Welcome!

Here you will find daily **news and tutorials about R**, contributed by over 750 bloggers. There are many ways to **follow us -**
By e-mail:

Your e-mail here

Subscribe

46037 readers
BY FEEDBURNER

On Facebook:

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

- Data Scientist @ Garching bei München, Bayern, Germany
- Software Developer
- Senior Quantitative Analyst, Data Scientist
- R data wrangler
- Senior Data Scientist

Search & Hit Enter

Popular Searches

- googlevis
- heatmap
- twitter
- latex
- sales forecasting

- sql
- web Scraping
- eof
- hadoop
- Jeff Hemsley
- random forest
- 3 d clusters
- anova
- blotter
- boxplot
- coplot
- decision tree
- discriminant
- financial
- ggplot background grid colour
- how to import image file to r
- maps
- purrr
- rattle
- Trading
- bar chart
- barplot
- Binary
- climate
- contingency table data frame

Recent Posts

- Unconf projects 5: mwparser, Gargle, arresteddev
- R Interface to Spark
- Data Science for Business – Time Series Forecasting Part 2: Forecasting with timekit
- Run massive parallel R jobs cheaply with updated doAzureParallel package
- Introduction to Set Theory and Sets with R
- Campaign Response Testing no longer published on Udemy
- Neural networks Exercises (Part-1)
- Introducing the MonteCarlo Package
- Words growing or shrinking in Hacker News titles: a tidy analysis
- Test-driving Microsoft Cognitive Toolkit in R using reticulate
- Deep Learning with R
- Add P-values and Significance Levels to ggplots
- UK R Courses
- Unconf projects 4: cityquant, notary,

packagemetrics, pegax
- Tic Tac Toe Part 3: The Minimax Algorithm

Other sites

- Jobs for R-users
- SAS blogs

R Function of the Day: cut

September 23, 2009
By erik

(This article was first published on **sigmafield - R**, and kindly contributed to R-bloggers)

10 SHARES Share Tweet

This post originally appeared on my WordPress blog on September 23, 2009. I present it here in its original form.

The **R Function of the Day** series will focus on describing in *plain language* how certain R functions work, focusing on simple examples that you can apply to gain insight into your own data.

Today, I will discuss the **cut** function.

What situation is cut useful in?

In many data analysis settings, it might be useful to break up a continuous variable such as age into a categorical variable. Or, you might want to classify a categorical variable like year into a larger bin, such as 1990-2000. There are many reasons *not* to do this when performing regression analysis, but for simple displays of demographic data in tables, it could make sense. The **cut** function in R makes this task simple!

How do I use cut?

First, we will simulate some data from a hypothetical clinical trial that includes variables for patient ID, age, and year of enrollment.

```
> ## generate data for clinical trial example
> clinical.trial <-
    data.frame(patient = 1:100,
               age = rnorm(100, mean = 60, sd = 8),
               year.enroll = sample(paste("19", 85:99, sep = ""),
                 100, replace = TRUE))
> summary(clinical.trial)
    patient            age          year.enroll
 Min.   :  1.00   Min.   :41.18   1991   :12
 1st Qu.: 25.75   1st Qu.:52.99   1988   :11
 Median : 50.50   Median :60.08   1985   : 9
 Mean   : 50.50   Mean   :59.67   1993   : 7
 3rd Qu.: 75.25   3rd Qu.:65.67   1995   : 7
 Max.   :100.00   Max.   :76.40   1997   : 7
                                  (Other):47
```

Now, we will use the **cut** function to make age a factor, which is what R calls a categorical variable. Our first example calls cut with the **breaks** argument set to a single number. This method will cause **cut** to break up age into 4 intervals. The default labels use standard mathematical notation for open and closed intervals.

```
> ## basic usage of cut with a numeric variable
> c1 <- cut(clinical.trial$age, breaks = 4)
> table(c1)
c1
  (41.1,50]   (50,58.8] (58.8,67.6] (67.6,76.4]
          9          34          41          16
> ## year.enroll is a factor, so must convert to numeric first!
> c2 <- cut(as.numeric(as.character(clinical.trial$year.enroll)),
            breaks = 3)
> table(c2)
c2
(1985,1990] (1990,1994] (1994,1999]
         36          34          30
```

Well, the intervals that **cut** chose by default are not the nicest looking with the age example, although they are fine with the year example, since it was already discrete. Luckily, we can specify the exact intervals we want for age. Our next example shows how.

```
> ## specify break points explicitly using seq function
>
> ## look what seq does
> seq(30, 80, by = 10)
[1] 30 40 50 60 70 80
> ## cut the age variable using the seq defined above
> c1 <- cut(clinical.trial$age, breaks = seq(30, 80, by = 10))
> ## table of the resulting factor
> table(c1)
c1
(30,40] (40,50] (50,60] (60,70] (70,80]
      0       9      40      42       9
```

That looks pretty good. There is no reason that the breaks argument has to be equally spaced as I have done above. It could be any grouping that you want.

Finally, I am going to show you an example of a custom R function to categorize ages. It uses **cut** inside of it, but does some preprocessing and uses the **labels** argument to cut to make the output look nice.

```
age.cat <- function(x, lower = 0, upper, by = 10,
                   sep = "-", above.char = "+") {

 labs <- c(paste(seq(lower, upper - by, by = by),
                 seq(lower + by - 1, upper - 1, by = by),
                 sep = sep),
           paste(upper, above.char, sep = ""))

 cut(floor(x), breaks = c(seq(lower, upper, by = by), Inf),
     right = FALSE, labels = labs)
}
```

This function categorizes age in a fairly flexible way. The first assignment to **labs** inside the function creates a vector of labels. Then, the **cut** function is called to do the work, with the custom labels as an argument. Here are some examples using our simulated data from above. I am no longer going to save the results of the function calls to a variable and call **table** on them, but rather just nest the call to **age.cat** in a call to **table**. I previously did a post on the table function.

```
> ## only specifying an upper bound, uses 0 as lower bound, and
> ## breaks up categories by 10
> table(age.cat(clinical.trial$age, upper = 70))
  0-9 10-19 20-29 30-39 40-49 50-59 60-69   70+
    0     0     0     0     9    40    42     9
> ## now specifying a lower bound
> table(age.cat(clinical.trial$age, lower = 30, upper = 70))
30-39 40-49 50-59 60-69   70+
    0     9    40    42     9
> ## now specifying a lower bound AND the "by" argument
> table(age.cat(clinical.trial$age, lower = 30, upper = 70, by = 5))
30-34 35-39 40-44 45-49 50-54 55-59 60-64 65-69   70+
    0     0     3     6    22    18    22    20     9
```

Summary of cut

The cut function is useful for turning continuous variables into factors. You saw how to specify the number of cutpoints, specify the exact cutpoints, and saw a function built around **cut** that simplifies categorizing an age variable and giving it appropriate labels.

Tags:

- R

Comments are closed.

Search & Hit Enter

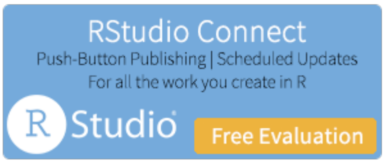
Recent popular posts

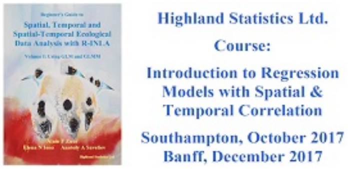
- Deep Learning with R
- Add P-values and Significance Levels to ggplots
- Introducing the MonteCarlo Package
- How to create dot-density maps in R

Most visited articles of the week

1. How to write the first for loop in R
2. Installing R packages
3. Using apply, sapply, lapply in R
4. How to Make a Histogram with Basic R
5. Tutorials for learning R
6. How to perform a Logistic Regression in R
7. Freedman's paradox
8. In-depth introduction to machine learning in 15 hours of expert videos
9. Shiny app to explore ggplot2

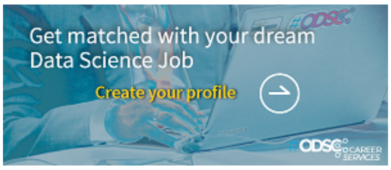
Sponsors


RStudio Connect
Push-Button Publishing | Scheduled Updates
For all the work you create in R
R Studio
Free Evaluation


Highland Statistics Ltd.
Course:
Introduction to Regression Models with Spatial & Temporal Correlation
Southampton, October 2017
Banff, December 2017

Register for free updates
in all areas of statistics
STATISTICS VIEWS
Bringing Statistics Together
WILEY

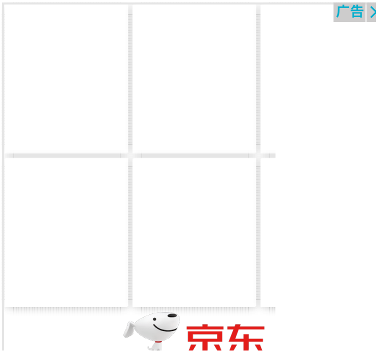
AI Learning Accelerator
gain AI Insights
enroll for free
#ODSC

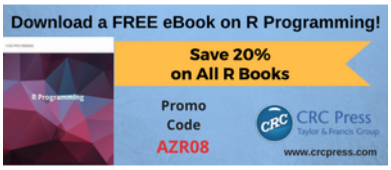

Get matched with your dream Data Science Job
Create your profile
#ODSC
ODSC CAREER SERVICES

R Courses for Professionals
Download R templates for machine learning
The fastest way to learn data science!
VIEW SNEAK PEAK
DATA SOCIETY

data science environment
Integrate Data Science into Business Processes

FANTASY FOOTBALL ANALYTICS
USING STATISTICS & R
TO IMPROVE YOUR FANTASY FOOTBALL TEAM


广告
京东


Download a FREE eBook on R Programming!
Save 20% on All R Books
Promo Code AZR08
CRC Press Taylor & Francis Group
www.crcpress.com

NYC DATA SCIENCE ACADEMY
Become a Data Scientist
Develop expertise in R, Python, Hadoop & Spark in just 12 Weeks
Apply for Data Science Bootcamp

AUTOMATIC MACHINE LEARNING
ML JAR
Train & tune predictive models with one click

Contact us if you wish to help support R-bloggers, and place **your banner here**.

Jobs for R users

- Data Scientist @ Garching bei München, Bayern, Germany
- Software Developer
- Senior Quantitative Analyst, Data Scientist
- R data wrangler
- Senior Data Scientist
- Manager, Statistical Consulting & Data Science
- Financial Controller

Search & Hit Enter

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.