Probabilistic Learning of Treatment Trees in Cancer

8/6/2021

Contents

Introduction

Motivation: The superior therapeutic effects of synergistic combination therapy have been discovered in treating multiple types of cancers and it is of high clinical value to accurately identify the synergistic combinations via shared or similar underlying biological mechanisms. However, not all possible drug combinations can be realistically tested on patients in real clinical trials. Cancer researchers are increasingly relying on pre-clinical systems such as patient-derived xenografts (PDX), that has recently emerged as a unique study design evaluating multiple treatments administered to samples from the same human tumor implanted into mice. Owing to the high fidelity of PDX systems to mimic human tumors, they can be used to guide the discovery of the most effective combination therapies in a cost-effective manner. Based on the PDX data, the central scientific question we address is how to identify promising combinations of candidate treatments for further clinical evaluation along with their plausible biological mechanisms.

We address this unmet translational research need by proposing a novel Bayesian probabilistic tree-based framework, referred to as treatment trees (Rx-tree), to investigate the hierarchical relationships between treatments. In particular, we infer treatment cluster trees, propose and estimate a metric of mechanistic similarity through the integrated posterior co-clustering probability (iPCP) for any combination of treatments. Ackowledging the uncertainties in the tree-structure, the Rx-tree visualizes the global hierarchy among all treatments, and the iPCP further quantifies the mechanism similarity for any subset of treatments assigned.

The model estimation is decoupled into (i) Euclidean parameters estimation by the approximated Bayesian computation (ABC) and (ii) tree parameters estimation by Metropolis-Hastings algorithm (MH) and we implemented four R scripts, names as "ABC_SyntheticData.R", "Inference.R", "Tree_Summary.R", and "Visualization.R" to estimate the Rx-tree and the corresponding iPCP. To conduct the ABC, we first run the "ABC_SyntheticData.R" to generate the synthetic data with a large sample size N^{syn}. Given the synthetic data and the observed data, "Inference.R" compares the summary statistics of synthetic data and the observed data, infers the posterior samples of Euclidean parameters and summarize the Euclidean parameters. Conditional on the posterior median of the Euclidean parameters, the "Inference.R" follows the MH algorithm and generates the posterior tree samples. Given posterior tree samples, "Tree_Summary.R" summarizes the posterior tree samples with MAP tree and calculates the iPCP for any subset of treatments of interest and the "Visualization.R" visualizes the results from the "Tree_Summary.R" as the Figures 5 and 6 in the Main paper. We demonstrate the code in the following sections with the data from Novartis Institutes for Biomedical Research PDX Encyclopedia (NIBR-PDXE; Gao et al., 2015 and Rashid et al., 2020) and replicate the Figures 5 and 6 in the Main paper.

Downloading PDX Data

The data is available online in the supplementary material of Rashid et al., 2020 and we download the data "rawData.rda" in the directory "PDX_Data. The"rawData.rda" contains five cancers: Breast cancer (BRCA), Cutaneous Melanoma (CM, skin cancer), Colorectal cancer (CRC), Non-small Cell Lung Carcinoma (NSCLC), and Pancreatic Ductal Adenocarcinoma (PDAC). After re-scaling data and missing data imputation (see the Section Figure Replication) , we took the BRCA data as an example to demonstrate the code with a smaller number iterations.

ABC Stage

We generate the ABC synthetic data through the file "ABC_Synthetic.R" with the function "ABC_SyntheticData". In the function "ABC_SyntheticData", we need to specify the number of treatments (nLeaf), number of PDX (nPDX) and the synthetic data sample size (nSyn). For example, in the Section 4.1, we set nLeaf=20, nPDX=38 and nSyn=600,000. We run the code on multiple machine parallely. Here, we run the function with a smaller number of nSyn=10 and give a brief look at the output.

```
source("ABC_SyntheticData.R")
sim_sumstat<-ABC_SyntheticData(nLeaf=20, nPDX=38, nSyn= 10)
sim_sumstat$sim_sumstat
```

```
##    idx  c_hazard  sigma_sq Summary_sigma dist_rel10 dist_rel25 dist_rel50
## 1    1 1.1514172 0.8869150     0.7240720  4.4933014  5.7179972   6.435069
## 2    2 1.5297501 0.6918322     0.7535355  3.6742958  4.4552700   5.651253
## 3    3 1.2227546 0.6523519     0.6677832  2.7683629  4.2236425   4.734695
## 4    4 1.5986263 0.7112752     0.6607796  4.0499126  4.5300564   5.538059
## 5    5 1.1494156 1.0443791     1.0672047  3.0591458  4.7180804   7.440855
## 6    6 0.4686373 0.8632445     0.8835305  1.4721389  2.6344211   6.139441
## 7    7 0.5417292 1.2202076     1.1512982  0.6751074  1.5946160   4.578173
## 8    8 1.1177449 0.8202990     0.7310862  2.7438943  3.4657305   5.958763
## 9    9 0.1569783 1.4353699     1.6783871  0.1836193  0.2748856   5.769969
## 10  10 0.5618925 5.8466394     5.7397688  4.9001253 10.6806794  13.627817
##    dist_rel75 dist_rel90 hclu_tip_BrLen10 hclu_tip_BrLen25 hclu_tip_BrLen50
## 1    7.038312   7.471017     4.017442e-01     1.246329e+00      1.690254727
## 2    6.263723   6.778627     9.646980e-01     1.418356e+00      2.160194839
## 3    5.194151   5.854676     4.918655e-01     1.012029e+00      1.086220707
## 4    6.674311   7.144242     7.786809e-01     1.016738e+00      1.439239261
## 5    8.146195   8.775657     8.209224e-01     9.888955e-01      1.104425159
## 6    6.559721   6.733350     2.236611e-01     3.310860e-01      0.399933779
## 7    7.217682   7.546561     2.145277e-02     1.589408e-01      0.499792999
## 8    6.846570   8.286376     4.415839e-01     9.367753e-01      1.267992709
## 9    5.860688   5.960209     1.178881e-05     1.608183e-05      0.009123654
## 10  14.991104  19.234368     2.614837e-01     1.396692e+00      2.019671573
##    hclu_tip_BrLen75 hclu_tip_BrLen90
## 1         2.6369339        2.9325538
## 2         2.3228170        2.8862593
## 3         1.5883585        2.2819109
## 4         2.0538021        2.1850280
## 5         2.1512734        2.3952396
## 6         0.8587957        1.9820238
## 7         1.2390807        1.7187148
## 8         1.5135534        2.5236632
## 9         0.2054331        0.3397392
## 10        2.4719147        2.6949064
```

The output contains the $S^{(\sigma^2)}$ ("Summary_sigma"), $\mathbf{S}^{(c)}$ ("dist_rel" and "hclu_tip_BrLen") and the correspoding generating parameters of c ("c_hazard") and σ^2 ("sigma_sq"). We ran the code to generate the synthetic data for five cancers for the NIBR-PDXE with sample size $N^{syn} = 600,000$ used in the Main Paper. We stored the synthetic data in the folder "ABC_Synthetic_Data" and used the stored data for the following analysis.

Given the synthetic data, we compare the summary statistics of the synthetic data and the observed data and obtain the posterior samples. Based on the R package "abc", we specify the threshold parameter d and generate the regression adjusted posterior samples of Euclidean parameters through the function "abc_s2" and "abc_c" in the "Inference.R". The posterior summary of the Euclidean parameters is then calculated by the function "summary.abc". From the "summary.abc", we obtain the posterior median and we pass it to the next MH stage. We extract part of the code related to the ABC inference in the "Inference.R" and give an example code of the ABC inference with the following posterior summary.

```
source("Inference.R")
sim_sumstat<-readRDS(here("ABC_Synthetic_Data","BRCA_sim_sumstat.RDS"))
obsDf<-readRDS(here("PDX_Data","BRCA_BAR.RDS"))
post_s2<-abc_s2( simulation_sumstat =sim_sumstat, df = obsDf, d=0.005)
```

```
## Warning in abc(target = as.matrix(obs_sumstat["Summary_sigma"]), sumstat =
## as.matrix(simulation_sumstat[, : No parameter names are given, using P1, P2, ...
```

```
## Warning in abc(target = as.matrix(obs_sumstat["Summary_sigma"]), sumstat =
## as.matrix(simulation_sumstat[, : No summary statistics names are given, using
## S1, S2, ...
```

```
post_c<-abc_c( simulation_sumstat = sim_sumstat, df = obsDf, d=0.005)
```

```
## Warning in abc(target = as.matrix(obs_sumstat[1, c(paste0("hclu_tip_BrLen", : No
## parameter names are given, using P1, P2, ...
```

```
summary(post_s2)
```

```
## Call:
## abc(target = as.matrix(obs_sumstat["Summary_sigma"]), param = as.matrix(simulation_sumstat[,
##     "sigma_sq"]), sumstat = as.matrix(simulation_sumstat[, "Summary_sigma"]),
##     tol = d, method = "loclinear", transf = rep("none", 1))
## Data:
##  abc.out$adj.values (3000 posterior samples)
## Weights:
##  abc.out$weights
##
##                             P1
## Min.:                   0.9435
## Weighted 2.5 % Perc.:   1.2908
## Weighted Median:        1.7670
## Weighted Mean:          1.7910
## Weighted Mode:          1.7478
## Weighted 97.5 % Perc.: 2.4704
## Max.:                   3.6907
```

```
summary(post_c)
```

```
## Call:
## abc(target = as.matrix(obs_sumstat[1, c(paste0("hclu_tip_BrLen",
##     c(10, 25, 50, 75, 90)), paste0("dist_rel", c(10, 25, 50,
##     75, 90)))]), param = as.matrix(simulation_sumstat[, "c_hazard"]),
##     sumstat = simulation_sumstat[, c(paste0("hclu_tip_BrLen",
```

```
##         c(10, 25, 50, 75, 90)), paste0("dist_rel", c(10, 25,
##         50, 75, 90)))], tol = d, method = "loclinear", transf = rep("none",
##         1))
## Data:
##  abc.out$adj.values (3000 posterior samples)
## Weights:
##  abc.out$weights
##
##                            P1
## Min.:                  0.4825
## Weighted 2.5 % Perc.:  0.7665
## Weighted Median:       1.1934
## Weighted Mean:         1.2116
## Weighted Mode:         1.1670
## Weighted 97.5 % Perc.: 1.7425
## Max.:                  2.5933
```

MH Stage

We run the second stage MH algorithm by the function "Tr_twoStage" in the script "Inference.R". In the function "Tr_twoStage", we first specify the fixed Euclidean parameters from the ABC stage through the arguments "post_c" and "post_s2" and initialize the tree structure by the distance-based agglomerative hierarchical clustering with the linkage method specified by argument "hclust_method".

```
source("Inference.R")
postMed_c<-summary(post_c, print=F)["Weighted Median:",1]
postMed_s2<-summary(post_s2, print=F)["Weighted Median:",1]
MH_res<-Tr_twoStage(post_c=postMed_c,post_s2=postMed_s2,iteNum = 10,
                    obsDf = obsDf, hclust_method = "ward.D")
MH_res$llh
```

```
##    idx Accept   llh_Prop    q_ToOld   llh_old    q_ToNew        c   sigma2
## 1    1      0  -918.6092 -1.3738670 -894.4860 -1.9357218 1.193364 1.766953
## 2    2      0  -906.3442 -1.3738670 -894.4860 -1.6477089 1.193364 1.766953
## 3    3      1  -895.3476 -1.6163055 -894.4860 -1.6316077 1.193364 1.766953
## 4    4      0 -1199.4128 -2.0115351 -895.3476 -1.9970650 1.193364 1.766953
## 5    5      1  -894.8429 -1.9731120 -895.3476 -1.5877438 1.193364 1.766953
## 6    6      0  -900.5837 -1.9663451 -894.8429 -2.0871385 1.193364 1.766953
## 7    7      0  -898.1185 -1.5877438 -894.8429 -2.0053685 1.193364 1.766953
## 8    8      0  -907.5156 -1.5759968 -894.8429 -2.3299110 1.193364 1.766953
## 9    9      1  -893.2354 -1.8775508 -894.8429 -1.6912692 1.193364 1.766953
## 10  10      1  -892.9377 -0.5436262 -893.2354 -0.6136026 1.193364 1.766953
```

The function "Tr_twoStage" has two outputs: "llh" and "tree_lt". The "llh" is a numeric data frame showing the likelihood and the acceptance of each proposed tree and the "tree_lt" is the posterior tree samples.

Posterior Tree Summary

Two posterior tree summaries, MAP tree and iPCP, are developed in the file "Tree_Summary.R". Given the posterior tree samples with the corresponding likelihood, the MAP tree can be obtained by the function "getMAP" with the likelihood and the posterior tree samples through the arguments "llh_mat" and "postTr_lt" , respectively. For iPCP, the function "iPCP" calculates the iPCP with the input argument "treatments" to specify the the subsets of treatments of interest and "postTr_lt" for the posterior tree samples. We offer

the example code to calculate the MAP tree and the three-way iPCP for three PI3K inhibitors (BKM120, BYL719 and CLR457).

```
source("Tree_Summary.R")
MAP_Tr<-getMAP(llh_mat = MH_res$llh, postTr_lt = MH_res$tree_lt)
iPCP_res<-iPCP(treatments = c("BKM120","BYL719","CLR457"), postTr_lt = MH_res$tree_lt)
```

Figure Replication

We demonstrate the code for reproducing Figures 5 and 6 with a smaller number of iterations and all codes with a larger number of iterations are contained in the script named as "ACS_reproduce_submit.R". For each cancer, we first impute the missing data to generate the "cancer_BAR.RDS" files from the raw data "rawData.rda" and decide the number of treatments and PDX. Here, we take the BRCA as an example and the same process is applied to other cancers.

```
rawData<-load(here("PDX_Data","rawData.rda"))
cancer.type="BRCA"
outcome = "BAR"
suppressPackageStartupMessages(library("bnstruct"))

dat = split.cm.data$BRCA; clinical = dat[ , 1:17]
clinical$RespScaled = -clinical$RespScaled

new.resp.mat = matrix(NA, nrow = length(unique(clinical$Model)),
                      ncol = length(unique(clinical$Treatment)))
rownames(new.resp.mat) = unique(clinical$Model)
colnames(new.resp.mat) = unique(clinical$Treatment)
for (dim1 in 1:nrow(new.resp.mat)) {
  for (dim2 in 1:ncol(new.resp.mat)) {
    row = which(clinical$Model == rownames(new.resp.mat)[dim1] &
                  clinical$Treatment == colnames(new.resp.mat)[dim2])
    if (length(row) != 0) {
      new.resp.mat[dim1, dim2] = clinical$RespScaled[row]
    }
  }
}
clinical = new.resp.mat
df_out<-t(clinical)
df_out_<-df_out[,!apply(df_out,2,function(x) sum(is.na(x))/nrow(df_out)>0.4 )]
df_out_impute<-bnstruct::knn.impute(df_out_)
df_final<-df_out_impute[rownames(df_out_impute)!="untreated",] -
  matrix(rep(df_out_impute[rownames(df_out_impute)=="untreated",],
             nrow(df_out_impute)-1),
         ncol=ncol(df_out_impute),byrow=T)
#saveRDS(df_final,file=paste0(here("PDX_Data",cancer.type,,outcome,".RDS")))
```

We then run the two-stage algorithm with functions demonstrated above and summarize the posterior tree summary. Here, we run the algorithm with a smaller number of iteration. In the file, "ACS_reproduce_submit.R", we offer the code with a larger number of iterations.

```
source("ABC_SyntheticData.R")
source("Inference.R")
source("Tree_Summary.R")
obsDf<-readRDS(here("PDX_Data","BRCA_BAR.RDS"))
```

```
sim_sumstat<-ABC_SyntheticData(nLeaf=nrow(obsDf), nPDX=ncol(obsDf), nSyn= 10)
sim_sumstat<-readRDS(here("ABC_Synthetic_Data","BRCA_sim_sumstat.RDS"))
post_s2<-abc_s2( simulation_sumstat =sim_sumstat, df = obsDf, d=0.005)
post_c<-abc_c( simulation_sumstat = sim_sumstat, df = obsDf, d=0.005)
postMed_c<-summary(post_c, print=F)["Weighted Median:",1]
postMed_s2<-summary(post_s2, print=F)["Weighted Median:",1]
MH_res<-Tr_twoStage(post_c=postMed_c,post_s2=postMed_s2,iteNum = 10,
                    obsDf = obsDf, hclust_method = "ward.D")
```

The "Visualization.R" generates the Figure 5 and 6 in the Main Paper and visualizes the posterior tree summaries, the MAP tree and pairwise iPCP, from the "Tree_Summary.R". We demonstrate the code in the "Visualization.R" and replicate the Figures 5 and 6 in the Main Paper.

```
require(ggplot2)
suppressPackageStartupMessages(library("ggtree"))
require(ggnewscale)
```

```
## Loading required package: ggnewscale
```

```
require(cowplot)
```

```
## Loading required package: cowplot
```

```
require(reshape2)
```

```
## Loading required package: reshape2
```

```
require(stringr)
```

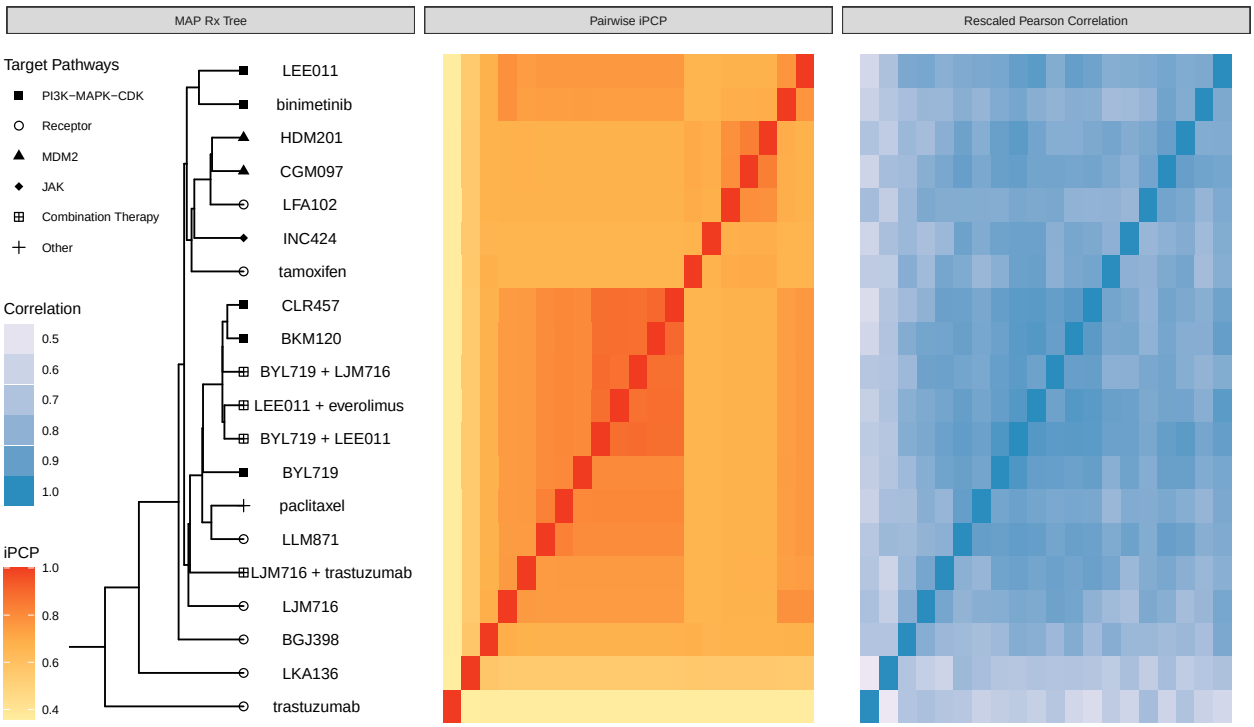
```
## Loading required package: stringr
```

```
target_mat<-readRDS(here("MH_posteriorTree","All_target.RDS"))
cancer.type<-"BRCA"
MAP_phylo4d<-readRDS(here("MH_posteriorTree",cancer.type,paste0(cancer.type,"_MAP.RDS")))
### extract the order of MAP tree to re-order the pairwise iPCP and correlation
apeMAP<-as(extractTree(MAP_phylo4d),"phylo")
is_tip <- apeMAP$edge[,2] <= length(apeMAP$tip.label)
ordered_tips <- apeMAP$edge[is_tip, 2]

iPCP_mat<-readRDS(here("MH_posteriorTree",cancer.type,paste0(cancer.type,"_pairIPCP.RDS")))
iPCP_mat_sym<-iPCP_mat[iPCP_mat$trt1 != iPCP_mat$trt2, c("trt2","trt1","iPCP")]
colnames(iPCP_mat_sym)<-c("trt1","trt2","iPCP")
iPCP_mat<-rbind(iPCP_mat,iPCP_mat_sym)
iPCP_mat$trt.x<-factor(iPCP_mat$trt2,levels=apeMAP$tip.label[ordered_tips])
iPCP_mat$trt.x_num<-as.numeric(iPCP_mat$trt.x)
raw_df<-readRDS(here("PDX_Data",paste0(cancer.type,"_BAR.RDS")))
smpCorr_sq<-matrix(NA,nrow=nrow(raw_df),ncol=nrow(raw_df))
rownames(smpCorr_sq)<-colnames(smpCorr_sq)<-rownames(raw_df)
for(idx1 in 1:(nrow(raw_df)-1)){
  for(idx2 in (idx1+1):(nrow(raw_df))){
    trt1<-rownames(smpCorr_sq)[idx1]
    trt2<-rownames(smpCorr_sq)[idx2]
    tmpCor<-cor(as.numeric(as.matrix(raw_df[idx1,-1])),as.numeric(as.matrix(raw_df[idx2,-1])))
    smpCorr_sq[idx1,idx2]<-smpCorr_sq[idx2,idx1]<-tmpCor
  }
}
diag(smpCorr_sq)<-1
```

```
smpCorr_sq_reSc<-(smpCorr_sq+1)/2
smpCorr_reSc<-reshape2::melt(smpCorr_sq_reSc)
smpCorr_reSc$trt.x<-factor(smpCorr_reSc$Var2,levels=apeMAP$tip.label[ordered_tips])
smpCorr_reSc$trt.x_num<-as.numeric(smpCorr_reSc$trt.x)

p <- ggtree(extractTree(MAP_phylo4d),ladderize = F)
if(cancer.type=="BRCA" | cancer.type=="PDAC"){
  p <- p %<+% target_mat + geom_tippoint(aes(shape=Target.Gr),size=2) +
    geom_tiplab(offset = .25, hjust = .3,size=3) +
    scale_shape_manual("Target Pathways",guide = guide_legend(order = 1),values=c(15,1,17,18,12,3))
}else if(cancer.type=="CRC"| cancer.type=="CM"){
  p <- p %<+% target_mat + geom_tippoint(aes(shape=Target.Gr),size=2) +
    geom_tiplab(offset = .25, hjust = .3,size=3) +
    scale_shape_manual("Target Pathways",guide = guide_legend(order = 1),values=c(15,1,17,16,12,3))
}else if(cancer.type=="NSCLC"){
  p <- p %<+% target_mat + geom_tippoint(aes(shape=Target.Gr),size=2) +
    geom_tiplab(offset = .25, hjust = .3,size=3) +
    scale_shape_manual("Target Pathways",guide = guide_legend(order = 1),values=c(15,1,17,12,3))
}
p1<-facet_plot(p=p,panel = "Pairwise iPCP", data = iPCP_mat, geom = geom_tile,
               mapping=aes(x = trt.x_num, fill = iPCP)) + labs(fill="iPCP") +
  scale_fill_gradientn(colors = RColorBrewer::brewer.pal(3, "YlOrRd"))
p2 <- p1 + ggnewscale::new_scale_fill()
p3<-facet_plot(p=p2 + xlim_tree(1.7) ,panel = "Rescaled Pearson Correlation",
               data = smpCorr_reSc, geom = geom_tile,
               mapping=aes(x = trt.x_num, fill = value)) + labs(fill="Correlation")+
  scale_fill_gradientn(colors = RColorBrewer::brewer.pal(3, "PuBu"),guide = guide_legend(order = 3))
p_final<-p3+theme(legend.position=c(.06, .5),text=element_text(size=9))
p_final<-facet_labeller(p_final, c(Tree = "MAP Rx Tree"))
p_final
```



```r
cancer.type<-"BRCA"
iPCP_mat<-readRDS(here("MH_posteriorTree",cancer.type,paste0(cancer.type,"_pairIPCP.RDS")))
iPCP_mat$name=paste0(iPCP_mat$trt1,", ",iPCP_mat$trt2)
iPCP_threshold<-0.7
iPCP_mat<-iPCP_mat[iPCP_mat$iPCP>=iPCP_threshold & iPCP_mat$trt1 != iPCP_mat$trt2,]
iPCP_mat$cmb<-ifelse(str_detect(iPCP_mat$trt1,"\\+") | str_detect(iPCP_mat$trt2,"\\+"),
                     "Combination Therapy","Monotherapy")

iPCP_mono<-iPCP_mat[!str_detect(iPCP_mat$trt1,"\\+") & !str_detect(iPCP_mat$trt2,"\\+"),]
iPCP_mono$name<- sapply(iPCP_mono$name,
                        function(x) paste(sort(unlist(str_split(x,", "))),collapse = ", "))
iPCP_cmb<-iPCP_mat[str_detect(iPCP_mat$trt1,"\\+") & str_detect(iPCP_mat$trt2,"\\+"),]
iPCP_cmb$name<- sapply(iPCP_cmb$name,
                       function(x) paste(sort(unlist(str_split(x,", "))),collapse = ", "))
iPCP_final<-rbind(iPCP_mono,iPCP_cmb)
g<-ggplot(iPCP_final,aes(y=reorder(name,iPCP),x=iPCP,fill=cmb))+geom_col()+
  labs(y="Treatments Pairs",fill="Treatment Type",x="iPCP") + coord_cartesian(xlim=c(0,1)) +
  theme(text = element_text(face="bold"))
g
```