Chapter 9 examples

Brooke Anderson

4/15/2020

```
library(tidyverse)
```

Example—bacterial abundance

This is used in section 9.3. It covers a study that compared healthy to diseased (irritable bowel syndrome) mice. It's looking at the bacterial abundance in these two groups, particularly if there's evidence of important differences between the healthy and diseased mice.

This data is stored in a binary R file format (".rds"). You can read it into R with the function `readRDS`:

```
ibd_chip <- readRDS("data/vsn28Exprd.rds")
```

This data is in a matrix. Each column is a sample and each rows gives observations, so this is another example of the "transposed" data format we're coming across for a lot of these biological datasets in R. All rows expect the last one give the measurements in each sample for a specific bacterial taxon, so looking across each row provides an estimate of how the abundance of bacteria in that taxon vary across the samples. Based on the text in the book, these abundance measurements have already been normalized in some way (maybe this is why they don't show up as whole-number counts?).

Some information is stored in the rownames (a identifier of the bacterial taxon given on that row), so if you change this to a dataframe at any point, you'll probably want to bring the rownames into a column (`rownames_to_column` from the `tibble` package, for example) so you don't lose that information.

```
ibd_chip %>%
  head()
```

```
##                                      20CF     20DF     20MF     20PF       CC
## 01010101000000.2104_gPM_GC     5.822202 6.013525 6.193154 5.995809 5.939815
## 01010101000000.2141_gPM_GC     6.628111 6.791654 6.894708 6.698952 6.649734
## 01010101000000.2147_pPM_GC     7.718877 7.837144 7.919016 7.728473 7.434145
## 01010101000000.2154_gPM_GC     6.373156 6.813640 6.690627 6.796760 6.654098
## 01010101000000.2196_gPM_GC     6.709530 6.830658 6.711614 6.810716 6.646262
## 01010101000000.2202_gPM_dPM_GC 7.171106 7.697832 7.595051 7.481985 7.163951
##                                        CD       CM CNTR23CF CNTR23DF CNTR23MF
## 01010101000000.2104_gPM_GC     5.879038 6.054405 5.948460 6.177409 5.877702
## 01010101000000.2141_gPM_GC     6.816117 6.866086 6.558888 7.111481 6.500045
## 01010101000000.2147_pPM_GC     7.717579 7.672562 7.441511 7.957593 7.490416
## 01010101000000.2154_gPM_GC     6.658794 6.871937 6.276523 6.616352 6.579366
## 01010101000000.2196_gPM_GC     6.743989 6.897269 6.503804 6.971761 6.355477
## 01010101000000.2202_gPM_dPM_GC 7.441376 7.504790 7.249389 7.693170 7.173640
##                                CNTR23PF   CNTRCF   CNTRDF   CNTRMF   CNTRPF
## 01010101000000.2104_gPM_GC     5.812016 5.999729 6.230501 5.969050 6.053401
## 01010101000000.2141_gPM_GC     6.681861 6.979696 7.001440 6.382498 6.558707
## 01010101000000.2147_pPM_GC     7.567559 7.669541 7.928936 7.427211 7.414472
## 01010101000000.2154_gPM_GC     6.430767 6.942562 6.844614 6.391119 6.483173
## 01010101000000.2196_gPM_GC     6.606087 6.842585 6.848668 6.582288 6.869764
## 01010101000000.2202_gPM_dPM_GC 7.120038 7.535775 7.850615 7.069872 7.167207
##                                      CP IBS1_CntrCb IBS1_CntrPb IBS1_IBSCb
## 01010101000000.2104_gPM_GC     5.899210    6.021985    6.074105   6.130299
## 01010101000000.2141_gPM_GC     6.732701    6.816760    6.854775   7.091984
## 01010101000000.2147_pPM_GC     7.541400    7.743678    7.784393   7.895193
## 01010101000000.2154_gPM_GC     6.523063    6.736578    6.773245   7.057359
## 01010101000000.2196_gPM_GC     6.785729    6.887595    6.763405   6.915904
## 01010101000000.2202_gPM_dPM_GC 7.183156    7.624855    7.374476   7.721901
##                                IBS1_IBSPb     IBSC     IBSD     IBSM     IBSP
## 01010101000000.2104_gPM_GC       6.065744 5.985233 5.931086 6.134504 6.362426
## 01010101000000.2141_gPM_GC       6.809185 6.920211 6.610850 6.815327 6.931352
## 01010101000000.2147_pPM_GC       7.768787 7.866416 7.681480 7.830903 8.232905
## 01010101000000.2154_gPM_GC       6.597753 6.782565 6.778239 6.652730 7.034562
## 01010101000000.2196_gPM_GC       6.838210 6.894402 6.750959 6.970233 7.047048
## 01010101000000.2202_gPM_dPM_GC   7.389213 7.689195 7.428264 7.443049 7.889873
##                                      IC       ID       IM       IP
## 01010101000000.2104_gPM_GC     5.932038 6.067206 5.955856 5.819750
## 01010101000000.2141_gPM_GC     6.927258 6.794931 6.714941 6.610351
## 01010101000000.2147_pPM_GC     7.644741 7.792662 7.733240 7.494777
## 01010101000000.2154_gPM_GC     6.777446 6.610261 6.627344 6.410889
## 01010101000000.2196_gPM_GC     6.805910 6.850387 6.772824 6.637868
## 01010101000000.2202_gPM_dPM_GC 7.434901 7.375826 7.406864 7.172607
```

You can check out the size of the data:

```r
ibd_chip %>%
  dim()
```

```
## [1] 8635   28
```

It looks like there are 28 samples and that they measured 8635 observations for each (although we'll see in a minute that one of these was just the day the sample was analyzed).

They ask us to look at the last row of the data. You can use `tail` (from base R) to do this (it defaults to show the last 6 rows):

```r
ibd_chip %>%
  tail()
```

```
##                                  20CF     20DF     20MF     20PF       CC
## bm-026.1.sig_st              7.299308 7.275802 7.383103 7.342093 7.197265
## bm-125.1.sig_st              8.538857 8.998562 9.296096 9.024030 8.572679
## bru.tab.d.HIII.Con32.sig_st  6.802736 6.777566 6.859950 6.835387 6.687917
## bru.tab.d.HIII.Con323.sig_st 6.463604 6.501139 6.611851 6.574264 6.403534
## bru.tab.d.HIII.Con5.sig_st   5.739235 5.666060 5.831079 5.740588 5.582382
## day                          2.000000 2.000000 2.000000 2.000000 1.000000
##                                    CD       CM CNTR23CF CNTR23DF CNTR23MF
## bm-026.1.sig_st              7.181504 7.287361 7.185804 7.543725 7.320221
## bm-125.1.sig_st              8.658590 8.685400 8.696692 9.488539 8.788726
## bru.tab.d.HIII.Con32.sig_st  6.749703 6.822694 6.724367 6.974028 6.827416
## bru.tab.d.HIII.Con323.sig_st 6.526329 6.542377 6.474240 6.740075 6.518457
## bru.tab.d.HIII.Con5.sig_st   5.647223 5.600590 5.764725 5.856314 5.897300
## day                          1.000000 1.000000 2.000000 2.000000 2.000000
##                              CNTR23PF   CNTRCF   CNTRDF   CNTRMF   CNTRPF
## bm-026.1.sig_st              7.282073 7.447955 7.411746 7.257611 7.193387
## bm-125.1.sig_st              8.558453 8.803169 9.250040 8.734481 8.528806
## bru.tab.d.HIII.Con32.sig_st  6.791515 6.941121 6.903581 6.719014 6.763368
## bru.tab.d.HIII.Con323.sig_st 6.426744 6.713179 6.634948 6.533171 6.478214
## bru.tab.d.HIII.Con5.sig_st   5.695705 5.785868 5.787694 5.813347 5.731513
## day                          2.000000 2.000000 2.000000 2.000000 2.000000
##                                    CP IBS1_CntrCb IBS1_CntrPb IBS1_IBSCb
## bm-026.1.sig_st              7.187511    7.353554    7.501938    7.439939
## bm-125.1.sig_st              8.536959    9.058916    9.067296    9.165668
## bru.tab.d.HIII.Con32.sig_st  6.686230    6.856511    6.888071    6.869919
## bru.tab.d.HIII.Con323.sig_st 6.415102    6.608229    6.710256    6.646177
## bru.tab.d.HIII.Con5.sig_st   5.602820    5.774074    5.817978    5.721342
## day                          1.000000    3.000000    3.000000    3.000000
##                              IBS1_IBSPb     IBSC     IBSD     IBSM     IBSP
## bm-026.1.sig_st                7.423275 7.256487 7.405401 7.403899 7.598086
## bm-125.1.sig_st                8.966644 8.933226 8.954909 8.996059 9.463854
## bru.tab.d.HIII.Con32.sig_st    6.846558 6.809883 6.794731 6.845082 6.994816
## bru.tab.d.HIII.Con323.sig_st   6.682649 6.552750 6.628656 6.651153 6.844713
## bru.tab.d.HIII.Con5.sig_st     5.816053 5.721133 5.756525 5.756231 5.919351
## day                            3.000000 2.000000 2.000000 2.000000 2.000000
##                                    IC       ID       IM       IP
## bm-026.1.sig_st              7.109487 7.355406 7.242506 7.106093
## bm-125.1.sig_st              8.686113 8.785689 8.805357 8.444407
## bru.tab.d.HIII.Con32.sig_st  6.740634 6.787609 6.745646 6.645080
## bru.tab.d.HIII.Con323.sig_st 6.481958 6.538410 6.493336 6.398438
## bru.tab.d.HIII.Con5.sig_st   5.593096 5.674684 5.705359 5.624223
## day                          1.000000 1.000000 1.000000 1.000000
```

The `day` row gives the day (day 1 or day 2) that the sample is collected. In this section, we'll be looking at whether the data shows evidence of batch effects by day (i.e., that the samples collected on the same day tend to be more similar to each other than when comparing samples analyzed on two different days).

To get a summary of the `day` information, one approach is to transpose the data (`t`), change it to a tibble (`as_tibble`), then pull out the `day` column as a vector and pipe it through `summary` to get a summary of that vector:

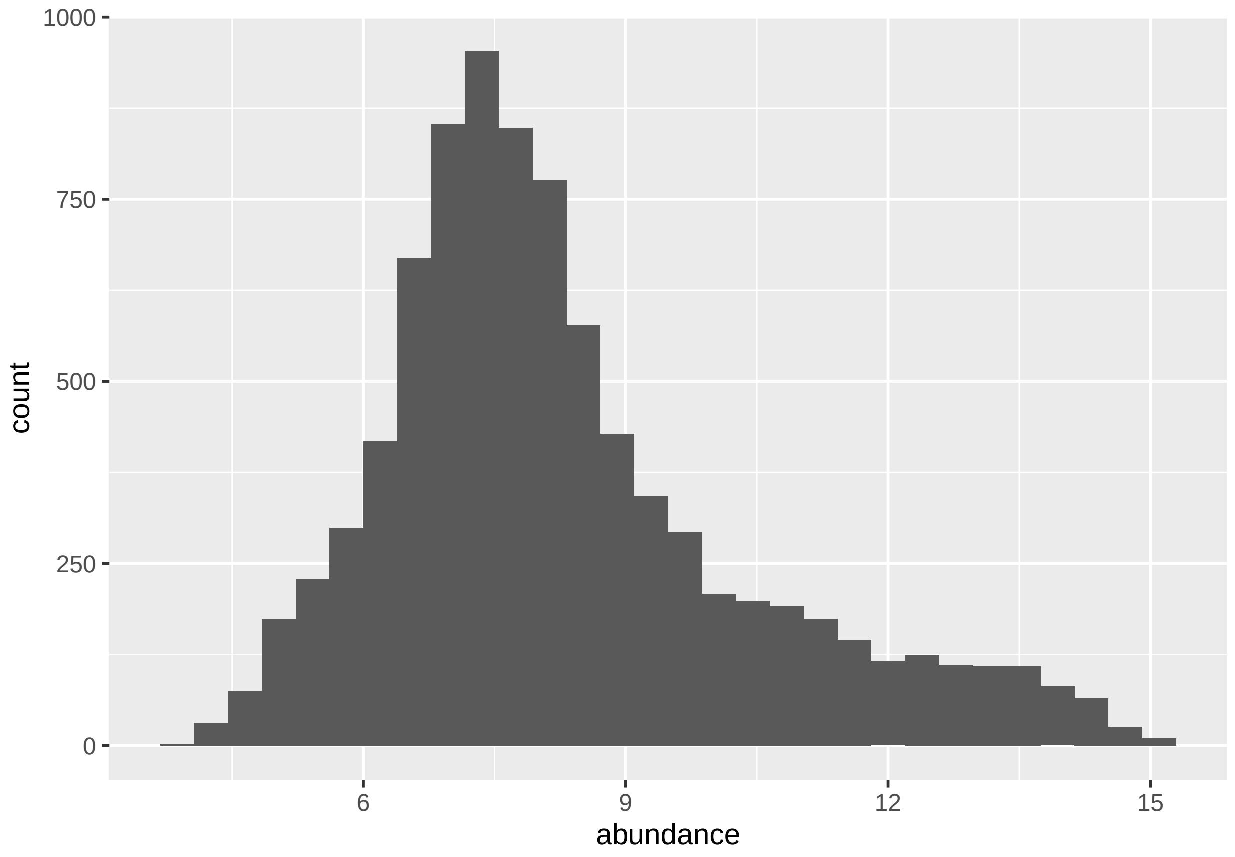
```
ibd_chip %>%
  t() %>%
  as_tibble() %>%
  pull(day) %>%
  summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   1.857   2.000   3.000
```

You can see that it looks like the data were collected across three days, so this variable can take the values 1 (first day), 2 (second day), or 3 (third day).

To explore the data, it might be interesting to first see how the average bacterial abundance (averaged across samples) varies across all the 8,000-some taxa measured:

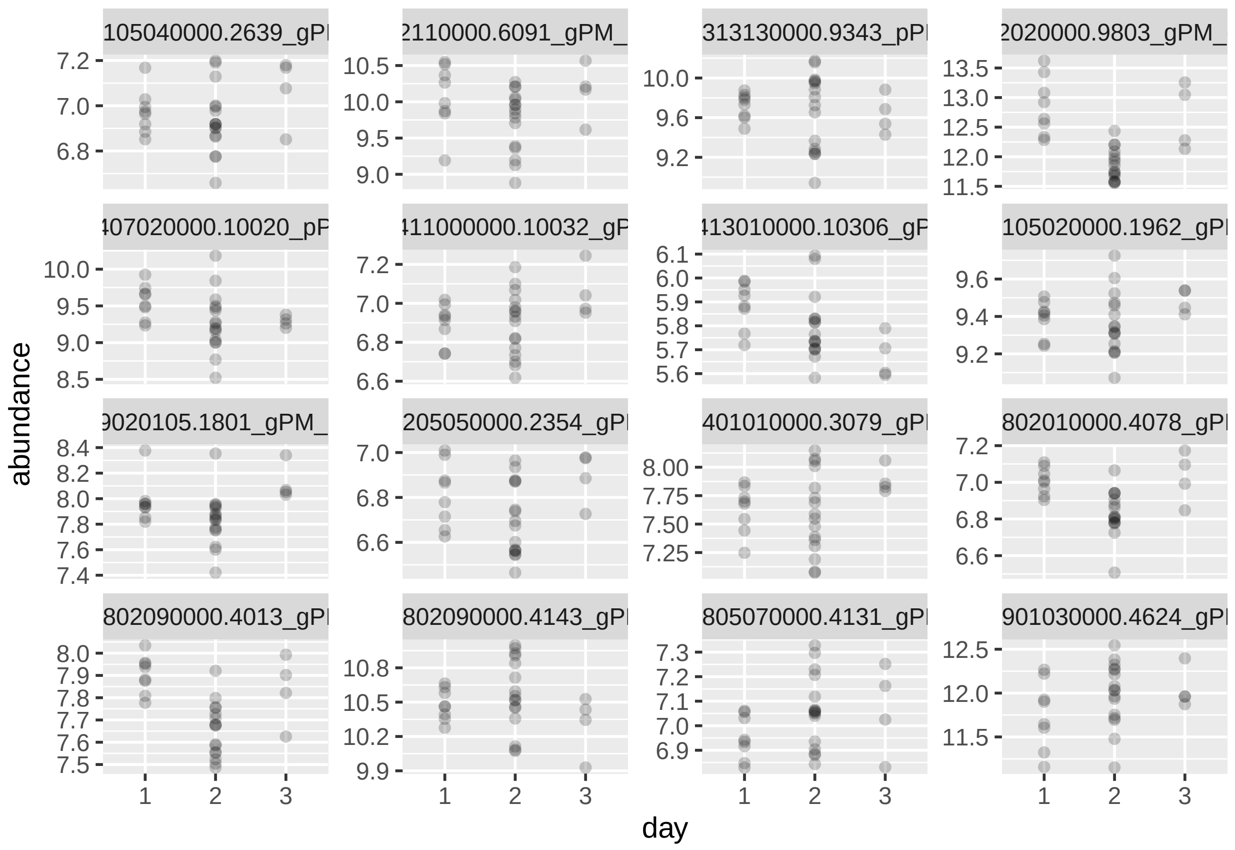
```
ibd_chip %>%
  t() %>% # Transpose the data
  as_tibble() %>%
  select(-day) %>% # Get rid of the `day` column for this analysis
  summarize_all(.funs = mean) %>% # Get the mean value for each column
  pivot_longer(everything(), # Here, the `everything()` means to pivot all the columns
               names_to = "taxon", values_to = "abundance") %>%
  ggplot(aes(x = abundance)) +
  geom_histogram()
```



It looks like the abundance measurements for most taxa are in the 7–8 range, all are positive, and a few are as high as 14 or 15.

We can take a look at how these measurements vary by day for a random sample of the bacterial taxa measured. In this pipeline, I've used nesting to allow for a random sample of 16 of the bacterial taxa and then used a plot with faceting by taxa to look at the measurements by day:

```
ibd_chip %>%
  t() %>% # Transpose the matrix
  as.data.frame() %>%  # Convert to a dataframe
  rownames_to_column(var = "sample_id") %>% # Make sure to keep the rownames (sample id now)
  pivot_longer(-c(sample_id, day),     # Pivot all the taxon columns (but not sample id or day)
               names_to = "taxon", values_to = "abundance") %>%
  group_by(taxon) %>%
  nest() %>% # Sample 20 random taxa from all that are measured
  ungroup() %>%
  sample_n(size = 16) %>%
  unnest(data) %>%
  mutate(day = as_factor(day)) %>% # Change day to a factor---better axis labels in this case
  ggplot(aes(x = day, y = abundance)) +
  geom_point(alpha = 0.2) +
  facet_wrap(~ taxon, ncol = 4, scales = "free_y")
```



In some cases, it does look like samples measured on the same day tend to group together in their measurements of abundance, so that abundance measuremens for some taxa tend to be more similar for samples measured on the same day compared to samples measured on different days.

In the text, they decided to do a rank-threshold transformation. The rules for that transform are:

1. Within each sample, rank all the taxa in terms of their abundance (1 to 8,635, with 1 as the most abundant)
2. For the first 3,000 in rank for that sample, replace the measured (normalized) abundance with a function of the rank (e.g., "3,000" for the most abundant, "2,999" for the next most abundant).
3. For the others, replace the measured (normalized) abundance with 1 (i.e., they all "tie" at a very low rank)

Here's an alternative way to do that compared to the code in the book (this includes filtering to only the samples taking on days 1 and 2, as they do in the book):

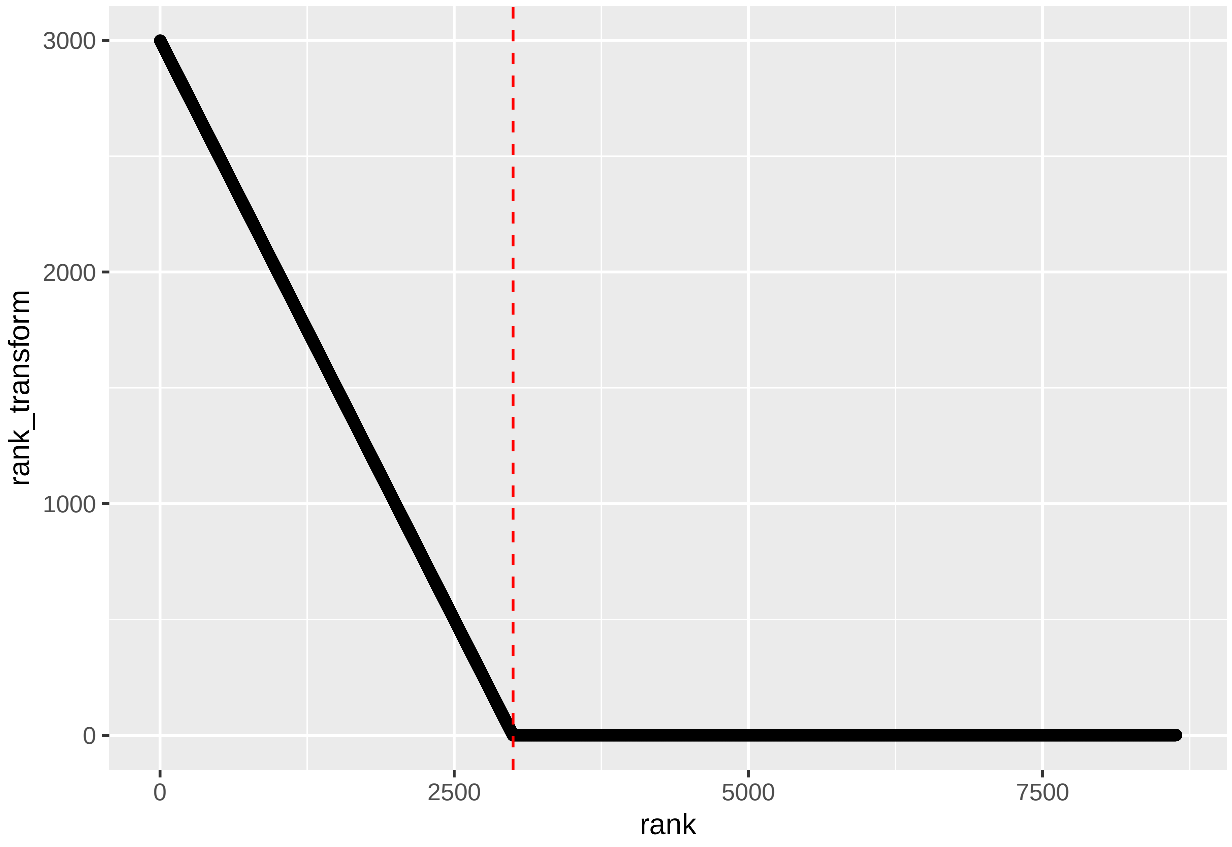
```
ibd_new <- ibd_chip %>%
  t() %>%
  as.data.frame() %>%
  rownames_to_column(var = "sample_id") %>%
  filter(day %in% c(1, 2)) %>% # Filter to samples analyzes on days 1 and 2
  mutate(day = as_factor(day)) %>% # Convert `day` to a factor (for visualization later)
  pivot_longer(-c(sample_id, day), names_to = "taxon", values_to = "abundance") %>%
  group_by(sample_id) %>%
  mutate(rank = rank(abundance)) %>% # Rank the abundance values in each sample
  ungroup() %>%
  mutate(rank_transform = case_when( # Create the rank-threshold transformation based on rank
    rank <= 3000 ~ 3000 - rank, # This is what you do if the rank is 3,000 or lower
    TRUE ~ as.numeric(1)         # This is what you do if it's not
  ))

ibd_new
```

```
## # A tibble: 207,216 x 6
##    sample_id day   taxon                              abundance  rank rank_transform
##    <chr>     <fct> <chr>                                  <dbl> <dbl>          <dbl>
##  1 20CF      2     01010101000000.2104_gPM_GC              5.82   671           2329
##  2 20CF      2     01010101000000.2141_gPM_GC              6.63  1735           1265
##  3 20CF      2     01010101000000.2147_pPM_GC              7.72  4185              1
##  4 20CF      2     01010101000000.2154_gPM_GC              6.37  1274           1726
##  5 20CF      2     01010101000000.2196_gPM_GC              6.71  1885           1115
##  6 20CF      2     01010101000000.2202_gPM_dPM_GC          7.17  2912             88
##  7 20CF      2     01010101000000.2216_pPM_GC              7.75  4251              1
##  8 20CF      2     01010101000000.2231_gPM_GC              6.15   999           2001
##  9 20CF      2     01010101000000.2233_gPM_GC              6.42  1352           1648
## 10 20CF      2     01010101000000.2236_gPM_GC              6.78  2021            979
## # ... with 207,206 more rows
```

You can check that you did this right by visualizing these rank-threshold transformation values for one of the samples:

```
ibd_new %>%
  filter(sample_id == first(sample_id)) %>% # Limit to values for the first sample
  ggplot(aes(x = rank, y = rank_transform)) +
  geom_point() +
  geom_vline(xintercept = 3000, col = "red", linetype = 2) # Add a line to show the threshold
```



Then, they ask you to run a PCA of the data, but using these rank-threshold transformation values instead of the original values:

```
library(ade4)
pca_result <- ibd_new %>%
  select(-c(abundance, rank_transform)) %>%
  pivot_wider(names_from = taxon, values_from = rank) %>%
  select(-sample_id, -day) %>%
  dudi.pca(scannf = FALSE, nf = 2) %>%
  pluck("li")
pca_result
```

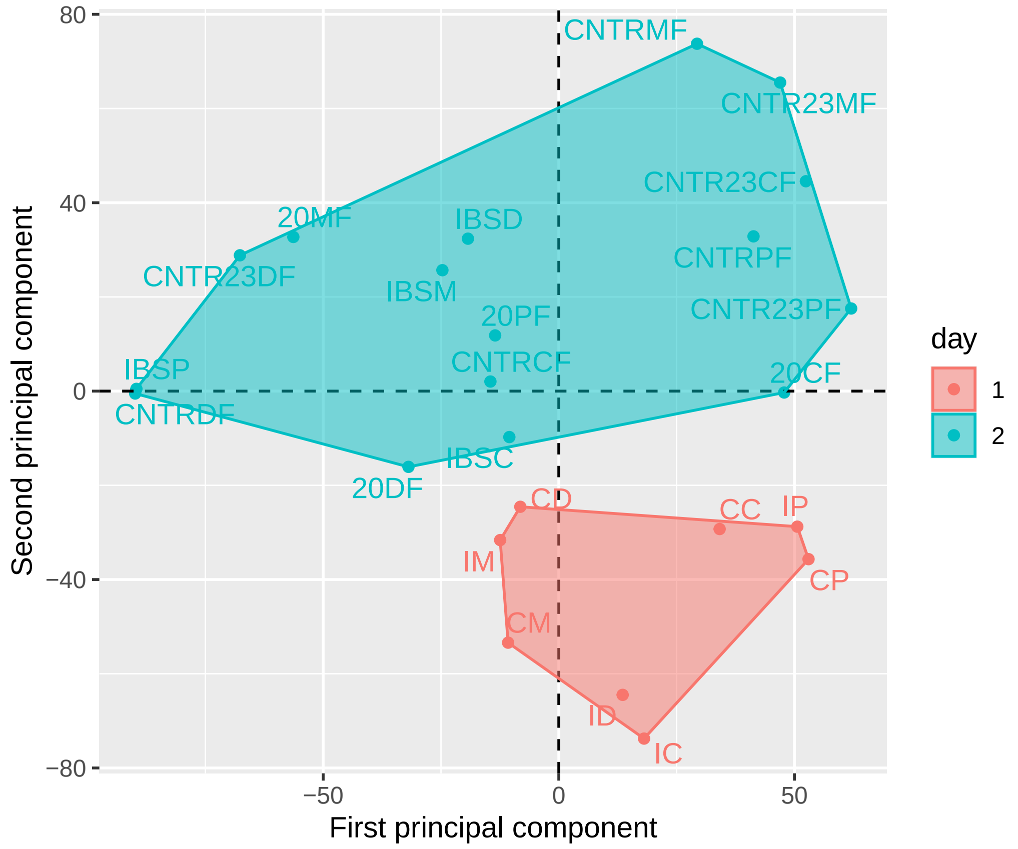
```
##        Axis1       Axis2
## 1   47.82118  -0.2958269
## 2  -31.89974 -16.0814136
## 3  -56.33141  32.7283799
## 4  -13.50589  11.8272180
## 5   34.12857 -29.2742311
## 6   -8.16916 -24.5519696
## 7  -10.76065 -53.4009773
## 8   52.43700  44.5617386
## 9  -67.66720  28.8372699
## 10  46.97636  65.5153093
## 11  62.04163  17.5427533
## 12 -14.51042   2.0380965
## 13 -89.89473  -0.5092561
## 14  29.33527  73.7506893
## 15  41.31576  32.8592479
## 16  52.98975 -35.6752368
## 17 -10.49521  -9.7513844
## 18 -19.28030  32.3644922
## 19 -24.70345  25.6763068
## 20 -89.63101   0.4823280
## 21  18.08760 -73.7629588
## 22  13.55093 -64.4854236
## 23 -12.44168 -31.6224886
## 24  50.60680 -28.7726628
```

```
pca_result <- ibd_new %>%
  group_by(sample_id) %>%
  slice(1) %>%
  ungroup() %>%
  select(sample_id, day) %>%
  bind_cols(pca_result)
pca_result
```

```
## # A tibble: 24 x 4
##    sample_id day    Axis1   Axis2
##    <chr>     <fct>  <dbl>   <dbl>
##  1 20CF      2      47.8   -0.296
##  2 20DF      2     -31.9  -16.1
##  3 20MF      2     -56.3   32.7
##  4 20PF      2     -13.5   11.8
##  5 CC        1      34.1  -29.3
##  6 CD        1      -8.17 -24.6
##  7 CM        1     -10.8  -53.4
##  8 CNTR23CF  2      52.4   44.6
##  9 CNTR23DF  2     -67.7   28.8
## 10 CNTR23MF  2      47.0   65.5
## # ... with 14 more rows
```

Here's an alternative way to create the plot shown in Figure 9.11:

```
library(ggrepel)
# Calculate the hull for each group
hull <- pca_result %>%
  group_by(day) %>%
  slice(chull(Axis1, Axis2))
# Create the plot and add the hull
ggplot(pca_result, aes(x = Axis1, y = Axis2, color = day, fill = day)) +
  geom_point() +
  geom_vline(xintercept = 0, linetype = 2) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_text_repel(aes(label = sample_id), show.legend = FALSE) + # Add labels for each sample
  coord_fixed() + # Forces anratio of x-unit distance to y-unit distance of 1
  labs(x = "First principal component",
       y = "Second principal component") +
  geom_polygon(data = hull, alpha = 0.5) # This adds the hull (shaded region)
```

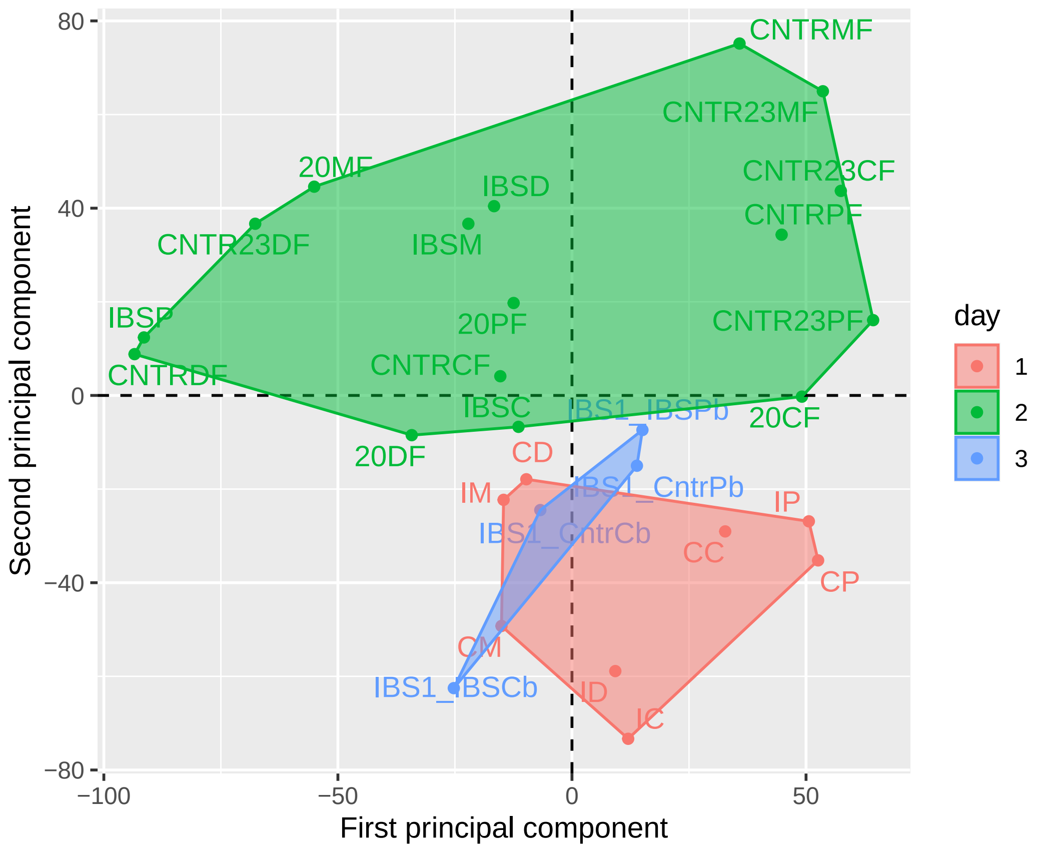


Evidently, they used measurements on the third day to try to tease apart why they see these differences between days 1 and 2. There are two difference in the days: (1) the protocol used and (2) provenences for the arrays (? I'm not completely sure what this means...). If we add in the measurements for day three, it might help tease apart differences (because they ran these with the protocol from one of the days and the array from the other day, so whichever group this new group “hangs out” with in the PCA will help us figure out what piece is explaining the variation).

```r
# Same as before, but keep day 3
ibd_new2 <- ibd_chip %>%
  t() %>%
  as.data.frame() %>%
  rownames_to_column(var = "sample_id") %>%
  mutate(day = as_factor(day)) %>% # Convert `day` to a factor (for visualization later)
  pivot_longer(-c(sample_id, day), names_to = "taxon", values_to = "abundance") %>%
  group_by(sample_id) %>%
  mutate(rank = rank(abundance)) %>% # Rank the abundance values in each sample
  ungroup() %>%
  mutate(rank_transform = case_when( # Create the rank-threshold transformation based on rank
    rank <= 3000 ~ 3000 - rank, # This is what you do if the rank is 3,000 or lower
    TRUE ~ as.numeric(1)        # This is what you do if it's not
  ))

# Re-run PCA and join with data on the day of measurement
pca_result2 <- ibd_new2 %>%
  select(-c(abundance, rank_transform)) %>%
  pivot_wider(names_from = taxon, values_from = rank) %>%
  select(-sample_id, -day) %>%
  dudi.pca(scannf = FALSE, nf = 2) %>%
  pluck("li")
pca_result2 <- ibd_new2 %>%
  group_by(sample_id) %>%
  slice(1) %>%
  ungroup() %>%
  select(sample_id, day) %>%
  bind_cols(pca_result2)

# Create the plot
hull2 <- pca_result2 %>%
  group_by(day) %>%
  slice(chull(Axis1, Axis2))
ggplot(pca_result2, aes(x = Axis1, y = Axis2, color = day, fill = day)) +
  geom_point() +
  geom_vline(xintercept = 0, linetype = 2) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_text_repel(aes(label = sample_id), show.legend = FALSE) +
  coord_fixed() +
  labs(x = "First principal component",
       y = "Second principal component") +
  geom_polygon(data = hull2, alpha = 0.5)
```



Since the samples from the third day are close to the ones from the first day, and since the samples for the third day were run using the protocol from day 1 but the assay from day 2, this result suggests that variation between the measurements on day 1 and day 2 result from differences in the protocol much more than differences in the array.