

Useful R code

Matthew Malishev^{1*}

¹ *Department of Biology, Emory University, 1510 Clifton Road NE, Atlanta, GA, USA, 30322*

Contents

Overview	2
Install dependencies	2
Attributes	2
Classes	2
D3 apps	2
Dataframes	3
Generic functions	3
ggplot functions	3
Messages	5
NAs	5
Packages	6
Plotting	6
Reading in files/data	11
Regular expressions (regex)	12
R Markdown	12
Subsetting	13
Web scraping	13

Date: 2018-12-12

R version: 3.5.0

*Corresponding author: matthew.malishev@gmail.com

This document can be found at <https://github.com/darwinanddavis/UsefulCode>

Overview

This document outlines some useful R code for plotting, cool functions, and other random tidbits.

Install dependencies

Attributes

Access structural attributes of unique classes, such as raster and ggmap.

```
# Normal example
df <- data.frame("X"=c(1:5), "Y"=c(6:10))
str(df)
df$X

# `attr` method
require(ggmap)
map <- get_map("Atlanta", zoom=12, source="stamen", maptype="toner-lines")
str(map)
attr(map, "bb")$ll.lat
```

Classes

Convert character to factor to numeric without conversion error

```
read.table(f, header=T, sep=",", row.names=NULL, stringsAsFactors=FALSE, strip.white=TRUE)
f$V2<-as.numeric(f$V2)
```

See call options for class

```
methods(class="estUDm")
```

Set dynamic input for variable / assign variable to char vector

```
shadedens<-function(shadedens){ # set shade density to clumped (to match food) or sparse
  if (shadedens == "Random"){
    NLCommand("set Shade-density \"Random\" ")
  }else{
    NLCommand("set Shade-density \"Clumped\" ")
  }
}
shadedens("Clumped") # set clumped resources
```

D3 apps

Interactive network plots using d3

```
# Load package
install.packages("networkD3")
library(networkD3)

# Load energy projection data
URL <- "https://cdn.rawgit.com/christophergandrud/networkD3/master/JSONdata/energy.json"
Energy <- jsonlite::fromJSON(URL)
```

```

# Now we have 2 data frames: a 'links' data frame with 3 columns (from, to, value), and a 'nodes' data frame
head(Energy$links)
head(Energy$nodes)

# Thus we can plot it
sankeyNetwork(Links = Energy$links, Nodes = Energy$nodes, Source = "source",
              Target = "target", Value = "value", NodeID = "name",
              units = "TWh", fontSize = 12, nodeWidth = 30)

?sankeyNetwork

```

Dataframes

Optimal empty data frame

```

df <- data.frame(Date=as.Date(character()),
                 X=numeric(),
                 Y=integer(),
                 stringsAsFactors=FALSE)

```

Add df cols with mutate

```

require(dplyr)
df <- data.frame("a"=rnorm(10),"b"=(1:20))
df %>%
  mutate(
    "c"=rnorm(20),
    b = b *67
  )

```

Change df column names

```

colnames(data)[c(1,2,3)] <- c("TimeStamp","Lat","Long")

```

Generic functions

Generic useful functions that I can't place under any other headings here

```

# dput() for converting outputs such as copied text or data tables into vectors
xx <- "Some copied text or table from the internet"
dput(xx)

```

ggplot functions

Remove annoying stock gridlines from plot window

```

plot + theme_bw() +
  theme(panel.border = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# alternative (after loading ggridges library)
theme_ridges(grid=F,center_axis_labels = T)

```

Setting global graphics theme for ggplot

```

plot_it_gg <- function(bg,family){ # bg = colour to plot bg, family = font family
  theme_tufte(base_family = family) +
  theme(panel.border = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = bg,
                                         colour = bg),
        plot.background = element_rect(fill=bg)
  ) +
  theme(axis.line = element_line(color = "white")) +
  theme(axis.ticks = element_line(color = "white")) +
  theme(plot.title = element_text(colour = "white")) +
  theme(axis.title.x = element_text(colour = "white"),
        axis.title.y = element_text(colour = "white")) +
  theme(axis.text.x = element_text(color = "white"),
        axis.text.y = element_text(color = "white")) +
  theme(legend.key = element_rect(fill = bg)) + # fill bg of legend
  theme(legend.title = element_text(colour="white")) + # legend title
  theme(legend.text = element_text(colour="white")) # legend labels
}

```

Put plot in function to take dynamic data inputs

Ref: <http://jcborras.net/carpet/visualizing-political-divergences-2012-local-elections-in-helsinki.html>

```

hr.mass.plot <- function(d) {
  p <- ggplot(d, aes(HR, Mass, color = colfunc)) +
    geom_density_2d(data=d, aes(x = HR, y = Mass),
                   stat = "density2d",position="identity",
                   color=adjustcolor("orange",alpha=0.8), size=1.5, contour = T, lineend="square",line
  p <- p + geom_point(data=d, aes(x = HR, y = Mass),
                     color=colfunc,
                     fill=colfunc) +
    scale_color_manual(values = magma(8))
  p <- p + scale_y_continuous(limits=c(-200,200), name="Mass lost (g)")
  p <- p + scale_x_continuous(limits=c(0,0.35),name=expression("Home range area (km^2)"))
  p <- p + theme_classic()
  print(p)
}
hr.mass.plot(d)

```

Using ggplot when looping through for loop and saving to dir

```

pdf("mypdf.pdf",onfile = T)
for(i in 1:3){
  par(bty="n", las = 1)
  grid.arrange(
    ggplot(data, aes(x = X, y = Y, fill=..x..)) + # geom_density_ridges()
    # scale = overlap
    geom_density_ridges_gradient(scale = 5, size=0.2,color="black", rel_min_height = 0.01,panel_scaling
    geom_density_ridges(scale = 5, size=0.2,color="black", rel_min_height = 0.01,fill="white",alpha=0.2
    # geom_density_ridges(scale = 5, size=0.2,color="white", rel_min_height = 0.01,fill=col,alpha=0.5)
    scale_fill_viridis(name = "Diameter", alpha=0.1, option = "magma",direction=-1) + # "magma", "infer
    xlim(c(0,25)) +
    labs(title = paste0("Title_",i)) +
    xlab("X") +

```

```

    ylab("Y") +
    # plot_it_gg("white")
  )
} # end loop
dev.off()

```

Messages

Display status message of progress

```

for(i in 1:10) {
  Sys.sleep(0.2)
  # Dirk says using cat() like this is naughty ;- )
  #cat(i, "\r")
  # So you can use message() like this, thanks to Sharpie's
  # comment to use appendLF=FALSE.
  message(i, "\r", appendLF=FALSE) # appendLF = new line
  flush.console()
}

```

Display popup progress bar

```

require(tcltk)
pb <- tkProgressBar("test progress bar", "Some information in %",
  0, 100, 50)
Sys.sleep(0.5)
u <- c(0, sort(runif(20, 0, 100)), 100)
for(i in u) {
  Sys.sleep(0.1)
  info <- sprintf("%d%% done", round(i))
  setTkProgressBar(pb, i, sprintf("test (%s)", info), info)
}
Sys.sleep(5)
close(pb)

```

NAs

Replace NAs with 0's

```
df[is.na(df)] <- 0
```

Replace X values less than given value (V) with 0

```
df$X[df$X<V] <- 0
```

Check for NAs

```
sapply(df, function(x) sum(is.na(x)))
```

Replace NaN and Inf values with NA

```
df$col1[which(!is.finite(df$col1))] <- NA
```

Fill in missing data values in sequence with NA

```

# /Users/malishev/Documents/Manuscripts/Chapter4/Sims/Chapter4_figs.R
library(zoo)

```

```

data <- data.frame(index = c(1:4, 6:10),
  data = c(1.5,4.3,5.6,6.7,7.1,12.5,14.5,16.8,3.4))
#you can create a series
z <- zoo(data$data, data$index)
#end extend it to the grid 1:10
z <- merge(zoo(,1:10), z)

#worked example
# fill in missing Tb values
minTb.d <- zoo(minTb$Tick,minTb$Days)
minTb.d <- merge(zoo(NULL,1:days), minTb.d) # make the minTb series match the temp series (117 days)
minTb.d <- as.numeric(minTb.d) # = time individuals reached VTMIN in ticks
minTb <- minTb.d - temp$Tick # get diff between starting time and time to reach VTMIN
minTb <- minTb/2 # convert ticks to minutes
minTb <- minTb/60 #convert to hours
minTb <- data.frame("Days"=1:days,"Time"=minTb)

# then fill in missing values
approx(minTb$Time,method = "linear")

```

Remove rows with NA

```
data <- data[!is.na(data$X),]
```

Packages

rLandsat

Sourcing, requesting, and downloading NASA Landsat 8 satellite data.

Radix

Improved RMarkdown output and interaction.

rpanel

Reference guide

Create interactive GUI control toggles from R. Like an early Shiny.

Plotting

Plot one plot window above and two below

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
```

Bookend axis ticks for plot E.g. at 0 and 100 when data is 1:99

```
axis(1,at=c(0,length(loco$X)),labels=c("", ""))# bookending axis tick marks
```

Optimal legend formatting for base

```

legend("right",legend=c("Small","Intermediate","Large"),col=c(colfunc[colvec[1:3]]),
  bty="n",pch=20,pt.cex=1.5,cex=0.7,y.intersp = 0.5, xjust = 0.5,
  title="Size class",title.adj = 0.3,text.font=2,
  trace=T,inset=0.1)

```

Plot inset plot in current plot (<https://stackoverflow.com/questions/17041246/how-to-add-an-inset-subplot-to-topright-of-an-r-plot>)

```

# calculate position of inset
plotdim <- par("plt") # get plot window dims as fraction of current plot dims
xleft  = plotdim[2] - (plotdim[2] - plotdim[1]) * 0.5
xright = plotdim[2]  #
ybottom = plotdim[4] - (plotdim[4] - plotdim[3]) * 0.5  #
ytop    = plotdim[4]  #

# set position for plot inset
par(fig = c(xleft, xright, ybottom, ytop), mar=c(0,0,0,0), new=TRUE)

boxplot(Eggs~Size, data=meso2,
        col=adjustcolor(colfunc[colvec[1:3]], alpha=0.5),
        notch = T, xlab="Week", ylab="Diameter (mm)",
        xaxs = "i", yaxs = "i"
        )

```

Interactive plots with rCharts (javascript and d3 viz)

<http://ramnathv.github.io/rCharts/>

```

require(devtools)
install_github('rCharts', 'ramnathv')

```

Cluster plot

<https://rpubs.com/dgrtwo/technology-clusters>

```

library(readr)
library(dplyr)
library(igraph)
library(ggraph)
library(ggforce)

# This shared file contains the number of question that have each pair of tags
# This counts only questions that are not deleted and have a positive score
tag_pair_data <- read_csv("http://varianceexplained.org/files/tag_pairs.csv.gz")

relationships <- tag_pair_data %>%
  mutate(Fraction = Cooccur / Tag1Total) %>%
  filter(Fraction >= .35) %>%
  distinct(Tag1)

v <- tag_pair_data %>%
  select(Tag1, Tag1Total) %>%
  distinct(Tag1) %>%
  filter(Tag1 %in% relationships$Tag1 |
         Tag1 %in% relationships$Tag2) %>%
  arrange(desc(Tag1Total))

a <- grid::arrow(length = grid::unit(.08, "inches"), ends = "first", type = "closed")

set.seed(2016)

relationships %>%
  graph_from_data_frame(vertices = v) %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha = Fraction), arrow = a) +

```

```
geom_node_point(aes(size = Tag1Total), color = "lightblue") +
geom_node_text(aes(size = Tag1Total, label = name), check_overlap = TRUE) +
scale_size_continuous(range = c(2, 9)) +
ggforce::theme_no_axes() +
theme(legend.position = "none")
```

Define global plotting graphics function.

The plot_it.R function is updated on the [plot_it Github page](#).

```
require(ggplot2)
require(ggthemes)
### set plotting params
plot_it <- function(manuscript,bg,cp1,cp2,alpha,family){ # plotting function (plot for MS or not, set b
  graphics.off()
  if(manuscript==0){
    if(bg=="black"){
      colvec <-- magma(200,1) # plot window bg # USES <- OPERATOR
      par(bg = colvec[1],col.axis="white",col.lab="white",col.main="white",fg="white",bty="n",las=1,mar=
      border=adjustcolor("purple",alpha=0.5)
    }else{
      colvec <-- bpy.colors(200) # plot window bg # USES <- OPERATOR
      par(bg = colvec[1],col.axis="white",col.lab="white",col.main="white",fg="white",bty="n",las=1,mar=
      border=adjustcolor("blue",alpha=0.5)
    }
  }else{
    # graphics.off()
    par(bty="n",las=1,family=family)
    colv<-"white"
  }
  # color palettes
  # ifelse(manuscript==1,colvec<-adjustcolor(brewer.pal(9,cp1)[9], alpha = alpha),colvec <- adjustcolor
  # colfunc <- colorRampPalette(brewer.pal(9,cp1),alpha=alpha)
  cp1_info <- brewer.pal.info[cp1,]$maxcolors
  cp2_info <- brewer.pal.info[cp2,]$maxcolors
  colv <- brewer.pal(cp1_info,cp1) # USES <- OPERATOR
  colv2 <- brewer.pal(cp2_info,cp2) # USES <- OPERATOR
}

# Setting ggplot theme graphics
plot_it_gg <- function(bg){ # bg = colour to plot bg, family = font family
  if(bg=="white"){
    bg <- "white"
    fg <- "black"
    theme_tufte(base_family = "HersheySans") +
      theme(panel.border = element_blank(),panel.grid.major = element_blank(),panel.grid.minor = elemen
      theme(axis.line = element_line(color = fg)) +theme(axis.ticks = element_line(color = fg)) + theme
  }
}# end gg

### Set plotting function

require("RCurl")
script <- getURL("https://raw.githubusercontent.com/darwinanddavis/plot_it/master/plot_it.R", ssl.verif
eval(parse(text = script))
```



```
cat("plot_it( \n0 for presentation, 1 for manuscript, \nset colour for background, \nset colour palette
plot_it(0,"blue","Spectral","Greens",1,"mono") # set col function params
plot_it_gg("white") # same as above
```

Make plot cycle on one page

```
plot(m_abundance$gam,pages=1)
```

Get plot summaries and values from plot

```
plot.gam(m_abundance$gam,shade=T,pages=1,seWithMean = T)[1] # everything
plot.gam(m_abundance$gam,shade=T,pages=1,seWithMean = T)[1][[1]]$x #subset x
plot.gam(m_abundance$gam,shade=T,pages=1,seWithMean = T)[1][[1]]$fit #get values to produce fit curve
```

Package for stock world maps

```
#worldmap
library(choroplethrMaps)
```

Circle packing, tree, dendrogram, network plots

```
# dendrogram tree nested bubble circle packing network
# https://www.r-graph-gallery.com/313-basic-circle-packing-with-several-levels/

# circle packing plot
# Libraries
p <- c("ggraph","igraph","tidyverse","DeducerSpatial","Rcpp","car")
install.packages(p,dependencies = T)
lapply(p,library,character.only=T)

# We need a data frame giving a hierarchical structure. Let's consider the flare dataset:
edges=flare$edges
# edges cols = character

# Usually we associate another dataset that give information about each node of the dataset:
vertices = flare$vertices
# vertices cols = character, numeric, character

# Create a subset of the dataset (I remove 1 level)
edges = flare$edges %>% filter(to %in% from) %>% droplevels()
vertices = flare$vertices %>% filter(name %in% c(edges$from, edges$to)) %>% droplevels()
vertices$size=runif(nrow(vertices))

# Then we have to make a 'graph' object using the igraph library:
mygraph <- graph_from_data_frame( edges, vertices=vertices )

# circle packing
ggraph(mygraph, layout = 'circlepack', weight="size",sort.by=NULL,direction="out") +
  geom_node_circle(aes(fill=depth)) +
  geom_node_text(aes(label=shortName, filter=leaf, fill=depth, size=size)) + # add text
  # geom_node_label(aes(label=shortName, filter=leaf, size=size)) + # add text boxes
  theme_void() +
  # theme(legend.position="F") + #show legend
  scale_fill_viridis(alpha=0.5,direction=-1,option="magma") +
  # scale_fill_distiller(palette = "Blues")
```

```

#circular dendo
str(mygraph)
ggraph(mygraph, layout='dendrogram', circular=T) +
  geom_edge_diagonal(flipped=F,
                    label_colour = "black",
                    label_alpha = 1,
                    angle_calc = "rot",
                    force_flip = TRUE, label_dodge = NULL, label_push = NULL,
                    show.legend = NA) +

  theme_void() +
  # theme(legend.position="none") +
  scale_fill_distiller(palette = "Blues")

# tree map
ggraph(mygraph, 'treemap', weight = 'size') +
  geom_node_tile(aes(fill = depth), size = 0.25) +
  theme_void() +
  theme(legend.position="none")

# circular partition
ggraph(mygraph, 'partition', circular = TRUE) +
  geom_node_arc_bar(aes(fill = depth), size = 0.25) +
  theme_void() +
  theme(legend.position="none")

# node
ggraph(mygraph) +
  geom_edge_link() +
  geom_node_point() +
  theme_void() +
  theme(legend.position="none")

```

Insert an animal silhouette into a plot

```

#1. Get image from http://www.phylopic.org
library(png)
ima <- readPNG("thething.png")
plot(1:3,1:3)
rasterImage(image=ima, xleft=2,ybottom=1.8,
            xright=2.7,ytop=2.7)

```

Create an empty plot window

```

# 1
plot(0,type='n',axes=FALSE,ann=FALSE)
# 2
plot(1, type="n", xlab="", ylab="", xlim=c(0, 10), ylim=c(0, 10))
# 3
plot.new()

```

Set color gradient, palette for smoothing data points

```

require(RColorBrewer)

alpha <- 0.8 # transparency (0 to 1 value)
set.seed(5000)

```

```

rr <- rnorm(5000)

# user defined gradient
col<-colorRampPalette(c("steelblue","lightblue","orange","red")) # set your own col gradient with as many colors as you want
colfunc <- col(length(rr))[as.numeric(cut(rr,breaks = length(rr)))] # define breaks in col gradient
plot(rr,col=colfunc,pch=20)

# gradient from palette
display.brewer.all()
col <- "Greens"
col<-colorRampPalette(brewer.pal(brewer.pal.info[col,]$maxcolors,col)) # col gradient
colfunc <- col(length(rr))[as.numeric(cut(rr,breaks = length(rr)))] # define breaks in col gradient
plot(rr,col=colfunc,pch=20)

```

Reading in files/data

Read in file manually

```

get.file.vol <- read.table(file.choose())#read file manually
v.file <- get.file.vol[1:100,1]#get the volume

```

Loop through files from dir and append to list

```

# reading in spdf (hrpath) files from drive
setwd("/Users/camel/Desktop/Matt2016/Manuscripts/MalishevBullKearney/Resubmission/2016/barcoo sims/barcoo")
file.list<-list.files()
hrs75<-as.list(rep(1,100)) # empty list
for (f in 1:100){
  load(file.list[f])
  hrs75[f]<-hrpath
}

# working version
#converting spdf into mcp(spdf,100,unout="m2)
ghr<-list()
for (i in hrs75[1:10]) {
  m<-mcp(i,100,unout='m2')
  ghr<-c(ghr,m)
};ghr

```

Read in PDF files from online source in R and save to drive.

```

# from https://github.com/ropensci/pdftools

require(pdftools)
url <- "https://raw.githubusercontent.com/darwinanddavis/499R/master/exp_pop_growth.pdf"
dir <- "FOLDER ON YOUR COMPUTER WHERE YOU WANT THE FILE SAVED"
f <- "NAME OF THE FILE"
f <- paste0(f,".pdf")

# run all this
download.file(url, paste0(dir,"/",f), mode = "wb")
txt <- pdf_text(paste0(dir,"/",f))

```

```

# first page text
page <- 1 # enter the page number
cat(txt[page])

toc <- pdf_toc(paste0(dir,"/",f))

require(jsonlite)
# Show as JSON
jsonlite::toJSON(toc, auto_unbox = TRUE, pretty = TRUE)

# show author, version, etc
info <- pdf_info(f)

# renders pdf to bitmap array
bitmap <- pdf_render_page(f, page = 1)

# save bitmap image
png::writePNG(bitmap, "page.png")
jpeg::writeJPEG(bitmap, "page.jpeg")
webp::write_webp(bitmap, "page.webp")

```

Regular expressions (regex)

Get just numbers or characters

```

vec <- "16-Feb-2018 20:08:04 PM"
vecN <- gsub("[^[:digit:]]", "", vec); vec; print(paste0("Just numbers: ",vecN))
vecC <- gsub("[[:digit:]]", "", vec); vec; print(paste0("Just characters: ", vecC))

# with tidyr. requires data frame
require(tidyr)
df <- data.frame(N1=c("APPLE348744", "BANANA77845", "OATS2647892", "EGG98586456"))
print("tidyr doesn't work with strings separated by spaces")
df %>%
  separate(N1, into = c("text", "num"), sep = "(?<=[A-Za-z])(?=[0-9])")

```

Insert or replace a character in a string at a specific location

```

require(stringi)
vec <- "ABCEF"
stri_sub(vec, 4, 2) <- "d"
print(paste0("Original: ABCEF")); print(paste0("New: ",vec))

```

Testing regex expressions and their output

<https://regex101.com/r/ksY7HU/2>

R Markdown

Hide unwanted code output, such as inherent examples for functions

```

# ``{r, cache = TRUE, tidy = TRUE, lazy = TRUE, results='markup'}``

```

Subsetting

Select specific rows E.g. select rows of sfeed_move not in foodh

```
library(sqldf)
a1NotIna2_h <- sqldf('SELECT * FROM sfeed_move EXCEPT SELECT * FROM foodh')
a1NotIna2_l <- sqldf('SELECT * FROM sfeed_move EXCEPT SELECT * FROM foodl')
# select rows from sfeed_move that also appear in foodh
a1Ina2_h <- sqldf('SELECT * FROM sfeed_move INTERSECT SELECT * FROM foodh')
a1Ina2_l <- sqldf('SELECT * FROM sfeed_move INTERSECT SELECT * FROM foodl')
```

Count occurrences of values in data frame

```
table(unlist(df$X))
```

Remove a specific column from a data frame

```
within(df, rm("Col1"))
```

Web scraping

Scraping web tables

http://web.mit.edu/~r/current/arch/i386_linux26/lib/R/library/XML/html/readHTMLTable.html%5Bhttp://web.mit.edu/~r/current/arch/i386_linux26/lib/R/library/XML/html/readHTMLTable.html%5D

```
library(XML)
readHTMLTable()
```

Scraping Twitter timelines

See complete example at <http://varianceexplained.org/r/trump-tweets/>

```
# https://cran.r-project.org/web/packages/twitterR/
library(dplyr)
library(purrr)
library(twitterR)
```