intro to CoDa

What is CoDa?

Compositional data (CoDa) are data where the data are constrained to a constant sum. In more formal terms, values in the dataset are limited to strictly positive values with a maximum. Some simple examples would be budgets where the total amount of money to be spent is a constant, or elections where a vote for one candidate means a vote against another.

CoDa have significant problems in interpretation.

In the simplest case, we have two categories, and all data points must be either in Trump or Clinton. There are three things we know about this data: the number of votes for Trump, the number of votes for Clinton, and the total number of votes. Knowing any two of these allows us to know the third. So, if we know that 10 people voted, and that 6 voted Trump, we don't need to know anything else to infer Clinton's votes. Thus, the Trump and Clinton votes are perfectly negatively correlated when expressed as a proportion of the total (See fivethirtyeight.com). This is known as the negative correlation bias.

Adding one more category, Trump, Clinton and Johnson (did you know there was a third candidate?), and we change the picture a bit. Now if we know how Trump and Johnson did we can infer how Clinton fared. That the data change with the addition of a category is called the subcomposition problem: our inference is not necessarily the same with different numbers of categories.

The third problem is that of spurious correlation. Here, while we have a bias to negative correlation, we can still observe strong positive or negative correlations between randomly chosen variables, and worse, the variables that correlate differ in different sub-compositions.

Why do you care?

Sequencing is a constant sum operation. The same library sequenced on a MiSeq or a NextSeq, or a HiSeq will give different numbers of reads: thus the total is irrelevant. That is, we have a constant sum problem.

Illustration of the problems.

It is assumed that the output from a high-throughput sequencing experiment represents in some way the underlying abundance of the input DNA molecules. This is not necessarily the case as explained by the following thought experiment.

Figure 1 shows two idealized experiments with four different ways of looking at the exact same data: the top row shows the data as linear points, the bottom row shows the same data after a log transform. The constrained input shows the case where the total count of all nucleic acid species in the input is constrained to a constant sum. The unconstrained input shows the case where the total sum is not constrained to a constant sum. These are modelled as a time series, but any process would produce the same results, and in practice we will likely only be comparing the first and last points (before and after some intervention). These are shown in two different ways. The first is linear counts "input", the second is proportions.

Constrained datasets occur if the increase or decrease in any component is exactly compensated by the increase or decrease of one or more others. Here the total count remains constant across all experimental conditions. Examples of constrained datasets would include allele frequencies at a locus where the total has to be 1, and the RNA-seq where the induction of genes occurs in a steady- state cell culture. In this case, any process, such as sequencing that generates a constant sum simply recapitulates the data with sampling error. The unspoken assumption in most high throughput experimental designs is that this assumption is true — it is not!

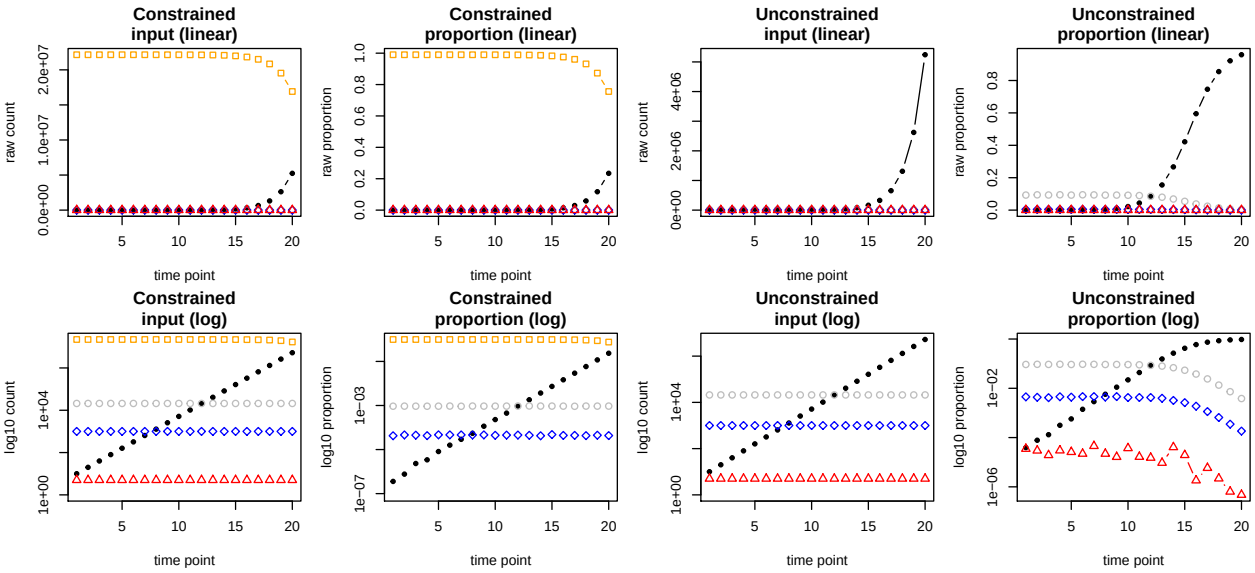

Figure 1: High-throughput sequencing affects the shape of the data differently on constrained and unconstrained data. The two left panels show the absolute number of reads in the input tube for 20 steps where the green and black OTUs are changing abundance by 2-fold each step. The gray, blue and red OTUs are held at a constant number in each step in both cases. The second column shows the output in proportions (or ppm, or FPKM) after random sampling to a constant sum, as occurs on the sequencer. The orange OTU in the constrained data set is much more abundant than any other, and is changing to maintain a constant number of input molecules. Samples in the two right columns are the same values plotted on a log scale on the Y-axis for convenience. Note how the constrained data is the same before and after sequencing while the unconstrained data is severely distorted.

An unconstrained dataset results if the total count is free to vary. Examples of unconstrained datasets include ChIP-Seq, RNA-seq where we are examining two different conditions or cell populations, metagenomics, etc. Importantly, 16S rRNA gene sequencing analyses are almost always free to vary; that is, the total bacterial load is rarely constant in an environment. Thus, the unconstrained data type would be the predominant type of data that would be expected.

The relative abundance panels on the right side of Figure 1 shows the result of random sampling with a defined maximum value in these two types of datasets. This random sampling reflects the data that results from high throughput sequencing where the total number of reads is constrained by the instrument capacity. The data is represented as a proportion, but scales to parts per million or parts per billion without changing the shape. Here we see that the shape of the data after sequencing is very similar to the input data in the case of constrained, but is very different in the case of non-constrained data. In the unconstrained dataset, observe how the blue and red features appear to be constant over the first 10 time points, but then appear to decrease in abundance at later time points. Conversely, the black feature appears to increase linearly at early time points, but appears to become constant at late time points. Obviously, we would misinterpret what is happening if we compared early and late timepoints in the unconstrained dataset. It is also worth noting how the act of random sampling makes the proportional abundance of the rare OTU species uncertain in both the constrained and unconstrained data, but has little effect on the relative apparent effect on the relative abundance of OTUs with high counts.

Questions

- Why are these constrained or unconstrained?
- Where is the negative correlation bias?
- What is the subcompositional effect?
- Where is the spurious correlation here?

But we can fix this, right?

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.4-1
```

```
## Warning: non-integer data: divided by smallest positive value
```

```
total: divide by margin total (default MARGIN = 1).
max: divide by margin maximum (default MARGIN = 2).
freq: divide by margin maximum and multiply by the number of non-zero items,
    so that the average of non-zero entries is one
    (Oksanen 1983; default MARGIN = 2).
normalize: make margin sum of squares equal to one (default MARGIN = 1).
range: standardize values into range 0 ... 1 (default MARGIN = 2).
    If all values are constant, they will be transformed to 0.
chi.square: divide by row sums and square root of column sums,
    and adjust for square root of matrix total (Legendre & Gallagher 2001).
    When used with the Euclidean distance, the distances should be similar
    to the Chi-square distance used in correspondence analysis. However,
```

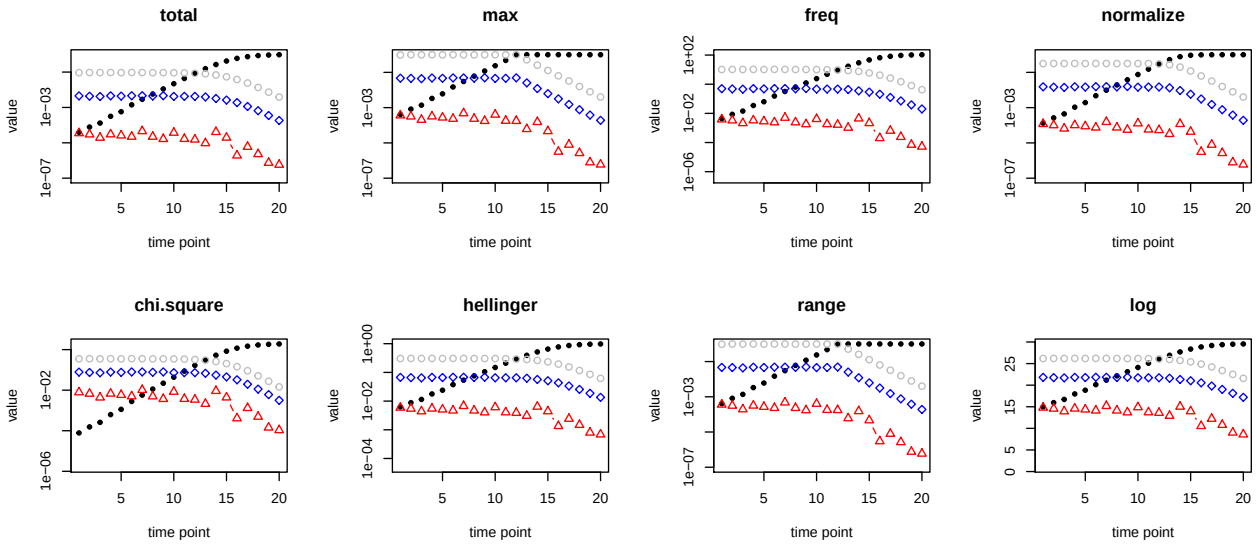

Figure 2: Most common transformations fail to correct for the shape of the data. Data transformations are from the vegan R package.

```
    the results from cmdscale would still differ, since CA is a weighted
    ordination method (default MARGIN = 1).
hellinger: square root of method = "total" (Legendre & Gallagher 2001).
log: logarithmic transformation as suggested by Anderson et al. (2006):
    log_b (x) + 1 for x > 0, where b is the base of the logarithm; zeros are
    left as zeros. Higher bases give less weight to quantities and more to
    presences, and logbase = Inf gives the presence/absence scaling. Please
    note this is not log(x+1). Anderson et al. (2006) suggested this for their
    (strongly) modified Gower distance, but the standardization can be used
    independently of distance indices.
```

So why is this?

```
Tiger <- round(runif(100, 1800,2200))
Ladybug <- round(runif(100, 8000,12000))
Alien <- round(runif(100, 450,550))

d <- data.frame(cbind(Tiger, Ladybug, Alien))

d.rare <- codaSeq.rarefy(d, n=1000, samples.by.row=FALSE)

if(plot==TRUE) pdf("tiger_count.pdf", height=4, width=14)
par(mfrow=c(1,4),mar=c(5,5,4,1))
plot(d$Tiger, d$Ladybug, main=round(cor(d$Tiger, d$Ladybug), 2), cex.lab=1.8,
    cex.main=2, pch=19,
    col=rgb(1,0,0,0.5), xlab="Tiger", ylab="Ladybug")
plot(d$Tiger, d$Alien, main=round(cor(d$Tiger, d$Alien), 2), cex.lab=1.8,
    cex.main=2, pch=19,
    col=rgb(1,0,0,0.5), xlab="Tiger", ylab="Alien")
plot(d$Ladybug, d$Alien, main=round(cor(d$Ladybug, d$Alien), 2), cex.lab=1.8,
```

```
    cex.main=2, pch=19, col=rgb(1,0,0,0.5), xlab="Ladybug", ylab="Alien")
scatterplot3d(d, type="h", lty.hplot=3, angle=40, pch=19, cex.lab=1.5)
```

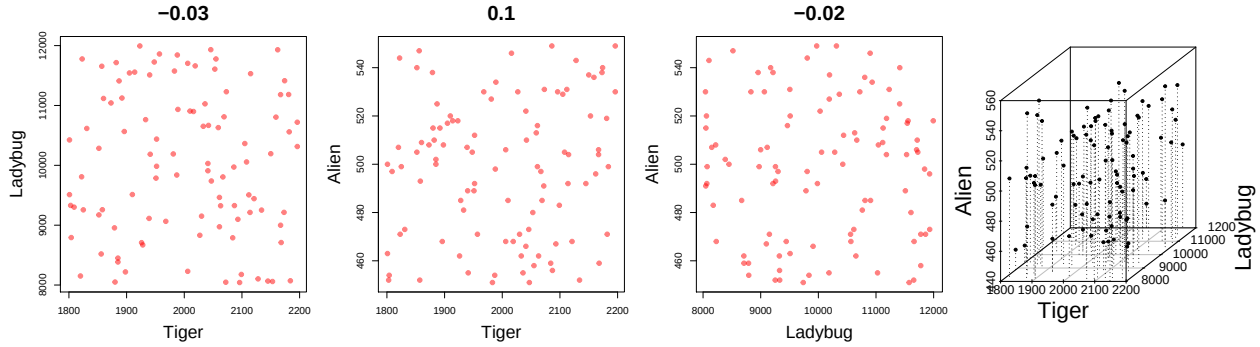

Figure 3: Scatter plots and volume plot of 100 randomly generated tiger, ladybug and space alien counts. It is obvious that there is no correlation between the values, as expected for randomly generated data.

```
if(plot==TRUE) dev.off()
```

s

```
if(plot==TRUE) pdf("tiger_prop.pdf", height=7, width=9)
par(mfrow=c(2,3),mar=c(5,5,4,1))
plot(d.rare$Tiger, d.rare$Ladybug, main=round(cor(d.rare$Tiger, d.rare$Ladybug), 2),
    cex.main=2, pch=19, col=rgb(1,0,0,0.5), cex.lab=1.8, xlab="Tiger", ylab="Ladybug")
plot(d.rare$Tiger, d.rare$Alien, main=round(cor(d.rare$Tiger, d.rare$Alien), 2),
    cex.main=2, pch=19, col=rgb(1,0,0,0.5), cex.lab=1.8, xlab="Tiger", ylab="Alien")
plot(d.rare$Ladybug, d.rare$Alien, main=round(cor(d.rare$Ladybug, d.rare$Alien), 2),
    cex.main=2, pch=19, col=rgb(1,0,0,0.5), cex.lab=1.8, xlab="Ladybug", ylab="Alien")
scatterplot3d(d.rare, type="h", lty.hplot=3, cex.lab=1.5, angle=40, pch=19)
plot(acomp(d.rare), scale=T, center=T, pch=19, cex=1, axes=FALSE)
if(plot==TRUE) dev.off()
```

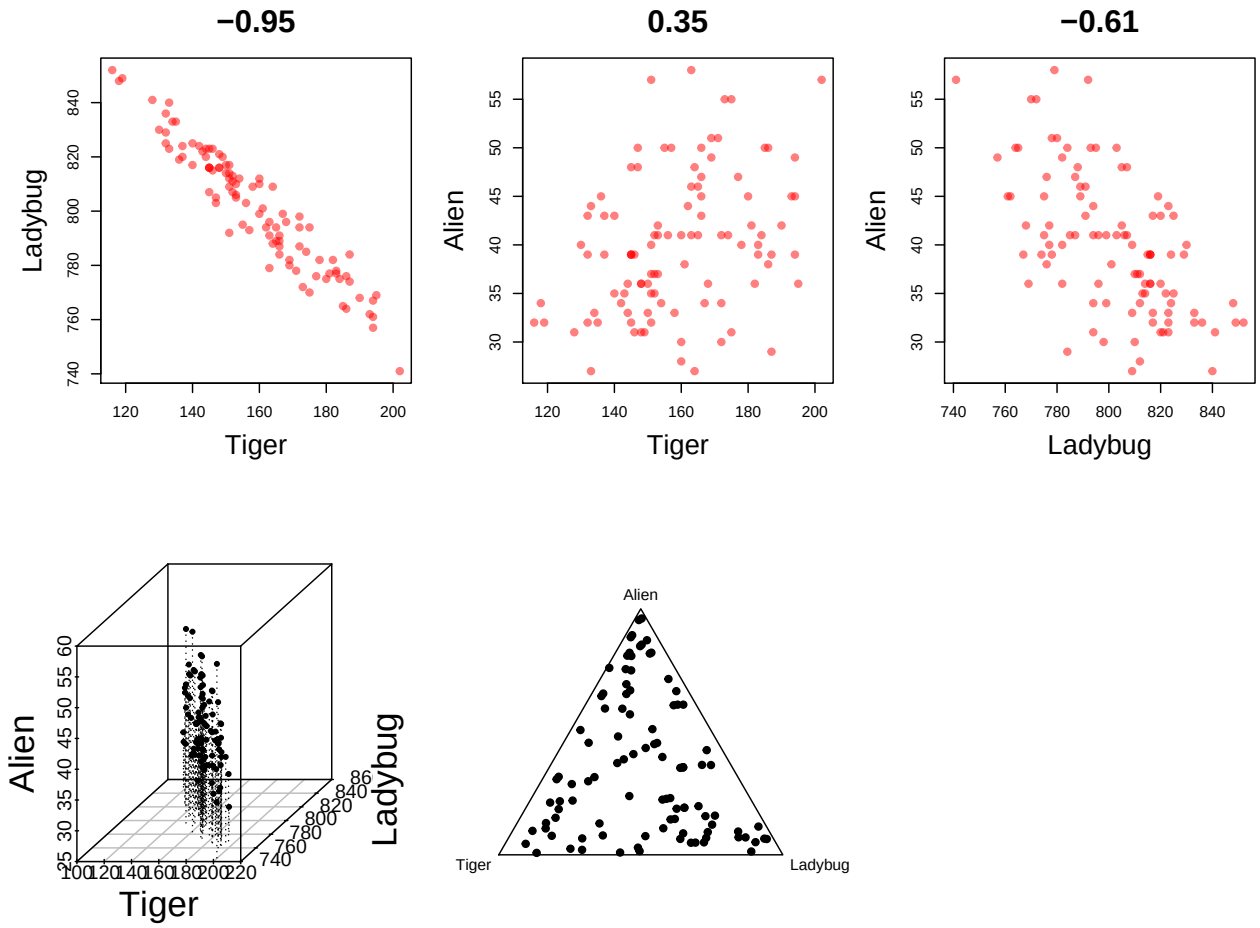

Figure 4: test