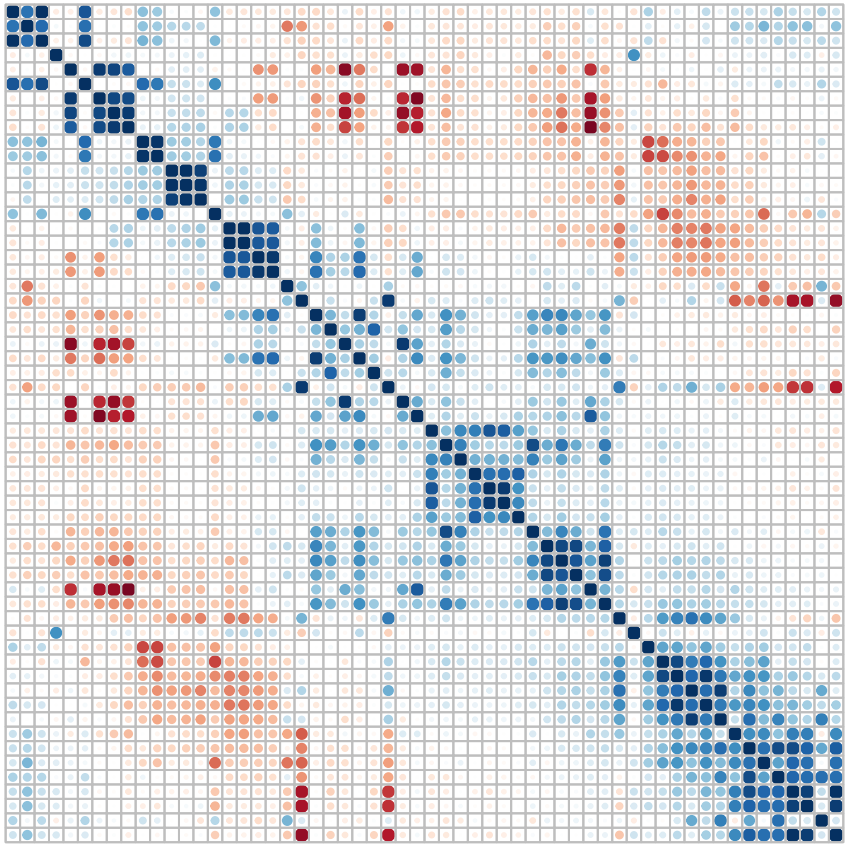
corr_rndf

Random forest base on clear from correlated variables.

Data pre-processing

First, loock at correlation between our 58 soil and clime variables

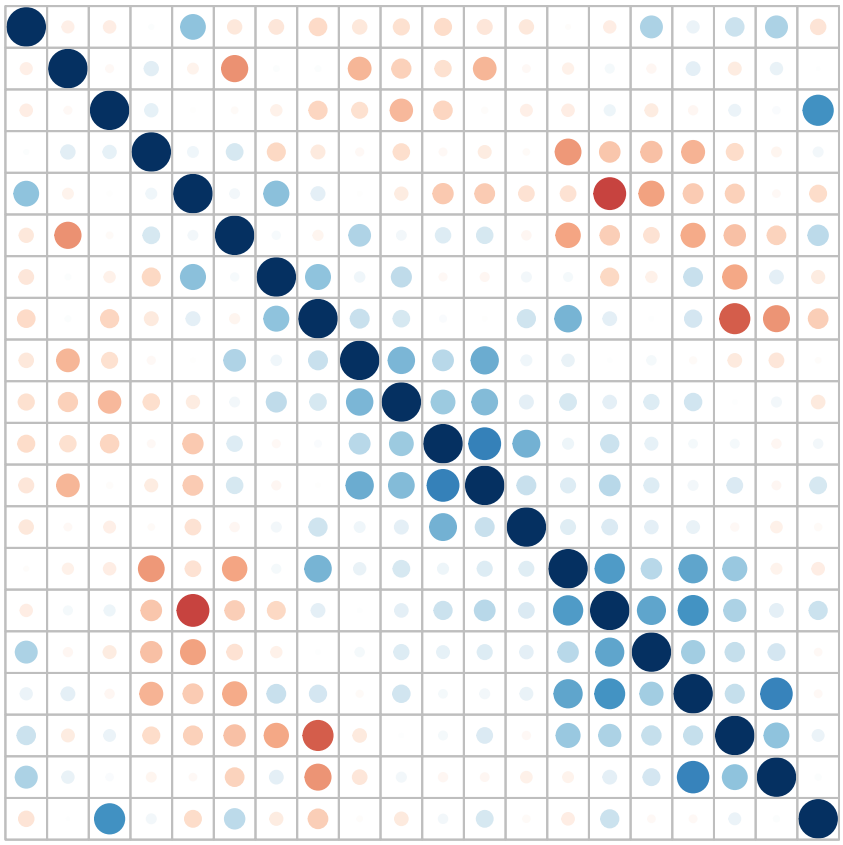
```
M <- cor(just.parametrs)
corrplot(M, order = "AOE", cl.pos = "n", tl.pos = "n")
```



We have many strong corellated variables which may decreas model accuracy. Delete correlated using caret library.

```
cor.var <- caret::findCorrelation(M, cutoff = 0.7)
clear.just.parametrs <- just.parametrs[,-cor.var]

corrplot(cor(clear.just.parametrs), order = "AOE", cl.pos = "n", tl.pos = "n")
```



We have next variables for model

```
names(clear.just.parametrs)
```

```
##  [1] "bio2"       "bio3"       "bio4"       "bio8"       "bio10"
##  [6] "bio13"      "bio15"      "bio18"      "solar_max"  "wind_mean"
## [11] "vapr_mean"  "T_GRAVEL"   "T_SAND"     "T_CEC_SOIL" "T_CACO3"
## [16] "S_GRAVEL"   "S_CEC_CLAY" "S_BS"       "S_CASO4"    "S_ESP"
```

Next step is transform admixture vectors to factor variables. Most simple way for achieved that is choose class for samples according to some threshold.

```
lbound <- .7
groups <- apply(all_data[, c(1:20)], 1, function(current.row) {
    group <- ((current.row > lbound) * c(1:20))[current.row > lbound]
    if (length(group) == 0) {
        return(NA)
    }
    return(group)
})
str(groups)
```

```
##  int [1:1048] 17 17 NA NA NA NA 8 8 8 8 ...
```

```
prepared.data <- data.frame(group = groups)
prepared.data <- na.omit(cbind(prepared.data, clear.just.parametrs))
prepared.data$group <- factor(prepared.data$group)
str(prepared.data)
```

```
## 'data.frame':    771 obs. of  21 variables:
##  $ group     : Factor w/ 20 levels "1","2","3","4",..: 17 17 8 8 8 8 8 10 10 10 ...
##  $ bio2      : int  63 63 87 85 87 85 100 115 115 115 ...
##  $ bio3      : int  37 37 30 30 30 30 31 29 29 29 ...
##  $ bio4      : int  3830 3830 7068 6994 7068 6985 7580 9960 9960 9960 ...
##  $ bio8      : int  69 69 164 158 164 158 179 200 200 200 ...
##  $ bio10     : int  158 158 164 158 164 158 179 220 220 220 ...
##  $ bio13     : int  134 134 85 88 85 89 124 104 104 104 ...
##  $ bio15     : int  31 31 40 41 40 39 41 26 26 26 ...
##  $ bio18     : int  189 189 241 251 241 254 355 298 298 298 ...
##  $ solar_max : int  19635 19635 19769 19718 19769 19766 19995 21829 21829 21829 ...
##  $ wind_mean : num  5.11 5.11 3.66 3.92 3.66 ...
##  $ vapr_mean : num  1.117 1.117 0.856 0.838 0.856 ...
##  $ T_GRAVEL  : num  4.5 4.5 6.6 6.6 6.6 6.6 6.5 3.75 3.75 3.75 ...
##  $ T_SAND    : num  40 40 43 43 43 43 41.5 32.5 32.5 32.5 ...
##  $ T_CEC_SOIL: num  13.8 13.8 12.2 12.2 12.2 ...
##  $ T_CACO3   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ S_GRAVEL  : num  9.25 9.25 13.25 13.25 13.25 ...
##  $ S_CEC_CLAY: num  45.5 45.5 44.2 44.2 44.2 ...
##  $ S_BS      : num  83.2 83.2 74.2 74.2 74.2 ...
##  $ S_CASO4   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ S_ESP     : num  1.25 1.25 1.75 1.75 1.75 1.75 1.5 2.75 2.75 2.75 ...
##  - attr(*, "na.action")= 'omit' Named int  3 4 5 6 20 21 32 37 39 44 ...
##   ..- attr(*, "names")= chr  "3" "4" "5" "6" ...
```

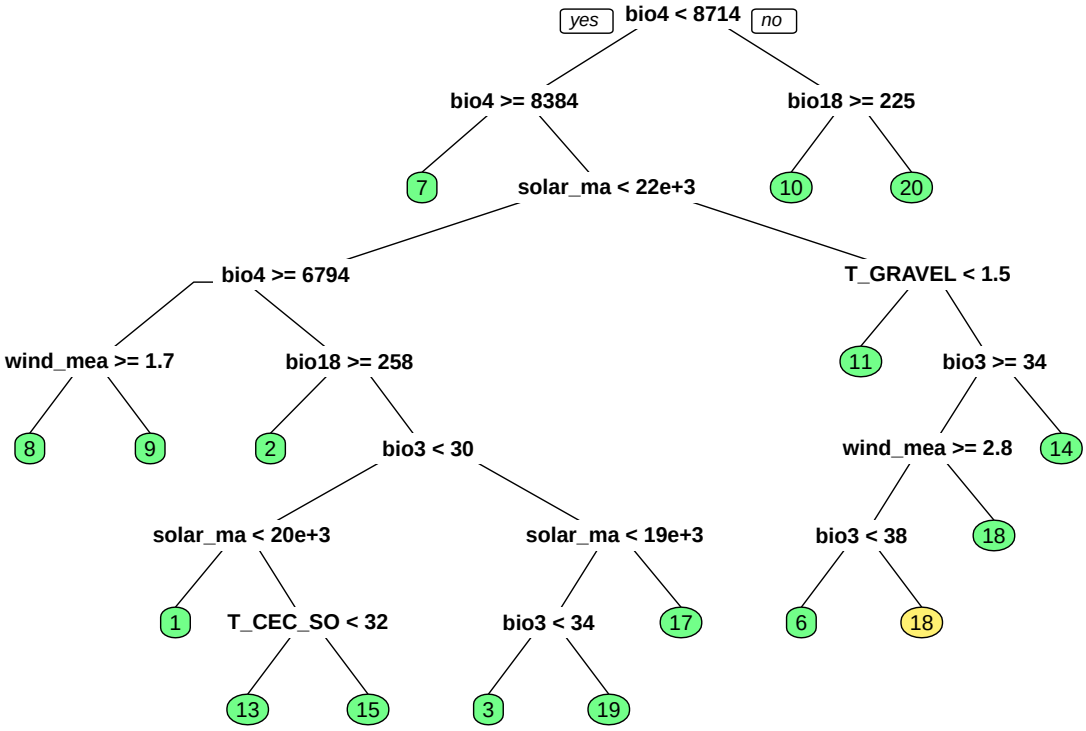
Prediction models base on original data.

Now make random forest base on this preprocessed data but first thing create simple design tree.

```
set.seed(666)

train.indexes <- sample(1:nrow(prepared.data), 0.5 * nrow(prepared.data))
train <- prepared.data[train.indexes, ]
test <- prepared.data[-train.indexes, ]

res.tree <- rpart(group ~ ., data = train, method = 'class')
prp(res.tree, box.palette="RdYlGn")
```



And calculate accuracy of a desigion tree

```
t_pred <- predict(res.tree, test, type = 'class')
confMat <- table(test$group, t_pred)
sum(diag(confMat)) / sum(confMat)
```

```
## [1] 0.7020725
```

Not bad, but what about random forest?

```
res.forest <- randomForest(group ~ ., data = train)
t_pred.forest <- predict(res.forest, test)
confMat <- table(test$group, t_pred.forest)
sum(diag(confMat)) / sum(confMat)
```

```
## [1] 0.8056995
```

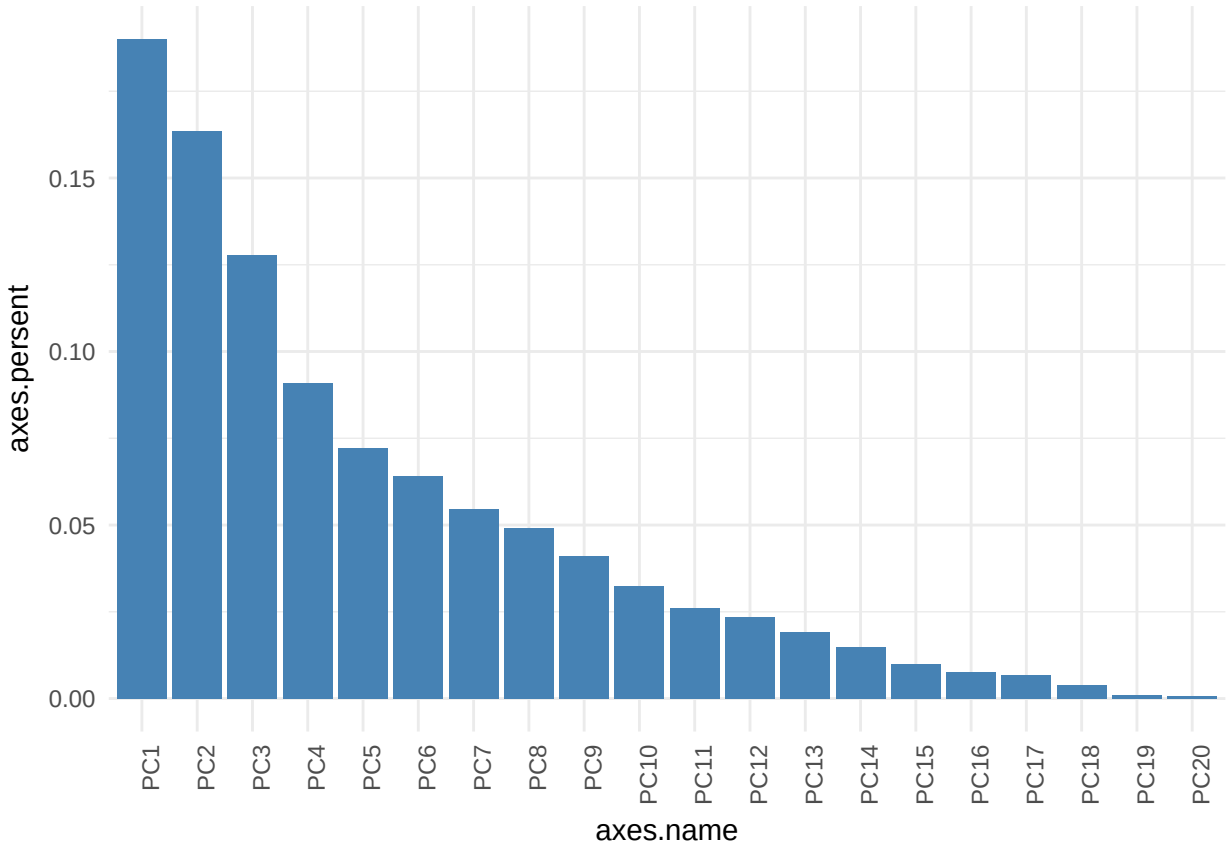
PCA for pre - processed data

Look at PCA of our dataset.

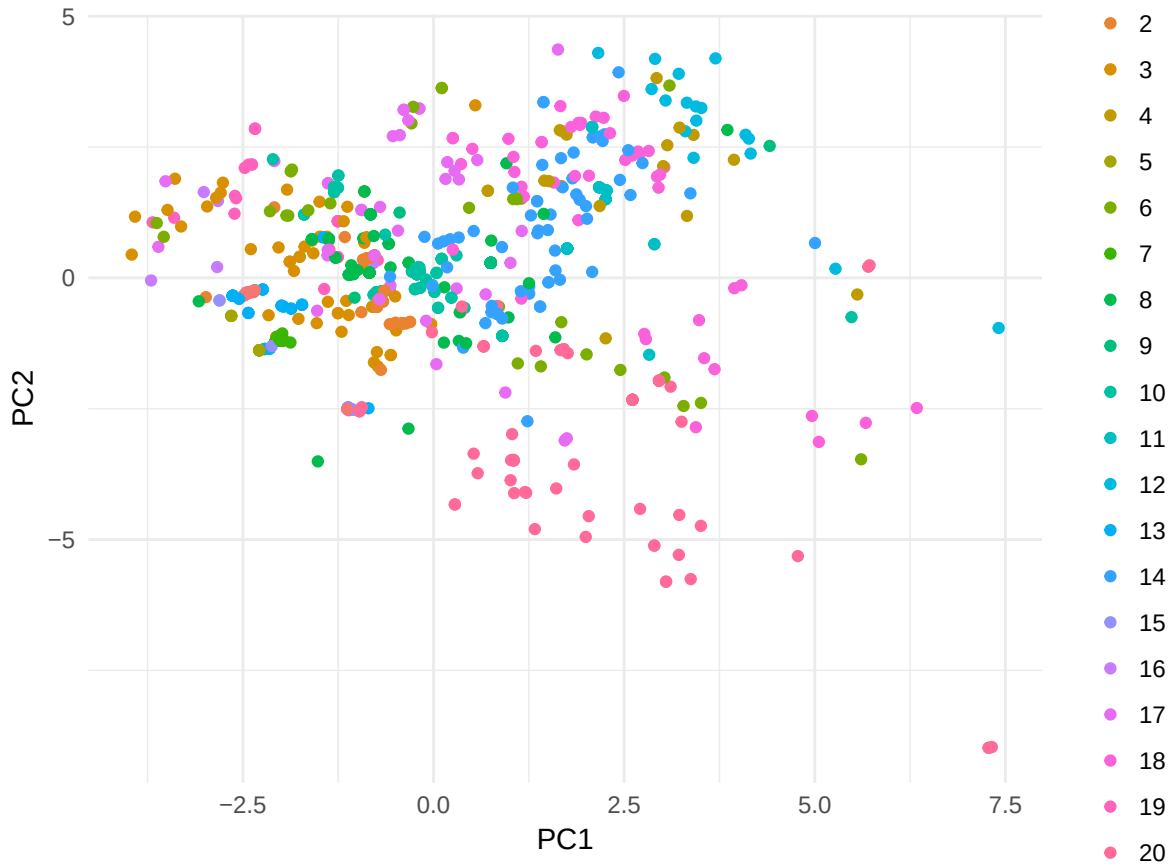
```
clear.param.pca <- prcomp(prepared.data[,-1], scale=T, center=T)
imp <- summary(clear.param.pca)$importance
pca.axes.data <- data.frame(
        axes.name = sort(names(imp[2, ])),
        axes.persent = as.vector(imp[2, ]))

levels(pca.axes.data$axes.name) <- unlist(lapply(1:20, function(i) {paste0("PC", toString(i))}))
```

```
ggplot(pca.axes.data,
    aes(x = axes.name, y = axes.persent)) +
    geom_bar(stat="identity", fill="steelblue") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
pca.result <- as.data.frame(cbind(prepared.data$group, clear.param.pca$x))
ggplot(pca.result, aes(x = PC1, y = PC2, colour = factor(V1))) +
    geom_point()+
    labs(x = 'PC1', y = 'PC2') +
    theme_minimal()
```



Apply random forest model to PCA component

```
# train.pca.data <- pca.result[train.indexes, ]
# test.pca.data <- pca.result[-train.indexes, ]
#
# pca.res.forest <- randomForest(V1 ~ ., data = train.pca.data)
# pca.t_pred.forest <- predict(pca.res.forest, test.pca.data)
# pca.confMat <- table(test.pca.data$V1, pca.t_pred.forest)
# sum(diag(pca.confMat)) / sum(pca.confMat)
```