Useful R code

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

Contents

Date: 2019-03-22
`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)
```

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

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

Add df cols with `mutate`

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

Change `df` column names

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

Remove multiple columns from df

```r
### Remove multiple NA columns
rm_cols <- grep("NA", names(tt), ignore.case = F)
df[,colnames(df[,rm_cols])] <- list(NULL)
```

Generic functions

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

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

Round up integers to optimal rounded value

```r
nn <- c(46,11,23)
round_any(nn,10)
round_any(nn,10,ceiling)
round_any(nn,10,floor)
```

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 = "blac
# 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",onefile = 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()
```

Lists

Find maximum value in entire list

```
master <- list(1:10,100,rnorm(12))
do.call(max,master)
```

Plot all elements in a list

```
xx <- list(sample(5,1000,replace=T),rnorm(1000),sample(50,1000,replace=T))
plot(unlist(xx),type="l")
```

Loops

Save loop output in master list

```
pars <- seq(0,1,0.5)
master <- list()
t_list <- list()
for (p in 1:length(pars)){
  for(t in 5){
    tt <- rnorm(1000*t)
    t_list[t] <-tt
  }
  master[[length(master)+1]] <- t_list # store in master list
}
```

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

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

Circle packing, tree, dendogram, network plots

```r
# dendogram 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
```

```r
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

```r
#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

```r
# 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

```r
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 ma
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)
```

Add plot point every nth element

```
n <- 3
plot(runif(10, 0, 1), type = "o", pch = c(20, rep(NA, n)))
```

Create function to make line as default type in plot

```
lplot <- function(...) plot(..., type="l")
lplot(runif(200))
```

Stack dataframe columns automatically in plot

```
head(outplot)
  # time          N        P        S        I
# 1 0.00 200.000000 200.0000 20.00000 2.000000
# 2 0.01  78.245140 177.1952 20.58217 2.067159
# 3 0.02  34.785145 168.9650 21.12174 2.136073
dats <- zoo(outplot)
plot(dats)
```

Make 3D scatterplot

```
require(scatterplot3d)
xx <- rnorm(1000)
yy <- runif(1000)
dens <- c(rep(0.0001,500),rep(1,500))
controls <- runif(3)
add.control <- 1
dens_val <- 1*10^-10 # 0 or 1*10^-10. value to knock out blanket of colour on plot surface
#linear model of r/ship between coords
dens_lm <- lm(dens ~ xx + yy)

xlim <- c(min(xx),max(xx)); ylim <- c(min(yy),max(yy)); zlim=c(min(dens),max(dens)) # set lims
colv <- "Blues"
colvv<-colorRampPalette(brewer.pal(brewer.pal.info[colv,]$maxcolors,colv)) # col gradient
colvv<-colorRampPalette(c("steelblue","lightblue","orange","red")) # set your own col gradient with as
# colvv<-colorRampPalette(magma(length(dens))) # set your own col gradient with as many colours as you

# set col palette
colfunc <- colvv(length(dens))[as.numeric(cut(dens,breaks = length(dens)))] # define breaks in col grad
bg <- bpy.colors(1)
alpha <- 0.8

# pdf(paste0(plot.dir,strat,"_",density,"_",stage,"_kudspdf.pdf"),width=8.27,height=11.69,paper="a4r")
scatterplot3d(x=xx,y=yy,z=dens,
              # color=ifelse(col_heat==1, adjustcolor(colfunc, alpha=1),adjustcolor("lightgreen",alpha=
              color=ifelse(dens<=dens_val,adjustcolor(ifelse(bg==bpy.colors(1),bpy.colors(1),"white"),a
              # col.axis="light green",
              las=1,
              pch=15,
              type="p",
              lty.hplot = 1,
              xlim=xlim,
              ylim=ylim,
              zlim=zlim,
              xlab="X",
              ylab="Y",
```

```
              zlab="Density",
              main="Main",
              box=F,
              lty.axis=par(1),
              grid=F,
              col.grid = adjustcolor("gray",1),
              lty.grid=par(3),
              #cex.symbols=dens*3,
              #cex.symbols = ifelse(z<=0,0,0.5),
              # highlight.3d=T, # ignores color arg if T
              # angle=70,
              axis=T
              # add below part to end of scatterplot3d plot
)#$plane3d(dens_lm, # add 3d linear model plane. # ??plane3d(Intercept, x.coef = NULL, y.coef = NULL, l
#            lty="dashed",
#          lty.box = NULL,
#          draw_lines = F, draw_polygon = T,
#          polygon_args = list(border = NA, col = adjustcolor("light green",alpha=0.4)))
# add control dates
if(add.control==1){par(new=T); scatterplot3d(x=rep(0,length(controls)),y=controls,z=rep(max(dens),lengt
```

Adding title from separate list to plot in loop (ggplot)

```
# plot all sim results in one window
gspl <- list()
ttl_list <- c("cerc","food", "juv", "adult", "infec", "infec (shed)", "host L", "parasite mass")

# choose sim to plot
global_sim_plot <- global_detritus

for(g in 1:10){
  gspl[[g]] <- ggplot() +
  geom_line(data = y_m, aes(x = rep.int(1:n.ticks,max(L1)) , y = value, group = L1, colour=factor(L1)),
  # scale_color_manual(values = viridis(length(mm))) +
  #linetype=y_m$L1) +
  theme_tufte() +
    labs(title=ttl_list[g],x="",y="") +
    if(g==length(global_sim_plot)){
      theme(legend.title=element_text(size=0.2),
            legend.text=element_text(size=0.2)) +
        theme(legend.position = "top")
      labs(x="Time")
    }else{
      theme(legend.position="none")
    }
    }
# +  geom_text(x=,y=,label = max(value),check_overlap = TUE)
do.call(grid.arrange,gspl) # plot in one window
```

```
plot(rnorm(1000),
     xlab=expression(paste("X values"^2)),
     ylab=expression(paste("Y values"^3,hat(beta)))
)
```

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/barc
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)
```

```r
# 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

```r
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

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

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

Math notation in R Markdown

```
x=y $x = y$
x<y $x < y$
x>y $x > y$
x y $x \le y$
x y $x \ge y$
xn  $x^{n}$
xn  $x_{n}$
x     $\overline{x}$
x̂  $\hat{x}$
x̃  $\tilde{x}$
ab  $\frac{a}{b}$
 f x    $\frac{a}{b}$
 f x    $\displaystyle \frac{a}{b}$
(nk)    $\binom{n}{k}$
x1+x2+ +xn  $x_{1} + x_{2} + \cdots + x_{n}$
```

```
x1,x2,…,xn  $x_{1}, x_{2}, \dots, x_{n}$
x= x1,x2,…,xn $\mathbf{x} = \langle x_{1}, x_{2}, \dots, x_{n}\rangle$
x A $x \in A$
|A| $|A|$
x A $x \in A$
A B $x \subset B$
A B $x \subseteq B$
A B $A \cup B$
A B $A \cap B$
X   (n, )    $X \sim {\sf Binom}(n, \pi)$

P(X x)=   (x,n, )    $\mathrm{P}(X \le x) = {\tt pbinom}(x, n, \pi)$
P(A B)  $P(A \mid B)$
P(A B)  $\mathrm{P}(A \mid B)$
{1,2,3} $\{1, 2, 3\}$
sin(x)  $\sin(x)$
log(x)  $\log(x)$
 ba $\int_{a}^{b}$
( baf(x)dx) $\left(\int_{a}^{b} f(x) \; dx\right)$
[ ϖ-ϖf(x)dx]    $\left[\int_{\-infty}^{\infty} f(x) \; dx\right]$
F(x)|ba $\left. F(x) \right|_{a}^{b}$
 bx=af(x)   $\sum_{x = a}^{b} f(x)$
 bx=af(x)   $\prod_{x = a}^{b} f(x)$
limx→ϖf(x)  $\lim_{x \to \infty} f(x)$
limx→ϖf(x)  $\displaystyle \lim_{x \to \infty} f(x)$
```

Greek Letters

```
A  $\alpha A$
N  $\nu N $
B  $\beta B$
Ξ  $\xi\Xi$
Γ  $\gamma \Gamma$
oO  $o O$ (omicron)
Δ  $\delta \Delta$
Π  $\pi \Pi$
 E $\epsilon \varepsilon E$
 P $\rho\varrho P$
Z  $\zeta Z \sigma \,\!$
Σ   $\sigma \Sigma$
H  $\eta H$
T  $\tau T$
 Θ $\theta \vartheta \Theta$
ϒ  $\upsilon \Upsilon$
I  $\iota I$
 Φ $\phi \varphi \Phi$
K  $\kappa K$
X  $\chi X$
Λ  $\lambda \Lambda$
Ψ  $\psi \Psi$
M  $\mu M$
 Ω$\omega \Omega$
```

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/twitteR/
library(dplyr)
library(purrr)
library(twitteR)
```