Using the zinck package

Soham Ghosh

2024-12-16

About zinck

Microbiome data is high-dimensional, compositional, and zero-inflated which makes false discovery rate (FDR)-controlled taxonomic variable selection difficult. We propose **zinck**, a novel knockoff-based framework tailored for microbiome data exploiting a flexible generative model. It can properly capture the zero-inflation and complex dependence structure among microbes, enjoying the property of simultaneous variable selection and FDR control. Its potential efficacy is illustrated using various simulated and real data experiments.

The hierarchical structure of zinck

Before going into the structure of zinck, we define the following parameters:

1. Number of biological samples: D, number of microbial features: p, number of latent clusters K and the sequencing depth of the d^{th} sample is given by N_d, where $d = 1, \dots, D$.
2. $\theta_d = (\theta_{d1}, \dots, \theta_{dK})'$ is the vector of cluster mixing probabilities for the d^{th} biological sample, where $d \in 1, 2, \dots, D$
3. $\beta_k = (\beta_{k1}, \dots, \beta_{k(p-1)})'$ and $\beta_{kp} = 1 - \sum_{i=1}^{p-1} \beta_{ki}$ is the vector of feature proportions for each cluster $k = 1, 2, \dots, K$.
4. $\mathbf{z}_d = \left(z_{d1}, \dots, z_{dN_d}\right)'$ is the vector of cluster assignments for the d^{th} biological sample. For instance, $z_{dn} = k$ implies that the n^{th} sequencing read in the d^{th} sample belongs to the k^{th} cluster.
5. $\mathbf{w}_d = \left(w_{d1}, \dots, w_{dN_d}\right)'$ is the vector of the features drawn for each sequencing read for the d^{th} biological sample.
6. $\alpha = (\alpha, \dots, \alpha)^{K \times 1}$: symmetric hyperparameter of the Dirichlet prior of θ_d.
7. $\pi_k = (\pi_{k1}, \dots, \pi_{k(p-1)})'$: hyperparameter of the ZIGD distribution specifying the probability of being a structural zero for the j^{th} feature in the k^{th} subcommunity, where $j = 1, \dots, p$ and $k = 1, \dots, K$.
8. $\mathbf{a} = (a, \dots, a)^{(p-1) \times 1}$ and $\mathbf{b} = (b, \dots, b)^{(p-1) \times 1}$ are the symmetric hyperparameter vectors on the ZIGD of β_k.

It is to be noted that ZIGD here refers to the zero-inflated Generalized Dirichlet distribution.

zinck is a probabilistic hierarchical model with the following specification:

$$
\begin{aligned}
w_{dn} | z_{dn}, \beta_{z_{dn}} &\sim \text{Multinomial}(\beta_{z_{dn}}) \\
\beta_{z_{dn}} | \pi, \mathbf{a}, \mathbf{b} &\sim \text{ZIGD}\left(\pi_{z_{dn}}, a, b\right) \\
z_{dn} | \theta_d &\sim \text{Multinomial}(\theta_d) \\
\theta_d &\sim \text{Dirichlet}(\alpha)
\end{aligned}
$$

We denote the elements of the observed sample taxa matrix $\mathbf{X}^{D\times p}$ by (x_{dj}) as the observed read count of the j^{th} taxon for the d^{th} subject:

$$x_{dj} = \sum_{n=1}^{N_d} \mathbb{1}_{\{w_{dn}=j\}}$$

Generating knockoffs using zinck

We can exploit the knockoff generative model to learn the structure of the sample-feature count matrix $\mathbf{X}$ and generate a valid knockoff copy, which is further used for FDR-controlled feature selection. We fit the augmented LDA model to the microbiome count data matrix $\mathbf{X}$. The latent parameters of the model namely, θ_d and β_k are learnt by approximating their joint posterior distribution via the Automatic Differentiation Variational Inference (ADVI) algorithm or by drawing MCMC samples using a Collapsed Gibbs sampler. We use the learnt parameters $\tilde{\theta}_d, \tilde{\beta}_k$, for $d = 1, 2, \ldots D$ and $k = 1, 2, \ldots, K$ to generate a knockoff copy. For each sample d, we first sample a cluster allocation z_{dn} from Multinomial$(1, \tilde{\theta}_d)$ and then we sample a feature w_{dn} from the selected cluster z_{dn}, that is, $w_{dn} \sim$ Multinomial$(1, \tilde{\beta}_{z_{dn}})$. Finally, we form the knockoff matrix $\tilde{\mathbf{X}}^{D\times p} = \{\tilde{x}_{dj}\}$ by cumulating all the taxa read counts per subjects as illustrated previously.

We demonstrate the ability of zinck to capture the compositional and highly sparse nature of microbiome count data by comparing the heatmaps of the original sample taxa matrix $\mathbf{X}$ with its high quality knockoff copy, $\tilde{\mathbf{X}}$. The CRC dataset contain 574 subjects (290 CRC cases and 284 controls) and 849 gut bacterial species. Our analysis focuses on 401 species with an average relative abundance greater than 0.2 across all subjects. Then, we consider a toy setting with 20 randomly selected samples and 30 randomly selected CRC taxa at the species level. The sample library sizes vary between 17 and 4588, within an average library size of 1329. The zero-inflation level in the original sample taxa matrix is 0.51.

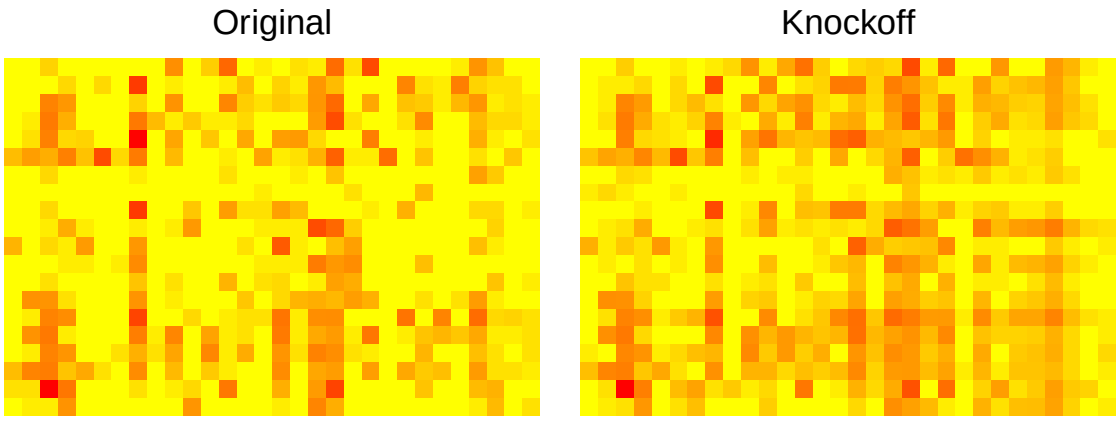
```
load("/Users/Patron/Documents/zinck research/count.RData")
load("/Users/Patron/Documents/zinck research/meta.RData")

norm_count <- count/rowSums(count)
col_means <- colMeans(norm_count > 0)
indices <- which(col_means > 0.2)
sorted_indices <- indices[order(col_means[indices], decreasing=TRUE)]
dcount <- count[,sorted_indices][,1:400]

set.seed(123)
selected_rows <- sample(1:nrow(dcount), 20)      ## Randomly select 20 subjects
selected_cols <- sample(1:ncol(dcount),30)       ## Randomly select 30 taxa
X <- dcount[selected_rows,selected_cols]       ## Resulting OTU matrix of dimensions 20*30
```

Now, we learn the compositional, zero-inflated structure of this generated matrix using the `zinck.fit()` function from the `zinck` package. Then, we will use the posterior estimates of the latent parameters to generate a knockoff copy for the original matrix. The knockoff generation is achieved by utilizing the function `generateKnockoff()`.

```
model_zinck <- fit.zinck(X, num_clusters=13, method="ADVI", seed=2,
                         boundary_correction = TRUE,prior_ZIGD = TRUE)
Theta <- model_zinck$theta
Beta <- model_zinck$beta
X_zinck <- generateKnockoff(X,Theta,Beta,seed=2)
```

We will now visualize the heatmaps of the original matrix and its corresponding knockoff copy. The function applies an arcsinh transformation to the data for normalization and better visualization of abundance patterns and zero inflation within the sample taxa matrix.

```
heat1 <- draw_heatmap(X, "Original")

rownames(X_zinck) <- rownames(X)
heat2 <- draw_heatmap(X_zinck, "Knockoff")

plot_grid(heat1, heat2, ncol = 2, align="v")
```



It is evident from the above heatmaps that the knockoff copy is almost indistinguishable from the original matrix! This underscores the fact that the knockoff copy preserves the underlying structure of the observed sample taxa count matrix.

The complete zinck workflow

We take the colon cancer (CRC) dataset which encompasses five distinct studies each originating from different countries. We calculated the average proportions of species across all 574 samples from five metagenomic studies and retained the top $p = 100$ most abundant species. Then, we randomly sampled $D = 500$ samples from the cohort of 574 observations and normalized their raw OTU counts row-wise, ensuring each row sums to 1. To avoid zero probabilities during multinomial sampling, a small constant (0.5) was added to the raw OTU counts before normalization.

Next, we paired the calculated true proportions Π with sequencing depths resampled from the original dataset. Each sequencing depth was independently resampled and paired with a proportion vector from a different subject. Using these paired proportions and sequencing depths, we generated the taxa count matrix $\mathbf{X}^{D\times p}$ via multinomial sampling. For each sample $d = 1, 2, \ldots, 500$, the count vector corresponding to the d^{th} row of $\mathbf{X}$ was generated as:

$$\mathbf{X}_d \sim \text{Multinomial}(N_d, \Pi_d),$$

where N_d is the sequencing depth for sample d, and Π_d is the vector of true proportions for sample d with Π_{dj} (j = 1...p) as its element.

Given the simulated count, among the $p = 100$ taxa, the first 30 were assumed to be associated with the outcome, while the remaining 70 taxa had no effect. To simulate outcomes reflecting realistic microbiome data, signal strengths s for the top 30 bio-markers were defined as:

$$s_j \overset{d}{=} \begin{cases} (2Z_j - 1)\frac{U_1}{\sqrt{C_j}}, & \text{for continuous outcomes,} \\ (2Z_j - 1)\frac{U_2}{\sqrt{C_j}}, & \text{for binary outcomes,} \end{cases}$$

where $U_1 \sim \text{Uniform}(1.5, 3)$ and $U_2 \sim \text{Uniform}(3, 10)$ represent the signal magnitudes, $Z_j \sim \text{Bernoulli}(0.5)$ determines the sign of s_j (+1 or −1), and C_j is the column-wise average abundance of the j^{th} bio-marker in Π.

```r
generate_data <- function(p,seed){
  dcount <- count[,order(decreasing=T,colSums(count,na.rm=T),apply(count,2L,paste,collapse=''))]
  ## ordering the columns w/ decreasing abundance
####### Randomly sampling patients from 574 observations #######
  set.seed(seed)
  norm_count <- count/rowSums(count)
  col_means <- colMeans(norm_count > 0)
  indices <- which(col_means > 0.2)
  sorted_indices <- indices[order(col_means[indices], decreasing=TRUE)]
  if(p %in% c(100,200,300,400)){
    dcount <- count[,sorted_indices][,1:p]
    sel_index <- sort(sample(1:nrow(dcount), 500))
    dcount <- dcount[sel_index,]
    original_OTU <- dcount + 0.5
    seq_depths <- rowSums(original_OTU)
    Pi = sweep(original_OTU, 1, seq_depths, "/")
    n = nrow(Pi)

    col_abundances = colMeans(Pi)

    ##### Generating continuous responses ######
    set.seed(1)
    signals = (2 * rbinom(30, 1, 0.5) - 1) * runif(30, 1.5, 3)
    kBeta = c(signals / sqrt(col_abundances[1:30]), rep(0, p - 30))
    eps=rnorm(n,mean = 0, sd=1)
    Y <- Pi^2 %*% (kBeta/2) + Pi %*% kBeta + eps

    ##### Generating binary responses #####
    set.seed(1)
    signals = (2 * rbinom(30, 1, 0.5) - 1) * runif(30, 3, 10)
    kBeta = c(signals / sqrt(col_abundances[1:30]), rep(0, p - 30))
    pr = 1/(1+exp(-(Pi^2 %*% (kBeta/2) + Pi %*% kBeta)))
    Y_bin = rbinom(n,1,pr)

    ######### Generate a copy of X #########
    X <- matrix(0, nrow = nrow(Pi), ncol = ncol(Pi))
    nSeq <- seq_depths
    # Loop over each row to generate the new counts based on the multinomial distribution

    set.seed(1)

    for (i in 1:nrow(Pi)) {
      X[i, ] <- rmultinom(1, size = nSeq, prob = Pi[i, ])
    }
  } else if (p %in% c(200,300,400)) {
    print("Enter p within 100 to 400")
  }
  colnames(X) <- colnames(Pi)
  return(list(Y = Y, X = X, Y_bin = Y_bin, index = 1:30))
}

ntaxa = 100 # Change to p = 200, 300, 400 accordingly.
```

```
X <- generate_data(p=ntaxa, seed=1)$X
Y <- generate_data(p=ntaxa, seed=1)$Y
```

Outcomes were generated using the following model:

$$g(\mathbb{E}(y_d)) = \sum_{j=1}^{p} \left\{ \Pi_{dj}^2 \times \left(\frac{s_j}{2}\right) + \Pi_{dj} \times s_j \right\}$$

where $g(\cdot)$ is the identity link for continuous outcomes and the logit link for binary outcomes. We replicate a single iteration of this setting for continuous outcomes. Now, we learn the latent zinck parameters by fitting the augmented ZIGD model on $\mathbf{X}$. We use the posterior estimates of θ and β.

```
fit <- fit.zinck(X,num_clusters = 15,method="ADVI",seed=12,elbo_samples = 100)
theta <- fit$theta
beta <- fit$beta
```

We use the posterior estimates of these parameters to generate the knockoff copy.

```
X_tilde <- zinck::generateKnockoff(X,theta,beta,seed=1) ## getting the knockoff copy
```

Now that we have generated the knockoff copy, we will fit a model associating the response $\mathbf{Y}$ and the augmented set of covariates $[\mathbf{X}, \tilde{\mathbf{X}}]^{D\times 2p}$. Here, we assume that the data generating mechanism is known, that is the response is linearly related to the covariates. Thus, we fit a Random Forest model and compute the importance statistics for each feature. Then, at a target FDR level of 0.2, we detect the non-zero features.

```
cts_rf <- suppressWarnings(zinck.filter(X,X_tilde,Y,model="Random Forest",fdr=0.2,offset=0,mtry=200,
                                        seed=12,metric = "Accuracy", rftuning = TRUE))
index_est <- cts_rf[["selected"]]
```

We can evaluate the performance of our filter by comparing with the true non-zero signals.

```
index <- 1:30
### Evaluation metrics ###
FN <- sum(index %in% index_est == FALSE) ## False Negatives
FP <- sum(index_est %in% index == FALSE) ## False Positives
TP <- sum(index_est %in% index == TRUE) ## True Positives

# ## False Discovery Proportion ##
estimated_FDR <- FP/(FP+TP)
print(estimated_FDR)
## [1] 0.1111111

## Power ##
estimated_power <- TP/(TP+FN)
print(estimated_power)
## [1] 0.5333333
```

Real Data Analysis

We applied `zinck` to meta-analyze five metagenomic studies of CRC. The five studies correspond to five different countries for the CRC data, which are named "AT" (Australia), "US" (USA), "CN" (China), "DE" (Germany), and "FR" (France). The sample sizes are 109, 127, 120, 114, and 104, respectively, and the number of cases and controls is roughly balanced in each study. We focus on all the species whose relative abundances are more than 0.2.

```
norm_count <- count/rowSums(count)
col_means <- colMeans(norm_count>0)
indices <- which(col_means > 0.2)
sorted_indices <- indices[order(col_means[indices],decreasing = TRUE)]
dcount <- count[,sorted_indices]

X <- dcount
Y <- ifelse(meta$Group=="CRC",1,0)
```

We train the `zinck` model on X using ADVI with an optimal number of clusters (20). We generate the knockoff matrix $\tilde{X}$ by plugging in the learnt parameters β and θ into the generative model.

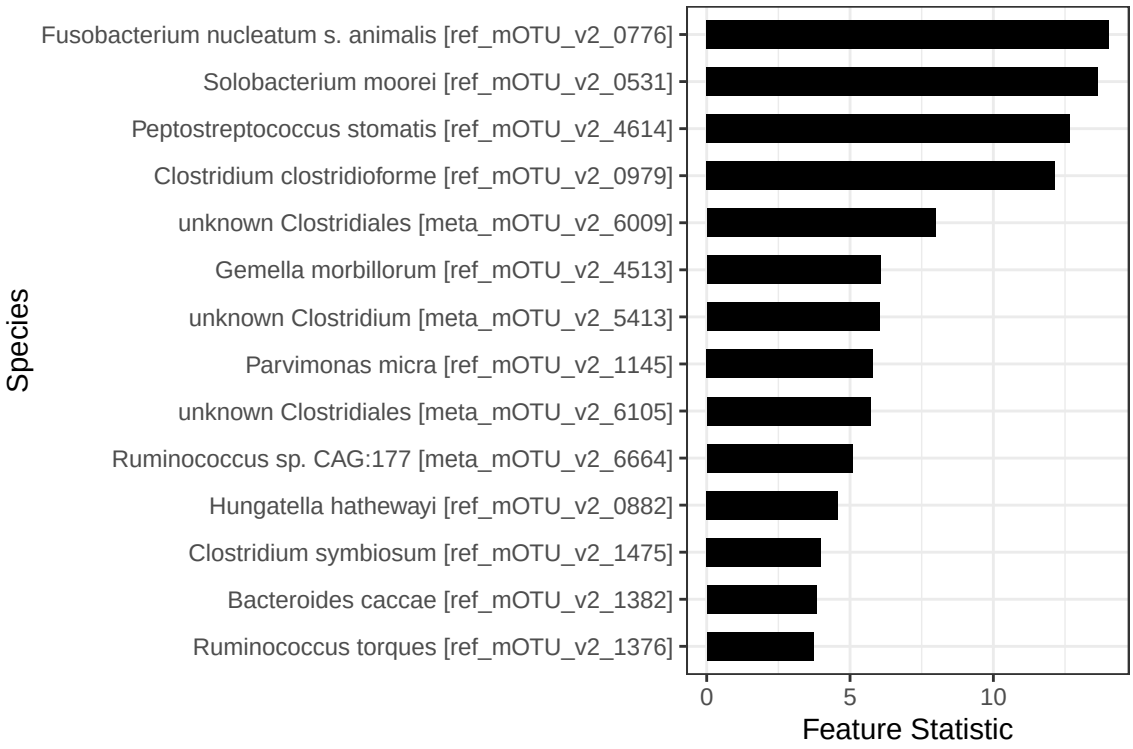
```
fitCRC <- fit.zinck(X,num_clusters=20,method="ADVI",seed=1,boundary_correction = TRUE,
                    elbo_samples = 100,importance_resampling = FALSE,
                    prior_ZIGD = TRUE)
theta <- fitCRC$theta
beta <- fitCRC$beta
X_tilde <- zinck::generateKnockoff(X,theta,beta,seed=1) ## Generating the knockoff copy
```

The selected species at a target FDR level of 0.1 are displayed below.

```
filter_zinck <- zinck.filter(as.matrix(X),as.matrix(X_tilde),as.factor(Y),
                model="Random Forest",offset = 1,seed=1,mtry=28,
                rftuning=TRUE,metric = "Gini")

selected_species <- filter_zinck$selected


## Importance scores ##
W <- filter_zinck$W

## Threshold ##
T <- filter_zinck$T

names <- colnames(X[,which(W>=T)])

### Creating the data frame with Feature Importance Statistics
data.species <- data.frame(
  impscores = sort(W[which(W>=T)], decreasing=FALSE) ,
  name = factor(names, levels = names),
  y = seq(length(names)) * 0.9
)
```

We can visualize the importance of these selected genera using the Feature Statistics obtained by contrasting the original and the knockoff features.

```
data.species <- data.frame(impscores=sort(W[which(W>=T)],decreasing = FALSE),
                           name=factor(names, levels=names), y=seq(length(names))*0.9)

plot.species <- ggplot(data.species)+geom_col(aes(impscores,name),
                                              fill="black",width=0.6)+theme_bw()+
  ylab("Species")+xlab("Feature Statistic")

plot.species
```



Choice of K?

An integral part of the zinck workflow is to choose an optimal value for k (number of clusters). We use the Jensen-Shannon (JS) divergence criterion for choosing the optimal number of clusters k for zinck. We can acheive this by fitting the zinck model for a range of k values. Finally, we choose the k that maximizes the average JS divergence. This signifies that for that model, the clusters are, on average, the most distinct from each other.

Here is an illustration for the CRC genus level data-set with $n = 574$ patients and $p = 133$ genera. We vary k from 8 to 17 and obtain the optimal k which maximizes the JS divergence.

```
load("/Users/Patron/Documents/zinck research/meta.RData")
dcount <- count.genus[,order(decreasing=T,colSums(count.genus,na.rm=T),
                             apply(count.genus,2L,paste,collapse=''))]
## ordering the columns w/ decreasing abundance
X <- dcount
kmin =8
kmax=17
K_values <- seq(kmin, kmax, by=1 )


## Run this command to find the optimal K value ##
#optimal_k(X,kmin=8,kmax=17,seed_list=list(1,11,1,1,1,1,1,1,1,1))

## On running optimal_k(), the Jensen-Shannon divergence values
js_values <- c(0.266242705918838,0.373342570843659,0.36512930638952,0.375253322971532,0.430181098394843

data <- data.frame(K_values, js_values)
p <- ggplot(data, aes(x = K_values, y = js_values)) +
    geom_line() +
```

```
    geom_point(size = 4) +
    labs(
      title = "Optimal Number of clusters based on Jensen-Shannon Divergence",
      x = "Number of clusters (K)",
      y = "Average JS Divergence"
    ) +
    theme_minimal() +
    theme(
      plot.title = element_text(hjust = 0.5),
      panel.grid.major = element_line(size = 0.5, linetype = 'solid', color = "grey"),
      panel.grid.minor = element_line(size = 0.5, linetype = 'solid', color = "grey")
    )
print(p)
```

It is clear from the Divergence plot that k=16 is optimal for fitting the zinck model to the CRC genus level data.

References

1. Latent dirichlet allocation Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003).
2. Zero-inflated generalized Dirichlet multinomial regression model for microbiome compositional data analysis Tang, Z. Z., & Chen, G. (2019).
3. Automatic Differentiation Variational Inference Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., & Blei, D. (2017).]
4. A Zero-Inflated Latent Dirichlet Allocation Model for Microbiome Studies Deek, R., and Li, H. (2021).