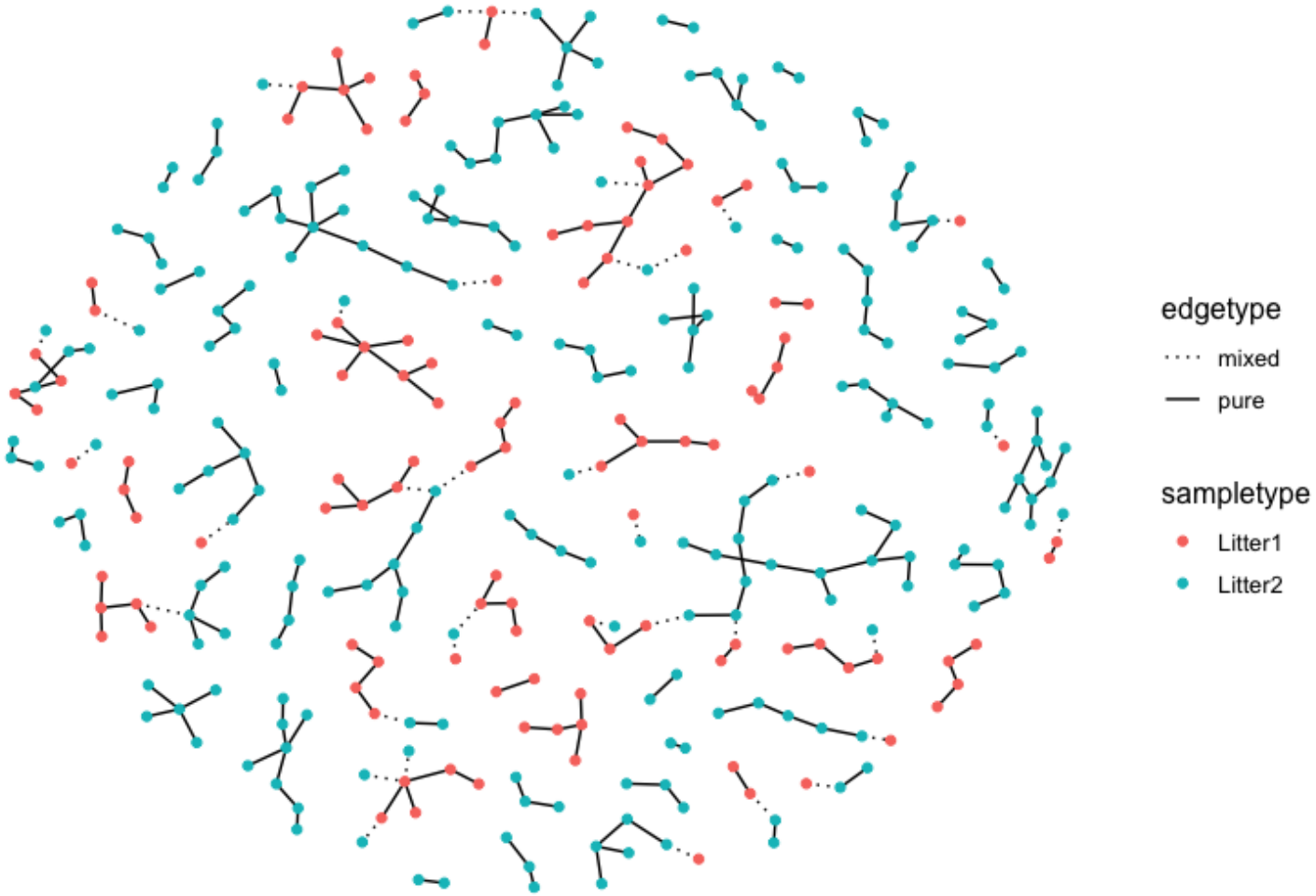
Sometimes it will preferable to adjust the permutation distribution to account for known structure between the covariates.

Friedman–Rafsy test with nested covariates

individual mice (the host_subject_id variable).

a litter (the family_relationship variable) effect ?

```
plot_test_network(gtnn1)
```

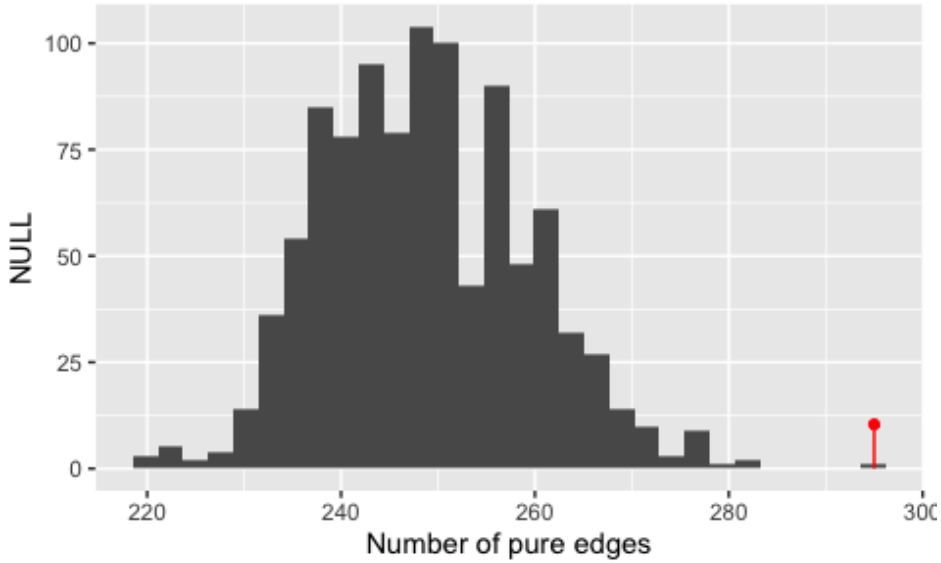


We permute the `family_relationship` labels but keep the `host_subject_id` structure intact.

```
gt = graph_perm_test(ps1, "family_relationship",
          grouping = "host_subject_id",
          distance = "jaccard", type = "mst", nperm= 1000)
gt$pval
```

```
## [1] 0.001998002
```

```
plot_permutations(gt)
```




```
gtnn1 = graph_perm_test(ps1, "family_relationship",
                        grouping = "host_subject_id",
                        distance = "jaccard", type = "knn", knn = 1)
gtnn1$pval
```

```
## [1] 0.004
```

The dual graph

- Between samples through their shared taxa.
- Between taxa: do some of the taxa co-occur more often than one would expect ?

Microbial ‘communities’ as they assemble in the microbiome.

Transpose the data.

Note: always prefer Jaccard to correlation networks.

Useful Packages for Incorporating Graphs into an Analysis

Important examples of graphs and useful R packages

ggnetwork and **igraph**

Combining graphs with statistical data

structSSI and **phyloseq**. **bioNet**

A full list packages that deal with graphs and networks is available here:
http://www.bioconductor.org/packages/release/BiocViews.html#___GraphAndNetwork

References

Benjamini, Yoav, and Marina Bogomolov. 2014. “Selective Inference on Multiple Families of Hypotheses.” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76 (1). Wiley Online Library: 297–318.

Benjamini, Yoav, and Daniel Yekutieli. 2003. “Hierarchical Fdr Testing of Trees of Hypotheses.” Technical report, Department of Statistics; Operations Research, Tel Aviv University.

Friedman, Jerome H, and Lawrence C Rafsky. 1979. “Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two-Sample Tests.” *The Annals of Statistics*. JSTOR, 697–717.

Sankaran, Kris, and Susan Holmes. 2014. “StructSSI: Simultaneous and Selective Inference for Grouped or Hierarchically Structured Data.” *Journal of Statistical Software* 59 (1): 1–21.

Schilling, Mark F. 1986. “Multivariate Two-Sample Tests Based on Nearest Neighbors.” *Journal of the American Statistical Association* 81 (395). Taylor & Francis: 799–806.