



R news and tutorials contributed by hundreds of R bloggers

- [Home](#)
- [About](#)
- [RSS](#)
- [add your blog!](#)
- [Learn R](#)
- [R jobs](#) 
- [Contact us](#)

Welcome!

 Follow @rbloggers

80.2k followers

Here you will find daily **news and tutorials about R**, contributed by hundreds of bloggers. There are many ways to **follow us** -

[By e-mail:](#)


51451 readers

BY FEEDBURNER

[On Facebook:](#)



R blog...
77K likes

 Like Page

Be the first of your friends to like this

If you are an R blogger yourself you are invited to [add your own R content feed to this site](#) (Non-English R bloggers should add themselves- [here](#))

[Jobs for R-users](#)

- [Lecturer of Statistical and Data Sciences](#)
- [Senior Research Data Scientist](#)
- [Data Scientist Position for Developing Software and](#)

[Tools in](#)

[Genomics, Big](#)

[Data and](#)

[Precision](#)

[Medicine](#)

- [Data Consultant 1](#)
[@ Cheney,](#)
[Washington,](#)
[United States](#)
- [Senior Scientist,](#)
[Translational](#)
[Informatics @](#)
[Vancouver, BC,](#)
[Canada](#)

Recent Posts

- [Data science](#)
[trainings in Berlin](#)
[& Hamburg](#)
- [Working with](#)
[Statistics Canada](#)
[Data in R, Part 4:](#)
[Canadian Census](#)
[Data – cancensus](#)
[Package Setup](#)
- [SMC on the 2019-](#)
[2020 nCoV](#)
[outbreak](#)
- [rWind is working](#)
[again!](#)
- [Dataviz Workshop](#)
[at RStudio::conf](#)
- [Get Better: R for](#)
[absolute beginners](#)
- [What and who is](#)
[IT community?](#)
[What does it take](#)
[to be part?](#)
- [How is](#)
[information gain](#)
[calculated?](#)
- [Lasso Regression](#)
[\(home made\)](#)
- [Hyperparameter](#)
[tuning and](#)
[#TidyTuesday](#)
[food consumption](#)
- [rstudio::conf 2020](#)
[Videos](#)
- [Getting started in](#)
[R markdown](#)
- [RStudio 1.3](#)
[Preview:](#)
[Configuration and](#)
[Settings](#)
- [Efficient Data](#)
[Management in R](#)
- [Clustered](#)
[randomized trials](#)
[and the design](#)
[effect](#)

Other sites

- [SAS blogs](#)

- [Jobs for R-users](#)

How to implement Random Forests in R

January 9, 2018
By [Perceptive Analytics](#)



[This article was first published on [R-posts.com](#), and kindly contributed to [R-bloggers](#)]. (You can report issue about the content on this page [here](#))

Want to share your content on R-bloggers? [click here](#) if you have a blog, or [here](#) if you don't.



Imagine you were to buy a car, would you just go to a store and buy the first one that you see? No, right? You usually consult few people around you, take their opinion, add your research to it and then go for the final decision. Let’s take a simpler scenario: whenever you go for a movie, do you ask your friends for reviews about the movie (unless, off-course it stars one of your favorite actress)?

Have you ever wondered why do we ask multiple people about their opinions or reviews before going for a movie or before buying a car or may be, before planning a holiday? It’s because review of one person may be biased as per her preference; however, when we ask multiple people we are trying to remove bias that a single person may provide. One person may have a very strong dislike for a specific destination because of her experience at that location; however, ten other people may have very strong preference for the same destination because they have had a wonderful experience there. From this, we can infer that the one person was more like an exceptional case and her experience may be one of a case.

Another example which I am sure all of us have encountered is during the interviews at any company or college. We often have to go through multiple rounds of interviews. Even though the questions asked in multiple rounds of interview are similar, if not same – companies still go for it. The reason is that they want to have views from multiple recruitment leaders. If multiple leaders are zeroing in on a candidate then the likelihood of her turning out to be a good hire is high.

In the world of analytics and data science, this is called ‘ensembling’. Ensembling is a “type of supervised learning technique where multiple models are trained on a training dataset and their individual outputs are combined by some rule to derive the final output.”

Let’s break the above definition and look at it step by step.

When we say multiple models are trained on a dataset, same model with different hyper parameters or different models can be trained on the training dataset. Training observations may differ slightly while sampling; however, overall population remains the same.

“Outputs are combined by some rule” – there could be multiple rules by which outputs are combined. The most common ones are the average (in terms of numerical output) or vote (in terms of categorical output). When different models give us numerical output, we can simply take the average of all the outputs and use the average as the result. In case of categorical output, we can use vote – output occurring maximum number of times is the final output. There are other complex methods of deriving at output also but they are out of scope of this article.

Random Forest is one such very powerful ensembling machine learning algorithm which works by creating multiple decision trees and then combining the output generated by each of the decision trees. Decision tree is a classification model which works on the concept of information gain at every node. For all the data points, decision tree will try to classify data points at each of the nodes and check for information gain at each node. It will then classify at the node where information gain is maximum. It will follow this process subsequently until all the nodes are exhausted or there is no further information gain. Decision trees are very simple and easy to understand models; however, they have very low predictive power. In fact, they are called weak learners.

Random Forest works on the same weak learners. It combines the output of multiple decision trees and then finally come up with its own output. Random Forest works on the same principle as Decision Tress; however, it does not select all the data points and variables in each of the trees. It randomly samples data points and variables in each of the tree that it creates and then combines the output at the end. It removes the bias that a decision tree model might introduce in the system. Also, it improves the predictive power significantly. We will see this in the next section when we take a sample data set and compare the accuracy of Random Forest and Decision Tree.

Now, let's take a small case study and try to implement multiple Random Forest models with different hyper parameters, and compare one of the Random Forest model with Decision Tree model. (I am sure you will agree with me on this – even without implementing the model, we can say intuitively that Random Forest will give us better results than Decision Tree). The dataset is taken from UCI website and can be found on this link. The data contains 7 variables – six explanatory (Buying Price, Maintenance, NumDoors, NumPersons, BootSpace, Safety) and one response variable (Condition). The variables are self-explanatory and refer to the attributes of cars and the response variable is 'Car Acceptability'. All the variables are categorical in nature and have 3-4 factor levels in each.

Let's start the R code implementation and predict the car acceptability based on explanatory variables.

```
1 # Data Source: https://archive.ics.uci.edu
2
3 install.packages("randomForest")
4 library(randomForest)
```

```
1 # Load the dataset and explore
2 data1 <- read.csv(file.choose(), header =
3
4 head(data1)
5
6 str(data1)
7
8 summary(data1)
```

```
1 > head(data1)
2   BuyingPrice Maintenance NumDoors NumPer
3   1         vhigh         vhigh      2
4   2         vhigh         vhigh      2
5   3         vhigh         vhigh      2
6   4         vhigh         vhigh      2
7   5         vhigh         vhigh      2
8   6         vhigh         vhigh      2
9 > str(data1)
10 'data.frame':   1728 obs. of  7 variables
11 $ BuyingPrice: Factor w/ 4 levels "high"
12 $ Maintenance: Factor w/ 4 levels "high"
13 $ NumDoors    : Factor w/ 4 levels "2","3
14 $ NumPersons  : Factor w/ 3 levels "2","4
15 $ BootSpace   : Factor w/ 3 levels "big",
16 $ Safety      : Factor w/ 3 levels "high"
```

```

17 $ Condition : Factor w/ 4 levels "acc",
18 > summary(data1)
19 BuyingPrice Maintenance NumDoors NumP
20 high :432 high :432 2 :432 2
21 low :432 low :432 3 :432 4
22 med :432 med :432 4 :432 more
23 vhigh:432 vhigh:432 5more:432

```

Now, we will split the dataset into train and validation set in the ratio 70:30. We can also create a test dataset, but for the time being we will just keep train and validation set.

```

1 # Split into Train and Validation sets
2 # Training Set : Validation Set = 70 : 30
3 set.seed(100)
4 train <- sample(nrow(data1), 0.7*nrow(data1))
5 TrainSet <- data1[train,]
6 ValidSet <- data1[-train,]
7 summary(TrainSet)
8 summary(ValidSet)

1 > summary(TrainSet)
2 BuyingPrice Maintenance NumDoors NumP
3 high :313 high :287 2 :305 2
4 low :292 low :317 3 :300 4
5 med :305 med :303 4 :295 more
6 vhigh:299 vhigh:302 5more:309
7 > summary(ValidSet)
8 BuyingPrice Maintenance NumDoors NumP
9 high :119 high :145 2 :127 2
10 low :140 low :115 3 :132 4
11 med :127 med :129 4 :137 more
12 vhigh:133 vhigh:130 5more:123

```

Now, we will create a Random Forest model with default parameters and then we will fine tune the model by changing 'mtry'. We can tune the random forest model by changing the number of trees (ntree) and the number of variables randomly sampled at each stage (mtry). According to Random Forest package description:

Ntree: Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.

Mtry: Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (\sqrt{p} where p is number of variables in x) and regression ($p/3$)

```

1 # Create a Random Forest model with default parameters
2 model1 <- randomForest(Condition ~ ., data = data1)
3
4 > model1
5
6 Call:
7 randomForest(formula = Condition ~ ., data = data1)
8
9 Type of random forest: classification
10 Number of trees: 500
11 No. of variables tried at each split: 2
12
13 OOB estimate of error rate: 3.64%
14 Confusion matrix:
15
16      acc good unacc vgood class.error
17 acc    253     7     4     0  0.04166667
18 good     3    44     1     4  0.15384615
19 unacc   18     1    837     0  0.02219626
20 vgood    6     0     0    31  0.16216216

```


By default, number of trees is 500 and number of variables tried at each split is 2 in this case. Error rate is 3.6%.

```
1 # Fine tuning parameters of Random Forest
2 model2 <- randomForest(Condition ~ ., data
3 model2

1 > model2
2
3 Call:
4 randomForest(formula = Condition ~ ., da
5               Type of random forest: cla
6               Number of trees: 500
7 No. of variables tried at each split: 6
8
9               OOB estimate of error rate: 2.32
10 Confusion matrix:
11      acc good unacc vgood class.error
12 acc    254    4     6     0  0.03787879
13 good     3   47     1     1  0.09615385
14 unacc   10    1   845     0  0.01285047
15 vgood    1    1     0    35  0.05405405
```

When we have increased the mtry to 6 from 2, error rate has reduced from 3.6% to 2.32%. We will now predict on the train dataset first and then predict on validation dataset.

```
1 # Predicting on train set
2 predTrain <- predict(model2, TrainSet, typ
3 # Checking classification accuracy
4 table(predTrain, TrainSet$Condition)
```

```
1 > table(predTrain, TrainSet$Condition)
2
3 predTrain acc good unacc vgood
4      acc    264    0     0     0
5      good     0   52     0     0
6      unacc     0    0   856     0
7      vgood     0    0     0    37
```

```
1 # Predicting on Validation set
2 predValid <- predict(model2, ValidSet, typ
3 # Checking classification accuracy
4 mean(predValid == ValidSet$Condition)
5 table(predValid, ValidSet$Condition)
```

```
1 > mean(predValid == ValidSet$Condition)
2 [1] 0.9884393
3 > table(predValid, ValidSet$Condition)
4
5 predValid acc good unacc vgood
6      acc   117    0     2     0
7      good    1   16     0     0
8      unacc    1    0   352     0
9      vgood    1    1     0    28
```

In case of prediction on train dataset, there is zero misclassification; however, in the case of validation dataset, 6 data points are misclassified and accuracy is 98.84%. We can also use function to check important variables. The below functions show the drop in mean accuracy for each of the variables.

```
1 # To check important variables
2 importance(model2)
3 varImpPlot(model2)
```

```
1 > importance(model2)
2
3      acc      good      unacc
4 BuyingPrice 143.90534 80.38431 101.06518 6
5 Maintenance 130.61956 77.28036  98.23423 4
6 NumDoors     32.20910 16.14126  34.46697 1
```

```
6 NumPersons 142.90425 51.76713 178.96850 4
7 BootSpace 85.36372 60.34130 74.32042 5
8 Safety 179.91767 93.56347 207.03434 9
```

```
1 > varImpPlot(model2)
2 <img data-attachment-id="419" data-permalink="https://www.kdnuggets.com/2018/07/random-forest-accuracy.html" data-rs="2" data-bbox="21 54 497 86"/>
```

Now, we will use 'for' loop and check for different values of mtry.

```
1 # Using For loop to identify the right mtry
2 a=c()
3 i=5
4 for (i in 3:8) {
5   model3 <- randomForest(Condition ~ ., data = TrainSet,
6   predValid <- predict(model3, ValidSet,
7   a[i-2] = mean(predValid == ValidSet$Condition)
8 }
9
10 a
11
12 plot(3:8,a)
```

```
1 > a
2 [1] 0.9749518 0.9884393 0.9845857 0.988439
3 >
4 > plot(3:8,a)
5 <img data-attachment-id="420" data-permalink="https://www.kdnuggets.com/2018/07/random-forest-accuracy.html" data-rs="2" data-bbox="21 303 497 375"/>
```

From the above graph, we can see that the accuracy decreased when mtry was increased from 4 to 5 and then increased when mtry was changed to 6 from 5. Maximum accuracy is at mtry equal to 8.

Now, we have seen the implementation of Random Forest and understood the importance of the model. Let's compare this model with decision tree and see how decision trees fare in comparison to random forest.

```
1 # Compare with Decision Tree
2
3 install.packages("rpart")
4 install.packages("caret")
5 install.packages("e1071")
6
7 library(rpart)
8 library(caret)
9 library(e1071)
10
11 # We will compare model 1 of Random Forest
12
13 model_dt = train(Condition ~ ., data = TrainSet, method = "rpart")
14 model_dt_1 = predict(model_dt, data = TrainSet)
15 table(model_dt_1, TrainSet$Condition)
16
17 mean(model_dt_1 == TrainSet$Condition)
18
19 > table(model_dt_1, TrainSet$Condition)
20
21 model_dt_1 acc good unacc vgood
22 acc 241 52 132 37
23 good 0 0 0 0
24 unacc 23 0 724 0
25 vgood 0 0 0 0
26
27 >
28 > mean(model_dt_1 == TrainSet$Condition)
29 [1] 0.7981803
```

On the training dataset, the accuracy is around 79.8% and there is lot of misclassification. Now, look at the validation dataset.

```
1 # Running on Validation Set
2 model_dt_vs = predict(model_dt, newdata = ValidationSet)
```

```
3 | table(model_dt_vs, ValidSet$Condition)
4 |
5 | mean(model_dt_vs == ValidSet$Condition)

1 | > table(model_dt_vs, ValidSet$Condition)
2 |
3 | model_dt_vs acc good unacc vgood
4 |      acc   107   17   58    28
5 |      good    0    0    0     0
6 |      unacc   13    0  296     0
7 |      vgood    0    0    0     0
8 | >
9 | > mean(model_dt_vs == ValidSet$Condition)
10| [1] 0.7764933
```

The accuracy on validation dataset has decreased further to 77.6%.

The above comparison shows the true power of ensembling and the importance of using Random Forest over Decision Trees. Though Random Forest comes up with its own inherent limitations (in terms of number of factor levels a categorical variable can have), but it still is one of the best models that can be used for classification. It is easy to use and tune as compared to some of the other complex models, and still provides us good level of accuracy in the business scenario. You can also compare Random Forest with other models and see how it fares in comparison to other techniques. Happy Random Foresting!!

Author Bio:

This article was contributed by [Perceptive Analytics](#). Chaitanya Sagar, Prudhvi Potuganti and Saneesh Veetil contributed to this article.

Perceptive Analytics provides data analytics, data visualization, business intelligence and reporting services to e-commerce, retail, healthcare and pharmaceutical industries. Our client roster includes Fortune 500 and NYSE listed companies in the USA and India.

 Share

 Tweet

To **leave a comment** for the author, please follow the link and comment on their blog:
[R-posts.com](#).

[R-bloggers.com](#) offers **daily e-mail updates** about [R](#) news and tutorials about [learning R](#) and many other topics. [Click here if you're looking to post or find an R/data-science job](#).

Want to share your content on R-bloggers? [click here](#) if you have a blog, or [here](#) if you don't.

If you got this far, why not **subscribe for updates** from the site?
Choose your flavor: [e-mail](#), [twitter](#), [RSS](#), or [facebook](#)...

 Like 237

 Share

 Tweet

 Share

Comments are closed.

Search R-bloggers

Go

Most visited articles of the week

1. [5 Ways to Subset a Data Frame in R](#)

2. [How to write the first for loop in R](#)
3. [R – Sorting a data frame by the contents of a column](#)
4. [Efficient Data Management in R](#)
5. [Installing R packages](#)
6. [Animated Plots using ggplot and gganimate](#)
7. [Stata to R cheatsheet for Econometrics](#)
8. [In-depth introduction to machine learning in 15 hours of expert videos](#)
9. [Date Formats in R](#)

Sponsors

Submit your abstract for the Enterprise Applications of the R Language Conference (EARL) before 31/3/20


London 8-10 September, 2020



 DataCamp

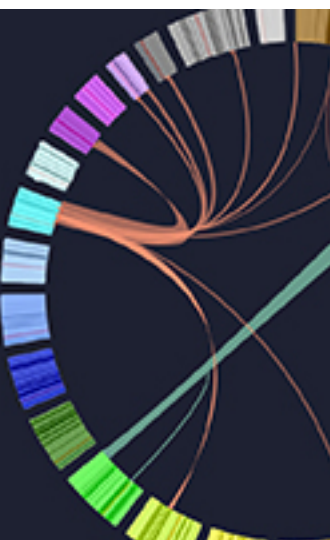
Learn **R**
by doing.



 plotly | Dash for R

Deliver Discovery
with interactive analytic apps

[Download the white paper](#)



**MACHINE
LEARNING
WEEK**

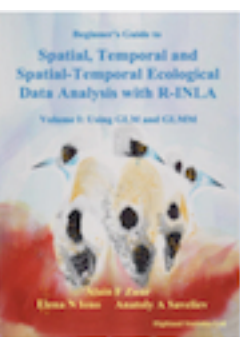
**Workshop
Machine Learning
with R: A Hands-On
Introduction**

Robert Muenchen
University of Tennessee



TRY **R** Studio Team

DOWNLOAD QUICKSTART VM



Beginner's Guide to
**Spatial, Temporal and
Spatial-Temporal Ecological
Data Analysis with R-INLA**

Zuur, Ieno, Saveliev



QUANTIDE
knowledge from data

Quantide: statistical consulting and training

**Apply [R]
Programming**

Build Applications with Shiny

40% Off Deal
Ends Friday



TRAININGS: R & PYTHON

Einführung & Machine Learning

BERLIN HAMBURG

April 2020

Oktober 2020

Jetzt anmelden



TRAINING: R, SCALA, STAN

STATWORX

Data Science Service

Data Science | Consulting | Development | Training

Try the FASTEST ML
for R



YOTTAMINE
ANALYTICS

Click for a Free Trial

SIGMA

SIGMA

ALPHIEN

BUILD QUANT STRATEGIES
JOIN COMPETITIONS
EARN LICENSING FEES

VISIT THE QUANT LAB

Our ads respect your privacy. Read our [Privacy Policy page](#) to learn more.

[Contact us](#) if you wish to help support R-bloggers, and place **your banner here**.

Submit your abstract for the Enterprise Applications of the R Language Conference (EARL) before 31/3/20

London 8-10 September, 2020

EARL
conference

MANGO
SOLUTIONS

DataCamp

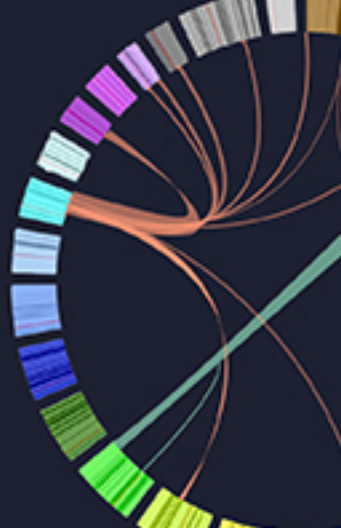
Learn R
by doing.



Deliver Discovery

with interactive analytic apps

[Download the white paper](#)



MACHINE
LEARNING
WEEK

Workshop Machine Learning with R: A Hands-On Introduction

Robert Muenchen
University of Tennessee



TRY **R** Studio Team

[DOWNLOAD QUICKSTART VM](#)



Beginner's Guide to Spatial, Temporal and Spatial-Temporal Ecological Data Analysis with R-INLA

Zuur, Ieno, Saveliev



QUANTIDE
knowledge from data

Quantide: statistical consulting and training

Apply [R] Programming

Build Applications with Shiny

40% Off Deal
Ends Friday



ODSC
BOSTON

TRAININGS: R & PYTHON

Einführung & Machine Learning

BERLIN HAMBURG

April 2020

Oktober 2020

[Jetzt anmelden](#)



COST PLUS
WORLD MARKET

Shop the world close to home!
Discover food, decor and more.



WEBSITE



DIRECTIONS



jumping rivers

STATWORX

Data Science Service

Data Science | Consulting | Development | Training



Try the FASTEST ML
for **R**

Click for a Free Trial

YOTTAMINE
ANALYTICS



ALPHIEN

BUILD QUANT STRATEGIES

JOIN COMPETITIONS

EARN LICENSING FEES

VISIT THE QUANT LAB

 python-bloggers.com
([python/data-science news](http://python-bloggers.com/python/data-science-news)).

- [New improved cdata instructional video](#)
- [Data re-Shaping in R and in Python](#)
- [sklearn Pipe Step Interface for vtreat](#)
- [Biomedical Data Science Textbook Available](#)
- [MinIO for Machine Learning Model Storage using Python](#)
- [New vtreat Feature: Nested Model Bias Warning](#)
- [New Timings for a Grouped In-Place Aggregation Task](#)

[Jobs for R users](#)

- [Lecturer of Statistical and Data Sciences](#)
- [Senior Research Data Scientist](#)
- [Data Scientist Position for Developing Software and Tools in Genomics, Big Data and Precision Medicine](#)
- [Data Consultant 1 @ Cheney, Washington, United States](#)
- [Senior Scientist, Translational Informatics @ Vancouver, BC, Canada](#)
- [Senior Principal Data Scientist @ Mountain View, California, United States](#)
- [Technical Research Analyst – New York, U.S.](#)

[Full list of contributing R-bloggers](#)

R-bloggers was founded by [Tal Galili](#), with gratitude to the [R](#) community.

Is powered by [WordPress](#) using a [bavotasan.com](#) design.

Copyright © 2020 **R-bloggers**. All Rights Reserved. [Terms and Conditions](#) for this website









