Data for Clustering

	6AM	A1M	A2MG	AAGP	AAT	Abeta42	ACETA	ACP	ACTH	Active B12	AFP	ALB	ALP	ALT	AMH	AMIK	AM
3170001157	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	400
3170001168	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	0	0	600
3170001174	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	600
3170002932	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3170004091	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3170006485	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3170006528	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3170006615	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
3170006948	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3170006955	0	0	0	400	200	0	0	0	0	0	0	0	8000	6240	0	0	0
3170006986	0	0	0	0	0	0	0	0	0	0	0	0	400	0	0	0	200
3170007470	0	0	0	0	0	0	195	0	0	0	0	13650	12728	14243	0	0	0
3170008470	0	0	0	0	0	0	0	0	0	0	0	0	2000	2500	0	0	0
3170008566	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Row: Accounts
Col: Parameters
Cell: Number of Reportables

```
ChannelFillNested=reactive({
  RevenueReportables_Channelfill=input$RevenueReportables_Channelfill
  flexiPcSpreadSub=flexiPcSpreadSub()
  NestedCluster=input$NestedCluster
  # cutreeParam=as.numeric(input$cutreeParam)
  PotentialScorethreshold=as.numeric(input$PotentialScore)
  ClusterSize=as.numeric(input$ClusterSize)
  if(NestedCluster==TRUE)
  {
    flexiPCSpreadSubM=flexiPcSpreadSub[,names(which(colSums(flexiPcSpreadSub[,setdiff(colnames(flexiPcSpreadSub),c('Customer.Num'))])>0))]
    if (nrow(flexiPCSpreadSubM)>=3)
    {
      flexiPCSpreadSubM[flexiPCSpreadSubM<0]=0
      flexiPCSpreadSubM=scale(flexiPCSpreadSubM)

      #### Parameter Clustering
      paramClust=hclust(dist(t(flexiPCSpreadSubM)),method = 'ward.D2')

      Channelfill=data.frame()
      for (cutreeParam in c(0.1, 0.4, 0.7,0.85, 1, 1.3,-0.1,-0.4, -0.7, -1, -1.3))
      {

        h = (mean(paramClust$height+cutreeParam*sd(paramClust$height)))
        if (h<0)
        {
          h=0
        }

        membParam <- cutree(paramClust, h = (mean(paramClust$height+cutreeParam*sd(paramClust$height))))
        membParamTab=table(membParam)

        #### Channel filling activities. Find islands
        flexiChannelFillA_Param_tmp=NULL

        for (i in which(membParamTab>=ClusterSize))
        {
          clusterParam=names(membParamTab[i])
          paramMemb=names(which(membParam==clusterParam))
          flexiPcSpreadSubMSub=flexiPCSpreadSubM[,paramMemb]
          #### Hospital Clustering
          hospClust=hclust(dist(flexiPcSpreadSubMSub),method = 'ward.D2')

          flexiChannelFillA_Hosp_tmp=NULL

          for (cutreeHosp in c(0.1, 0.4, 0.7,0.85, 1, 1.3,-0.1,-0.4, -0.7, -1, -1.3))
          {
            h = (mean(hospClust$height+cutreeHosp*sd(hospClust$height)))
            if (h<0)
            {
              h=0
            }
            membHosp <- cutree(hospClust, h = h)
            hospClustOrder=hospClust$labels[hospClust$order]
            membHospTab=table(membHosp)

            flexiChannelFillA=NULL

            for (j in which(membHospTab>=ClusterSize))
            {
              clusterHosp=names(membHospTab[j])
              hospMemb=names(which(membHosp==clusterHosp))
              flexiPcSpreadSubSub=flexiPcSpreadSub[which(rownames(flexiPcSpreadSub) %in% hospMemb),paramMemb]
              flexiPcSpreadSubSub=flexiPcSpreadSubSub[,names(which(colSums(flexiPcSpreadSubSub)!=0)),drop=FALSE]

              if (nrow(flexiPcSpreadSubSub)>0)
              {
                if (length(which(flexiPcSpreadSubSub==0))!=nrow(flexiPcSpreadSubSub)*ncol(flexiPcSpreadSubSub) & length(which(flexiPcSpreadSubSub!=0))!=
                {

                PotentialScore=round(length(which(flexiPcSpreadSubSub!=0))/length(which(flexiPcSpreadSubSub==0)),2)

                  if (PotentialScore>PotentialScorethreshold)
                  {
                    fillTmp=data.frame(which(flexiPcSpreadSubSub ==0, arr.ind = T))
                    fillTmp$Param=colnames(flexiPcSpreadSubSub)[fillTmp$col]
                    fillTmp$Customer.Num=rownames(flexiPcSpreadSubSub)[fillTmp$row]
                    paramMedian=apply(flexiPcSpreadSubSub,2,function(x) median(x[which(x>0)]))
                    fillTmp$ExpAvgReportables=paramMedian[fillTmp$Param]
                    fillTmp$PotentialScore=PotentialScore
                    fillTmp$scoreDetails=sprintf("%d = %d vs %d",nrow(flexiPcSpreadSubSub)*ncol(flexiPcSpreadSubSub),length(which(flexiPcSpreadSubSub!=
                    fillTmp$clusterParamList=list(paramMemb)
                    fillTmp$clusterCustomerList=list(hospMemb)
                    fillTmp$row=NULL
                    fillTmp$col=NULL
                    names(fillTmp)[which(names(fillTmp)=='ExpAvgReportables')]=paste0('ExpAvg',RevenueReportables_Channelfill)
                    fillTmp=fillTmp[which(fillTmp[,paste0('ExpAvg',RevenueReportables_Channelfill)]>0),]
                    flexiChannelFillA=rbind(flexiChannelFillA,fillTmp)
                  }
                }
              }
            }

            flexiChannelFillA_Hosp_tmp=rbind(flexiChannelFillA_Hosp_tmp,flexiChannelFillA)
          }

          flexiChannelFillA_Param_tmp=rbind(flexiChannelFillA_Param_tmp,flexiChannelFillA_Hosp_tmp)
        }

        Channelfill=rbind(Channelfill,flexiChannelFillA_Param_tmp)
      }
      ChannelFillNested=Channelfill
      ChannelFillNested <- ChannelFillNested[order(ChannelFillNested$Param,ChannelFillNested$Customer.Num, -(ChannelFillNested$PotentialScore) ), ]
      ChannelFillNested=ChannelFillNested[!duplicated(ChannelFillNested[,c('Param','Customer.Num')]),]
    }else{
      ChannelFillNested=NULL
    }
  }
})
```

Inputs

Data cleaning and scaling

apply hierarchical clustering on Parameters

run different height of Dendrogram to get different clusters of Parameters

For each of the parameter group, if number of parameters >=ClusterSize

Hierarchical clustering will applied to all Accounts within this group of parameters

Same here, run different height of Dendrogram to get different clusters of Accounts

Now within each of the parameter group, we will have multiple groups of Accounts

Under the specific parameter group, for each group of the account, we will do the channel filling process

within 1 group of parameter and 1 group of account, check if this cluster have all 0, or all have reportables, we will not do channel filling on this cluster.

Number of cells have reportables / Number of cells have no reportables

if greater than threshold

fill in the empty cell by column meadian