Data Manipulation in R Tips

Use below code to manipulate “(31-Aug-2010) 14:00:41.386 2 into date time format in r

op <- options(digits.secs=3)

as.POSIXct(str1, format = "(%d-%b-%Y) %H:%M:%OS")

#[1] "2010-08-31 14:00:41.385 IST"

## Remove NA’s from data frame

na.omit(data\_frame\_name)

## Extract time from Date time to plot on x-axis

Recreate the (first two rows of) the dataframe in the original post:

foo <- data.frame(start.time = c("2012-02-06 15:47:00",

"2012-02-06 15:02:00",

"2012-02-22 10:08:00"),

duration = c(1,2,3))

Convert to POSIXct and POSIXt class (two ways to do this)

# using base::strptime

t.str <- strptime(foo$start.time, "%Y-%m-%d %H:%M:%S")

# using lubridate::ymd\_hms

library(lubridate)

t.lub <- ymd\_hms(foo$start.time)

Now, extract time as decimal hours

# using base::format

h.str <- as.numeric(format(t.str, "%H")) +

as.numeric(format(t.str, "%M"))/60

# using lubridate::hour and lubridate::minute

h.lub <- hour(t.lub) + minute(t.lub)/60

Demonstrate that these approaches are equal:

identical(h.str, h.lub)

Then choose one of above approaches to assign decimal hour to foo$hr:

foo$hr <- h.str

# If you prefer, the choice can be made at random:

foo$hr <- if(runif(1) > 0.5){ h.str } else { h.lub }

then plot using the ggplot2 package:

library(ggplot2)

qplot(foo$hr, foo$duration) +

         scale\_x\_datetime(labels = "%S:00")

## Find correlations of all numeric attributes in dataframe and remove NA’s

## cor(norm\_1hz\_data[sapply(norm\_1hz\_data, is.numeric)], use = "complete")

## Remove numbering on x-axis or y-axis:

plot(1:10, xaxt='n')

plot(1:10, yaxt='n')

axis(side = 2, 1:2, labels = levels(data\_test\_flight$HBV51OpenDmd))

# add axis command to add custom limits to y axis &[x axis => side=1]

If you want to remove the labels as well:

plot(1:10, xaxt='n', ann=FALSE)

plot(1:10, yaxt='n', ann=FALSE)

## Remove duplicate values

mydata <- mydata[,!duplicated (mydata)]

## Remove highly correlated variables(corr >0.99)

tmp <- cor(data)

> tmp[upper.tri(tmp)] <- 0

> diag(tmp) <- 0

# Above two commands can be replaced with

# tmp[!lower.tri(tmp)] <- 0

#

>

> data.new <- data[,!apply(tmp,2,function(x) any(x > 0.99))]

## Store R output in a text file

Run sink(“xsdfg.txt”) #diverting output to txt file

source(filename,echo=TRUE) # run R script

sink() # to stop diverting output to text file

check for hbv dmd signal when flight number is 1

table(rt\_engine\_data$HBV51OpenDmd[rt\_engine\_data$ft\_num == 1])

# base R

crime.ny.2005 <- crime.ny.2005[order(crime.ny.2005$Count,

decreasing=TRUE), ]

# dplyr

crime.ny.2005 <- arrange(crime.ny.2005, desc(Count))

The base R solution ranks each row by value of "Count" in decreasing order, and uses the rank vector to subset the "crime.ny.2005" data frame. The dplyr solution appears to be about 20% faster.

## number of state changes of vector/list :

> v=c(0,1,1,1,0,0,0)

> sum(abs(diff(v)))

[1] 2

## Remove duplicate columns dataframe

norm\_1hz\_data <- norm\_1hz\_data[,!duplicated(colnames(norm\_1hz\_data))]

# duplicated function returns true/false value

# [Difference between | and || in R](http://stackoverflow.com/questions/20122192/difference-between-and-in-r)

|  |  |
| --- | --- |
|  | The short operates element-wise on vectors and returns a vector of the same size as the input vectors. If necessary it recycles the shorter vector:  > c(FALSE, FALSE) | c(TRUE, FALSE)  [1] TRUE FALSE  The long form only consideres the first element of each vecotr and returns a length-one logical vector.  > c(FALSE, FALSE) || c(FALSE, TRUE, FALSE)  [1] FALSE  Typically whenever you have an if-statement, you need a length-one logical vector as a condition. Since || is faster than |, this version should be preferred. |

## Find relative density by grouping year,area,species

Rel density = density/sum of density of grouped data by year,area,species

|  |  |
| --- | --- |
|  | In base R it is something like that:  df$sumval <- tapply(df$density, list(df$year, df$area, df$species), sum )  df$perce <- df$density / df$sumval |

# different ways of ifelse implementation

## old way:

mydata$Agegroup1<-0

for (i in 1:10){

if(mydata$Age[i]>10 & mydata$Age[i]<20){

mydata$Agegroup1[i]<-1

}

if(mydata$Age[i]>=20){

mydata$Agegroup1[i]<-2

}

}

## Faster ways:

##Faster way to do the agegroup example above

mydata$Agegroup2<-ifelse(mydata$Age>10 & mydata$Age<20,1, ifelse(mydata$Age>20, 2,0))

##Or use cut - remember to change first to character, then to numeric

mydata$Agegroup3<-as.numeric(as.character(cut(mydata$Age, c(0,10,20,100),labels=0:2)))

# Recoding NA’s with other values

##recoding

mydata$Height<-ifelse(is.na(mydata$Height),9999,mydata$Height)

# find mean w.r.t other attribute value

mydata$meanweight.bysex<-ifelse(mydata$Sex==0, mean(mydata$Weight[mydata$Sex==0], na.rm=TRUE),mean(mydata$Weight[mydata$Sex==1], na.rm=TRUE))

# subsetting data

## different ways

subset(data,data$Loci %in% data$Loci[1:10])

## ##Subset by Age and Sex, selecting only ID, Age, and Weight columns

sub.data2<-subset(mydata, Age>50 & Sex==0, select=c(ID, Age, Weight))

##Subset rows where age>50, sex==0, and columns 1-3

sub.data4<-mydata[mydata$Age>50 & mydata$Sex==0, c(1:3)]

## ##Subset by Age all columns between ID and Sex

sub.data2<-subset(mydata, Age>50, select=c(ID:Sex))

## ##Subset all rows and everything EXCEPT columns 190-200

sub.data6<-mydata[,-c(190:200)]

## subset every nth element with different interval each time

foo2 = function(data, nSubsets, nSkip){

lapply(1:nSubsets, function(n) data[seq(n, NROW(data), by = nSkip),])

}

foo2(mtcars, 5, 15)

<http://stackoverflow.com/questions/7049467/select-several-subsets-by-taking-different-row-interval-and-appy-function-to-all>

## subset every nth element from a dataframe

a <- 1:120

b <- a[seq(1, length(a), n)]

# Plots

facet\_grid(Station~.)+

## change background colour in plot function

par(bg = "yellow")

plot(1:10, type = "n")

## Plot columns against each other

##Plot 3 columns against each other

plot(mydata[,c(2,4,5)])

## Plot values from 2 data frames in same graph

4 ways to do

* p1=ggplot(rt\_engine\_data,aes\_string("time\_sec","P30V"))+geom\_point()+scale\_size\_area()+

ggtitle("all data P30V overlay")+scale\_colour\_brewer(palette = "Set1")+theme\_bw()+theme(axis.text = element\_text(angle = 45, hjust = 1))

* p1 + ggplot(lt\_engine\_data,aes(time\_sec,P30V))+geom\_point(aes(P30V,colour = "blue"))
* (p2 <- ggplot(rt\_engine\_data,aes(time\_sec,P30V)) + geom\_point() + geom\_step(data = lt\_engine\_data))
* (p3 <- ggplot(NULL,aes(time\_sec,P30V)) + geom\_point(data = rt\_engine\_data)+geom\_step(data = lt\_engine\_data))
* (p4 <- ggplot() + geom\_point(data = rt\_engine\_data,aes(x=day\_mon\_y,y=P30V),color = 'blue') + geom\_point(data = lt\_engine\_data,aes(x=day\_mon\_y,y=P30V),color = 'red',shape = 4)+theme\_bw()+theme(axis.text = element\_text(angle = 45, hjust = 1)))

When plot object is placed in brackets, plot is actually printed without need to explicitly print graph

## Plot variables from subset of data

with(dfr[dfr$var3 < 155,], plot(var1, var2)) should do the trick.

**Edit** regarding multiple conditions:

with(dfr[(dfr$var3 < 155) & (dfr$var4 > 27),], plot(var1, var2))

library(ggplot2)

ggplot(subset(dat,ID %in% c("P1" , "P3"))) +

geom\_line(aes(Value1, Value2, group=ID, colour=ID))

Using subset the argument of geom\_line(Note I am using plyr package to use the special .function).

library(plyr)

ggplot(data=dat)+

geom\_line(aes(Value1, Value2, group=ID, colour=ID),

,subset = .(ID %in% c("P1" , "P3")))

You can also use the complementary subsetting:

subset(dat,ID != "P2")

## plot multiple subsets of the same object:

myplot<-ggplot(df)+geom\_line(aes(Value1, Value2, group=ID, colour=ID))

myplot %+% subset(df, ID %in% c("P1","P3"))

myplot %+% subset(df, ID %in% c("P2"))

## Multiple plots based on factor value of column

|  |  |
| --- | --- |
| ccepted | You can use facet\_grid or facet\_wrap to split up graphs by factors.  ggplot(mydata, aes(Var1, Var2)) + geom\_point() + facet\_grid(~ Variety)  or, on separate plots, just use a simple loop  for (var in unique(mydata$Variety)) {  dev.new()  print( ggplot(mydata[mydata$Variety==var,], aes(Var1, Var2)) + geom\_point() )  } |

## Plot colours for different range of variable

data.frame of value between -10 to 10, my data.frame has 2 columns. I need to give color to points which have values more than 8 or less than -8.

dat <- data.frame(p, g)

dat$colors <- 1

dat[which(dat$p < (-8) | dat$p > 8),"colors"] <- 0

library(ggplot2)

ggplot(dat, aes(x=g, y=p, group=colors)) + geom\_point(aes(color=colors))

## Dual Axis in a Plot

library(ggplot2)

library(gtable)

library(grid)

grid.newpage()

# two plots

p1 <- ggplot(mtcars, aes(mpg, disp)) + geom\_line(colour = "blue") + theme\_bw()

p2 <- ggplot(mtcars, aes(mpg, drat)) + geom\_line(colour = "red") + theme\_bw() %+replace%

theme(panel.background = element\_rect(fill = NA))

# extract gtable

g1 <- ggplot\_gtable(ggplot\_build(p1))

g2 <- ggplot\_gtable(ggplot\_build(p2))

# overlap the panel of 2nd plot on that of 1st plot

pp <- c(subset(g1$layout, name == "panel", se = t:r))

g <- gtable\_add\_grob(g1, g2$grobs[[which(g2$layout$name == "panel")]], pp$t,

pp$l, pp$b, pp$l)

# axis tweaks

ia <- which(g2$layout$name == "axis-l")

ga <- g2$grobs[[ia]]

ax <- ga$children[[2]]

ax$widths <- rev(ax$widths)

ax$grobs <- rev(ax$grobs)

ax$grobs[[1]]$x <- ax$grobs[[1]]$x - unit(1, "npc") + unit(0.15, "cm")

g <- gtable\_add\_cols(g, g2$widths[g2$layout[ia, ]$l], length(g$widths) - 1)

g <- gtable\_add\_grob(g, ax, pp$t, length(g$widths) - 1, pp$b)

# draw it

grid.draw(g)

## ggplot right side axis

**require**(gtable)

*# top plot*

p1 <- **ggplot**(mtcars, **aes**(mpg, disp)) + **geom\_line**(colour = 'blue') +

**background\_grid**(minor = 'none')

g1 <- **switch\_axis\_position**(p1, 'xy') *# switch both axes*

g1 <- **gtable\_squash\_rows**(g1, **length**(g1$height)) *# set bottom row to 0 height*

p2 <- **ggplot**(mtcars, **aes**(mpg, qsec)) + **geom\_line**(colour = 'green') + **ylim**(14, 25) +

**background\_grid**(minor = 'none')

g2 <- **ggplotGrob**(p2)

g2 <- **gtable\_add\_cols**(g2, g1$widths[5:6], 4) *# add the two additional columns that g1 has*

g2 <- **gtable\_squash\_rows**(g2, 1:2) *# set top two rows to 0 height*

**plot\_grid**(g1, g2, ncol = 1, align = 'v')

## ggplot subplots

library(grid)

library(gtable)

p1 <-ggplot(subset(dly\_51,ft\_num == 2),aes(x=time\_sec,y=delay))+geom\_point()+ylim(0,100)+ggtitle("hbv 51 flight 2 delay vs time")+theme\_bw()+theme(axis.text = element\_text(angle = 45, hjust = 1))

p2 <- ggplot(subset(dly\_51,ft\_num == 2),aes(x=time\_sec,y=act\_hbv51))+geom\_point()+theme\_bw()+theme(axis.text = element\_text(angle = 45, hjust = 1))

grid.newpage()

grid.draw(rbind(ggplotGrob(p1),ggplotGrob(p2),size = "last"))

## Change colour of plot depending on atrribute value

#Change the color depending on the sex

plot(mydata$Weight, mydata$Height, xlab="Weight (lbs)", ylab="Height (inches)", xlim=c(80,200), ylim=c(55,75), main="Height vs Weight", pch=2, cex.main=1.5, frame.plot=FALSE, , col=ifelse(mydata$Sex==1, "red", "blue"))

## Facet\_grid / right axis

#For more attributes in one graph

Use in ggplot object: facet\_grid(~.HBV51OpenDmd) or facet\_grid(HBV51OpenDmd ~ HBV52OpenDmd) or facet\_grid(HBV51OpenDmd+HBV52OpenDmd ~ HBV53OpenDmd)

## add legend

##Add a legend to the above plot

legend(80, 75, pch=c(2,2), col=c("red", "blue"), c("Male", "Female"), bty="o", cex=.8, box.col="darkgreen")

##This adds the same legend but uses "topleft" instead of points (80,75)

legend("topleft", pch=c(2,2), col=c("red", "blue"), c("Male", "Female"), bty="o", cex=.8, box.col="darkgreen")

## Abline

##Add an orange vertical line for the mean of Weight

abline(v=mean(mydata$Weight, na.rm=TRUE), col="orange")

##Linear regression line of Height on Age

reg<-lm(Height~Age, data=mydata)

##Add the regression line to the plot

abline(reg)

## Text addition

##Add text to the plot to describe the orange line

text(140,73, cex=.8, pos=4, "Orange line is\n sample average\n weight")

##Add text to the plot describing the regression line

text(0,72, paste("Height ~ ",round(reg$coef[1],2),"+",round(reg$coef[2],2),"\*Age"), pos=4, cex=.8)

## ggplot new graph type

d$weekend = 'weekday'

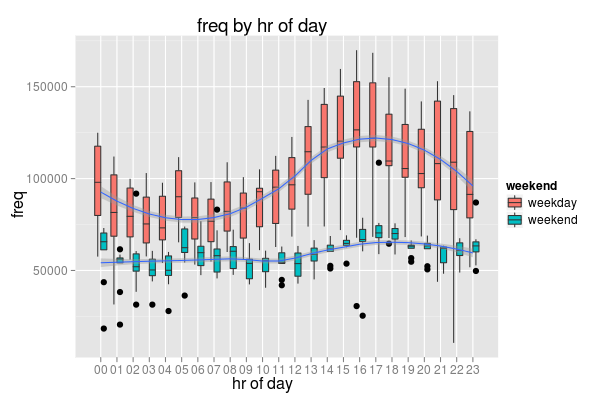
d[d$dow=='Sat'|d$dow=='Sun',]$weekend = 'weekend' # terrible style :(

ggplot(d,aes(hr,freq))

+ geom\_boxplot(aes(fill=weekend))

+ geom\_smooth(aes(group=weekend))

+ xlab('hr of day') + ylab('freq') + opts(title='freq by hr of day')



## Ggplot increase number of colors in pa;ette by mapping

# full data plots #

library(RColorBrewer)

colourCount = length(unique(rt\_engine\_data$ft\_num))

getPalette = colorRampPalette(brewer.pal(12, "Set3"))

ggplot(data = rt\_engine\_data,aes(NHV,P30V,color = factor(ft\_num)))+geom\_line(aes(group = factor(ft\_num)))+

scale\_fill\_manual(values = getPalette(colourCount))

## ggplot add labels to lines/line names

library(directlabels)

#ggplot(data = rt\_engine\_data,aes(NHV,P30V,color = factor(ft\_num)))+geom\_line(aes(group = factor(ft\_num)))+scale\_color\_discrete(guide = 'none')+

# scale\_x\_discrete(expand = c(0,1))+geom\_dl(aes(label = ft\_num),method = list(dl.combine("first.points","last.points"),cex = 0.8))

ggplot(data = p30\_b5,aes(NHV,P30V,color = factor(ft\_num)))+geom\_line(aes(group = factor(ft\_num)))+scale\_color\_discrete(guide = 'none')+

scale\_x\_discrete(expand = c(0,1))+geom\_dl(aes(label = ft\_num),method = list(dl.combine("first.points","last.points"),cex = 0.8))

## Overlay P30( attribute) for all factors in other attribute(like flight\_number)

ggplot(data = rt\_1hz\_data,aes(x=hr\_min,y=P30V,color = ft\_num))+geom\_line(aes(group = ft\_num))+geom\_point()+

scale\_color\_gradientn(colours=rainbow(7))

## Two facets ggplot

cspdistbv <- cbind.data.frame(lanem=cspdistbv[,1], cspdistbv[,2])

And then

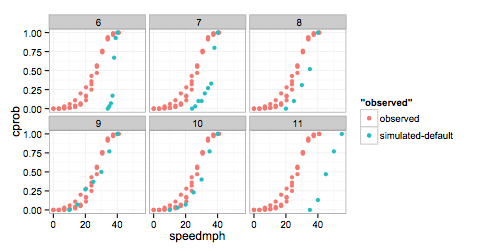
cb1 <- ggplot() + geom\_point(data = spdistbc, mapping = aes(x=speedmph,

y = cprob, color = 'observed')) + facet\_wrap(~Lane) + theme\_bw()

cb2 <- cb1 + geom\_point(data = cspdistbv, mapping = aes(x = speedmph,

y = prob, color = 'simulated-default')) + facet\_wrap(~lanem)

should work



# Create Factor variables

for(i in 1:5){births[,i]<-factor(births[,i], levels=c(1,2,3),labels=c("live birth","lost","still birth"))}

# Apply function

## Apply with custom function on row

##use Apply to get the sum of the live births by counting over a row the number of times it sees "live birth"

births$childcount<-apply(births[,1:5], MARGIN=1, function(x) {sum(x=="live birth", na.rm=TRUE)})

## apply with custom function on column

##Apply to this original data, across the columns, the function where it takes the columns and replaces it with NA if it equals 99 or -99, and leaves it alone otherwise

newdata<-apply(originaldata[,c(2,4:6)], MARGIN=2, function(x) {ifelse(x==99 | x==-99, NA,x)})

# ggplot in a loop

When in a for loop, you have to explicitly print your resulting ggplot object :

for (i in 1:5) {

print(ggplot(df,aes(x,y))+geom\_point())

}

# Remove variables

## Based on pattern

### Variables

If I want to remove all variables starting with A like A001,ABC,AXE etc

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |  | | Use  rm(list = ls()[grep("A", ls())]) |

### Data frames

paste0("data\_",seq(1,3,1))

# makes multiple data.frame names with sequential number

rm(list=paste0("data\_",seq(1,3,1))

# above code removes data\_1~data\_3

### Functions

rm(function\_name)

# ERRORS

## Dplyr

### Error: invalid subscript type 'double'

Fix: add quote to attribute name in group\_by command as in norm\_1hz\_data %>% group\_by(norm\_1hz\_data$'ft\_num')