All_figures

L.Bystrykh, M.Belderbos

Here are all figures for the manuscript. Note that conditions for the beta distribution remain the same. First we load only those packages which are needed for the Figure 1. For some reason function melt is interfered by some packages. So we try to avoid it.

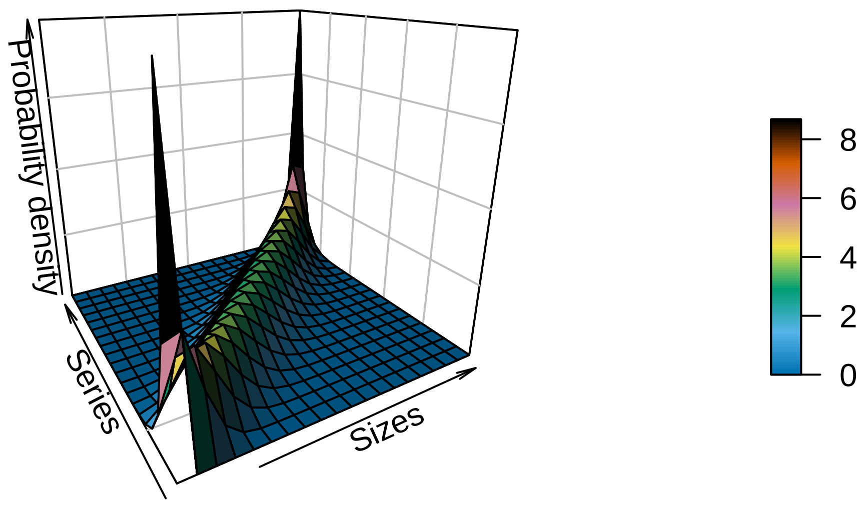
```
library(ggplot2) #graphics
library(plot3D) #for mesh and static 3D
library(RColorBrewer) #for custom colors
library(reshape) #for melt
```

This is the version of showing beta distribution series as a surface plot for this we need a meshgrid, then we generate probability density function and all three coordinates are ready to use

```
M<-mesh(seq(1,20,length.out=20), #series ID
        seq(0,1,length.out=20)) #beta distribution values
y <-M$y
x <-M$x
z=c()
for (i in 1:20){
  z=rbind(z,dbeta(seq(0,1,length.out=20),i,21-i))
}
```

No we plot it as static 3D. This is essentially Figure 1A

```
surf3D(x, y, z,
       colvar = z,
       colkey = list(side = 4,plot = TRUE,length = 0.5,width =
                                                   0.8,dist = 0.01, shift = 0, addlines = FALSE, col.clab = NUL
                                                 = NULL, line.clab = NULL, adj.clab = NULL, font.clab = NULL),
         shade = 0.5,
       box = TRUE, bty = "b2", phi = 20, theta = -30,
       col = gg2.col(100), border = "black", facets = TRUE, clab = NULL,
       xlab = "Sizes", ylab = "Series", zlab = "Probability density",
       drawlabels = TRUE)
```



Next we make a series of heatmaps for variable beta distribution and number of clones. Here are data for the Ich heatmap

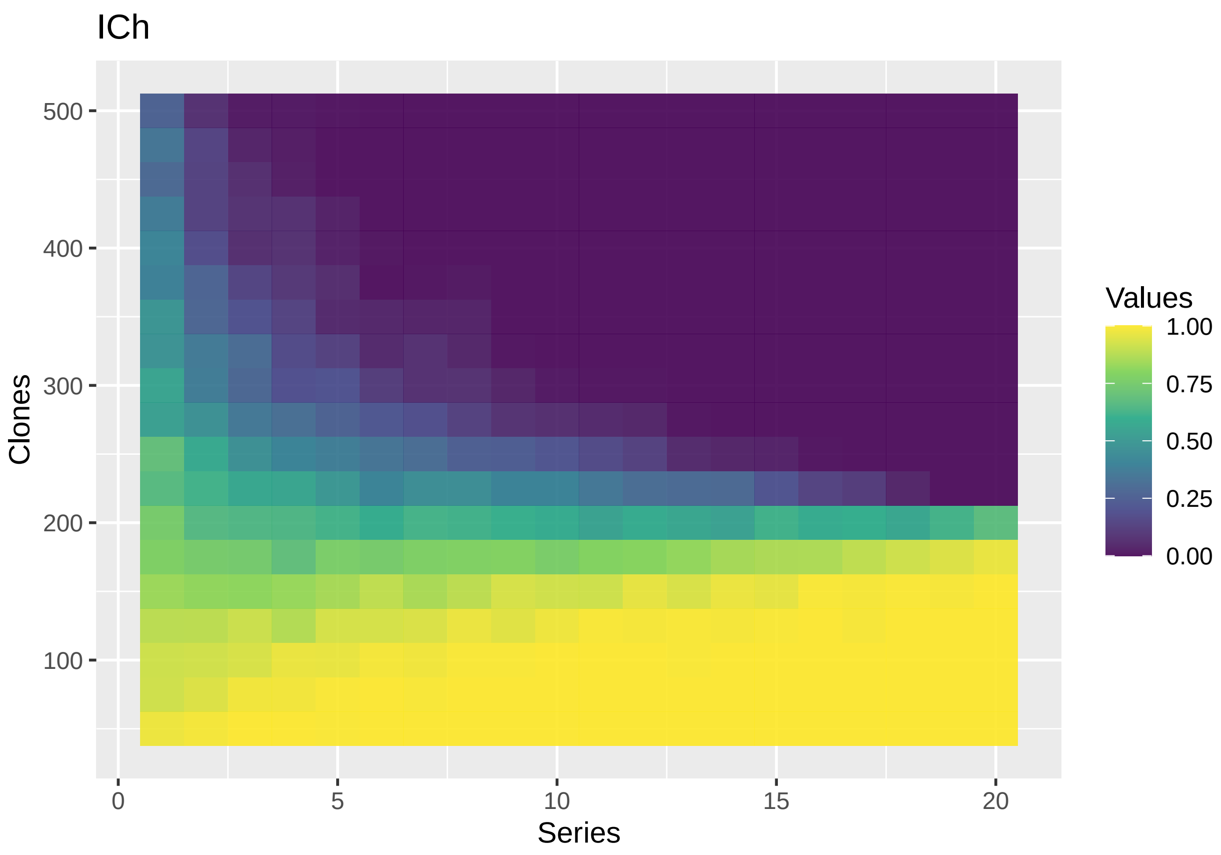
```
Ich_series=c()
for (clones in seq(50,500, 25)){
Ich=c()
for (i in 1:20){
  values=rbeta(clones,i,21-i)
  fractions<-values/sum(values)
  selected<-fractions[fractions>=0.005]
  Ich=append(Ich,sum(selected))
}
Ich_series=cbind(Ich_series,Ich)
}
```

Now we can make a plot

```
#rename column names, it will be shown on the figure
colnames(Ich_series)<-seq(50,500,25) #make proper col names
coul <- colorRampPalette(brewer.pal(8, "YlGnBu"))(20)
melted<-melt(Ich_series) #or any other series down the script
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
colnames(melted)<-c("Skewing","Clones","Values")
ggplot(melted, aes(Skewing, Clones, fill=Values))+
  geom_tile()+
  labs(title="ICh", x="Series", y="Clones")+
  scale_fill_viridis_c(alpha=0.9)#+
```



```
# theme_void() #this line defines whether to show coord values or not
```

Note the almost binary switch in Ich below clone number 200. It is related to the threshold 5% used. At this number almost all clones are counted. Further we do similar plots for other indexes. First we make a function for generating those data.

```
library(vegan) #for Shannon and Simpson
```

```
## Loading required package: permute

## Warning: package 'permute' was built under R version 4.1.2

## Loading required package: lattice

## This is vegan 2.5-7
```

```
library(reldist) #for gini index
```

```
## Warning: package 'reldist' was built under R version 4.1.2

## reldist: Relative Distribution Methods
## Version 1.7-0 created on 2021-11-10.
## copyright (c) 2003, Mark S. Handcock, University of California-Los Angeles
##  For citation information, type citation("reldist").
##  Type help(package="reldist") to get started.
```

```r
library(OTUtable) #for pielou
# make function with switch for index selection
indexes=function(data,type){
  switch(type,
         Shannon=diversity(data, index="shannon"),
         Simpson=diversity(data, index="simpson"),
         Pielou=pielou(data),
         Gini=gini(data))
}
#make function for data series
data_series=function(kind){
  data=c()
  for (clones in seq(50,500, 25)){
    dat=c()
    for (i in 1:20){
      values=rbeta(clones,i,21-i)
      indx=indexes(values, kind)
      dat=append(dat, indx)
    }
    data=cbind(data,dat)
  }
  colnames(data)<-seq(50,500,25)
  return(data)
}
```

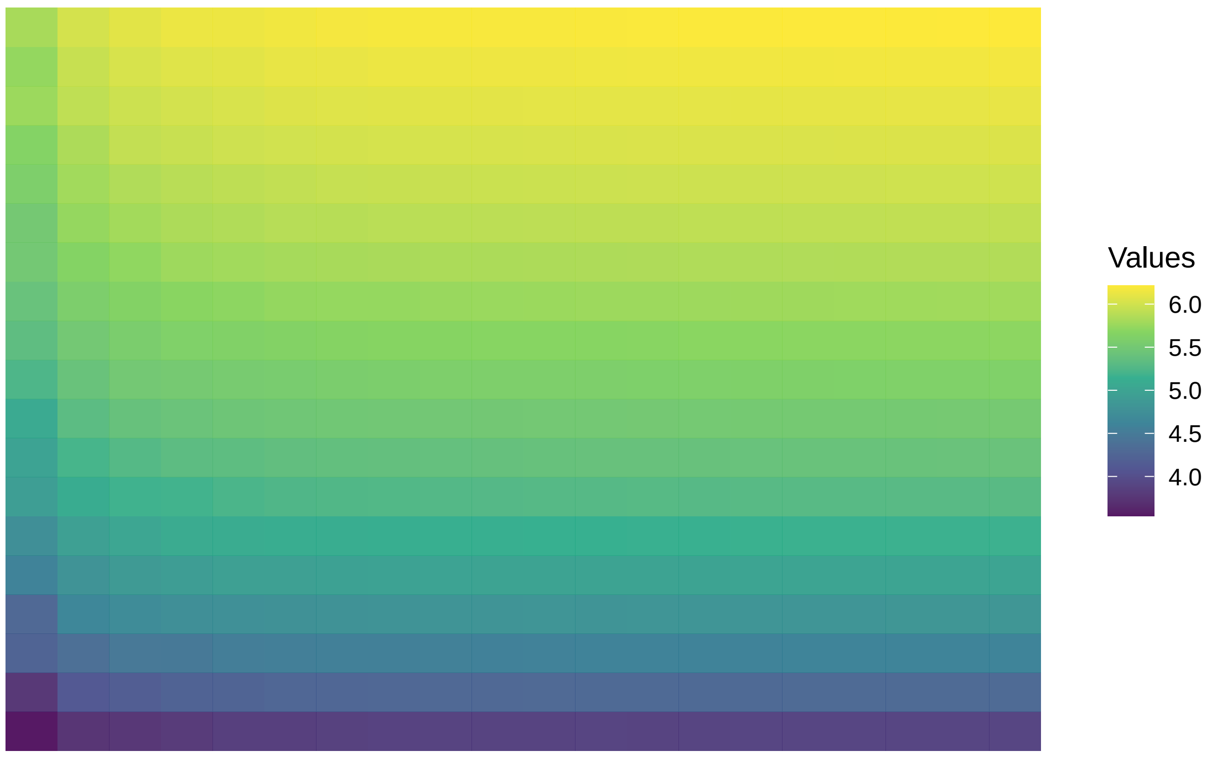
Now we can use it for all four indexes

```r
#visualise any of them as below or use series above
name="Shannon" #or Simpson, Gini, Pielou
series<-data_series(kind=name)
#reformat data with melt, I cannot fix the warning
melted<- melt(series)   #or any other series down the script
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```r
colnames(melted)<-c("Skewing","Clones","Values")
ggplot(melted, aes(Skewing, Clones, fill=Values))+
  geom_tile()+
  labs(title=name, x=element_blank(), y=element_blank())+
  scale_fill_viridis_c(alpha=0.9)+
  theme_void()
```

Shannon



If you want heatmap for other index, just change the name above to “Simpson”, “Gini” or “Pielou” and re-run the block. Next we move to the regression model. We will need all data for the heatmap series.

```
#generate data series for Shannon, Simpson, Gini, Pielou indexes
Sha_series<-data_series("Shannon")
Sim_series<-data_series("Simpson")
Pie_series<-data_series("Pielou")
Gin_series<-data_series("Gini")
```

We also load the remaining packages needed for this part

```
library(textshape) #for flatten
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:textshape':
##
##     combine
```

```
## The following object is masked from 'package:reshape':
##
##     rename
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Next we convert 2D matrix data into a 1D lists

```
#first we have to convert data from table format to a simple series
#this is called "flatten"
Ich<-as.double(flatten(as.list(Ich_series)))
Sha<-as.double(flatten(as.list(Sha_series)))
Sim<-as.double(flatten(as.list(Sim_series)))
Pie<-as.double(flatten(as.list(Pie_series)))
Gin<-as.double(flatten(as.list(Gin_series)))
```

Next we use regular linear model function to check how well used indexes predict the Ich

```
#here we run a trivial linear model where we check whether Ich can be explained
#note: 0 means ignore intercept (there is no meaning for it)
fit<-lm(Ich~ +Sha+Sim+ Pie+ Gin)
#here we check which parameters (if any) were essential
summary(fit)
```

```
##
## Call:
## lm(formula = Ich ~ +Sha + Sim + Pie + Gin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41853 -0.08729  0.00984  0.09452  0.37429
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -40.69875    3.77903 -10.770   <2e-16 ***
## Sha          -0.93685    0.03233 -28.979   <2e-16 ***
## Sim          47.40822    4.13714  11.459   <2e-16 ***
## Pie          -1.01178    1.12389  -0.900    0.369
## Gin           0.13216    0.16908   0.782    0.435
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1451 on 375 degrees of freedom
## Multiple R-squared:  0.8801, Adjusted R-squared:  0.8789
## F-statistic: 688.4 on 4 and 375 DF,  p-value: < 2.2e-16
```

```
anova(fit)
```

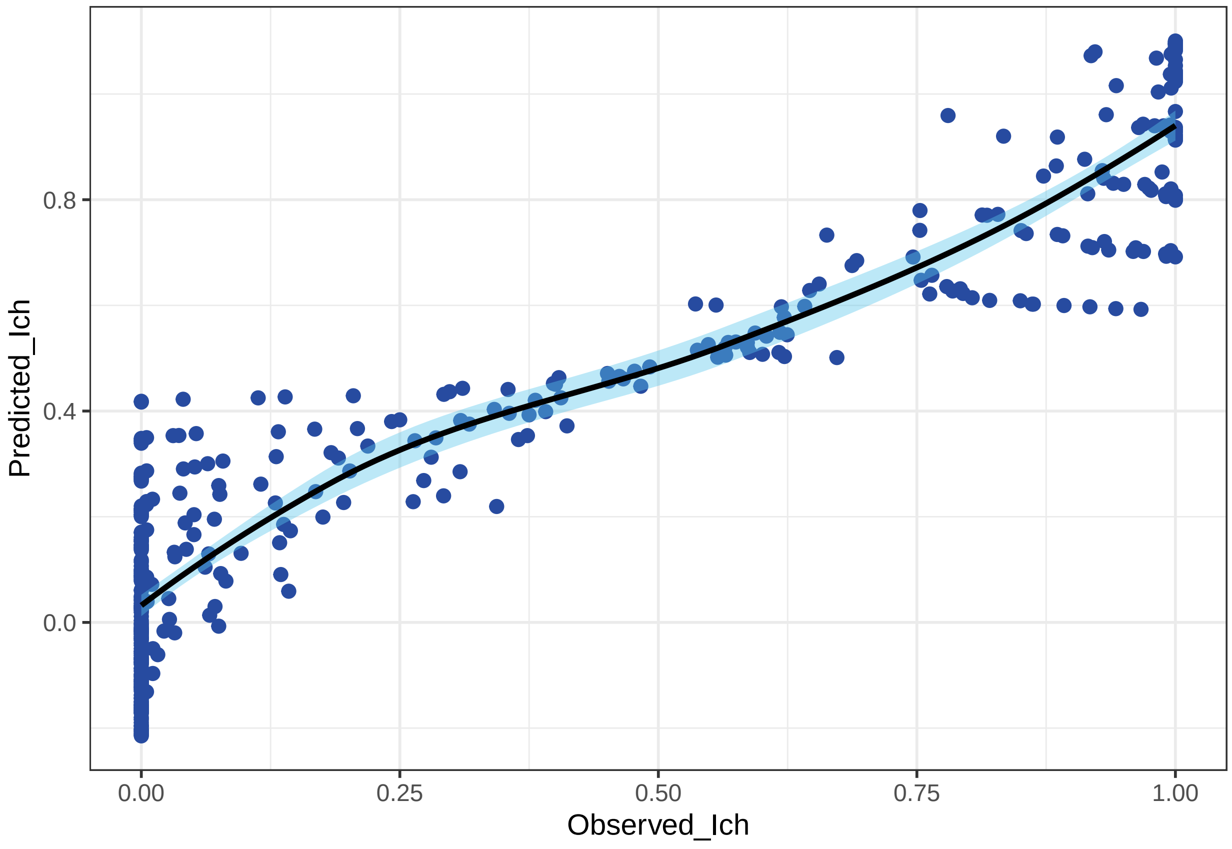
```
## Analysis of Variance Table
##
```

```
## Response: Ich
##             Df Sum Sq Mean Sq  F value    Pr(>F)
## Sha          1 54.996  54.996 2613.138 < 2.2e-16 ***
## Sim          1  2.596   2.596  123.351 < 2.2e-16 ***
## Pie          1  0.351   0.351   16.697 5.367e-05 ***
## Gin          1  0.013   0.013    0.611    0.4349
## Residuals 375  7.892   0.021
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Obviously, Ich is not the same as ano of the indexes. However it is not completely different either. R-squared value shows how well the selected model explains the Ich. We can also visualise this using two plots below

```
####Figure 2A----
#the same in ggplot style
data<-data.frame(cbind(Ich,predict(fit)))
colnames(data)<-c("Observed_Ich","Predicted_Ich")
ggplot(data,aes(x=Observed_Ich, y=Predicted_Ich))+
  geom_point(size = 2, color = "#274ba0")+
  geom_smooth(color = "black", fill = "#59C7EB")+
  theme_bw()
```

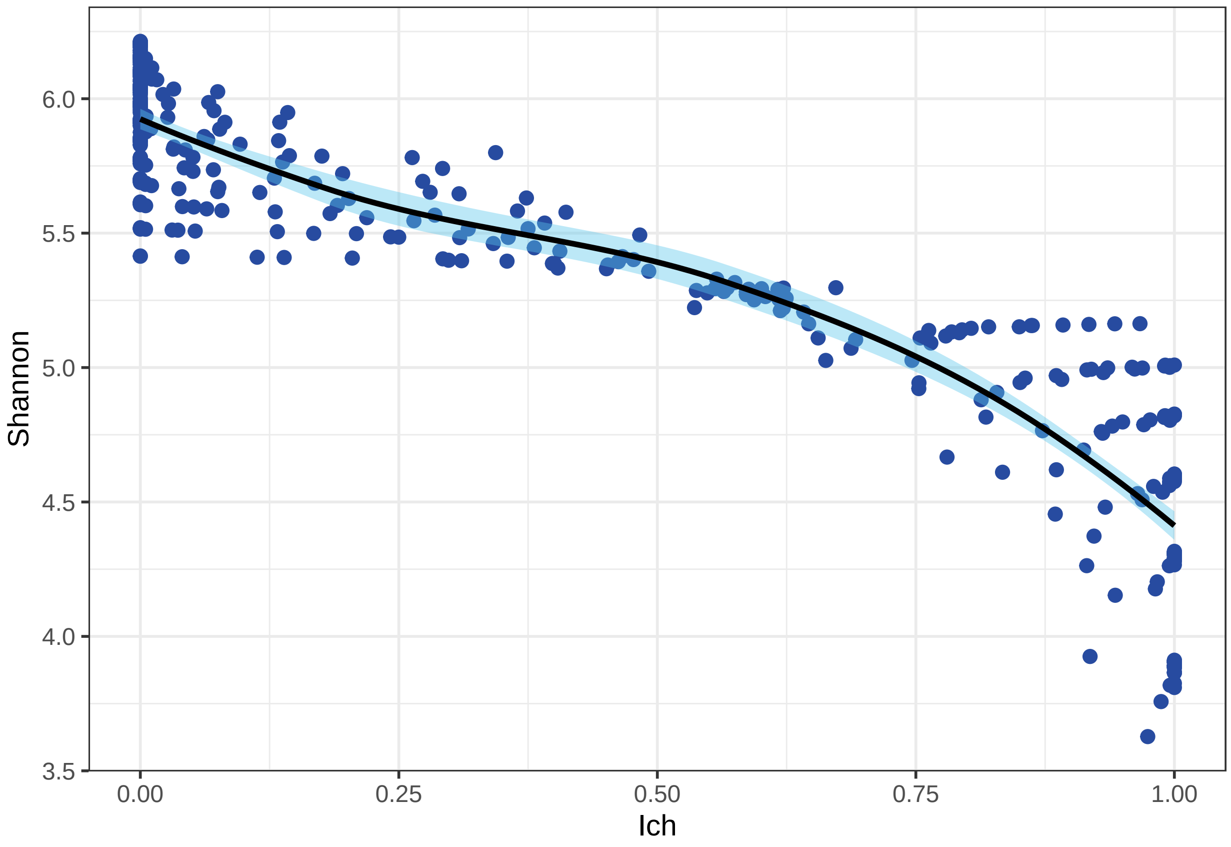
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



And the last figure

```
data2<-data.frame(cbind(Ich,Sha))
colnames(data2)<-c("Ich","Shannon")
ggplot(data2, aes(x=Ich, y=Shannon))+
  geom_point(size = 2, color = "#274ba0")+
  geom_smooth(color = "black", fill = "#59C7EB")+
  theme_bw()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



#this were all figures for the manuscript