Supervised normalisation can bias microbiome analyses

2023-08-03

A preprint was recently posted to bioRxiv challenging the methods and conclusions of this paper in Nature.

This followed on from a recent preprint and rebuttal from the same authors published earlier this year.

Since the most recent preprint, a further rebuttal has now been posted on GitHub.

Before I get into discussing the issue of normalisation in microbiome studies, I would like to start by thanking the authors of both the original study and the countering preprints for their transparency in making the data and the associated analytical code publicly accessible. This open approach to science is critical to accurately assessing the arguments from both perspectives.

This blog post does not attempt to consider all the points made in both the original paper or the countering preprints and rebuttals. Instead, I focus on the issue of controlling for unwanted batch effects and the importance of study design.

Summary

The original paper searched for microbial DNA and RNA in whole-genome and whole-transcriptome sequencing studies from The Cancer Genome Atlas (TCGA) data set which includes 33 types of cancer. The study claimed to identify

"unique microbial signatures in tissue and blood within and between most major types of cancer"

A criticism outlined in the second pre-print by Gihawi, was that:

"errors in the genome database and the associated computational methods led to millions of false positive findings of bacterial reads across all samples, largely because most of the sequences identified as bacteria were instead human"

The authors of the preprint also noted that:

"errors in transformation of the raw data created an artificial signature, even for microbes with no reads detected, tagging each tumor type with a distinct signal that the machine learning programs then used to create an apparently accurate classifier"

While the TCGA study was primarily designed to focus on human genomics and transcriptomics, its design makes it challenging to discern reliable microbial signatures. This is due to the confounding correlations between unintended batch effects—like hospital, sequencing center, and body site—and the primary variable of interest: cancer.

It has been suggested that the contamination identified in the pre-print does not necessarily invalidate the findings as it does not matter if the reads were assigned perfectly as long as a signature was still present. Indeed, I would be surprised if cancer associated human reads were responsible for all the strong signals observed in the original paper.

However, there are many other batch effects in the TCGA data that correlate with particular cancer types. As the authors of the original paper state:

"The possibility of sample contamination during collection, processing, and sequencing limits these investigations, as procedural controls have rarely been implemented in cancer genomics projects."

To address these concerns, the authors of the Nature paper invested significant effort and employed multiple strategies to mitigate unwanted batch effects.

In theory, with a good study design including many control samples, it should be possible to correct for batch effects including human contamination. However, when the there are confounding correlations between batch effects and the variable of interest this becomes very difficult.

In particular, the use of supervised normalisation techniques when such confounding correlations are present is problematic as it can generate artificial signatures.

In a response to the Gihawi et al. pre-print, Poore et al. identified a cancer-specific microbial signature in a subset of the TCGA data, which included three cancer types sequenced at Harvard Medical School. To tackle the issue of normalisation, they utilised the cleaned counts from Gihawi et al., without implementing any normalisation procedures.

While all samples were processed at HMS, they originated from different hospitals and included cancers from different body sites. Thus, it is not clear if the identified signal was due to the cancer type or remaining batch effects.

To investigate this further I wanted to consider three main points

1) **How robust is the supervised normalisation method to confounding batch effects?**
2) **Given the presence of confounding batch effects, can we use the 'normal' (non-cancerous) tissue control samples to account for unwanted batch effects in the original count data?**
3) **If we can, is there still a strong signal of cancer specific microbial signatures?**

1. Supervised normalisation

I have noticed an increasing number of microbiome papers using Supervised Normalisation and in particular the Supervised Normalisation of Microarrays (SNM) method described in Mecham et al., 2010 prior to training machine learning techniques.

As this method is supervised, it includes the variable of interest when controlling for unwanted variation. If care is not taken with the experimental design, this approach will artificially imprint the data with a signal of the variable of interest.

The problem of unintentionally imprinting the data becomes especially critical when there's a correlation between the undesired variables set to be removed and the target variable. For instance, if specific labs or hospitals (the unwanted sources of variation) predominantly sample or sequence a subset of cancers (the variable of interest).

In differential gene expression research, where many of these normalisation techniques originated, it's usually advised to include batch effects as a variable in the Generalised Linear Model (GLM) used to identify differentially expressed genes. The normalised expression values are typically only used to generate quality control plots to assess the influence of different variables, which helps circumvent potential data imprinting issues.

To demonstrate the problem of confounding variables we can simulate artificial microbiome data where there exists unwanted variation but **no** signal associated with the variable of interest.

Initially, we need to load some libraries.

```
library(tidyverse)
library(data.table)
library(scales)
library(caret)
library(splatter)
library(snm)
library(SCRuB)
```

```
library(edgeR)

set.seed(1234)
cols <- c('#e41a1c','#377eb8','#4daf4a','#984ea3')
```

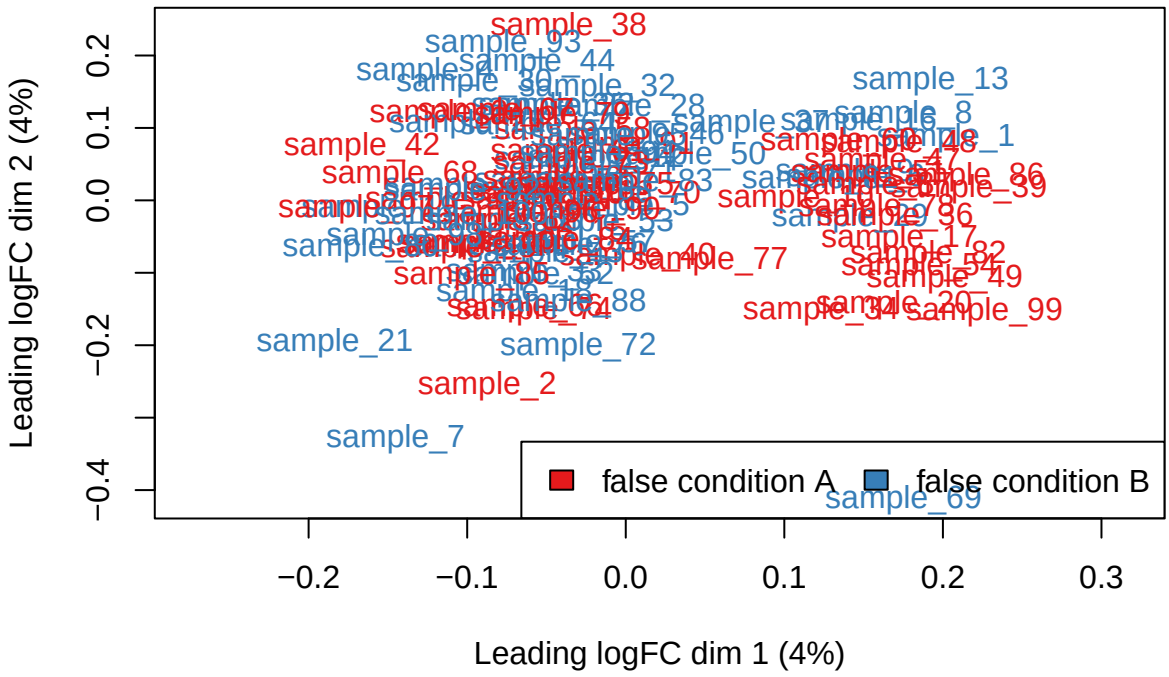
Let's start by generating simulated microbiome compositions using the **splatter** bioconductor package. **Splatter** was originally designed to simulate single cell data which happens to have a similar distribution to microbiome studies.

We'll consider 100 samples with up to 500 species in each sample. Each simulation is independent so there are no underlying groups or clusters of interest present. We add a simulated batch effect of 4 groups split roughly evenly over the 100 samples.

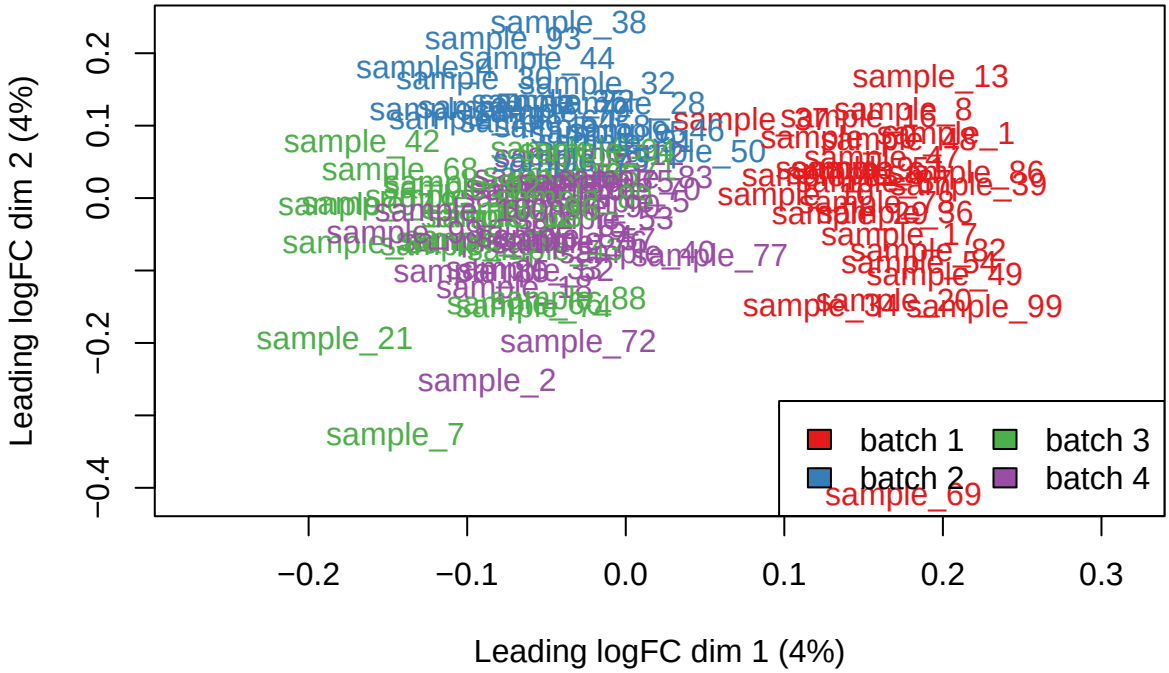
```
params <- newSplatParams(nGenes=500, lib.loc=12, lib.scale=0.5)

sim <- splatter::splatSimulate(group.prob = c(0.25, 0.25, 0.25, 0.25), method = "groups", params = para
                               verbose = FALSE)

count_matrix <- counts(sim)
colnames(count_matrix) <- paste('sample', 1:100, sep="_")
```

Let's arbitrarily divide the data into two groups of interest, potentially representing different cancer types. Since we haven't simulated any differences between these groups, we shouldn't expect them to cluster, which can be verified using an MDS plot.

```
groups <- tibble(
  sample=paste('sample', 1:100, sep="_"),
  batch = sim$Group,
  disease_type=sample(rep(c('a','b'), 50), 100, replace = FALSE) # sample is used to randomise the allo
)

# account for read depth and transform to log space
y <- edgeR::cpm(count_matrix, normalized.lib.sizes = TRUE, log = TRUE, prior.count = 1)

#Plot disease type
plotMDS(y, col=cols[factor(groups$disease_type)])
legend(x = "bottomright", legend = c("false condition A","false condition B"), fill= cols, ncol=2)
```




```
#Plot batch
plotMDS(y, col=cols[factor(groups$batch)])
legend(x = "bottomright", legend = paste("batch", 1:4), fill=cols, ncol=2)
```

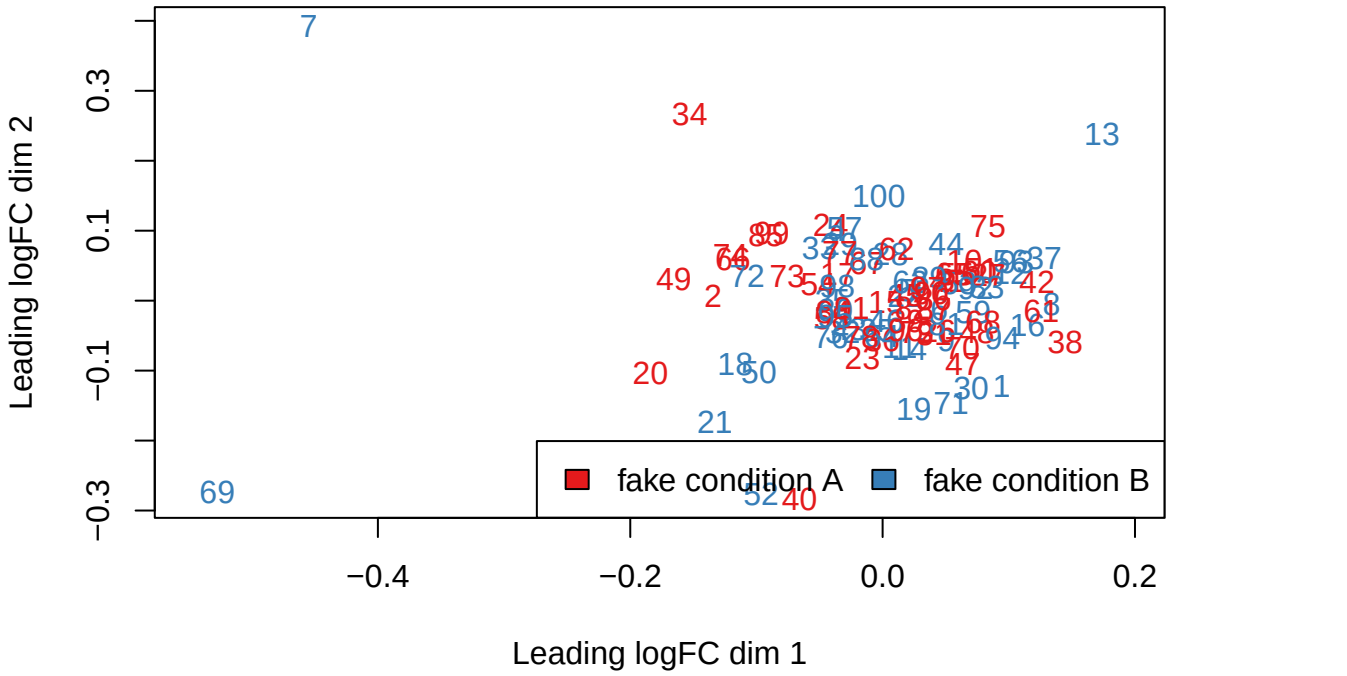


We can now apply the Supervised Normalisation of Microarrays method. A critical aspect of this approach is the incorporation of the variable of interest as an input, enhancing its capacity to preserve the associated signal. However, critically it assumes that **there is not a correlation between the unwanted batch effects and the variable of interest**. Thus, it is essential that the experimental design satisfies this condition. This is rarely the case in microbiome studies.

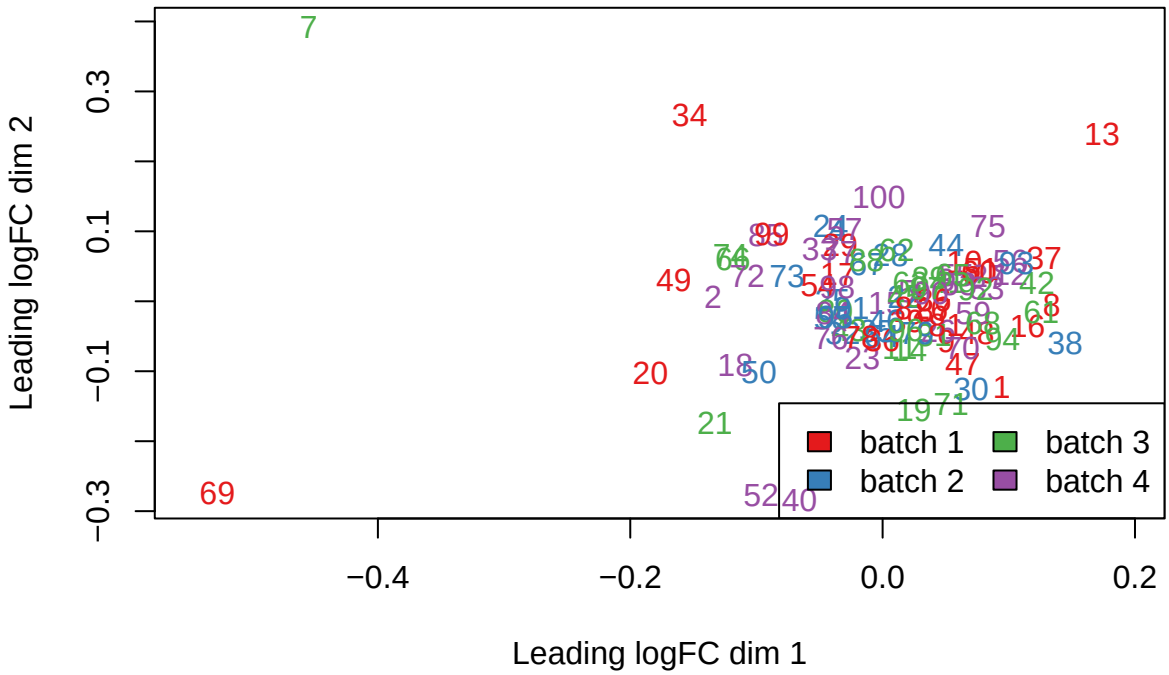
```
disease_matrix <- model.matrix(~disease_type, data=groups)
batch_matrix <- model.matrix(~batch, data=groups)

normalised <- snm(raw.dat = y,
                  bio.var = disease_matrix,
                  adj.var = batch_matrix,
                  rm.adj=TRUE,
                  verbose = TRUE,
                  diagnose = TRUE)
```

Let's plot the results using Multidimensional Scaling (MDS).

```
plotMDS(normalised$norm.dat, col=cols[factor(groups$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = c("fake condition A","fake condition B"), fill= cols, ncol=2)
```




```
plotMDS(normalised$norm.dat, col=cols[factor(groups$batch)],
        var.explained = FALSE)
legend(x = "bottomright", legend = paste("batch", 1:4), fill=cols, ncol=2)
```



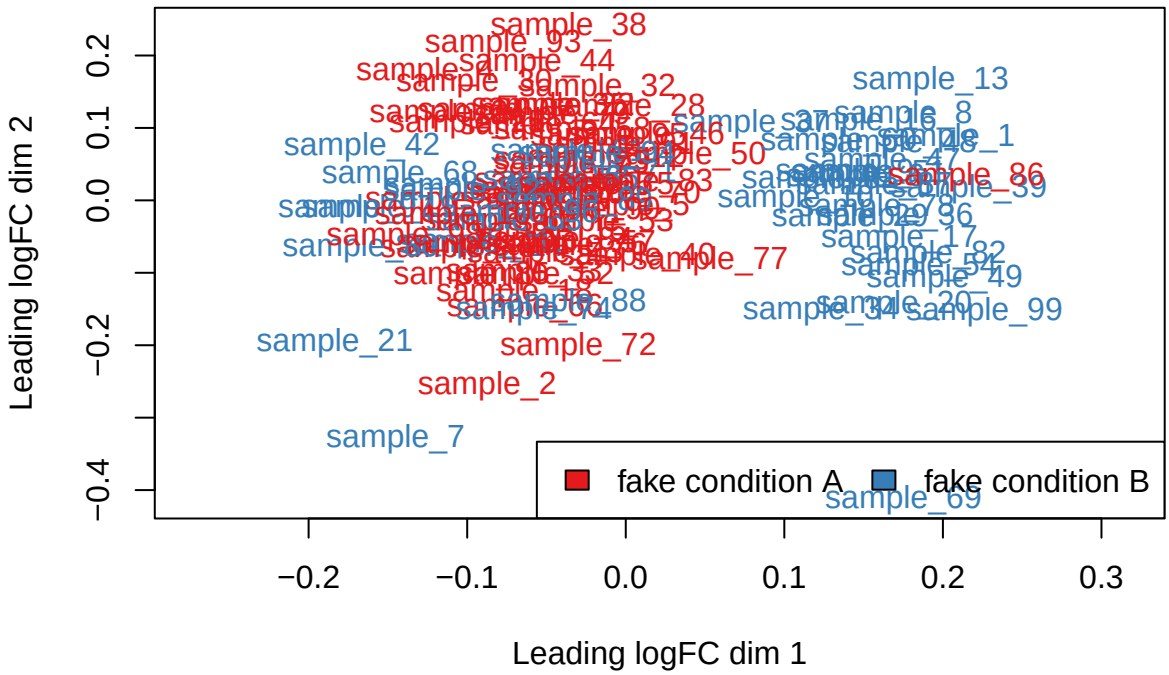
It's handled the batch correction well and has not added any obvious artificial signal. However, things change when we add some correlation between the unwanted variation (batches) and the fake variable of interest.

To add some correlation between the groups and the 'fake' variable of interest, we can increase the probability a sample is of a particular cancer type if occurs within a subset of batches.

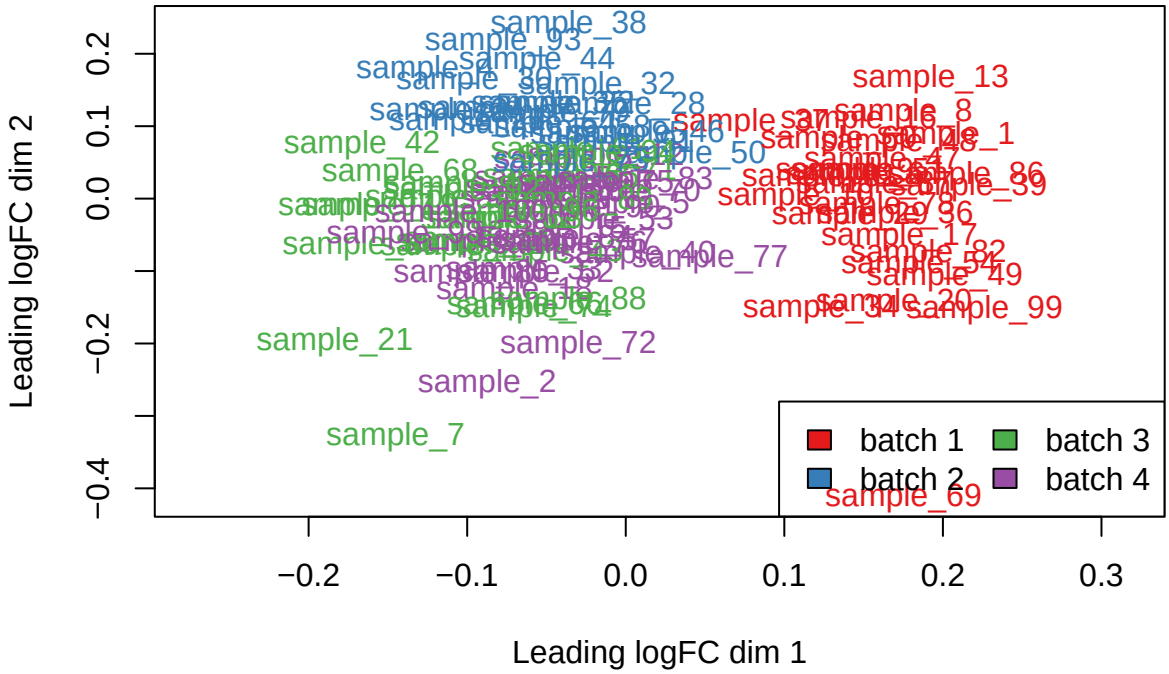
```
# a sample is 5 times more likely to be in condition 'b' if it belongs to an even batch number
fake_disease_type <- purrr::map_chr(as.numeric(sim$Group),
                                    ~ sample(c('a','b'), size = 1, prob = c(1, 5*(.x %% 2))))

groups <- tibble(
  sample=paste('sample', 1:100, sep="_"),
  batch = sim$Group,
  disease_type=fake_disease_type
)

disease_matrix <- model.matrix(~disease_type, data=groups)
batch_matrix <- model.matrix(~batch, data=groups)

# account for read depth and transform to log space
y <- edgeR::cpm(count_matrix, normalized.lib.sizes = TRUE, log = TRUE, prior.count = 1)
```

We can now check the un-normalised plots which look similar to before.

```
plotMDS(y, col=cols[factor(groups$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = c("fake condition A","fake condition B"), fill= cols, ncol=2)
```



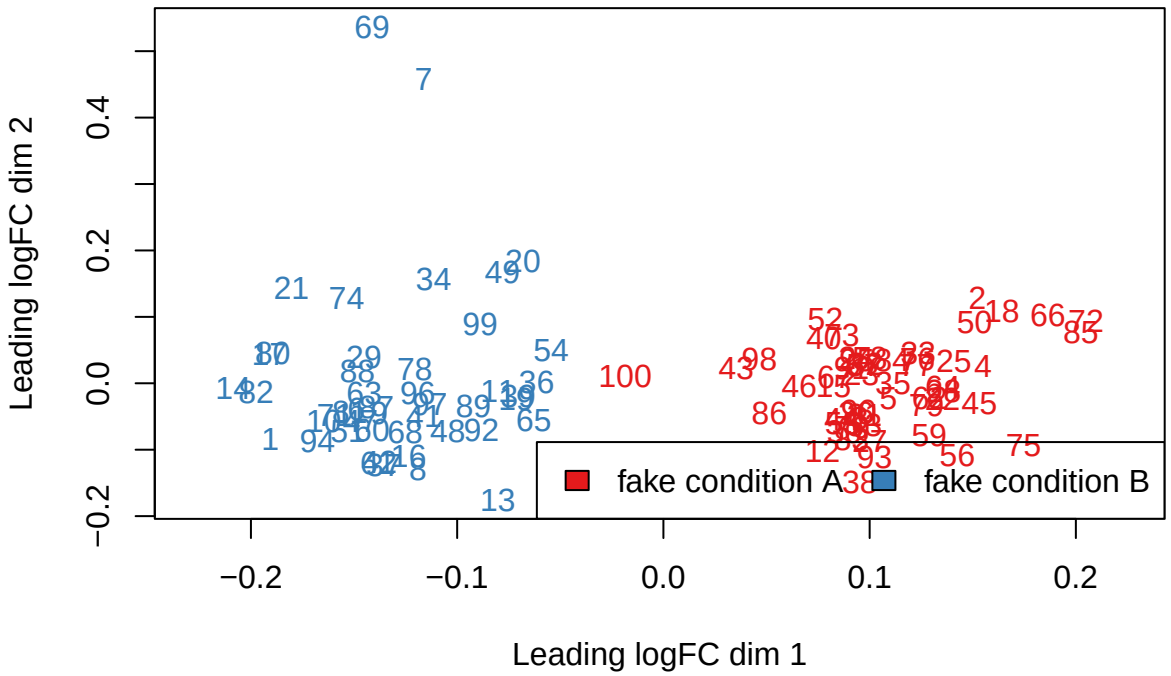

```
plotMDS(y, col=cols[factor(groups$batch)],
        var.explained = FALSE)
legend(x = "bottomright", legend = paste("batch", 1:4), fill=cols, ncol=2)
```



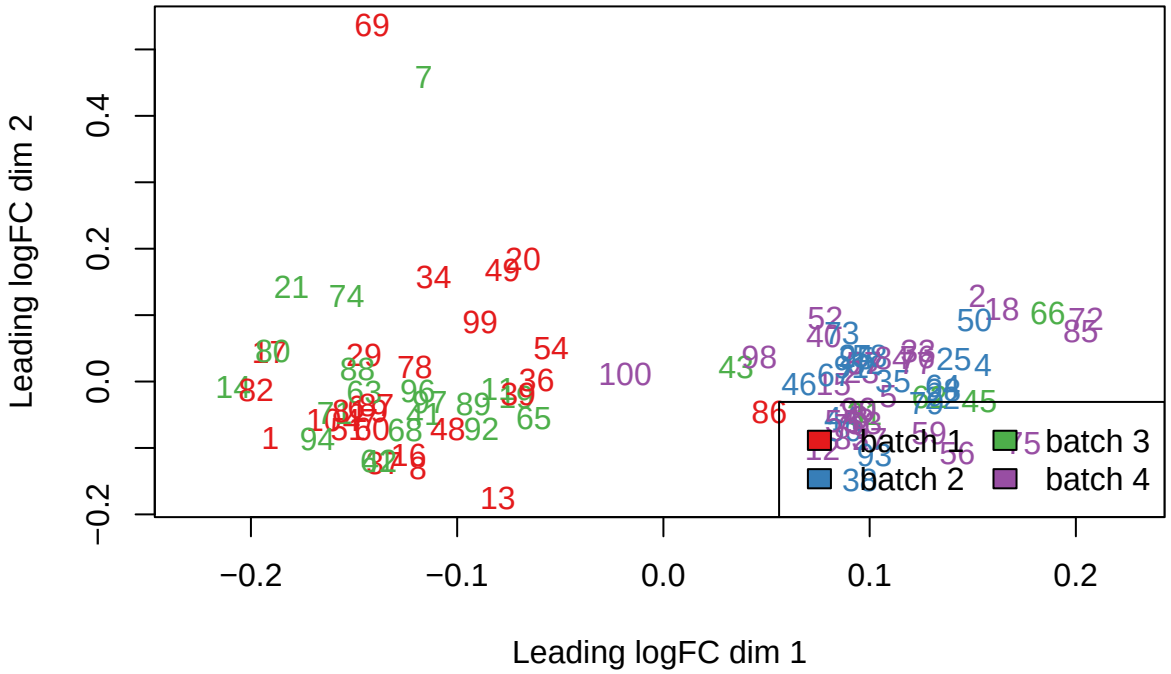
However, after normalisation, the data no longer clusters by batch. Instead, we see an unexpected clustering based on our simulated disease type!

```
normalised <- snm(raw.dat = y,
                  bio.var = disease_matrix,
                  adj.var = batch_matrix,
                  rm.adj=TRUE,
```

```
                    verbose = TRUE,
                    diagnose = TRUE)

plotMDS(normalised$norm.dat, col=cols[factor(groups$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = c("fake condition A","fake condition B"), fill= cols, ncol=2)
```




```
plotMDS(normalised$norm.dat, col=cols[factor(groups$batch)],
        var.explained = FALSE)
legend(x = "bottomright", legend = paste("batch", 1:4), fill=cols, ncol=2)
```



If we feed this imprinted data into a machine learning classifier, it will artificially increase it's accuracy and incorrectly identify a signal associated with the fake variable of interest.

2: Accounting for unwanted variation in the TCGA data

Knowing that the supervised normalisation of microbiome data is problematic, I was interested in examining the signals present within the TCGA data and whether it was possible to account for the unwanted batch effects without imprinting the data with an artificial signal.

Although, the contamination with human reads has received a lot of attention, it seems unlikely that mutations in human genomes associated with cancer would result in sufficiently different contamination profiles to distinguish all cancer types. It should be possible to account for such contamination in the normalisation stage if suitable control samples are available.

To investigate this further, I was interested in whether the observed cancer correlations remained after we accounted for microbial variation observed in the corresponding tissue normal samples for each site.

These 'normal' samples are not an ideal control. If cancer associated bacteria are also present in the normal tissue this would remove such a signal. Ideally, we would have a corresponding set of body site specific normal tissue from patients without cancer. However, by using the normal tissue samples from the TCGA dataset we should be able to account for many batch effects including whether the observed correlations could simply be driven by site specific bacteria.

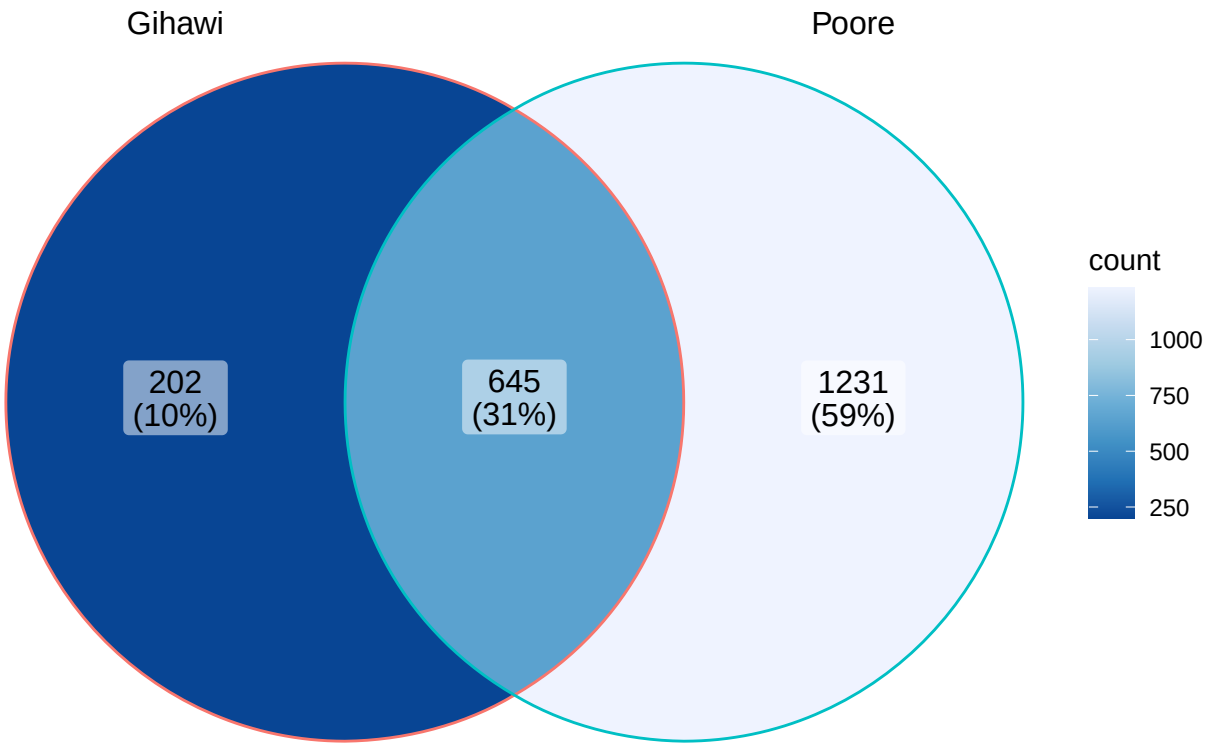
To start, lets load the original un-filtered count data from the original publication and the cleaned count data produced by Gihawi et al. in their recent preprint.

```
# Original counts with and without normalisation from Poore et al., 2020 https://www.nature.com/article
orignal_counts_raw <- read_csv("./Kraken-TCGA-Raw-Data-All-18116-Samples.csv")
colnames(orignal_counts_raw)[[1]] <- "Sample"
colnames(orignal_counts_raw) <- gsub(".*g__","",colnames(orignal_counts_raw))

meta_data <- read_csv("./tcga_metadata_poore_et_al_2020_1Aug23.csv")

meta_extra <- read_csv("./Metadata-TCGA-All-18116-Samples.csv")
colnames(meta_extra)[[1]] <- "Sample"

meta_data$gender <- meta_extra$gender[match(meta_data$sampleid, meta_extra$Sample)]
meta_data$platform <- meta_extra$platform[match(meta_data$sampleid, meta_extra$Sample)]
meta_data$portion_ffpe <- meta_extra$portion_is_ffpe[match(meta_data$sampleid, meta_extra$Sample)]
meta_data$experimental_strategy <- meta_extra$experimental_strategy[match(meta_data$sampleid, meta_extra
meta_data$tissue_source_site_label <- meta_extra$tissue_source_site_label[match(meta_data$sampleid, meta


#Load the updated counts from Gihawi et al., bioRxiv 2023.  https://github.com/yge15/Cancer_Microbiome_
gihawi_counts <- map(c("TableS8_BLCA.all.csv", "TableS9_HNSC_all.csv", "TableS10_BRCA_WGS.csv"), ~ {
                      df <- read_csv(.x)
                     colnames(df)[[1]] <- "Sample"
                     colnames(df) <- gsub("^g_","",colnames(df))
                     return(df)
                    })
common_species <- Reduce(intersect, map(gihawi_counts, colnames))
common_species <- common_species[common_species!="Homo"] # remove human reads

gihawi_counts <- map_dfr(gihawi_counts, ~ .x[,common_species])

#subset metadata and samples
```

```
meta_data <- meta_data[meta_data$sampleid %in% gihawi_counts$Sample,]
orignal_counts_raw <- orignal_counts_raw[orignal_counts_raw$Sample %in% gihawi_counts$Sample, ]
orignal_counts_raw <- orignal_counts_raw[, c(TRUE, colSums(orignal_counts_raw[,2:ncol(orignal_counts_raw
)

# Order all matrices to match the meta table
gihawi_counts <- gihawi_counts[match(meta_data$sampleid, gihawi_counts$Sample),]
orignal_counts_raw <- orignal_counts_raw[match(meta_data$sampleid, orignal_counts_raw$Sample),]
```

We can consider the intersection between the species observed in the original un-filtered data and the cleaned set from the preprint.

```
ggVennDiagram::ggVennDiagram(list(Gihawi=colnames(gihawi_counts)[-1],
                                  Poore=colnames(orignal_counts_raw)[-1])) +
  scale_fill_distiller(palette = 1)
```



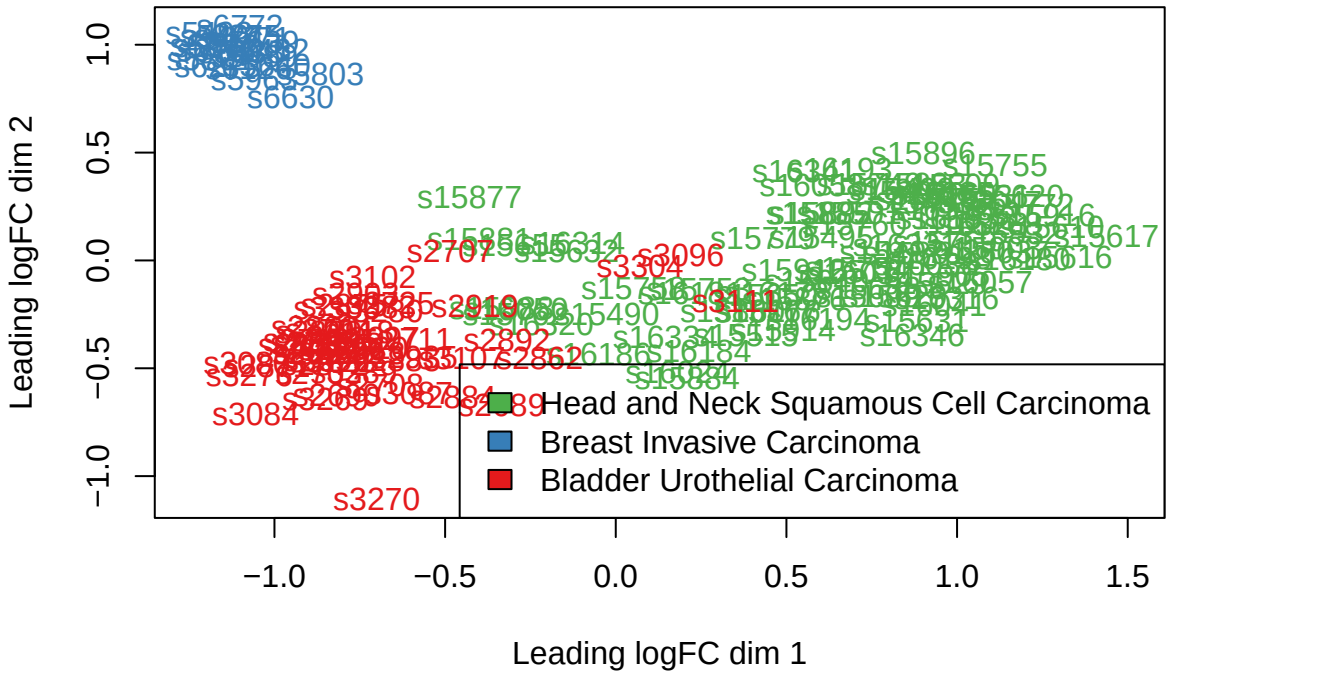
2.1 Analysis of Gihawi et al. filtered counts

In a rebuttal to the preprint of Gihawi, it was argued that using the updated counts of Gihawi et al., without normalisation still resulted in clustering by cancer type in samples sequenced at Harvard Medical School.

Indeed, if we create an MDS plot of this data we observe quite clear clusters.

```
filt_meta <- meta_data %>%
  filter(!grepl(".*Normal", sample_type)) %>%
  filter(data_submitting_center_label=="Harvard Medical School")

filt_counts <- gihawi_counts[gihawi_counts$Sample %in% filt_meta$sampleid, ]

stopifnot(all(filt_counts$Sample==filt_meta$sampleid))

filt_counts <- t(filt_counts[,-1])
```

```
colnames(filt_counts) <- filt_meta$sampleid

filt_counts <- filt_counts[, colSums(filt_counts)>100]
filt_meta <- meta_data[meta_data$sampleid %in% colnames(filt_counts), ]

y <- edgeR::cpm(filt_counts, normalized.lib.sizes = TRUE, log = TRUE, prior.count = 1)

plotMDS(y, col=cols[factor(filt_meta$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = unique(filt_meta$disease_type), fill=rev(cols[1:3]), ncol=1)
```



However, while the sequencing center is the same, other potential batch effects could be driving the signal including the hospital the samples originated from and the body site the cancer was located in.

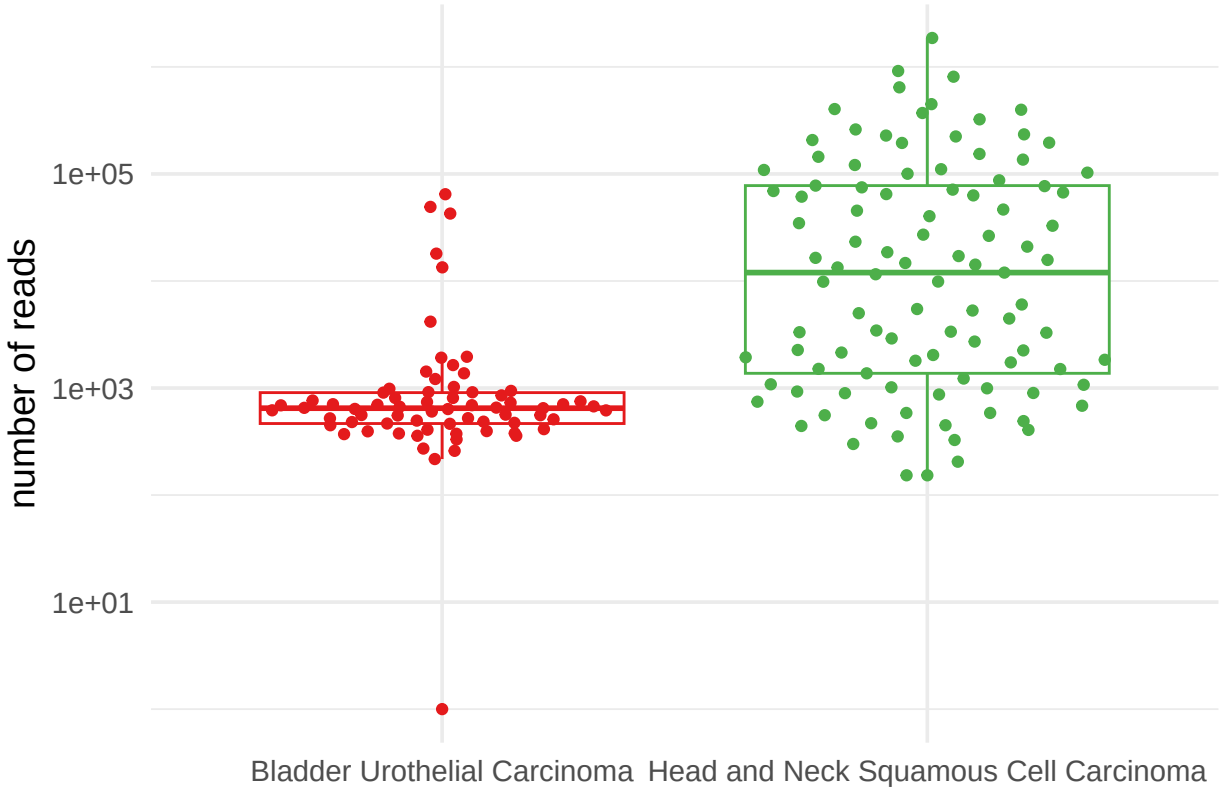
One option to account for these sources of unwanted variation is to use the associated non-cancer normal tissue samples as controls. We can use the excellent SCRuB decontamination method to do this.

As there are no normal tissue samples for the breast cancer samples at Harvard Medical School we restrict this analysis to the remaining two cancer types.

First, let's check the number of remaining reads in each sample

```
filt_meta <- meta_data %>%
  filter(data_submitting_center_label=="Harvard Medical School") %>%
  filter(sample_type!="Blood Derived Normal")

# split into normal and cancerous tissue samples
filt_meta$normal <- grepl(".*Normal", filt_meta$sample_type)

# Keep those where we have both tumor and normal tissue samples.
keep <- filt_meta %>%
  group_by(investigation, data_submitting_center_label) %>%
  dplyr::summarise(
    normal_and_cancer = length(unique(normal))>1,
```

```
    n_normal = sum(normal),
    n_cancer = sum(!normal)
  ) %>% filter(normal_and_cancer)

filt_meta <- filt_meta %>%
  filter(paste(investigation, data_submitting_center_label) %in% paste(keep$investigation, keep$data_su

filt_counts <- gihawi_counts[gihawi_counts$Sample %in% filt_meta$sampleid, ]
stopifnot(all(filt_counts$Sample==filt_meta$sampleid))

filt_counts <- t(filt_counts[,-1])
colnames(filt_counts) <- filt_meta$sampleid

pdf <- tibble(
  `cancer type`=filt_meta$disease_type,
  `number of reads` = colSums(filt_counts)
)

ggplot(pdf, aes(x=`cancer type`, y=`number of reads`, colour=`cancer type`)) +
  geom_boxplot(outlier.colour = NA) +
  ggbeeswarm::geom_quasirandom() +
  scale_y_log10() +
  scale_color_manual(values = cols[-2]) +
  theme_minimal(base_size = 14) +
  theme(legend.position = 'none') +
  xlab("")
```

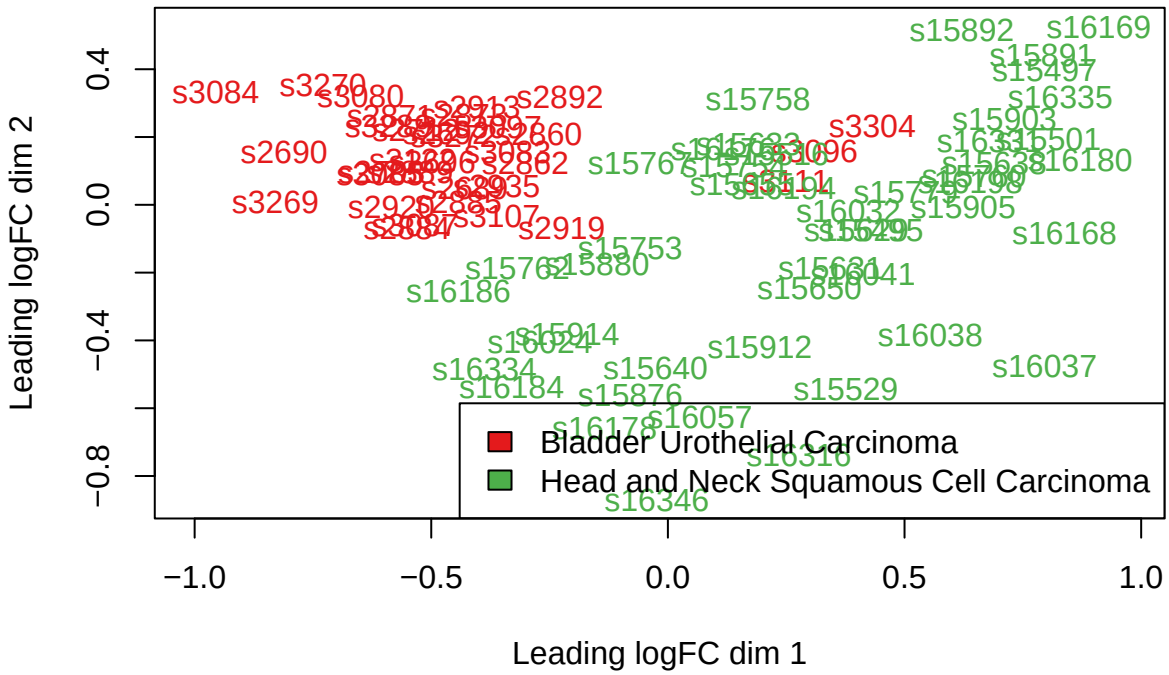


There are some very different library sizes depending on the cancer type. While it might be possible to

account for some of this by normalising for library size, it is unlikely to account for the increased number of potential species that could be observed at much higher read depths.

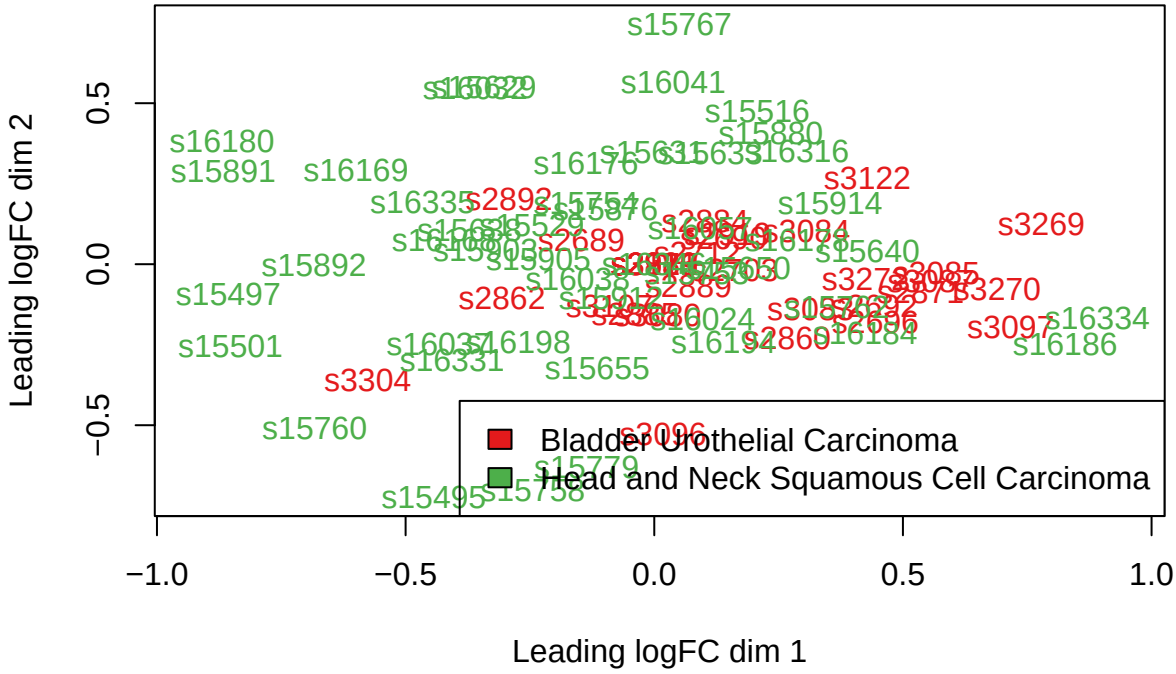
Instead, we restrict out analysis to samples with similar read counts of between 500 and 50,000 reads.

```
# Throw out samples with < 100 reads
filt_counts <- filt_counts[, (colSums(filt_counts)>500) & (colSums(filt_counts)<50000)]
filt_meta <- filt_meta[filt_meta$sampleid %in% colnames(filt_counts), ]

# Run the SCRuB algorithm on each cancer type
scrub_counts <- purrr::map2(keep$data_submitting_center_label, keep$investigation, ~{
  # print(paste(.x, .y, sep = " : "))

  k <- (filt_meta$investigation==.y) & (filt_meta$data_submitting_center_label==.x)

  case_counts <- t(filt_counts[, k & !filt_meta$normal])
  control_counts <- t(filt_counts[, k & filt_meta$normal])

  new_meta <- filt_meta[k,] %>%
    filter(!normal)

  # Account for site specific signatures using SCruB
  norm_counts <- t(SCRUB_no_spatial(case_counts, control_counts)$decontaminated_samples)

  stopifnot(all(colnames(norm_counts)==new_meta$sampleid))

  return(list(norm_counts=norm_counts, meta=new_meta))
})

scrub_meta <- map_dfr(scrub_counts, ~ .x$meta)
scrub_counts <- do.call(cbind, purrr::map(scrub_counts, ~ .x$norm_counts))

# Account for remaining unwanted variables including gender and the hospital where the sample was taken
y <- edgeR::cpm(scrub_counts, normalized.lib.sizes = TRUE, log = TRUE, prior.count = 1)

plotMDS(y, col=cols[-2][factor(scrub_meta$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = unique(scrub_meta$disease_type), fill=cols[-2], ncol=1)
```



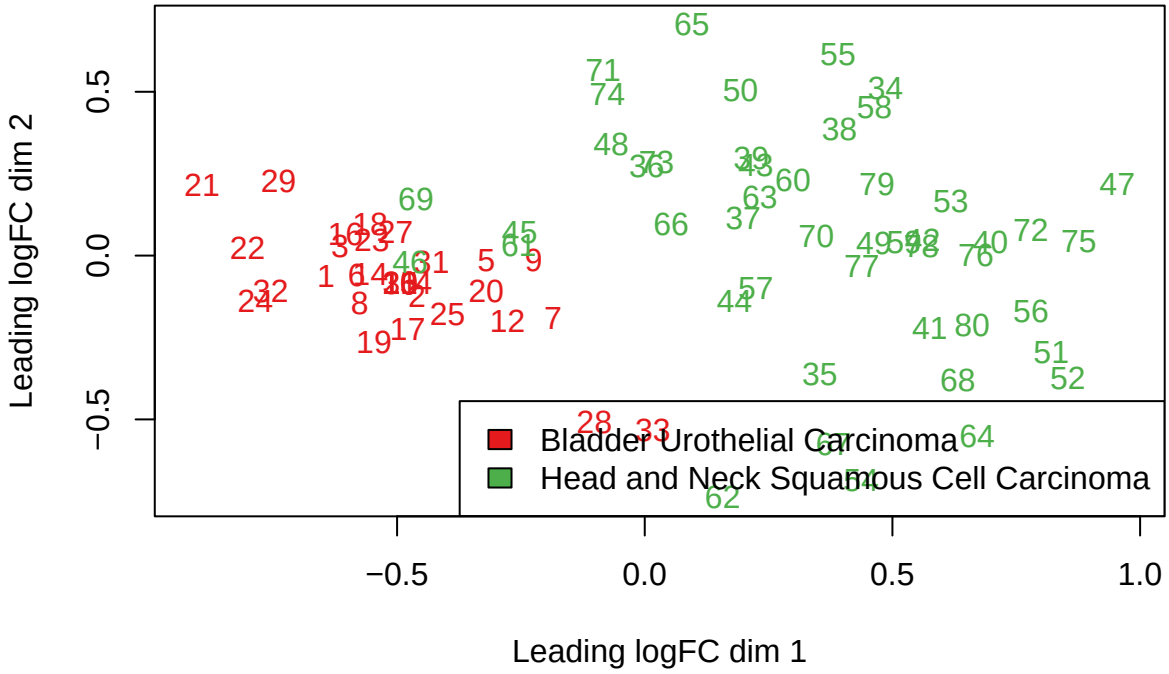
Once body site-specific microbial signatures are accounted for, the distinction between cancer types diminishes. Furthermore, if we factor in the hospital from which samples were collected—a potential source of contamination—no obvious signal remains.

```
y <- removeBatchEffect(y, batch = scrub_meta$tissue_source_site_label)

plotMDS(y, col=cols[-2][factor(scrub_meta$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = unique(scrub_meta$disease_type), fill=cols[-2], ncol=1)
```



Due to the correlation between hospital and cancer type, if we use Supervised Normalisation to account for hospital variation we **artificially** increase the separation between the two cancer types.

```
y <- removeBatchEffect(y, batch = scrub_meta$tissue_source_site_label)

disease_matrix <- model.matrix(~investigation, data=scrub_meta)
batch_matrix <- model.matrix(~tissue_source_site_label, data=scrub_meta)

normalised <- snm(raw.dat = y,
                  bio.var = disease_matrix,
                  adj.var = batch_matrix,
                  rm.adj=TRUE,
                  verbose = TRUE,
                  diagnose = TRUE)

plotMDS(normalised$norm.dat, col=cols[-2][factor(scrub_meta$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = unique(scrub_meta$disease_type), fill=cols[-2], ncol=1)
```



So far, we have shown that Supervised Normalisation can artificially imprint a data set with a signal if there is a correlation between the variable of interest and unwanted batch effects.

After employing rigorous normalisation methods to mitigate potential influences from body site and hospital batch effects, the distinction between Bladder Urothelial Carcinoma and Head and Neck Squamous Cell Carcinoma isn't evident. While this doesn't rule out unique microbial signatures in these samples, the TCGA study design complicates the task of detecting a robust signal.

2.2 Analysis of the original Poore et al. read counts

As we can use alternative normalisation strategies to account for batch effects within the TCGA data, I was interested in whether these would work on the original count data from the Poore et al., paper.

This presupposes that any cancer-associated human contaminant reads, as identified in the Gihawi et al. preprint, are unlikely to align with different bacterial species compared to non-cancerous human reads.

To investigate this, I started by considering the same samples sequenced at Harvard Medical School.

```r
filt_meta <- meta_data %>%
  filter(data_submitting_center_label=="Harvard Medical School") %>%
  filter(sample_type!="Blood Derived Normal")

# split into normal and cancerous tissue samples
filt_meta$normal <- grepl(".*Normal", filt_meta$sample_type)

# Keep those where we have both tumor and normal tissue samples.
keep <- filt_meta %>%
  group_by(investigation, data_submitting_center_label) %>%
  dplyr::summarise(
    normal_and_cancer = length(unique(normal))>1,
    n_normal = sum(normal),
    n_cancer = sum(!normal)
  ) %>% filter(normal_and_cancer)

filt_meta <- filt_meta %>%
  filter(paste(investigation, data_submitting_center_label) %in% paste(keep$investigation, keep$data_su

filt_counts <- orignal_counts_raw[orignal_counts_raw$Sample %in% filt_meta$sampleid, ]
stopifnot(all(filt_counts$Sample==filt_meta$sampleid))

filt_counts <- t(filt_counts[,-1])
colnames(filt_counts) <- filt_meta$sampleid

filt_meta <- filt_meta[filt_meta$sampleid %in% colnames(filt_counts), ]

# Run the SCRuB algorithm on each cancer type
scrub_counts <- purrr::map2(keep$data_submitting_center_label, keep$investigation, ~{
  # print(paste(.x, .y, sep = " : "))

  k <- (filt_meta$investigation==.y) & (filt_meta$data_submitting_center_label==.x)

  case_counts <- t(filt_counts[, k & !filt_meta$normal])
  control_counts <- t(filt_counts[, k & filt_meta$normal])

  new_meta <- filt_meta[k,] %>%
    filter(!normal)

  # Account for site specific signatures using SCruB
  norm_counts <- t(SCRUB_no_spatial(case_counts, control_counts)$decontaminated_samples)

  stopifnot(all(colnames(norm_counts)==new_meta$sampleid))

  return(list(norm_counts=norm_counts, meta=new_meta))
})

scrub_meta <- map_dfr(scrub_counts, ~ .x$meta)
scrub_counts <- do.call(cbind, purrr::map(scrub_counts, ~ .x$norm_counts))
```

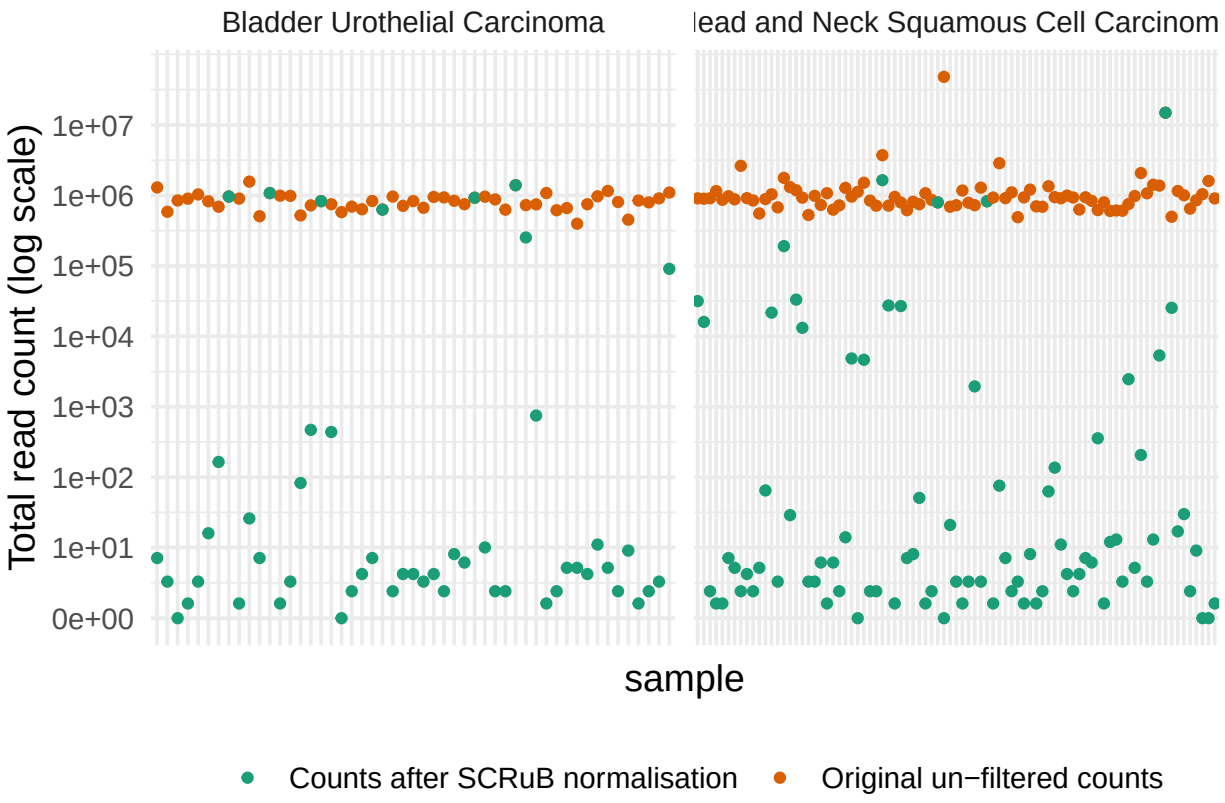
As the read counts are now much higher and similar between cancer types we do not need to filter out samples by read depth prior to running SCRuB.

However, the SCRuB algorithm now assigns the vast majority of read counts to potential contamination.

This demonstrates that using good control samples when performing normalisation and batch correction can account for even large issues with contamination, as long as that contamination is not correlated with the variable of interest.

We can look at the difference in read counts before and after using a log-scaled plot.

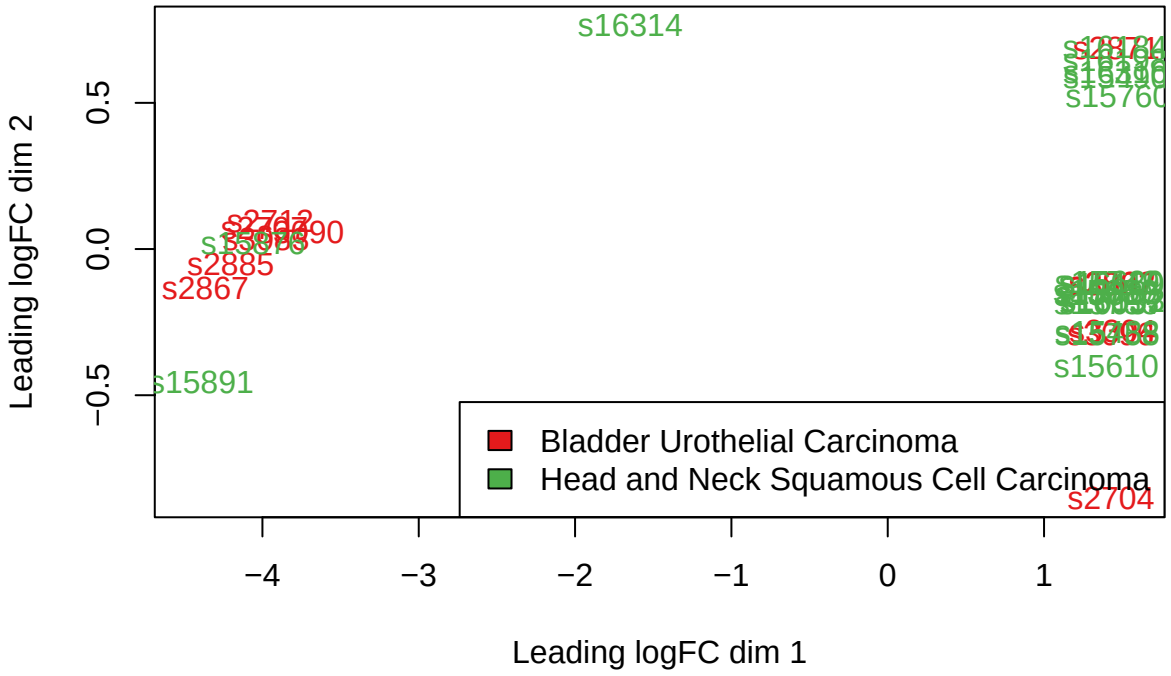
```
pdf <- tibble(
       sample=colnames(scrub_counts),
       center=scrub_meta$data_submitting_center_label,
       cancer_type=scrub_meta$disease_type,
       `Original un-filtered counts`=colSums(filt_counts)[match(colnames(scrub_counts), colnames(filt_c
       `Counts after SCRuB normalisation`=colSums(scrub_counts)) %>%
  pivot_longer(cols=c("Original un-filtered counts","Counts after SCRuB normalisation"))

ggplot(pdf, aes(x=sample, y=value, colour=name)) +
  geom_point() +
  # scale_y_log10() +
  scale_y_continuous(trans=scales::pseudo_log_trans(base = 10),
                     breaks=c(0, 10, 100,1000,1e4,1e5,1e6,1e7)) +
  facet_wrap(~cancer_type, scales = "free_x") +
  scale_color_brewer(type = 'q', palette = 2) +
  theme_minimal(base_size = 14) +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        legend.position="bottom",
        legend.title=element_blank()) +
  ylab("Total read count (log scale)")
```



After filtering out signals present in the tissue normal samples, some samples have very low read depths making them unsuitable for further analysis. After filtering these out, we can look at how the remaining

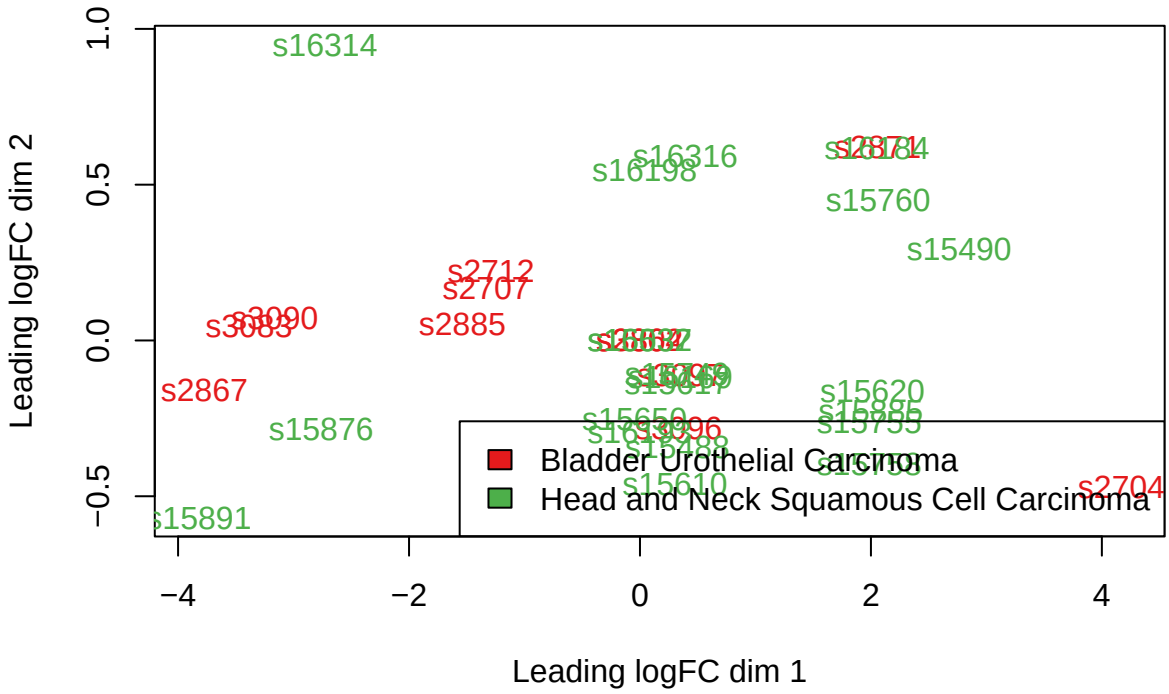
samples cluster together.

```
# Throw out samples with < 100 reads
scrub_counts <- scrub_counts[, colSums(scrub_counts)>100]
scrub_meta <- scrub_meta[scrub_meta$sampleid %in% colnames(scrub_counts), ]

# Account for remaining unwanted variables including gender and the hospital where the sample was taken
y <- edgeR::cpm(scrub_counts, normalized.lib.sizes = TRUE, log = TRUE, prior.count = 1)

plotMDS(y, col=cols[-2][factor(scrub_meta$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = unique(scrub_meta$disease_type), fill=cols[-2], ncol=1)
```



There is no longer any clustering by cancer type, but there does appear to be some clustering driven by a remaining batch effect. Accounting for the hospital removes this effect.

```
y <- removeBatchEffect(y, batch = scrub_meta$tissue_source_site_label)

plotMDS(y, col=cols[-2][factor(scrub_meta$disease_type)],
        var.explained = FALSE)
legend(x = "bottomright", legend = unique(scrub_meta$disease_type), fill=cols[-2], ncol=1)
```



While it is always better to start with cleaner data, this analysis demonstrates that a careful use of normalisation and batch correction techniques can help to correct for even large amounts of contamination.

As we have shown that we can account for most unwanted variation in the subset of samples from Harvard, even when using the original un-filtered count data, we can now investigate the signals present in the larger dataset originally published by Poore et al.

3. Re-analysis of Poore et al., across hospital sites and cancer types

To start, I load the full unfiltered TCGA data from the Poore et al., manuscript.

To streamline the analysis and avoid some undesired variables, I've limited the re-analysis to WGS samples sequenced using the Illumina HiSeq.

In order to implementation of the SCRuB algorithm, I further restrict the analysis to consider only those cancer type and sequencing institute pairings that have a minimum of three cancer tissue samples and three normal tissue control samples.

This results in a data set of 1,809 cancer samples across 16 cancer types.

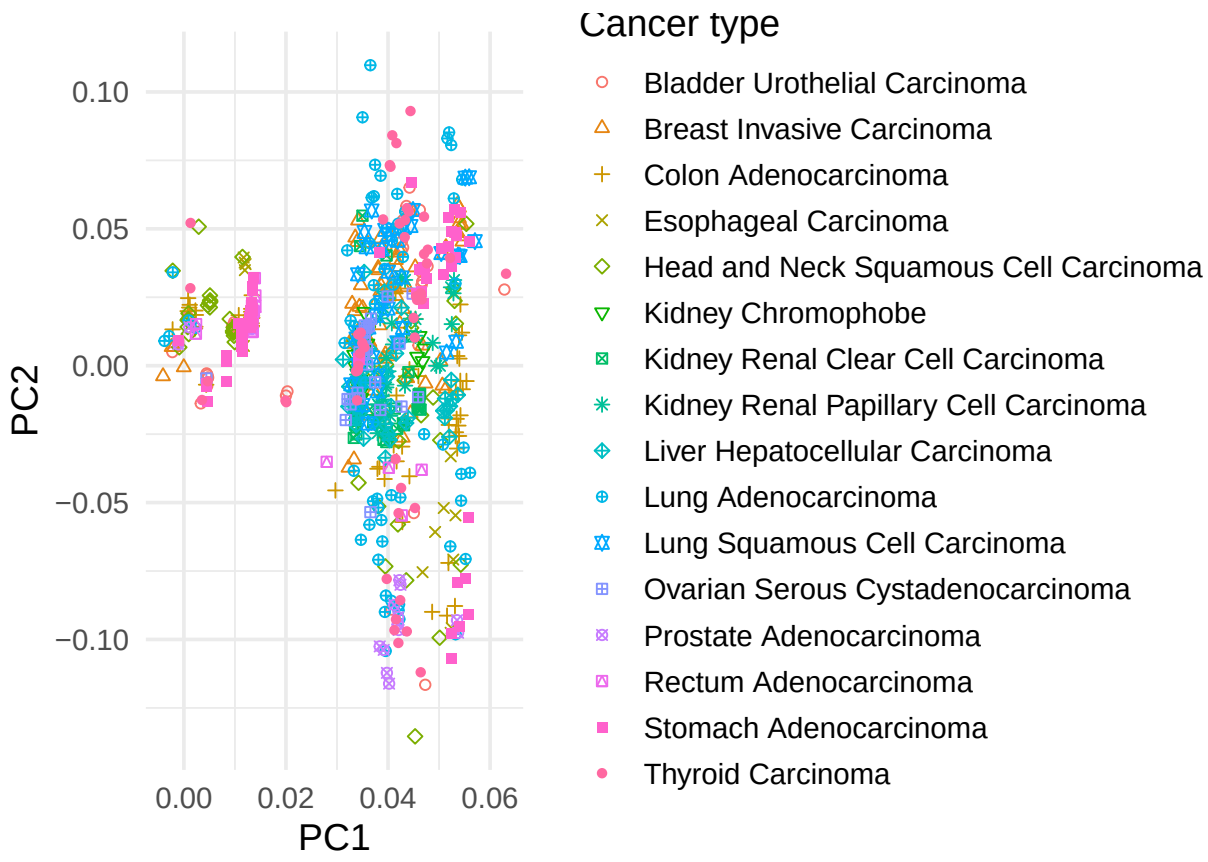
```
meta_data <- read_csv("./tcga_metadata_poore_et_al_2020_1Aug23.csv")
meta_extra <- read_csv("./Metadata-TCGA-All-18116-Samples.csv")
colnames(meta_extra)[[1]] <- "Sample"

meta_data$gender <- meta_extra$gender[match(meta_data$sampleid, meta_extra$Sample)]
meta_data$platform <- meta_extra$platform[match(meta_data$sampleid, meta_extra$Sample)]
meta_data$portion_ffpe <- meta_extra$portion_is_ffpe[match(meta_data$sampleid, meta_extra$Sample)]
meta_data$experimental_strategy <- meta_extra$experimental_strategy[match(meta_data$sampleid, meta_extra
meta_data$tissue_source_site_label <- meta_extra$tissue_source_site_label[match(meta_data$sampleid, meta

orignal_counts_raw <- read_csv("./Kraken-TCGA-Raw-Data-All-18116-Samples.csv")
colnames(orignal_counts_raw)[[1]] <- "Sample"
colnames(orignal_counts_raw) <- gsub(".*g__","",colnames(orignal_counts_raw))
```

```r
meta_data <- meta_data[match(orignal_counts_raw$Sample, meta_data$sampleid), ]

filt_meta <- meta_data %>%
  filter(sample_type!="Blood Derived Normal") %>%
  filter(platform=='Illumina HiSeq') %>%
  filter(portion_ffpe=='NO') %>%
  filter(experimental_strategy=='WGS')

# split into normal and cancerous tissue samples
filt_meta$normal <- grepl(".*Normal", filt_meta$sample_type)

# Keep those where we have both tumor and at least 3 normal tissue samples.
keep <- filt_meta %>%
  group_by(investigation, data_submitting_center_label) %>%
  dplyr::summarise(
    normal_and_cancer = length(unique(normal))>1,
    n_normal = sum(normal),
    n_cancer = sum(!normal)
  ) %>%
  filter(n_normal>3) %>%
  filter(n_cancer>3)

filt_meta <- filt_meta %>%
  filter(paste(investigation, data_submitting_center_label) %in% paste(keep$investigation, keep$data_su

filt_counts <- orignal_counts_raw[orignal_counts_raw$Sample %in% filt_meta$sampleid, ]
stopifnot(all(filt_counts$Sample==filt_meta$sampleid))

filt_counts <- t(filt_counts[,-1])
colnames(filt_counts) <- filt_meta$sampleid

filt_meta <- filt_meta[filt_meta$sampleid %in% colnames(filt_counts), ]

# Run the SCRuB algorithm on each cancer type
scrub_counts <- purrr::map2(keep$data_submitting_center_label, keep$investigation, ~{
  # print(paste(.x, .y, sep = " : "))

  k <- (filt_meta$investigation==.y) & (filt_meta$data_submitting_center_label==.x)

  case_counts <- t(filt_counts[, k & !filt_meta$normal])
  control_counts <- t(filt_counts[, k & filt_meta$normal])

  new_meta <- filt_meta[k,] %>%
    filter(!normal)

  # Account for site specific signatures using SCruB
  norm_counts <- t(SCRUB_no_spatial(case_counts, control_counts)$decontaminated_samples)

  stopifnot(all(colnames(norm_counts)==new_meta$sampleid))

  return(list(norm_counts=norm_counts, meta=new_meta))
})
```

```
scrub_meta <- map_dfr(scrub_counts, ~ .x$meta)
scrub_counts <- do.call(cbind, purrr::map(scrub_counts, ~ .x$norm_counts))

# Throw out samples with < 100 reads and species with < 10
scrub_counts <- scrub_counts[, colSums(scrub_counts)>100]
scrub_counts <- scrub_counts[rowSums(scrub_counts)>10, ]
scrub_meta <- scrub_meta[scrub_meta$sampleid %in% colnames(scrub_counts), ]
```

We can now generate some plots to look for evidence of clustering by type across a much larger subset of 1,809 samples from the TCGA data set

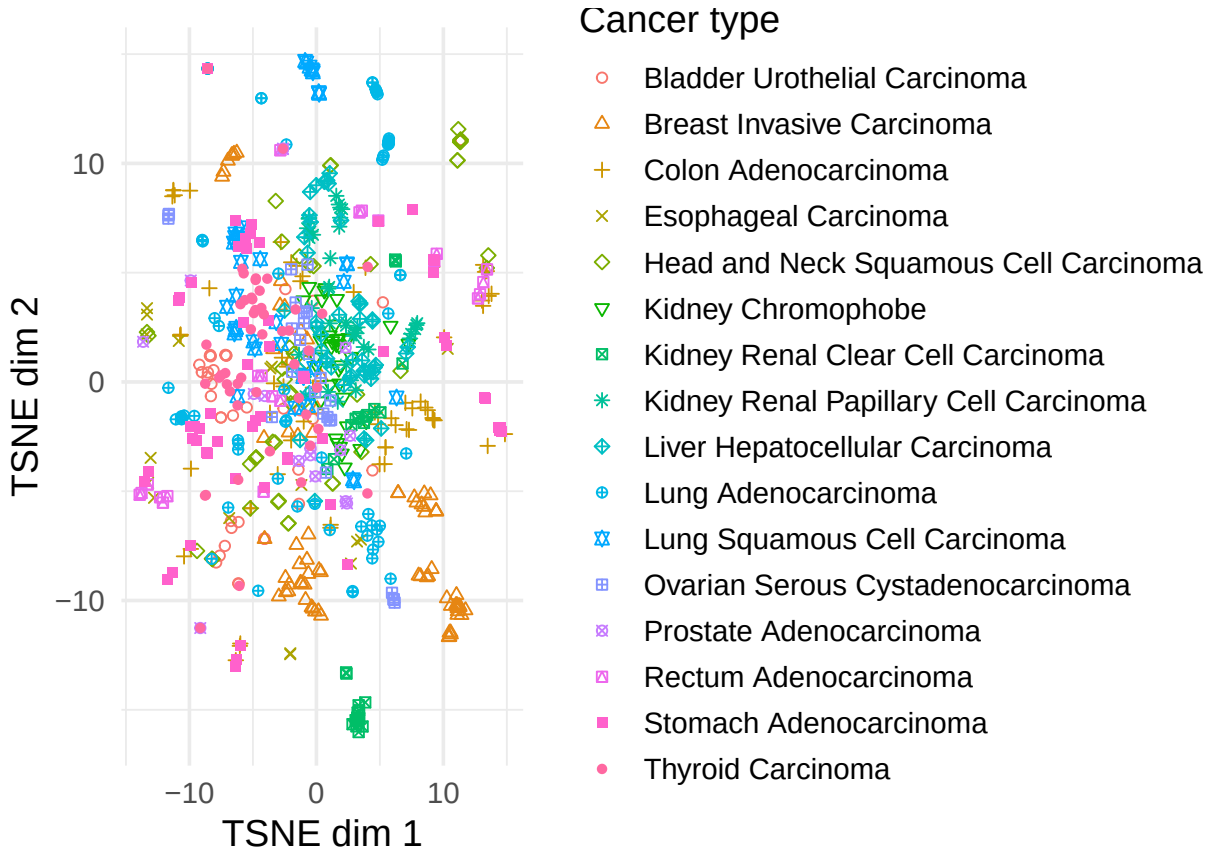
```
y <- edgeR::cpm(scrub_counts, normalized.lib.sizes = TRUE, log = TRUE, prior.count = 1)

# Account for remaining unwanted variables including gender and the hospital where the sample was taken
y <- removeBatchEffect(y, batch = scrub_meta$tissue_source_site_label)

pca_res <- prcomp(y)

pdf <- cbind(scrub_meta, pca_res$rotation[,1:10])

ggplot(pdf, aes(x=PC1, y=PC2, col=disease_type, shape=disease_type)) +
  geom_point() +
  scale_shape_manual(values =1:17) +
  theme_minimal(base_size = 14) +
  labs(colour="Cancer type") +
  labs(shape="Cancer type")
```



There appears to be a clustering along the first principal component. However, this was not correlated with cancer type or any of the other unwanted variables available in the metadata.

We can also consider a t-SNE plot which can sometimes reveal fine scale structure in high dimensional data sets.

```
tsne <- Rtsne::Rtsne(pca_res$rotation[,1:50], pca=FALSE, perplexity=50, check_duplicates=FALSE)
colnames(tsne$Y) <- paste('TSNE dim', 1:ncol(tsne$Y))

pdf <- cbind(scrub_meta, tsne$Y)

ggplot(pdf, aes(x=`TSNE dim 1`, y=`TSNE dim 2`, col=disease_type, shape=disease_type)) +
  geom_point() +
  scale_shape_manual(values =1:17) +
  theme_minimal(base_size = 14) +
  labs(colour="Cancer type") +
  labs(shape="Cancer type")
```



Again, although there appears to be some clustering it does not correlate strongly with cancer type.

Finally, an alternative method for determining if there is cancer associated clusters within this data is to build a machine learning model and estimate how accurately it can identify cancer types.

Here, I used the same Gradient Boosted Trees method as was used in the rebuttal to the preprint by Gihawi et al.

```
# Set up data sets and set seed
set.seed(1234)
trainX <- t(scrub_counts)
trainY <-  factor(gsub("^TCGA-","", scrub_meta$investigation))
```

```
# Set up model
xgbGrid <- data.frame(nrounds = 10,
                      max_depth = 4,
                      eta = .1,
                      gamma = 0,
                      colsample_bytree = .7,
                      min_child_weight = 1,
                      subsample = .8)


# as ctrl defines the cross validation sampling during training
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 1,
                     sampling = "up",
                     summaryFunction = multiClassSummary,
                     classProbs = TRUE,
                     verboseIter = FALSE,
                     savePredictions = TRUE,
                     allowParallel=TRUE)

set.seed(1234)
mlModel <- train(x = trainX,
                 y = trainY,
                 method = "xgbTree",
                 nthread = 1,
                 trControl = trainControl(method = "repeatedcv",
                                          number = 10,
                                          repeats = 1,
                                          sampling = "up",
                                          summaryFunction = multiClassSummary,
                                          classProbs = TRUE,
                                          verboseIter = FALSE,
                                          savePredictions = TRUE,
                                          allowParallel=TRUE),
                 tuneGrid = xgbGrid,
                 verbose = FALSE)


conf <- confusionMatrix(data = mlModel$pred$pred, reference = trainY, mode = "prec_recall")

conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##    0.064011380   -0.008282945    0.047069894    0.084717669    0.115220484
## AccuracyPValue  McnemarPValue
##    0.999998351            NaN
```

After controlling for sampling site and other unwanted batch effects, the machine learning model achieved an accuracy of 0.078 which is not significantly more than a null model.

Our approach to normalisation has been quite strict, so this analysis does not rule out an association between microbial reads and cancer types in this data set.

However, it does indicate that by using appropriate normalisation techniques we are able to account for

problems with contamination and database errors.

In the future, careful study designs that include body site specific but non-cancerous controls are necessary to accurately identify association between microbial DNA and cancer types.

Ultimately, to ascertain whether microbial DNA signatures can be utilised in cancer diagnostics, large comprehensive trials encompassing both cancer-afflicted and healthy patients will be required.

References

Sepich-Poore G. tcga_rebuttal: Re-analysis of data provided by Gihawi et al. 2023 bioRxiv. Github; Available: https://github.com/gregpoore/tcga_rebuttal

Gihawi A, Ge Y, Lu J, Puiu D, Xu A, Cooper CS, et al. Major data analysis errors invalidate cancer microbiome findings. *bioRxiv*. 2023. p. 2023.07.28.550993. doi:10.1101/2023.07.28.550993

Sepich-Poore GD, Kopylova E, Zhu Q, Carpenter C, Fraraccio S, Wandro S, et al. Reply to: Caution Regarding the Specificities of Pan-Cancer Microbial Structure. *bioRxiv*. 2023. p. 2023.02.10.528049. doi:10.1101/2023.02.10.528049

Gihawi A, Cooper CS, Brewer DS. Caution Regarding the Specificities of Pan-Cancer Microbial Structure. *bioRxiv*. 2023. p. 2023.01.16.523562. doi:10.1101/2023.01.16.523562

Austin GI, Park H, Meydan Y, Seeram D, Sezin T, Lou YC, et al. Contamination source modeling with SCRuB improves cancer phenotype prediction from microbiome data. *Nat Biotechnol*. 2023. doi:10.1038/s41587-023-01696-w

Poore GD, Kopylova E, Zhu Q, Carpenter C, Fraraccio S, Wandro S, et al. Microbiome analyses of blood and tissues suggest cancer diagnostic approach. *Nature*. 2020;579: 567–574. doi:10.1038/s41586-020-2095-1

Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol*. 2017;18: 174. doi:10.1186/s13059-017-1305-0

Mecham BH, Nelson PS, Storey JD. Supervised normalization of microarrays. *Bioinformatics*. 2010;26: 1308–1315. doi:10.1093/bioinformatics/btq118