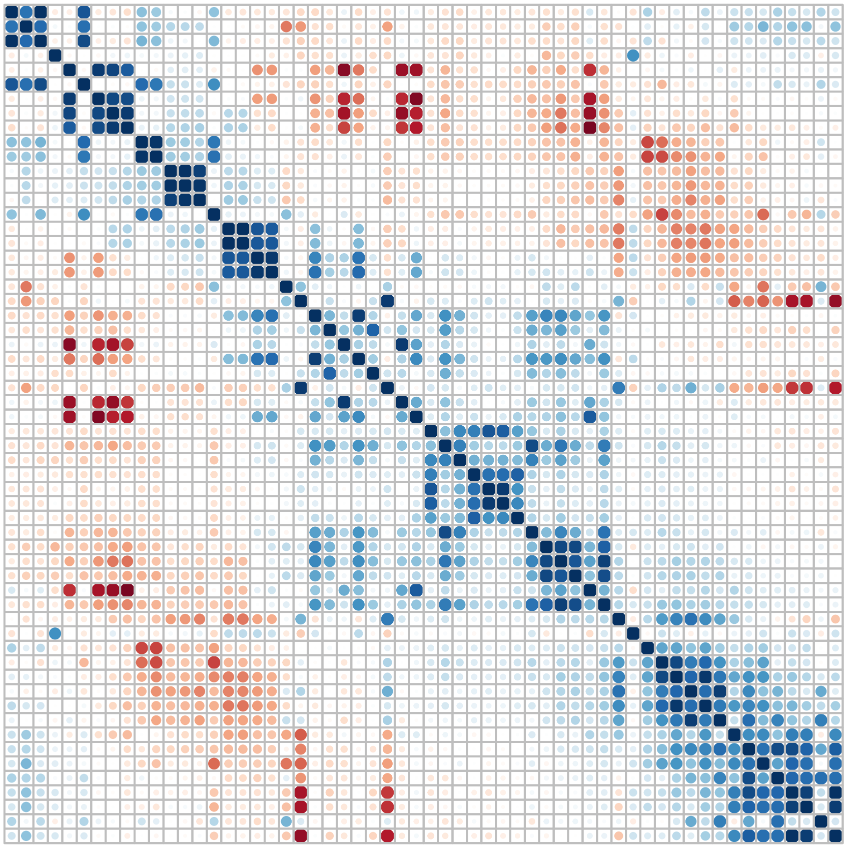
corr_rndf

Random forest base on clear from correlated variables.

Data pre-processing

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

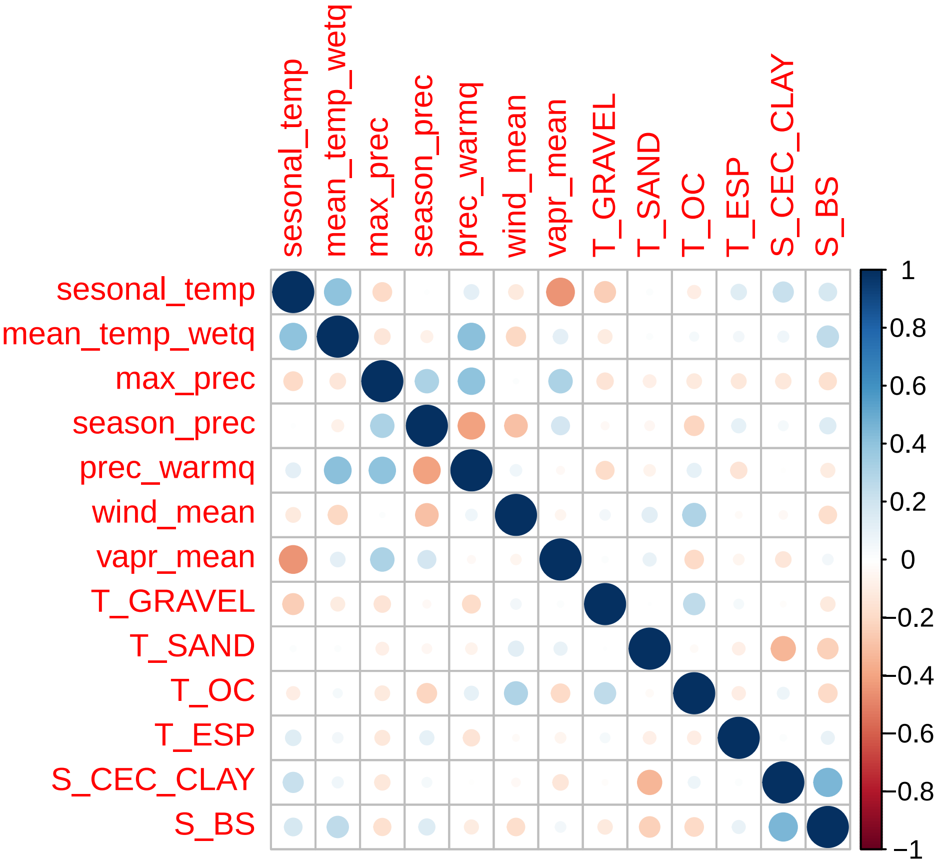
```
M <- cor(just.parametrs)
corrplot(M, order = "AOE", cl.pos = "n", tl.pos = "n")
```



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

```
cor.var <- caret::findCorrelation(M, cutoff = 0.5)
clear.just.parametrs <- just.parametrs[,-cor.var]

corrplot(cor(clear.just.parametrs))
```



We have next variables for model

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

```
##  [1] "sesonal_temp"   "mean_temp_wetq" "max_prec"       "season_prec"
##  [5] "prec_warmq"     "wind_mean"      "vapr_mean"      "T_GRAVEL"
##  [9] "T_SAND"         "T_OC"           "T_ESP"          "S_CEC_CLAY"
## [13] "S_BS"
```

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(1:14)], 1, function(current.row) {
    group <- ((current.row > lbound) * c(1:14))[current.row > lbound]
    if (length(group) == 0) {
        return(NA)
    }
    return(group)
})
str(groups)
```

```
##  int [1:1048] 6 6 6 NA NA NA 11 11 11 11 ...
```

```
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':    710 obs. of  14 variables:
##  $ group          : Factor w/ 14 levels "1","2","3","4",..: 6 6 6 11 11 11 11 11 7 7 ...
##  $ sesonal_temp   : int  3830 3830 5862 7068 6994 7068 6985 7580 9960 9960 ...
##  $ mean_temp_wetq: int  69 69 34 164 158 164 158 179 200 200 ...
##  $ max_prec       : int  134 134 79 85 88 85 89 124 104 104 ...
##  $ season_prec    : int  31 31 12 40 41 40 39 41 26 26 ...
##  $ prec_warmq     : int  189 189 209 241 251 241 254 355 298 298 ...
##  $ wind_mean      : num  5.11 5.11 3.39 3.66 3.92 ...
##  $ vapr_mean      : num  1.117 1.117 0.98 0.856 0.838 ...
##  $ T_GRAVEL       : num  4.5 4.5 11 6.6 6.6 6.6 6.6 6.5 3.75 3.75 ...
##  $ T_SAND         : num  40 40 78 43 43 43 43 41.5 32.5 32.5 ...
##  $ T_OC           : num  1.24 1.24 1.68 1.37 1.37 ...
##  $ T_ESP          : num  1.25 1.25 1.5 2.2 2.2 2.2 2.2 1.5 2 2 ...
##  $ S_CEC_CLAY     : num  45.5 45.5 40.5 44.2 44.2 ...
##  $ S_BS           : num  83.2 83.2 28.5 74.2 74.2 ...
##  - attr(*, "na.action")=Class 'omit'  Named int [1:338] 4 5 6 20 32 37 38 39 45 46 ...
##   .. ..- attr(*, "names")= chr [1:338] "4" "5" "6" "20" ...
```

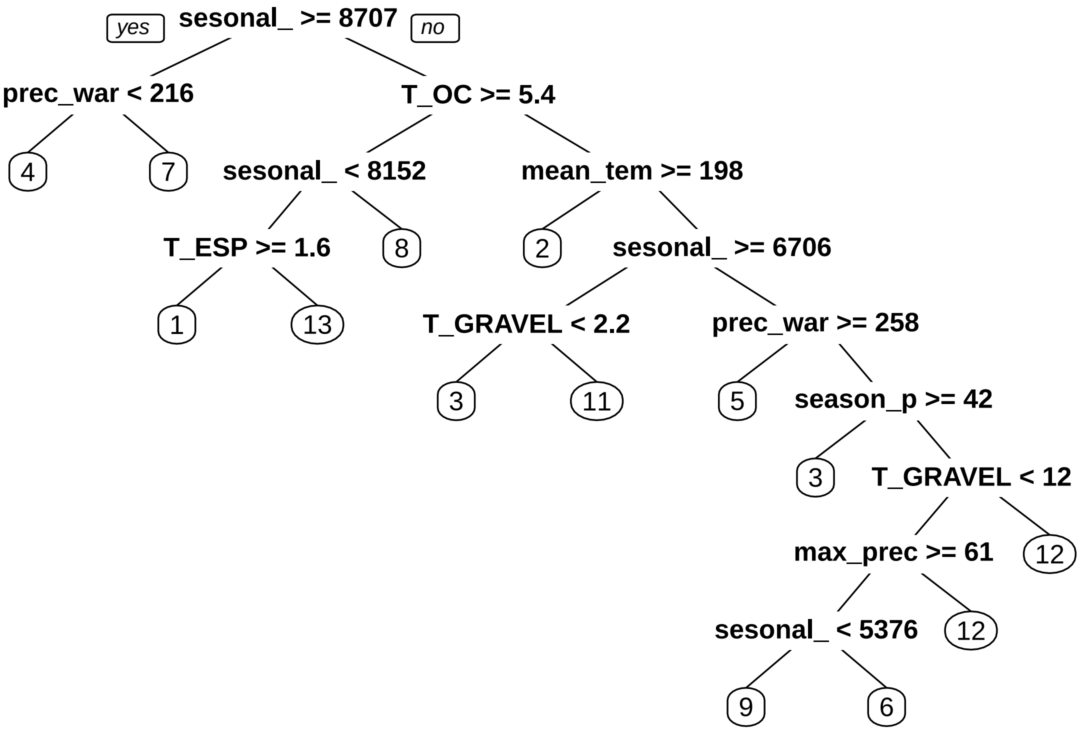
Prediction models base on original data.

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)
```



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)
```

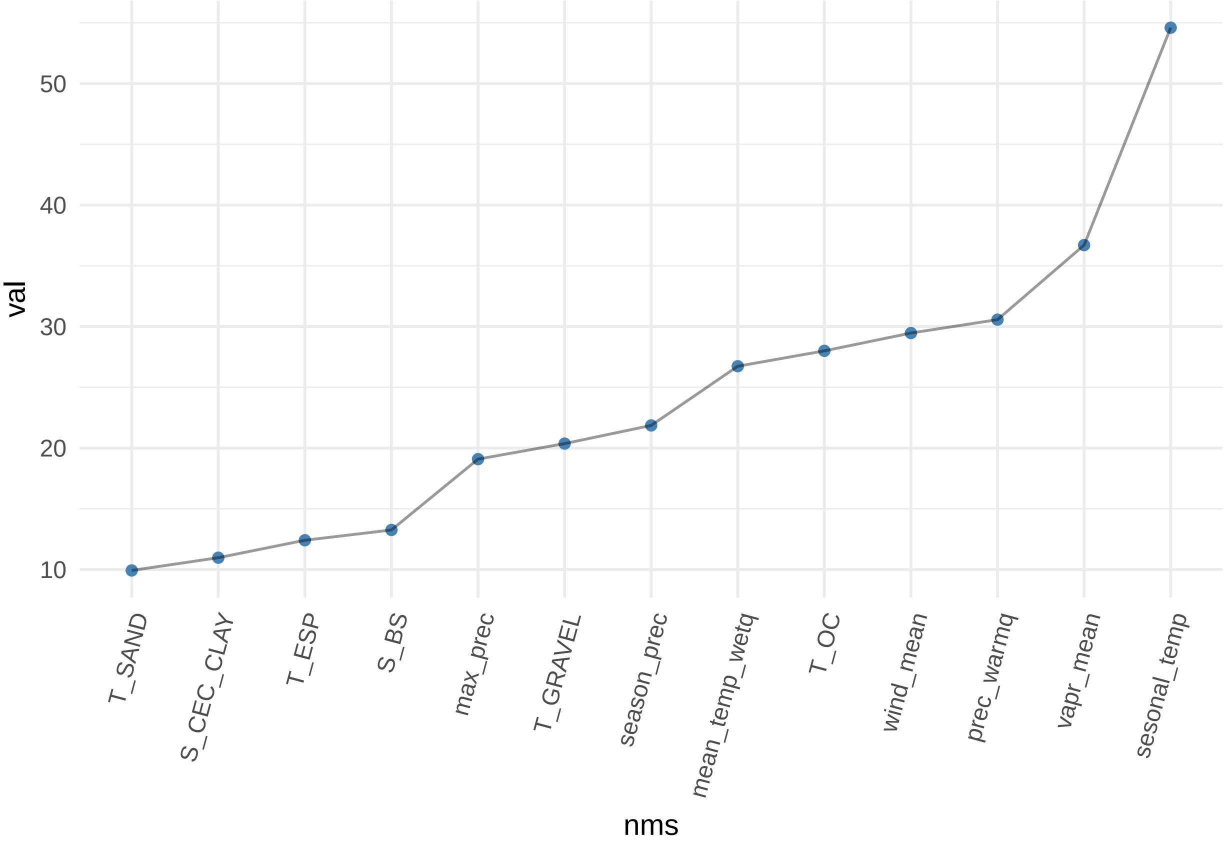
```
## [1] 0.7408451
```

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] 0.856338
```

```
important.dot.plot(res.forest)
```



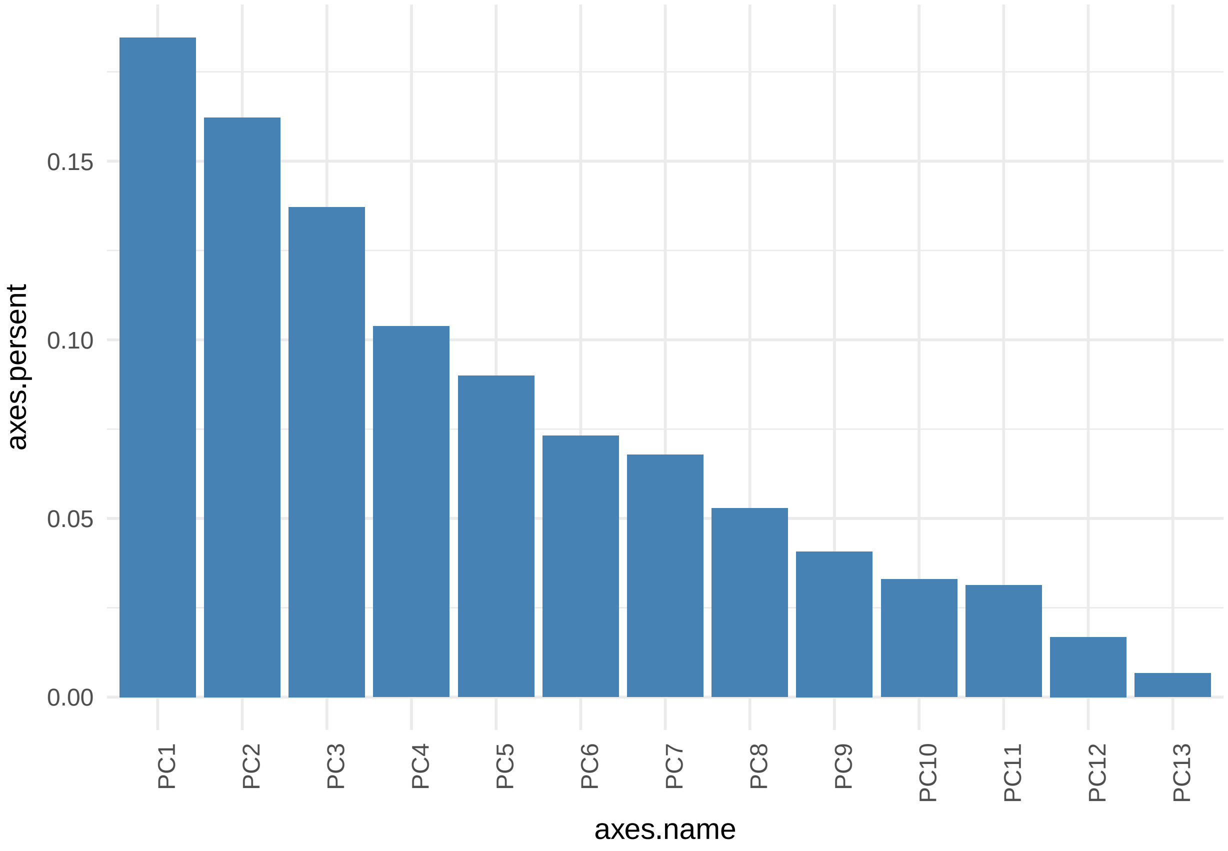
PCA for pre - processed data

Look at PCA of our dataset.

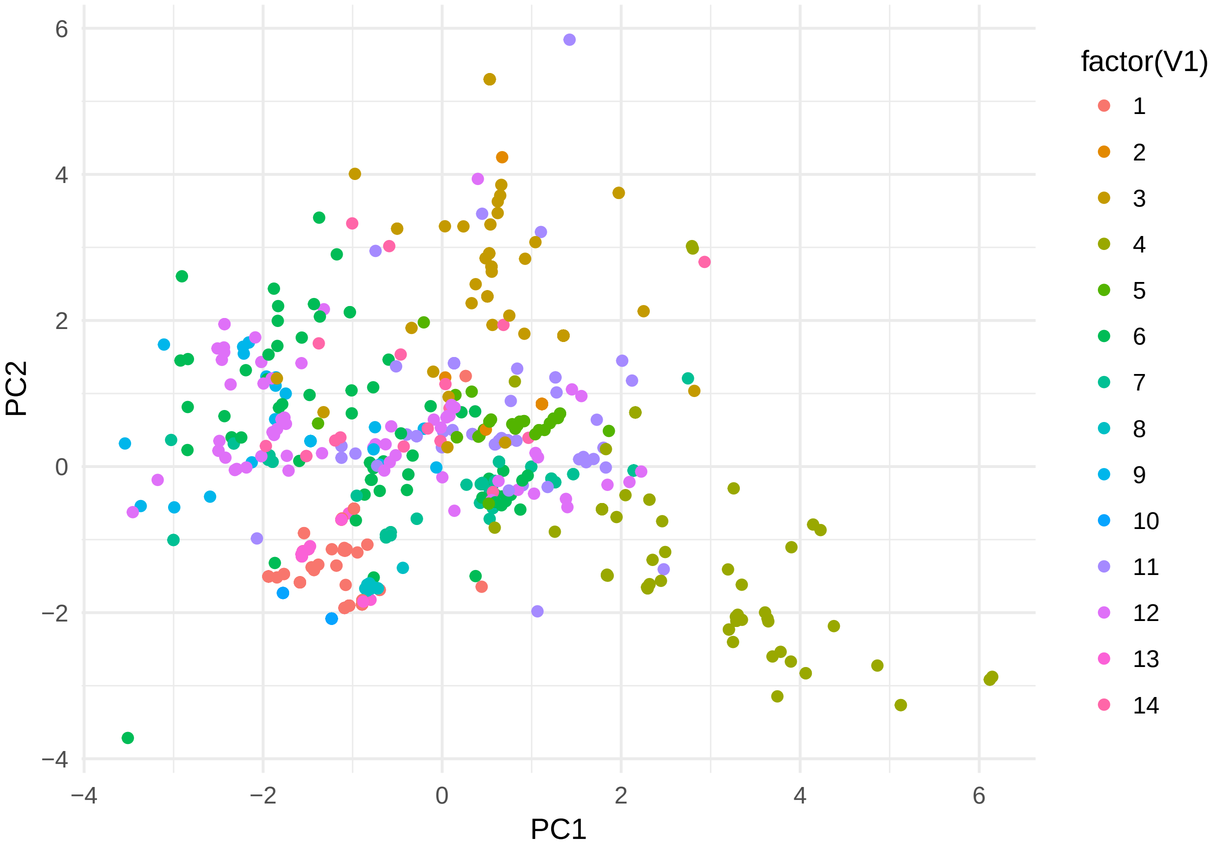
```
clear.param.pca <- prcomp(prepared.data[,-1], scale=T, center=T)
imp <- summary(clear.param.pca)$importance
pca.axes.data <- data.frame(
        axes.name = sort(names(imp[2, ])),
        axes.persent = as.vector(imp[2, ]))
```

```r
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))
```



```r
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, ]
train.pca.data$V1 <- factor(train.pca.data$V1)
test.pca.data$V1 <- factor(test.pca.data$V1)

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)
```

```
## [1] 0.8422535
```