randomForestSRC CHEAT SHEET

Basics
randomForestSRC is a fast OpenMP and memory efficient package for fitting random forests (RF) for univariate, multivariate, unsupervised, survival, competing risks, class imbalanced classification and quantile regression. A basic grow call is of the form:

\[ \text{rfsrc(formula, data, ntree, mtry, nodesize)} \]

Grow your RF through \text{rfsrc}, specify your model in \text{formula}, provide your data frame in \text{data} and tune your model via \text{ntree, mtry, nodesize}.

Specify a formula

<table>
<thead>
<tr>
<th>Survival</th>
<th>rfsrc(Surv(time, status) ~ ., data = veteran)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competing Risk</td>
<td>rfsrc(Surv(time, status) ~ ., data = wihs)</td>
</tr>
<tr>
<td>Regression</td>
<td>rfsrc(Ozone ~ ., data = aqquality)</td>
</tr>
<tr>
<td>Quantile Regression</td>
<td>quantreg(mpg ~ ., data = mtcars)</td>
</tr>
<tr>
<td>Classification</td>
<td>rfsrc(Surv(time, status) ~ ., data=veteran)</td>
</tr>
<tr>
<td>Imbalanced Two-Class</td>
<td>rfsrc(Multivar(mpg, cyl) ~ ., data = mtcars)</td>
</tr>
<tr>
<td>Multivariate</td>
<td>rfsrc(cbind(Species, Sepal.Length) ~ ., data=iris)</td>
</tr>
<tr>
<td>Reg</td>
<td>quantreg(cbind(mpg, cyl) ~ ., data = mtcars)</td>
</tr>
<tr>
<td>Mixed Regression</td>
<td>quantreg(cbind(Species, Sepal.Length) ~ ., data=iris)</td>
</tr>
<tr>
<td>Class</td>
<td>rfsrc(data = mtcars)</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>sidClustering(data = mtcars)</td>
</tr>
<tr>
<td>Breiman (Shi-Horvath)</td>
<td>sidClustering(data = mtcars, method = &quot;sh&quot;)</td>
</tr>
</tbody>
</table>

Tune mtry and nodesize

\[ \text{tune} \text{ Find the optimal mtry and nodesize tuning parameter for a random forest using out-of-bag (OOB) error} \]

\[ o < tune(quality ~ ., data) > o$optimal \]

\[ \text{nodesize} \text{ Find the optimal nodesize} \]

Grow

Convenient interface for growing a CART tree

\[ \text{rfsrc.cart(formula, data, ntree = 1, mtry = ncol(data), bootstrap = "none")} \]

Fast OpenMP parallel computing of random forests

\[ \text{rfsrc(formula, data, ntree = 500, mtry = NULL, ytry = NULL, nodesize = NULL, nodedepth = NULL, splitrule = NULL, nsplit = 10, importance = c(FALSE, TRUE, "none", "permute", "random", "anti"), ensemble = c("all", "oob", "inbag"), bootstrap = c("by.root", "none", "by.user"), sampertype = c("sxor", "swz"), samp = NULL, membership = FALSE, na.action = c("na.omit", "na.impute"), nimpute = 1, ntune = 250, cause, proximity = FALSE, distance = FALSE, forest.wt = FALSE, xvwt.wt = NULL, yvar.wt = NULL, split.wt = NULL, case.wt = NULL, forest = TRUE, var.used = c(FALSE, "all.trees", "by.tree"), split.depth = c(FALSE, "all.trees", "by.tree"), seed = NULL, do.trace = FALSE, statistics = FALSE, ...)} \]

Clean up and impute data

<table>
<thead>
<tr>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Choose your variables in formula and grow a tree.

\[ o < \text{rfsrc(y ~ a + z, data = dta, ntree = 1)} \]

Your outcome(s) will be saved in \text{o$x} and your predictors are in \text{o$x} from \text{dta} without missing values. To impute your data, use

\[ o < \text{impute(y ~ a + z, data = dta)} \]

\[ o < \text{rfsrc(y ~ a + z, data = dta, na.action = "na.impute")} \]

Inference from the Forest

Ensemble Predicted Value for Training Data

\[ o < \text{rfsrc(Ozone ~ ., data = aqquality)} \]

Inbag and out-of-bag (OOB) predicted values for the training dataset are in \text{o$predicted} and \text{o$predicted.oob}

Other Ensemble Values for Training Data

- For classification problem, we also have \$class and \$class.oob for class labels
- For survival problem, we have \$survival and \$survival.oob for survival function
  \$chf and \$chf.oob for cumulative hazard function
  \$cif and \$cif.oob for cumulative incidence function

Prediction Error for Assessing Model Performance

\[ o < \text{rfsrc(Species ~ ., data=iris, block.size=1)} \]

\[ o$err.rate \text{ returns tree cumulative OOB error rate; print(o) lists OOB error rate in the bottom; plot(o) plots OOB error rate along with number of trees; get.auc(y, prob) obtains the value of AUC (area under the ROC curve)} \]

\[ get.mv.error \text{ obtains error rate from a multivariate random forest} \]

CC by Min Lu Learn more with the randomForestSRC homepage • randomForestSRC version 2.12.0 • Updated: 2021-08
**Variable Selection**

**Variable Importance (VIMP)**

```r
o <- rfsrc(Species ~ ., iris, importance = TRUE)
```

Or

```r
obj <- rfsrc(Species ~ ., data = iris)
o <- vimp(obj)
```

```r
o$importance returns permutation VIMP and plot(o) plots VIMP when setting importance to "permute" or "TRUE" in rfsrc or using vimp
```

**Variable Selection and Hunting**

```r
var.select(formula, data, method) Variable selection or hunting by setting method
```

- **md** Minimal depth (default)
- **vh** Variable hunting
- **vh.vimp** Variable hunting with VIMP

**Partial Plot**

**Marginal Effect Plot**

```r
plot.variable(o, xvar.names)
```

**Partial Dependence Plot**

```r
plot.variable(o, xvar.names, partial = TRUE)
```

**Set surv.type for survival analysis:**

- **mort** Mortality
- **rel.freq** Relative frequency of mortality
- **surv** Predicted survival, where the predicted survival is for the time point specified using time
- **years.lost** The expected number of life years lost
- **cif** The cumulative incidence function
- **chf** The cumulative hazard function

**Get partial plot data** is a handy function that parses the output from "partial.rfsrc" in format suitable for plots

---

**Tree Visualization**

```r
get.tree extract a single tree from a forest and plot it on your browser
mtcars.unspv <- rfsrc(data = mtcars)
plot(get.tree(mtcars.unspv, 5))
```

**Split Statistics**

```r
stat.split acquires split statistic information. The end-cut preference (ECP) splitting property can be plotted
```

---

**Predict on New Data**

```r
o.pred <- predict(object = o, newdata)
```

Predicted values for the new dataset are in `o.pred$predicted`

```r
get.mv.predicted returns predicted value for multivariate regression analysis
```

**Minimal Depth**

```r
max.subtree extracts minimal depth and maximal subtree information used for variable selection and identifying interactions between variables
```

---

CC by Min Lu• Learn more with the randomForestSRC homepage • randomForestSRC version 2.12.0 • Updated: 2021-08