The `Rtropical` package

In this vignette, we will demonstrate the main capabilities of the `Rtropical` package. We'll show the pipeline to analyze phylogenetic tree data with these methods.

We start by importing the `Rtropical` library.

```
library(Rtropical)
library(ape)
library(e1071)
```

Tropical SVM

We will carry out tropical SVM with the simulated tree data in the package. This data set contains 300 trees with the first 150 assumed coming from one category and the rest from the other. We firstly prepare the data by transforming it into data matrix and splitting it into training and testing set.

```
set.seed(101)
data(sim_trees)
treevecs = do.call("rbind", lapply(sim_trees, vec.fun))
labels = as.factor(rep(c(1, 2), each = nrow(treevecs)/2))

# generate training data set
trn_ind = sample(1: nrow(treevecs), nrow(treevecs)*0.8)
x = treevecs[trn_ind, ]
y = labels[trn_ind]

# generate testing data set
newx = treevecs[-trn_ind, ]
newy = labels[-trn_ind]
```

Next, we run the tropical svm.

```
# run tropical svm
start = Sys.time()
trop_fit <- tropsvm(x, y, auto.assignment = TRUE)
end = Sys.time()
# predict for testing data
trop_pred <- predict(trop_fit, newx)
# compute classification accuracy
sum(as.vector(trop_pred) == newy)/length(newy)
#> [1] 0.45
print(paste("The running time is: ", round(end - start, digits = 3), "s", sep = ""))
#> [1] "The running time is: 0.295s"
```

The accuracy seems to be worse because this function does not tune for a good classification method. In the cases when `tropsvm` fails, we recommend to use `cv.tropsvm`. This function automatically carries out cross-validation to improve performance but it can take longer.

```
# tropical svm with cross-validation
start = Sys.time()
```

```
cv_trop_fit <- cv.tropsvm(x, y, nassignment = 500, parallel = TRUE)
end = Sys.time()
cv_trop_pred <- predict(cv_trop_fit, newx)
# compute classification accuracy for testing data
sum(cv_trop_pred == newy)/length(newy)
#> [1] 0.9
print(paste("The running time is: ", round(end - start, digits = 3), "min", sep = ""))
#> [1] "The running time is: 4.523min"
```

We can also run `svm` from `e1071` as a comparison:

```
svm_fit <- svm(x, y)
svm_pred <- predict(svm_fit, newx)
sum(svm_pred == newy)/length(newy)
#> [1] 0.5833333
```

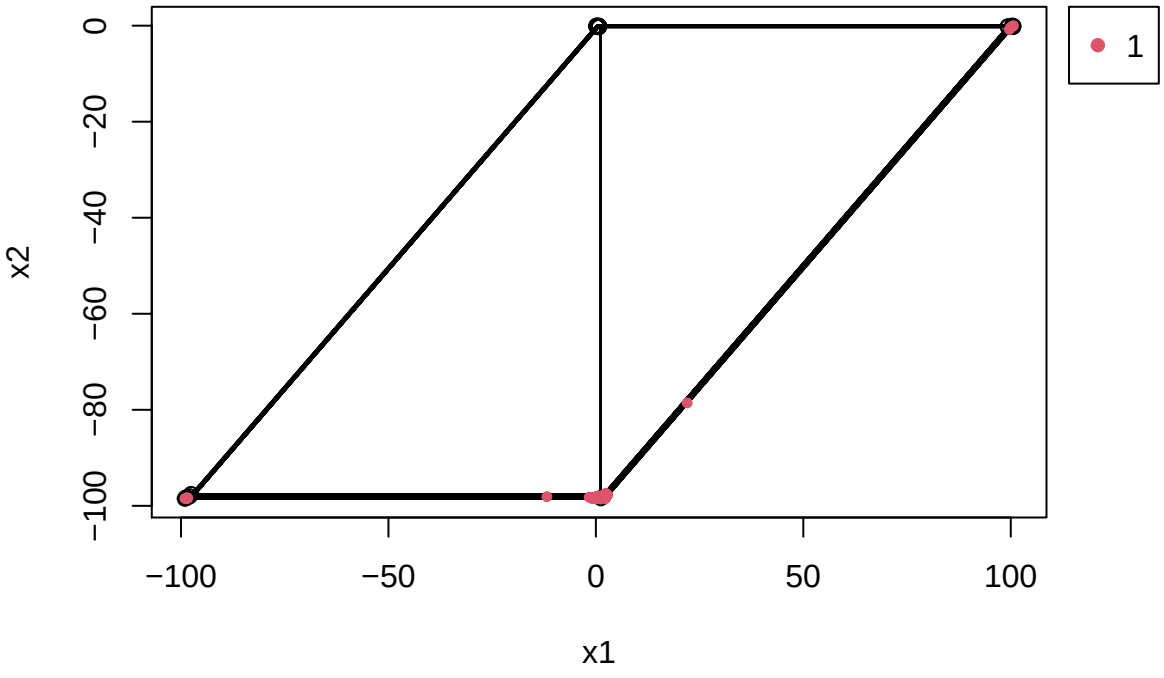
Tropical PCA

Now we analyze some actual phylogenetic tree data with tropical principal component analysis.

```
data(apicomplexa)
treevecs <- do.call("rbind", lapply(.compressTipLabel(apicomplexa), vec.fun))
pca_fit = troppca.poly(treevecs)
```

For second order principal component and the projection of data points, we can visualize on a 2D plane by isometric transformation.

```
plot(pca_fit)
```



Tropical Fermat-weber Point

Fermat-weber point can be regarded as the "tropical mean" of a data set. It minimizes the sum of distance to each point in a data set, which is also the zero-th order tropical principal component analysis. Unlike

tropical PCA with higher order, tropical fermat-weber point can be computed deterministically by `tropFW`.

```
tropFW(treevecs)
#> $fw
#>  [1] 0.350087 0.640838 0.680904 0.686252 0.973897 0.634037 0.649114 0.748865
#>  [9] 0.837696 0.820084 1.123781 0.754612 0.759945 0.857507 0.816025 1.039273
#> [17] 0.778886 0.789528 0.312610 1.155168 0.769828 0.780251 1.144641 0.760579
#> [25] 0.756364 1.101268 1.098565 0.000000
#>
#> $distsum
#> [1] 1228.93
```