for(i in 2:11){

if(i<10){

j<-paste("00",i,".csv",sep="")

print(j)

}

else if(i<100){

j<-paste("0",i,".csv",sep="")

print(j)

}

else{

j<-paste(i,".csv",sep="")

print(j)

}

}

**Attempt to extend to 332**

for(i in 4:332){

if(i<10){

j<-paste("00",i,".csv",sep="")

print(j)

}

else if(i<100){

j<-paste("0",i,".csv",sep="")

print(j)

}

else{

j<-paste(i,".csv",sep="")

print(j)

}

}

**Attempt to file.append**

for(i in 5:332){

if(i<10){

j<-paste("00",i,".csv",sep="")

file.append("mierda001.csv",j)

}

else if(i<100){

j<-paste("0",i,".csv",sep="")

file.append("mierda001.csv",j)

}

else{

j<-paste(i,".csv",sep="")

file.append("mierda001.csv",j)

}

}

**IT WORKS!**

Extract rows where id=11

ext1<-pol\_dat[pol\_dat["ID"]==11,]

**IT WORKS!**

sulf<-ext1[,"sulfate"]

good\_sulf<-complete.cases(sulf)

mean(good\_sulf)

[1] 0.2021908

> library("readxl")

> volumes<-read\_excel("To\_estimate\_line\_speeds.xls")

Error: `path` does not exist: ‘To\_estimate\_line\_speeds.xls’

> getwd()

[1] "C:/Users/javier.lores/Documents/Operations/Equipment"

> dir()

[1] "Line\_speeds.csv" "Speeds.csv"

[3] "To\_estimate\_line\_speeds.xlsx"

> volumes<-read\_excel("To\_estimate\_line\_speeds.xlsx")

> head(volumes)

# A tibble: 6 x 5

`Location Code` `Item No#` WeekC `Registering date`

<chr> <chr> <dttm> <dttm>

1 CALI FGCB50011 2017-12-31 00:00:00 2018-01-06 00:00:00

2 CALI FGCB50011 2018-02-11 00:00:00 2018-02-12 00:00:00

3 CALI FGCB50011 2018-02-18 00:00:00 2018-02-19 00:00:00

4 CALI FGCB50011 2018-02-18 00:00:00 2018-02-23 00:00:00

5 CALI FGCB50011 2018-03-04 00:00:00 2018-03-10 00:00:00

6 CALI FGCB50011 2018-03-25 00:00:00 2018-03-28 00:00:00

# ... with 1 more variable: `SumOfNet Weight` <dbl>

> head(volumes)

# A tibble: 6 x 5

`Location Code` `Item No#` WeekC `Registering date` `SumOfNet Weight`

<chr> <chr> <dttm> <dttm> <dbl>

1 CALI FGCB50011 2017-12-31 00:00:00 2018-01-06 00:00:00 9020

2 CALI FGCB50011 2018-02-11 00:00:00 2018-02-12 00:00:00 1880

3 CALI FGCB50011 2018-02-18 00:00:00 2018-02-19 00:00:00 1490

4 CALI FGCB50011 2018-02-18 00:00:00 2018-02-23 00:00:00 700

5 CALI FGCB50011 2018-03-04 00:00:00 2018-03-10 00:00:00 1720

6 CALI FGCB50011 2018-03-25 00:00:00 2018-03-28 00:00:00 1680

> mean(volumes[["SumOfNet Weight"]],na.rm=TRUE)

[1] 2477.513

> max(volumes[["SumOfNet Weight"]],na.rm=TRUE)

[1] 4545060

> min(volumes[["SumOfNet Weight"]],na.rm=TRUE)

[1] 1.125

>

**Programming assignment 1**

**Step by step…**

**Step 1**

pollutantmean<-function(directory) {

setwd("C:/Users/javier.lores/Documents/R")

d<-paste("C:/Users/javier.lores/Documents/R/",directory,sep="")

setwd(d)

file.copy("001.csv","pollution\_data.csv")

for(i in 2:332){

if(i<10){

j<-paste("00",i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

else if(i<100){

j<-paste("0",i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

else{

j<-paste(i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

}

df<-read.csv("pollution\_data.csv")

file.remove("pollution\_data.csv")

}

Warning message:

In mean.default(pivot$Experimental) :

argument is not numeric or logical: returning NA

> d <- c("5","7")

> str(d)

chr [1:2] "5" "7"

> e <- as.numeric(d)

> str(e)

num [1:2] 5 7

> mean(d)

[1] NA

Warning message:

In mean.default(d) : argument is not numeric or logical: returning NA

> mean(e)

[1] 6

NO:

> mean(as.numeric(as.character(filas[["nitrate"]])),na.rm=TRUE)

[1] 1572.815

**YES:**

**mean(as.numeric(as.character(filas[["nitrate"]])),na.rm=TRUE)**

**[1] 0.6530321**

**Step 2**

pollutantmean<-function(directory,pollutant="nitrate") {

setwd("C:/Users/javier.lores/Documents/R")

d<-paste("C:/Users/javier.lores/Documents/R/",directory,sep="")

setwd(d)

file.copy("001.csv","pollution\_data.csv")

for(i in 2:332){

if(i<10){

j<-paste("00",i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

else if(i<100){

j<-paste("0",i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

else{

j<-paste(i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

}

df<-read.csv("pollution\_data.csv")

file.remove("pollution\_data.csv")

m\_df<-mean(as.numeric(as.character(df[[pollutant]])),na.rm=TRUE)

m\_df

}

**Algo es algo…**

> pollutantmean("specdata")

[1] 1.702932

Warning message:

In mean(as.numeric(as.character(df[[pollutant]])), na.rm = TRUE) :

NAs introduced by coercion

>pollutantmean("specdata","sulfate")

[1] 3.189369

Warning message:

In mean(as.numeric(as.character(df[[pollutant]])), na.rm = TRUE) :

NAs introduced by coercion

**Step 3**

pollutantmean<-function(directory="specdata",pollutant="nitrate",ID=1:332) {

setwd("C:/Users/javier.lores/Documents/R")

d<-paste("C:/Users/javier.lores/Documents/R/",directory,sep="")

setwd(d)

file.copy("001.csv","pollution\_data.csv")

for(i in 2:332){

if(i<10){

j<-paste("00",i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

else if(i<100){

j<-paste("0",i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

else{

j<-paste(i,".csv",sep="")

file.append("pollution\_data.csv",j)

}

}

df<-read.csv("pollution\_data.csv")

file.remove("pollution\_data.csv")

df\_ID<-df[df["ID"]==ID,]

Mdf\_ID<-mean(as.numeric(as.character(df\_ID[[pollutant]])),na.rm=TRUE)

Mdf\_ID

}

Conclusion: df\_ID aparece con filas de menos, debe ser por ID character al hacer file.append versus 1:322 as number

Classes of all columns in a dataframe

> foo <- data.frame(c("a", "b"), c(1, 2))

> names(foo) <- c("SomeFactor", "SomeNumeric")

> lapply(foo, class)

$SomeFactor

[1] "factor"

$SomeNumeric

[1] "numeric"

cols = c(1, 3, 4, 5);

df[,cols] = apply(df[,cols], 2, function(x) as.numeric(as.character(x)));

Attempt… and it works

Assignment 1

Part 1

**pollutantmean<-function(directory="specdata",pollutant="nitrate",Ident=1:332) {**

**file.remove("pollution\_data.csv")**

**file.remove("ddff.csv")**

**file.remove("df\_ID.csv")**

**file.remove("Mdf\_ID.csv")**

**setwd("C:/Users/javier.lores/Documents/R")**

**d<-paste("C:/Users/javier.lores/Documents/R/",directory,sep="")**

**setwd(d)**

**file.copy("001.csv","pollution\_data.csv")**

**for(i in 2:332){**

**if(i<10){**

**j<-paste("00",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else if(i<100){**

**j<-paste("0",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else{**

**j<-paste(i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**}**

**ddff<-read.csv("pollution\_data.csv")**

**ddff[,4]<-as.numeric(as.character(ddff[,4]))**

**df\_ID<-subset(ddff,ID>=min(Ident) & ID<=max(Ident))**

**write.csv(ddff,"ddff.csv")**

**write.csv(df\_ID,"df\_ID.csv")**

**Mdf\_ID<-mean(as.numeric(as.character(df\_ID[[pollutant]])),na.rm=TRUE)**

**write.csv(Mdf\_ID,"Mdf\_ID.csv")**

**Mdf\_ID**

**}**

Done…

df\_ID<-subset(ddff,ID==Ident | ID==Ident+1)

df\_ID<-ddff[ddff["ID"]==ID,]

file.remove("pollution\_data.csv")

Mdf\_ID<-mean(as.numeric(as.character(df\_ID[[pollutant]])),na.rm=TRUE)

Mdf\_ID

media\_calc<-function(uno,dos,tres){

u<-3

d<-dos

tt<-tres

mn<-mean(c(uno\*u,dos,tres),na.rm=TRUE)

}

U, d, tt, mn do not exist once the function has finished the operation

Change class of a column in a dataframe

for (i in 2:3{

mydf[,i] <- as.factor(mydf[,i])

}

Assignment 1

PART 2

**complete<-function(directory="specdata",Ident=1:332) {**

**file.remove("pollution\_data.csv")**

**file.remove("com\_ddff.csv")**

**setwd("C:/Users/javier.lores/Documents/R")**

**d<-paste("C:/Users/javier.lores/Documents/R/",directory,sep="")**

**setwd(d)**

**file.copy("001.csv","pollution\_data.csv")**

**for(i in 2:332){**

**if(i<10){**

**j<-paste("00",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else if(i<100){**

**j<-paste("0",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else{**

**j<-paste(i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**}**

**ddff<-read.csv("pollution\_data.csv")**

**ddff[,4]<-as.numeric(as.character(ddff[,4]))**

**write.csv(ddff,"com\_ddff.csv")**

**x<-numeric(length(Ident))**

**y<-numeric(length(Ident))**

**for(k in 1:length(Ident)){**

**x[k]<-sum(complete.cases(subset(ddff,ID==Ident[k])))**

**y[k]<-Ident[k]**

**}**

**comp<-data.frame(y, x, stringsAsFactors=FALSE)**

**names(comp)<-c("ID","nobs")**

**print(comp)**

**}**

Fails for question 7

set.seed(42)

cc <- complete("specdata", 332:1)

use <- sample(332, 10)

print(cc[use, "nobs"])

To resolve issue on question 7… **IT WORKS!**

**complete<-function(directory="specdata",Ident=1:332) {**

**file.remove("pollution\_data.csv")**

**file.remove("com\_ddff.csv")**

**setwd("C:/Users/javier.lores/Documents/R")**

**d<-paste("C:/Users/javier.lores/Documents/R/",directory,sep="")**

**setwd(d)**

**file.copy("001.csv","pollution\_data.csv")**

**for(i in 2:332){**

**if(i<10){**

**j<-paste("00",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else if(i<100){**

**j<-paste("0",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else{**

**j<-paste(i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**}**

**ddff<-read.csv("pollution\_data.csv")**

**ddff[,4]<-as.numeric(as.character(ddff[,4]))**

**write.csv(ddff,"com\_ddff.csv")**

**x<-numeric(length(Ident))**

**y<-numeric(length(Ident))**

**for(k in Ident[1]:Ident[length(Ident)]){**

**x[k]<-sum(complete.cases(subset(ddff,ID==Ident[k])))**

**y[k]<-Ident[k]**

**}**

**comp<-data.frame(y, x, stringsAsFactors=FALSE)**

**names(comp)<-c("ID","nobs")**

**print(comp)**

**}**

Attempts, failures, ideas…

comp= data.frame(x = numeric(), y = numeric())

for(k in length(Ident)){

x[k]<-sum(complete.cases(subset(ddff,ID==k)))

y[k]<-k

}

comp<-data.frame(y, x, stringsAsFactors=FALSE)

INFO

Loop with the content of a variable, even in reverse order

prueba<-function(h){

for(i in h)

print(i)

}

> prueba(3:1)

[1] 3

[1] 2

[1] 1

>

f4 <- function(n) {

x <- numeric(n)

y <- character(n)

for (i in 1:n) {

x[i] <- i

y[i] <- i

}

data.frame(x, y, stringsAsFactors=FALSE)

}

n<-sum(complete.cases(subset(df,ID<=min(Ident) & ID>=max(Ident))))

Esto CUENTA los TRUEs de la lista de TRUEs y FALSEs

El total:

> sum(comp$nobs)

[1] 111802

PART 3

**corr<-function(directory="specdata",threshold=0){**

**Ident<-1:332**

**file.remove("pollution\_data.csv")**

**file.remove("ddff.csv")**

**file.remove("ddff\_comps.csv")**

**file.remove("correla.csv")**

**file.remove("for\_cor.csv")**

**file.remove("ks.csv")**

**setwd("C:/Users/javier.lores/Documents/R")**

**d<-paste("C:/Users/javier.lores/Documents/R/",directory,sep="")**

**setwd(d)**

**file.copy("001.csv","pollution\_data.csv")**

**for(i in 2:332){**

**if(i<10){**

**j<-paste("00",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else if(i<100){**

**j<-paste("0",i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**else{**

**j<-paste(i,".csv",sep="")**

**file.append("pollution\_data.csv",j)**

**}**

**}**

**ddff<-read.csv("pollution\_data.csv")**

**ddff[,2]<-as.numeric(as.character(ddff[,2]))**

**ddff[,3]<-as.numeric(as.character(ddff[,3]))**

**ddff[,4]<-as.numeric(as.character(ddff[,4]))**

**write.csv(ddff,"ddff.csv")**

**x<-numeric(length(Ident))**

**y<-numeric(length(Ident))**

**for(k in 1:length(Ident)){**

**x[k]<-sum(complete.cases(subset(ddff,ID==Ident[k])))**

**y[k]<-Ident[k]**

**}**

**comp<-data.frame(y, x, stringsAsFactors=FALSE)**

**names(comp)<-c("ID","nobs")**

**write.csv(comp,"comp\_ddff.csv")**

**as.character(t)**

**as.numeric(u)**

**as.numeric(v)**

**as.numeric(w)**

**m<-numeric()**

**m<-0**

**for(k in 1:nrow(ddff)){**

**if(is.na(ddff[k,2])==FALSE){**

**if(is.na(ddff[k,3])==FALSE){**

**m<-m+1**

**t[m]<-ddff[k,1]**

**u[m]<- ddff[k,2]**

**v[m] <- ddff[k,3]**

**w[m] <- ddff[k,4]**

**}**

**}**

**}**

**ddff\_comps<-data.frame(t, u, v, w, stringsAsFactors=FALSE)**

**names(ddff\_comps)<-c("Date","sulfate","nitrate","ID")**

**write.csv(ddff\_comps,"ddff\_comps.csv")**

**nodo<-numeric()**

**correla<-numeric()**

**q<-0**

**h<-0**

**for(k in 1:nrow(comp)){**

**if(comp[k,2]>=threshold){**

**cor\_s<-numeric(comp[k,2])**

**cor\_n<-numeric(comp[k,2])**

**for(o in 1:comp[k,2]){**

**cor\_s[o]<-ddff\_comps[o+h,2]**

**cor\_n[o]<-ddff\_comps[o+h,3]**

**}**

**h<-h+comp[k,2]**

**for\_cor<- matrix(c(cor\_s, cor\_n), nrow=length(cor\_s),ncol=2)**

**nodo[k]<-k**

**rm(cor\_s)**

**rm(cor\_n)**

**q<-q+1**

**write.csv(for\_cor,"for\_cor.csv")**

**correla[q]<-cor(for\_cor[,1],for\_cor[,2])**

**}**

**else{**

**h<-h+comp[k,2]}**

**}**

**write.csv(nodo,"ks.csv")**

**write.csv(correla,"correla.csv")**

**correla**

**}**

Pregunta 8

cr <- corr("specdata")

cr <- sort(cr)

set.seed(868)

out <- round(cr[sample(length(cr), 5)], 4)

print(out)

***> cr<-sort(cr)***

***> set.seed(868)***

***> out <- round(cr[sample(length(cr), 5)], 4)***

***> print(out)***

***[1] 0.2688 0.1127 -0.0085 0.4586 0.0447***

Pregunta 9

cr <- corr("specdata", 129)

cr <- sort(cr)

n <- length(cr)

set.seed(197)

out <- c(n, round(cr[sample(n, 5)], 4))

print(out)

***> cr <- corr("specdata", 129)***

***Warning messages:***

***1: In corr("specdata", 129) : NAs introduced by coercion***

***2: In corr("specdata", 129) : NAs introduced by coercion***

***3: In corr("specdata", 129) : NAs introduced by coercion***

***> cr <- sort(cr)***

***> n <- length(cr)***

***> set.seed(197)***

***> out <- c(n, round(cr[sample(n, 5)], 4))***

***> print(out)***

***[1] 243.0000 0.2540 0.0504 -0.1462 -0.1680 0.5969***

Ddff\_comps\_thr<-ddff[ddff[,ID>=threshold]

if(x[k]>=threshold){

vec[k]<-comp$ID

}

#laply para cor to new dataframe

z[k]<-y[k]

(subset(ddff,ID==y[k]))

ddffc<-subset(ddff,ddff[,2][!is.na(ddff[,2])] & ddff[,3][!is.na(ddff[,3])])

x[k]<-sum(complete.cases(subset(ddff,ID==Ident[k])))

y[k]<-Ident[k]

}

}

}

prueba<-function(x){

if(exampl$Date!=2003-01-01){

h<-1

print(h)

}

}

Remove zero 0

down vote[favorite](https://stackoverflow.com/questions/29639680/r-table-function-how-to-remove-0-counts)

1

I need to remove the rows from the table function output, which have 0 counts in all the columns. Is there any easy way to do that?

table(ds$animal,ds$gender)

\_\_\_ | M | F

Cat | 9 | 4

Dog | 0 | 0

Rat | 4 | 3

I just would like to see those rows:

\_\_\_ | M | F

Cat | 9 | 4

Rat | 4 | 3

Complete.cases

I'd like to remove the lines in this data frame that:

a) **contain NAs across all columns.** Below is my example data frame.

gene hsap mmul mmus rnor cfam

1 ENSG00000208234 0 NA NA NA NA

2 ENSG00000199674 0 2 2 2 2

3 ENSG00000221622 0 NA NA NA NA

4 ENSG00000207604 0 NA NA 1 2

5 ENSG00000207431 0 NA NA NA NA

6 ENSG00000221312 0 1 2 3 2

Basically, I'd like to get a data frame such as the following.

gene hsap mmul mmus rnor cfam

2 ENSG00000199674 0 2 2 2 2

6 ENSG00000221312 0 1 2 3 2

b) **contain NAs in only some columns**, so I can also get this result:

gene hsap mmul mmus rnor cfam

2 ENSG00000199674 0 2 2 2 2

4 ENSG00000207604 0 NA NA 1 2

6 ENSG00000221312 0 1 2 3 2

[r](https://stackoverflow.com/questions/tagged/r) [dataframe](https://stackoverflow.com/questions/tagged/dataframe) [filter](https://stackoverflow.com/questions/tagged/filter) [missing-data](https://stackoverflow.com/questions/tagged/missing-data) [r-faq](https://stackoverflow.com/questions/tagged/r-faq)

844down voteaccepted

Also check [complete.cases](http://stat.ethz.ch/R-manual/R-patched/library/stats/html/complete.cases.html) :

> final[complete.cases(final), ]

gene hsap mmul mmus rnor cfam

2 ENSG00000199674 0 2 2 2 2

6 ENSG00000221312 0 1 2 3 2

na.omit is nicer for just removing all NA's. complete.cases allows partial selection by including only certain columns of the dataframe:

> final[complete.cases(final[ , 5:6]),]

gene hsap mmul mmus rnor cfam

2 ENSG00000199674 0 2 2 2 2

4 ENSG00000207604 0 NA NA 1 2

6 ENSG00000221312 0 1 2 3 2

Your solution can't work. If you insist on using is.na, then you have to do something like:

> final[rowSums(is.na(final[ , 5:6])) == 0, ]

gene hsap mmul mmus rnor cfam

2 ENSG00000199674 0 2 2 2 2

4 ENSG00000207604 0 NA NA 1 2

6 ENSG00000221312 0 1 2 3 2

but using complete.cases is quite a lot more clear, and faster.

8down voteaccepted

you need to drop levels from the factor animal.

table(droplevels(ds$animal),ds$gender)

you can also just drop them from ds and then do the table

ds$anima <- droplevels(ds$animal)

with(ds, table(animal,gender))

here I used with because it prints headers.

regex

One of the most frequently used string recognition algorithms out there is regex and R implements regex.  However, users can often be frustrated with how despite taking examples verbatim from many sources such as stackoverflow they do not seem to work.  From my own experience, I have found that the largest issue is really about what characters need to be escaped from R.

For example:

**Listing all files whose names match a simple pattern.**

Looking at “/^.\*icon.\*\.png$/i” from

[http://stackoverflow.com/questions/4845125/regex-to-match-filename-containing-a-word-regardless-of-case](https://stackoverflow.com/questions/4845125/regex-to-match-filename-containing-a-word-regardless-of-case)

I was able to get “^.\*icon.\*.png$“ to work in R though I lost the case insensitivity.  I think including the “^.” ensures that only files in the current directory, not subdirectory are matched but I am not sure.

So, the following code will return a list of file names from the folder Clipart which match the pattern [anything]icon.png

list.files(“C:/Clipart/”, pattern=”^.\*icon.\*.png$”)

[1] “manicon.png”     “handicon.png”     “bookicon.png”

Looking at the original entry we can see that what was causing us problems was the attempt to escape the “^” which does not need to be escaped in R.

Before looking at another example lets modify the previous command slightly to show how we can make it match differently.

list.files(“C:/Clipart/”, pattern=”^.\*icon\*.\*.png$”)

[1] “manicon.png”     “handicon.png”     “bookicon.png”    “iconnew.png”

There are a lot of resources available for regex since it is really its own text matching language supported by many different programming languages.  A good introductory guide can be found:

<http://www.zytrax.com/tech/web/regex.htm>

or

[http://www.regular-expressions.info/tutorial.html](https://www.regular-expressions.info/tutorial.html)

> list.files("C:/Users/javier.lores/Documents/R/color", pattern="\*olor\*.\*.xlsx$")

[1] "Color1.xlsx" "color10 lots 2016-2023.xlsx" "Color11 Medium 5 belly.xlsx" "Color12 Regular Medium 6.xlsx"

[5] "Color13 Regular Medium 7.xlsx" "Color14 Regular Small 2.xlsx" "Color15 Regular Small 3.xlsx" "Color16 Large Golden Flake.xlsx"

[9] "Color2.xlsx" "Color3 Cracklin Medium Tender.xlsx" "Color4 Cracklin Short Strips.xlsx" "Color5 Cracklin Small Tender.xlsx"

[13] "Color6 Cracklin Strips.xlsx" "Color7 Gaytan Small 2.xlsx" "Color8 Ham Strips.xlsx" "color9 lots 1x9.xlsx"

28down vote

glob2rx() converts a pattern including a wildcard into the equivalent regular expression. You then need to pass this regular expression onto one of R's pattern matching tools.

If you want to match "blue\*" where \* has the usual wildcard, *not* regular expression, meaning we use glob2rx() to convert the wildcard pattern into a useful regular expression:

> glob2rx("blue\*")

[1] "^blue"

The returned object *is* a regular expression.

Given your data:

x <- c('red','blue1','blue2', 'red2')

we can pattern match using grep() or similar tools:

> grx <- glob2rx("blue\*")

> grep(grx, x)

[1] 2 3

> grep(grx, x, value = TRUE)

[1] "blue1" "blue2"

> grepl(grx, x)

[1] FALSE TRUE TRUE FALSE

As for the selecting rows problem you posted

> a <- data.frame(x = c('red','blue1','blue2', 'red2'))

> with(a, a[grepl(grx, x), ])

[1] blue1 blue2

Levels: blue1 blue2 red red2

> with(a, a[grep(grx, x), ])

[1] blue1 blue2

Levels: blue1 blue2 red red2

or via subset():

> with(a, subset(a, subset = grepl(grx, x)))

x

2 blue1

3 blue2

Color data analysis

> library(readxl)

> color\_data<-read\_xlsx("Color.xlsx")

> write.csv(color\_data,"color.csv")

closeAllConnections()

Un solo fichero de color…

**colo<-function(x=0){**

**archivos<-dir()**

**colr<-read\_xlsx(archivos[1])**

**write.csv(colr,"color\_d.csv")**

**rm(colr)**

**for(i in 2:length(archivos)){**

**colr<-read\_xlsx(archivos[i])**

**write.csv(colr,"colr.csv")**

**file.append("color\_d.csv","colr.csv")**

**rm(colr)**

**file.remove("colr.csv")**

**}**

**color\_d<-read.csv("color\_d.csv")**

**head(color\_d)**

**}**

write.csv(color\_data,"color\_data.csv")

length(colores)

}

hist(B, col = "red", breaks=6, xlim=c(0,max),   
main="My Histogram", las=2, xlab = "Values", cex.lab = 1.3)

Select files by pattern  
2down voteaccepted

You can list the files in your working directory with list.files(). It has an argument allowing you to specify a certain pattern:

list.files(pattern = "Alta Guajira")

# [1] "Alta Guajira Coord & elevation TS 0-4000.txt"

If you have a vector with the patterns in there, like:

patterns <- c("Alta Guajira", "Rio Tomo", "Rio Sogamoso")

sapply(patterns, function(x){list.files(pattern = x)})

Edit

If you already have the file names, you are basically looking for patterns in a character vector.

df <- c("Alta Guajira Coord & elevation TS 0-4000.txt","Baja Guajira Coord & elevation TS 0-4000.txt",

"Bajo Meta Coord & elevation TS 0-4000.txt", "Rio Arauca Coord & elevation TS 0-4000.txt",

"Rio Catatubo Coord & elevation TS 0-4000.txt", "Rio Cesar Coord & elevation TS 0-4000.txt",

"Rio Sogamoso Coord & elevation TS 0-4000.txt", "Rio Tomo Coord & elevation TS 0-4000.txt",

"Sabana de Bogota Coord & elevation TS 0-4000.txt", "Total Area Coord & elevation TS 0-4000.txt")

patterns <- c("Alta Guajira", "Rio Tomo", "Rio Sogamoso")

it <- 0

res <- c()

for(i in patterns){

it <- it + 1

res <- append(res, df[grepl(pattern = i, x = df)])

}

res

# [1] "Alta Guajira Coord & elevation TS 0-4000.txt" "Rio Tomo Coord & elevation TS 0-4000.txt"

# [3] "Rio Sogamoso Coord & elevation TS 0-4000.txt"

Or, alternatively:

res2 <- sapply(patterns, function(y){df[grepl(pattern = y, x = df)]})

lista<-list.files(pattern="olor")

class(lista)

[1] "character"

colr<-vector()

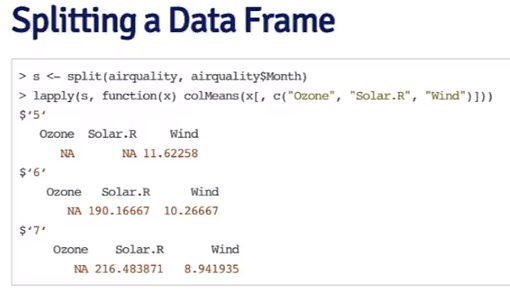
for (i in 1:length(lista)){

colr[i]<-read\_xlsx(lista[1])

}

lapply(

split and lapply



printmessage<-function(x){

if(x>0)

print("x is greater than zero")

else

print("x is equal to or less than zero")

invisible(x)

}

Digo yo: SET the SEED so that every time you run the random number you get the same result

lapply(unique\_vals,function(elem)elem[2])

anonymous function, created on the fly…

we can

| use vapply(flags, class, character(1)). The 'character(1)' argument tells R

| that we expect the class function to return a **character vector of length 1**

| when applied to EACH column of the flags dataset.

> table(flags$landmass)

1 2 3 4 5 6

31 17 35 52 39 20

> table(flags$animate)

0 1

155 39

Week 3

Quiz

> tapply(iris$Sepal.Length,iris$Species=="virginica",mean)

FALSE TRUE

5.471 6.588

> f<-gl(3,50)

> tapply(iris$Sepal.Length,f,mean)

1 2 3

5.006 5.936 6.588

> apply(iris[, 1:4], 2, mean)

Sepal.Length Sepal.Width Petal.Length Petal.Width

5.843333 3.057333 3.758000 1.199333

How can one calculate the average miles per gallon (mpg) by number of cylinders in the car (cyl)? Select all that apply.



apply(mtcars, 2, mean)

**Un-selected is correct**



sapply(split(mtcars$mpg, mtcars$cyl), mean)

> tapply(mtcars$mpg,mtcars$cyl,mean)

4 6 8

26.66364 19.74286 15.10000

> hps<-tapply(mtcars$hp,mtcars$cyl,mean)

> hps[3]-hps[1]

8

126.5779

> tapply(mtcars$hp,mtcars$cyl,mean)

4 6 8

82.63636 122.28571 209.21429

Whole quiz

Take a look at the 'iris' dataset that comes with R. The data can be loaded with the code:



1

2

library(datasets)

data(iris)

A description of the dataset can be found by running



1

?iris

There will be an object called 'iris' in your workspace. In this dataset, what is the mean of 'Sepal.Length' for the species virginica? **Please round your answer to the nearest whole number**.

(Only enter the numeric result and nothing else.)



**Correct Response**

To get the answer here, you can use 'tapply' to calculate the mean of 'Sepal.Length' within each species.

Question 2

Correct

1 / 1 points

## 2. Question 2

Continuing with the 'iris' dataset from the previous Question, what R code returns a vector of the means of the variables 'Sepal.Length', 'Sepal.Width', 'Petal.Length', and 'Petal.Width'?



rowMeans(iris[, 1:4])



colMeans(iris)



apply(iris, 2, mean)



apply(iris, 1, mean)



apply(iris[, 1:4], 2, mean)

**Correct**



apply(iris[, 1:4], 1, mean)

Question 3

Correct

1 / 1 points

## 3. Question 3

Load the 'mtcars' dataset in R with the following code



1

2

library(datasets)

data(mtcars)

There will be an object names 'mtcars' in your workspace. You can find some information about the dataset by running



1

?mtcars

How can one calculate the average miles per gallon (mpg) by number of cylinders in the car (cyl)? Select all that apply.



split(mtcars, mtcars$cyl)

**Un-selected is correct**



tapply(mtcars$mpg, mtcars$cyl, mean)

**Correct**



apply(mtcars, 2, mean)

**Un-selected is correct**



sapply(split(mtcars$mpg, mtcars$cyl), mean)

**Correct**



tapply(mtcars$cyl, mtcars$mpg, mean)

**Un-selected is correct**



mean(mtcars$mpg, mtcars$cyl)

**Un-selected is correct**



lapply(mtcars, mean)

**Un-selected is correct**



sapply(mtcars, cyl, mean)

**Un-selected is correct**



with(mtcars, tapply(mpg, cyl, mean))

**Correct**

Correct

1 / 1 points

## 4. Question 4

Continuing with the 'mtcars' dataset from the previous Question, what is the absolute difference between the average horsepower of 4-cylinder cars and the average horsepower of 8-cylinder cars?

(**Please round your final answer to the nearest whole number**. Only enter the numeric result and nothing else.)



**Correct Response**

Question 5

Correct

1 / 1 points

## 5. Question 5

If you run



1

debug(ls)

what happens when you next call the 'ls' function?



The 'ls' function will return an error.



Execution of 'ls' will suspend at the beginning of the function and you will be in the browser.

**Correct**



The 'ls' function will execute as usual.



You will be prompted to specify at which line of the function you would like to suspend execution and enter the browser.

Week 3

Programming assignment 2

Example provided:

makeVector <- function(x = numeric()) {

m <- NULL

set <- function(y) {

x <<- y

m <<- NULL

}

get <- function() x

setmean <- function(mean) m <<- mean

getmean <- function() m

list(set = set, get = get,

setmean = setmean,

getmean = getmean)

}

cachemean <- function(x, ...) {

m <- x$getmean()

if(!is.null(m)) {

message("getting cached data")

return(m)

}

data <- x$get()

m <- mean(data, ...)

x$setmean(m)

m

}

**Pruebesitas**

makeVector <- function(x = numeric()) {

m <- NULL

set <- function(y) {

x <<- y

m <<- NULL

}

get <- function() x

setmean <- function(mean) m <<- mean

getmean <- function() m

y<-list(set = set, get = get,

setmean = setmean,

getmean = getmean)

print(y)

}

**Otra pruebesita** ->> NO funciona

cachemean <- function(x, ...) {

makeVector <- function(x = numeric()) {

m <- NULL

set <- function(y) {

x <<- y

m <<- NULL

}

get <- function() x

setmean <- function(mean) m <<- mean

getmean <- function() m

list(set = set, get = get,

setmean = setmean,

getmean = getmean)

}

m <- x$getmean()

if(!is.null(m)) {

message("getting cached data")

return(m)

}

data <- x$get()

m <- mean(data, ...)

x$setmean(m)

m

}

**Otra pruebesita**:

Cachemean (makeVector(y))

write.csv(x,file="valor\_x.csv") después de m<-x$getmean

Error in as.data.frame.default(x[[i]], optional = TRUE) :

cannot coerce class ‘"function"’ to a data.frame

**Otra pruebesita**

makeVector <- function(x = numeric()) {

set <- function(y) {

x <<- y

}

get <- function() x

setmean <- function(mean) m <<- mean

getmean <- function() m

list(set = set, get = get,

setmean = setmean,

getmean = getmean)

}

> cachemean(makeVector(y))

[1] 7 8 9

Error: $ operator is invalid for atomic vectors

> r=matrix()

> r

[,1]

[1,] NA

> m\_inv<-NULL

> m<-mean

> m

function (x, ...)

UseMethod("mean")

<bytecode: 0x000000001a2e1df8>

<environment: namespace:base>

> m(3)

[1] 3

> m(c(3,5))

[1] 4

> m<-solve

> m(matrix(c(1,2,3,4),ncol=2,nrow=2))

[,1] [,2]

[1,] -2 1.5

Attempt

makeCacheMatrix <- function(x = matrix()) {

inv <- NULL

set <- function(y) {

x <<- y

inv <<- NULL

}

get <- function() x

setsolve <- function(solve) inv <<- solve

getsolve <- function() inv

list(set = set, get = get,

setsolve = setsolve,

getsolve = getsolve)

}

cacheSolve <- function(x, ...) {

inv <- x$getsolve()

if(!is.null(inv)) {

message("getting cached data")

return(inv)

}

else {

data <- x$get()

inv <- solve(data, ...)

x$setsolve(inv)

inv}

}

He quitado esto…

else if(x==y) {

message("getting cached data")

return(inv)

}

Week 4

Quiz

**Congratulations! You passed!**

Next Item

Question 1

Correct

1 / 1 points

## 1. Question 1

What is produced at the end of this snippet of R code?



1

2

set.seed(1)

rpois(5, 2)



It is impossible to tell because the result is random



A vector with the numbers 1, 1, 2, 4, 1

**Correct**

Because the `set.seed()' function is used, `rpois()' will always output the same vector in this code.



A vector with the numbers 1, 4, 1, 1, 5



A vector with the numbers 3.3, 2.5, 0.5, 1.1, 1.7

Question 2

Correct

1 / 1 points

## 2. Question 2

What R function can be used to generate standard Normal random variables?



dnorm



pnorm



rnorm

**Correct**

Functions beginning with the `r' prefix are used to simulate random variates.



qnorm

Question 3

Correct

1 / 1 points

## 3. Question 3

When simulating data, why is using the set.seed() function important? Select all that apply.



It ensures that the random numbers generated are within specified boundaries.

**Un-selected is correct**



It can be used to generate non-uniform random numbers.

**Un-selected is correct**



It can be used to specify which random number generating algorithm R should use, ensuring consistency and reproducibility.

**Correct**



It ensures that the sequence of random numbers is truly random.

**Un-selected is correct**

Question 4

Correct

1 / 1 points

## 4. Question 4

Which function can be used to evaluate the inverse cumulative distribution function for the Poisson distribution?



rpois



dpois



ppois



qpois

**Correct**

Probability distribution functions beginning with the `q' prefix are used to evaluate the quantile (inverse cumulative distribution) function.

Question 5

Correct

1 / 1 points

## 5. Question 5

What does the following code do?



1

2

3

4

set.seed(10)

x <- rep(0:1, each = 5)

e <- rnorm(10, 0, 20)

y <- 0.5 + 2 \* x + e



Generate uniformly distributed random data



Generate random exponentially distributed data



Generate data from a Poisson generalized linear model



Generate data from a Normal linear model

**Correct**

Question 6

Correct

1 / 1 points

## 6. Question 6

What R function can be used to generate Binomial random variables?



pbinom



rbinom

**Correct**



qbinom



dbinom

Question 7

Correct

1 / 1 points

## 7. Question 7

What aspect of the R runtime does the profiler keep track of when an R expression is evaluated?



the global environment



the function call stack

**Correct**



the working directory



the package search list

Question 8

Correct

1 / 1 points

## 8. Question 8

Consider the following R code



1

2

3

4

library(datasets)

Rprof()

fit <- lm(y ~ x1 + x2)

Rprof(NULL)

(Assume that y, x1, and x2 are present in the workspace.) Without running the code, what percentage of the run time is spent in the 'lm' function, based on the 'by.total' method of normalization shown in 'summaryRprof()'?



100%

**Correct**

When using `by.total' normalization, the top-level function (in this case, `lm()') always takes 100% of the time.



It is not possible to tell



23%



50%

Question 9

Correct

1 / 1 points

## 9. Question 9

When using 'system.time()', what is the user time?



It is the time spent by the CPU evaluating an expression

**Correct**



It is the time spent by the CPU waiting for other tasks to finish



It is a measure of network latency



It is the "wall-clock" time it takes to evaluate an expression

Question 10

Incorrect

0 / 1 points

## 10. Question 10

If a computer has more than one available processor and R is able to take advantage of that, then which of the following is true when using 'system.time()'?



user time is 0



elapsed time may be smaller than user time



elapsed time is 0



user time is always smaller than elapsed time

**This should not be selected**

I think it is the second. I need to re-watch the video.

type ls() to see a list of the variables in your workspace. Then, type

| rm(list=ls()) to clear your workspace.

| Whenever you're working with a new dataset, the first thing you should do is

| look at it! What is the format of the data? What are the dimensions? What are

| the variable names? How are the variables stored? Are there missing data? Are

| there any flaws in the data?

| Each of the functions we've introduced so far has its place in helping you to

| better understand the structure of your data. However, we've left the best

| for last....

Let's simulate rolling four six-sided dice: sample(1:6, 4, replace = TRUE)

| You are really on a roll!

| Now use replicate(100, rpois(5, 10)) to perform this operation 100 times.

| Store the result in a new variable called my\_pois.

| replicate() created a matrix, each column of which contains 5 random numbers

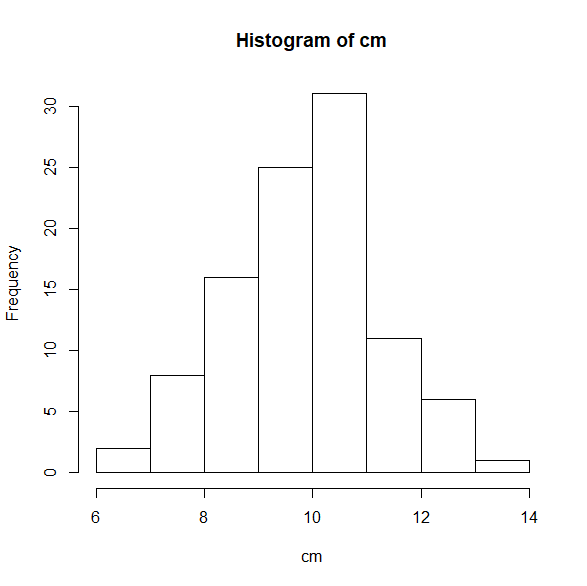
| generated from a Poisson distribution with mean 10. Now we can find the mean

| of each column in my\_pois using the colMeans() function. Store the result in

| a variable called cm.

| And let's take a look at the distribution of our column means by plotting a

| histogram with hist(cm).



| Looks like our column means are almost normally distributed, right? That's

| the Central Limit Theorem at work, but that's a lesson for another day!

| Simulation is practically a field of its own and we've only skimmed the

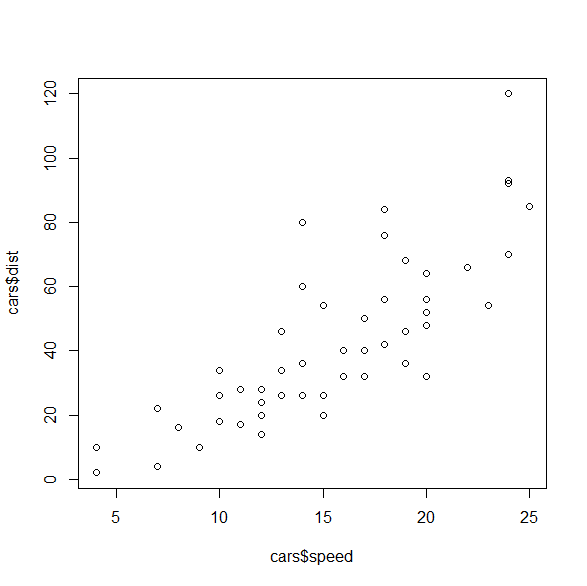
| surface of what's possible. I encourage you to explore these and other

| functions further on your own.

| These include lattice, ggplot2 and ggvis.

data(cars)

plot(x = cars$speed, y = cars$dist).



Check Value included in vector

> "NY" %in% outcome[,7]

[1] TRUE

grep("ttack",names(outcome))

[1] 11 12 13 14 15 16 29 30 31 32 33 34

install.packages("stringr")

condition<-c("heart attack")

> word(condition,1,2)

[1] "heart attack"

> word(condition,1,1)

[1] "heart"

Stack overflow  
34down vote

I would suggest the grep() function and some of its additional arguments that make it a pleasure to use.

grep("stringofinterest",names(dataframeofinterest),ignore.case=TRUE,value=TRUE)

without the argument value=TRUE you will only get a vector of index positions where the match occurred.

[share](https://stackoverflow.com/a/5672116)[improve this answer](https://stackoverflow.com/posts/5672116/edit)

Sort rows in a dataframe based on one column content

population[order(population$age),]

order(..., na.last = TRUE, decreasing = FALSE,

method = c("auto", "shell", "radix"))

sorted\_res<-outcome[order(outcome[,11],outcome[,2]),]

> str1 <- "How many words are in this sentence"

> sapply(strsplit(str1, " "), length)

[1] 7

my.data.frame <- subset(data , V1 > 2 | V2 < 4)

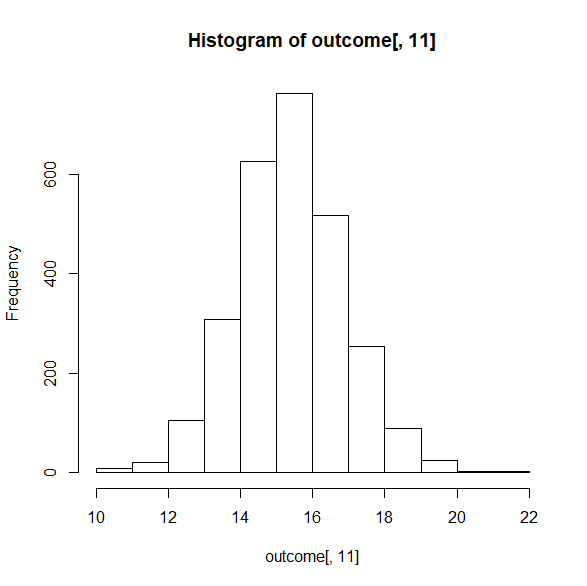
An alternative solution that mimics the behavior of this function and would be more appropriate for inclusion within a function body:

new.data <- data[ which( data$V1 > 2 | data$V2 < 4) , ]

if(length(grep(word(condition,1),names(outcome),ignore.case=TRUE))!=0& length(grep(word(condition,2),names(outcome),ignore.case=TRUE))!=0){…

**Programming assignment 3**

Part 1



Part 2

Best=function(state,condition){

best.R

**best<-function(state,condition){**

##Sorry that I took the liberty to change the name of the second function argument, for the sake of clarity

## Assuming that stringr library has been loaded because we will use the word() function

## CHECKS if State and condition are a match. If not, error message via stop function.

**acc\_cond<-c("Heart attack","Heart failure","Pneumonia")**

**pal<-sapply(strsplit(condition, " "), length)**

**if(length(grep(state,outcome[,7], ignore.case=TRUE))==0){**

**stop("Sorry, State is not a match")**

**} else {outcome\_s<-subset(outcome,State==state)**

**##Checks if the condition is pneumonia, heart attack or heart failure**

**if(pal==1){**

**if(length(grep(word(condition),names(outcome),ignore.case=TRUE))!=0){**

**colnum<-23**

**##because pneumonia is the condition**

**} else { stop("Sorry, condition is not a match")**

**}**

**} else if(pal>=3) { stop("Sorry, condition is not a match")**

**} else { if(length(grep(word(condition,1),names(outcome),ignore.case=TRUE))!=0& length(grep(word(condition,2),names(outcome),ignore.case=TRUE))!=0){**

**if(grep(word(condition,2),acc\_cond,ignore.case=TRUE)==1){**

**colnum<-11**

**} else { colnum<-17**

**}**

**}**

**else { stop("Sorry, condition is not a match")}**

**}**

**}**

**res<- subset(outcome\_s,select=c(2,7,colnum),na.rm=TRUE)**

**sorted\_res<-res[order(as.numeric(res[,3]),res[,1]),]**

**print(sorted\_res[1,1])**

**}**

**Assignment 3, part 3**

rankhospital.R

**rankhospital<-function(state,condition,num){**

**##Sorry that I took the liberty to change the name of the second function argument, for the sake of clarity**

**## Assuming that stringr library has been loaded because we will use the word() function**

**## CHECKS if State and condition are a match. If not, error message via stop function.**

**acc\_cond<-c("Heart attack","Heart failure","Pneumonia")**

**pal<-sapply(strsplit(condition, " "), length)**

**if(length(grep(state,outcome[,7], ignore.case=TRUE))==0){**

**stop("Sorry, State is not a match")**

**} else {outcome\_s<-subset(outcome,State==state)**

**##Checks if the condition is pneumonia, heart attack or heart failure**

**if(pal==1){**

**if(length(grep(word(condition),names(outcome),ignore.case=TRUE))!=0){**

**colnum<-23**

**##because pneumonia is the condition**

**} else { stop("Sorry, condition is not a match")**

**}**

**} else if(pal>=3) { stop("Sorry, condition is not a match")**

**} else { if(length(grep(word(condition,1),names(outcome),ignore.case=TRUE))!=0& length(grep(word(condition,2),names(outcome),ignore.case=TRUE))!=0){**

**if(grep(word(condition,2),acc\_cond,ignore.case=TRUE)==1){**

**colnum<-11**

**} else { colnum<-17**

**}**

**}**

**else { stop("Sorry, condition is not a match")}**

**}**

**}**

**res<- subset(outcome\_s,select=c(2,7,colnum),na.rm=TRUE)**

**write.csv(res,"res.csv")**

**clean\_res<-subset(res,res[,3]!="Not Available")**

**sorted\_res<-clean\_res[order(as.numeric(clean\_res[,3]),clean\_res[,1]),]**

**write.csv(sorted\_res,"sorted\_res.csv")**

**r\_s\_res<-sorted\_res**

**write.csv(r\_s\_res,"r\_s\_res.csv")**

**for(i in 1:nrow(r\_s\_res)){**

**r\_s\_res[i,4]<-i**

**colnames(r\_s\_res)[4]<-"Rank"**

**}**

**if(num=="best"){num<-as.numeric(1)**

**} else if (num=="worst") { num<-as.numeric(nrow(r\_s\_res)) } else {}**

**print(r\_s\_res[as.numeric(num),1])**

**}**

**rh("TX","heart failure",4)**

**Part 4 ranking hospitals in all states**

**Need to correct WORST!!**

**rankall <- function(condition, num = "best") {**

**##Sorry that I took the liberty to change the name of the first function argument, for the sake of clarity**

**## Assuming that stringr library has been loaded because we will use the word() function**

**## CHECKS if condition is a match. If not, error message via stop function.**

**pal<-sapply(strsplit(condition, " "), length)**

**##Checks if the condition is pneumonia, heart attack or heart failure**

**if(pal==1){**

**if(length(grep(word(condition),names(outcome),ignore.case=TRUE))!=0){**

**colnum<-23**

**##because pneumonia is the condition**

**} else { stop("Sorry, condition is not a match")**

**}**

**} else if(pal>=3) { stop("Sorry, condition is not a match")**

**} else { if(length(grep(word(condition,1),names(outcome),ignore.case=TRUE))!=0& length(grep(word(condition,2),names(outcome),ignore.case=TRUE))!=0){**

**if(grep(word(condition,2),acc\_cond,ignore.case=TRUE)==1){**

**colnum<-11**

**} else { colnum<-17**

**}**

**}**

**else { stop("Sorry, condition is not a match")}**

**}**

**x<-character()**

**y<-character()**

**for (i in 1:length(unique(outcome[,7]))) {**

**##subsetting main data frame by state**

**outc\_st<-subset(outcome,outcome[,7]==unique(outcome[,7])[i],na.rm=TRUE)**

**##Eliminating "Not Available" rows**

**cl\_outc\_st<-subset(outc\_st,outc\_st[,colnum]!="Not Available")**

**##In case num is best or worst**

**if(num=="best"){num1<-as.numeric(1)**

**} else if (num=="worst") { num1<-nrow(cl\_outc\_st)**

**} else {num1<-num}**

**##sorting subset by condition outcome**

**sor\_outc\_st<-cl\_outc\_st[order(as.numeric(cl\_outc\_st[,colnum]),cl\_outc\_st[,2]),]**

**##Selecting row in question**

**int\_res<-sor\_outc\_st[num1,]**

**write.csv(int\_res,"int\_res.csv")**

**x[i]<-int\_res[,2]**

**y[i]<- unique(outcome[,7])[i]**

**}**

**comp<-data.frame(x, y, stringsAsFactors=FALSE)**

**names(comp)<-c("Hospital","State")**

**write.csv(comp,"comp.csv")**

**comp\_s<-comp[order(comp$State),]**

**write.csv(comp\_s,"comp\_s.csv")**

**comp\_s**

**}**

r <- rankall("pneumonia", "worst")

as.character(subset(r, State == "NJ")$Hospital)

write.csv(sor\_outc\_st,paste("sor\_out\_st",i,".csv",sep=""))

print("Hola")

**Congratulations! You passed!**

Next Item

Question 1

Correct

1 / 1 points

**1. Question 1**

What result is returned by the following code?



1

best("SC", "heart attack")

Question 2

Correct

1 / 1 points

**2. Question 2**

What result is returned by the following code?



1

best("NY", "pneumonia")

Question 3

Correct

1 / 1 points

**3. Question 3**

What result is returned by the following code?



1

best("AK", "pneumonia")

Question 4

Correct

1 / 1 points

**4. Question 4**

What result is returned by the following code?



1

rankhospital("NC", "heart attack", "worst")

Question 5

Correct

1 / 1 points

**5. Question 5**

What result is returned by the following code?



1

rankhospital("WA", "heart attack", 7)

Question 6

Correct

1 / 1 points

**6. Question 6**

What result is returned by the following code?



1

rankhospital("TX", "pneumonia", 10)

Question 7

Correct

1 / 1 points

**7. Question 7**

What result is returned by the following code?



1

rankhospital("NY", "heart attack", 7)

Question 8

Correct

1 / 1 points

**8. Question 8**

What result is returned by the following code?



1

2

3

r <- rankall("heart attack", 4)

as.character(subset(r, state == "HI")$hospital)

Question 9

Correct

1 / 1 points

**9. Question 9**

What result is returned by the following code?



1

2

3

r <- rankall("pneumonia", "worst")

as.character(subset(r, state == "NJ")$hospital)

Question 10

Correct

1 / 1 points

**10. Question 10**

What result is returned by the following code?

For week commencing

cut(as.Date(fecha),"week",start.on.monday=FALSE)

Adding columns to a data table

DT[,w:=z^2]

DT[,m:={tmp<-(x+z);log2(tmp+5)}] ##Executes the last command, so adds another column m with the value of the last command ("it returns the evaluation of the last statement")

Install from swirl

install\_from\_swirl("Getting and Cleaning Data")

|  |  |
| --- | --- |
| Comparison {base} |  |

## Relational Operators

### Description

Binary operators which allow the comparison of values in atomic vectors.

### Usage

x < y

x > y

x <= y

x >= y

x == y

x != y

dplyr tidyr readr

The author of tidyr, Hadley Wickham, discusses his philosophy of tidy data in

| his 'Tidy Data' paper:

http://vita.had.co.nz/papers/tidy-data.pdf

# Don't change any of the code below. Just type submit()

# when you think you understand it. If you find it

# confusing, you're absolutely right!

result2 <-

arrange(

filter(

summarize(

group\_by(cran,

package),

count = n(),

unique = n\_distinct(ip\_id),

countries = n\_distinct(country),

avg\_bytes = mean(size)

),

countries > 60

),

desc(countries),

avg\_bytes

)

print(result2)

# Read the code below, but don't change anything. As

# you read it, you can pronounce the %>% operator as

# the word 'then'.

#

# Type submit() when you think you understand

# everything here.

result3 <-

cran %>%

group\_by(package) %>%

summarize(count = n(),

unique = n\_distinct(ip\_id),

countries = n\_distinct(country),

avg\_bytes = mean(size)

) %>%

filter(countries > 60) %>%

arrange(desc(countries), avg\_bytes)

# Print result to console

print(result3)

| To help drive the point home, let's work through a few more examples of chaining.

ID DateRange1Start DateRange1End Value1 DateRange2Start DateRange2End Value2 DateRange3Start DateRange3End Value3

1 1/1/90 3/1/90 4.4 4/5/91 6/7/91 6.2 5/5/95 6/6/96 3.3

reshape(dat, idvar="ID", direction="long",

varying=list(Start=c(2,5,8), End=c(3,6,9), Value=c(4,7,10)),

v.names = c("DateRangeStart", "DateRangeEnd", "Value") )

#-------------

ID time DateRangeStart DateRangeEnd Value

1.1 1 1 1/1/90 3/1/90 4.4

1.2 1 2 4/5/91 6/7/91 6.2

1.3 1 3 5/5/95 6/6/96 3.3

<http://garrettgman.github.io/tidying/>

**ext\_wh<-function(){**

**alsip<-read.csv("alsip ph count.csv")**

**alsip2a<-subset(alsip,select=c(1,2,3,4,5,6,7,8,16))**

**alsip2b<-subset(alsip,select=c(1,2,3,9,10,11,12,13,16))**

**alsip3a<-gather(alsip2a,"lot","lotnum",4:8)**

**write.csv(alsip3a,"alsip3a.csv")**

**alsip3b<-gather(alsip2b,"cs\_lot","cs",4:8)**

**write.csv(alsip3b,"alsip3b.csv")**

**##alsip4a<-subset(alsip3a,lotnum!="NA")**

**##alsip4b<-subset(alsip3b,cs!="NA")**

**alsip4a<-alsip3a**

**alsip4b<-alsip3b**

**write.csv(alsip4b,"alsip4b.csv")**

**write.csv(alsip4a,"alsip4a.csv")**

**alsip5a<-mutate(alsip4a,index=substr(alsip4a$lot,nchar(alsip4a$lot),nchar(alsip4a$lot)))**

**write.csv(alsip5a,"alsip5a.csv")**

**alsip5b<-mutate(alsip4b,index=substr(alsip4b$cs\_lot,nchar(alsip4b$cs\_lot),nchar(alsip4b$cs\_lot)))**

**write.csv(alsip5b,"alsip5b.csv")**

**alsip6<-merge(alsip5a,alsip5b,by=c("Load","index","X70"),all=FALSE)**

**write.csv(alsip6,"alsip6.csv")**

**## ~~alsip7<-group\_by(alsip6,tolower(Item),Load,lotnum)~~**

**alsip7<-group\_by(arrange(mutate(alsip6,ni=substr(Item.x,1,2)),ni,Load,lotnum),ni,Load,lotnum, tolower(Item.x),ignore.case=TRUE)**

**##(or toupper)**

**##solo al hacer la siguiente instrucción se ponen todos los items en mayúscula (o minúscula), curioso.**

**##Y el colname cambia a tolower(Item)**

**alsip8<-summarize(alsip7,tot=sum(cs))**

**write.csv(alsip8,"alsip8.csv")**

**alsip9<-mutate(alsip8,pounds=tot\*70)**

**alsip9bis<-alsip9[complete.cases(alsip9),]**

**alsip9\_sim<-select(alsip9bis,-ignore.case)**

**alsip9\_fin<-summarize(group\_by(alsip9\_sim,Load,lotnum),sum(tot),sum(pounds))**

**write.csv(alsip9bis,"alsip9bis.csv")**

**write.csv(alsip9\_sim,"alsip9\_sim.csv")**

**write.csv(alsip9\_fin,"alsip9\_fin.csv")**

**alsip9\_no\_load<-summarize(group\_by(alsip9bis,ni,lotnum),sum(tot),sum(pounds))**

**write.csv(alsip9\_no\_load,"alsip9\_no\_load.csv")**

**}**

mutate(alsip9\_no\_load,lb=

**Cleaning DATA week1  
Congratulations! You passed!**

Next Item

Question 1

Correct

1 / 1

point

## 1. Question 1

The American Community Survey distributes downloadable data about United States communities. Download the 2006 microdata survey about housing for the state of Idaho using download.file() from here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06hid.csv>

and load the data into R. The code book, describing the variable names is here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FPUMSDataDict06.pdf>

How many properties are worth $1,000,000 or more?



164



47



31



53

**Correct**

Question 2

Correct

1 / 1

point

## 2. Question 2

Use the data you loaded from Question 1. Consider the variable FES in the code book. Which of the "tidy data" principles does this variable violate?



Numeric values in tidy data can not represent categories.



Each variable in a tidy data set has been transformed to be interpretable.



Tidy data has no missing values.



Tidy data has one variable per column.

**Correct**

Question 3

Correct

1 / 1

point

## 3. Question 3

Download the Excel spreadsheet on Natural Gas Aquisition Program here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FDATA.gov_NGAP.xlsx>

Read rows 18-23 and columns 7-15 into R and assign the result to a variable called:



1

dat

What is the value of:



1

sum(dat$Zip\*dat$Ext,na.rm=T)

(original data source: <http://catalog.data.gov/dataset/natural-gas-acquisition-program>)



NA



0



154339



36534720

**Correct**

Question 4

Correct

1 / 1

point

## 4. Question 4

Read the XML data on Baltimore restaurants from here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Frestaurants.xml>

How many restaurants have zipcode 21231?



127

**Correct**



17



156



100

Question 5

Incorrect

0 / 1

point

## 5. Question 5

The American Community Survey distributes downloadable data about United States communities. Download the 2006 microdata survey about housing for the state of Idaho using download.file() from here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06pid.csv>

using the fread() command load the data into an R object



1

DT

The following are ways to calculate the average value of the variable



1

pwgtp15

broken down by sex. Using the data.table package, which will deliver the fastest user time?



mean(DT[DT$SEX==1,]$pwgtp15); mean(DT[DT$SEX==2,]$pwgtp15)

**This should not be selected**



sapply(split(DT$pwgtp15,DT$SEX),mean)



rowMeans(DT)[DT$SEX==1]; rowMeans(DT)[DT$SEX==2]



mean(DT$pwgtp15,by=DT$SEX)



DT[,mean(pwgtp15),by=SEX]



tapply(DT$pwgtp15,DT$SEX,mean)

## Sorting by Multiple Columns (chartio.com)

**In some cases, it may be desired to sort by multiple columns. Thankfully, doing so is very simple with the previously described methods.**

**To sort multiple columns using vector names, simply add additional arguments to the order() function call as before:**

***# Sort by vector name [z] then [x]***

**dataframe[**

**with(dataframe, order(z, x)),**

**]**

**Similarly, to sort by multiple columns based on column index, add additional arguments to order() with differing indices:**

***# Sort by column index [1] then [3]***

**dataframe[**

**order( dataframe[,1], dataframe[,3] ),**

**]**

**crsst<-function() {**

**##To turn results table into cross table values**

**##Update source file name!!**

**ovw<-read.csv("OvW WeekC Oct 14.csv")**

**colnames(ovw)<-c("LocTyp","Wc","OvW")**

**##Converting to.Date and sorting by descending date**

**wc<-as.Date(ovw$Wc,"%d-%b-%y")**

**ovw\_d<-data.frame(loc\_typ=ovw$LocTyp,wc=wc,sobre=ovw$OvW)**

**ovw\_ord<-arrange(ovw\_d,desc(wc),loc\_typ)**

**povw<- data.frame(loc\_typ=ovw\_ord$loc\_typ,wc=wc,sobre=ovw\_ord$sobre)**

**write.csv(povw,"povw.csv")**

**loc<-unique(povw[,1])**

**weekc<-unique(povw[,2])**

**write.csv(weekc,"weekc.csv")**

**##Initializing the dataframe**

**crsstbl<-data.frame(matrix(ncol=length(weekc),nrow=length(loc)))**

**for (i in 1:length(loc)){**

**for (j in 1:length(weekc)){**

**if(length(subset(povw,loc\_typ==loc[i]&wc==weekc[j])[,3])!=0){**

**crsstbl[i,j]<-subset(povw,loc\_typ==loc[i]&wc==weekc[j])[,3]**

**} else{crsstbl[i,j]<-"NA"}**

**}**

**}**

**for (i in 1:length(loc)){ rownames(crsstbl)[i]<-as.character(loc[i]) }**

**for (j in 1:length(weekc)){colnames(crsstbl)[j]<-as.character(weekc[j])}**

**write.csv(crsstbl,"crsstbl.csv")**

**}**

[**Check whether value exist in one data frame or not**](https://stackoverflow.com/questions/13774773/check-whether-value-exist-in-one-data-frame-or-not)

I have two data frames (A,B)having same column names(C), but can have different unique values in that column. I want to check if 'value' in column (C) in data frame (A) exists in data frame (B).

A = data.frame(C=c(1,2,3,4))

B = data.frame(C=c(1,3,4,7))

In above example, I want to check if '2' is present in B or not Is there any one liner without loop, as I have pretty big files and would have to check this at every line.

setdiff may be of interest – [James](https://stackoverflow.com/users/269476/james) [Dec 8 '12 at 13:01](https://stackoverflow.com/questions/13774773/check-whether-value-exist-in-one-data-frame-or-not#comment18943627_13774773)

Use %in% as follows

A$C %in% B$C

Which will tell you which values of column C of A are in B.

What is returned is a logical vector. In the specific case of your example, you get:

A$C %in% B$C

# [1] TRUE FALSE TRUE TRUE

Which you can use as an index to the rows of A or as an index to A$C to get the actual values:

# as a row index

A[A$C %in% B$C, ] # note the comma to indicate we are indexing rows

# as an index to A$C

A$C[A$C %in% B$C]

[1] 1 3 4 # returns all values of A$C that are in B$C

We can negate it too:

A$C[!A$C %in% B$C]

[1] 2   # returns all values of A$C that are NOT in B$C

If you want to know if a specific value is in B$C, use the same function:

2 %in% B$C # "is the value 2 in B$C ?"

# FALSE

A$C[2] %in% B$C # "is the 2nd element of A$C in B$C ?"

# FALSE

getwd()

[1] "C:/Users/javier.lores/OneDrive - EVANS FOOD PRODUCTS/Documents/R"

setwd("..")

getwd()

[1] "C:/Users/javier.lores/OneDrive - EVANS FOOD PRODUCTS/Documents"

setwd(".")

getwd()

[1] "C:/Users/javier.lores/OneDrive - EVANS FOOD PRODUCTS/Documents"

setwd("./R")

[1] "C:/Users/javier.lores/OneDrive - EVANS FOOD PRODUCTS/Documents/R"

# Append two more function calls to accomplish the following:

#

# 1. Use group\_by() (from dplyr) to group the data by part and

# sex, in that order.

#

# 2. Use mutate to add two new columns, whose values will be

# automatically computed group-by-group:

#

# \* total = sum(count)

# \* prop = count / total

#

sat %>%

select(-contains("total")) %>%

gather(part\_sex, count, -score\_range) %>%

separate(part\_sex, c("part", "sex")) %>%

group\_by(part,sex) %>%

mutate(total=sum(count),

prop=count/total

) %>% print

FROM

sat

# A tibble: 6 x 10

score\_range read\_male read\_fem read\_total math\_male math\_fem math\_total

<chr> <int> <int> <int> <int> <int> <int>

1 700-800 40151 38898 79049 74461 46040 120501

2 600-690 121950 126084 248034 162564 133954 296518

3 500-590 227141 259553 486694 233141 257678 490819

4 400-490 242554 296793 539347 204670 288696 493366

5 300-390 113568 133473 247041 82468 131025 213493

6 200-290 30728 29154 59882 18788 26562 45350

# ... with 3 more variables: write\_male <int>, write\_fem <int>,

# write\_total <int>

A tibble: 36 x 6

# Groups: part, sex [6]

score\_range part sex count total prop

<chr> <chr> <chr> <int> <int> <dbl>

1 700-800 read male 40151 776092 0.0517

2 600-690 read male 121950 776092 0.157

3 500-590 read male 227141 776092 0.293

4 400-490 read male 242554 776092 0.313

5 300-390 read male 113568 776092 0.146

6 200-290 read male 30728 776092 0.0396

7 700-800 read fem 38898 883955 0.0440

8 600-690 read fem 126084 883955 0.143

9 500-590 read fem 259553 883955 0.294

10 400-490 read fem 296793 883955 0.336

# ... with 26 more rows

a <- c(rep("A", 3), rep("B", 3), rep("C",2))

b <- c(1,1,2,4,1,1,2,2)

df <-data.frame(a,b)

duplicated(df)

[1] FALSE TRUE FALSE FALSE FALSE TRUE FALSE TRUE

> df[duplicated(df), ]

a b

2 A 1

6 B 1

8 C 2

> df[!duplicated(df), ]

a b

1 A 1

3 A 2

4 B 4

5 B 1

7 C 2

"C:/Users/JAVIER~1.LOR/ONEDRI~1/DOCUME~1/R/R-35~1.1"

HDF5 interface to R

### Installation

To install this package, start R and enter:

## try http:// if https:// URLs are not supported

source("https://bioconductor.org/biocLite.R")

biocLite("rhdf5")

### Documentation

To view documentation for the version of this package installed in your system, start R and enter:

browseVignettes("rhdf5")

Github App token

secret, I suppose

2abd9c85850d5ba43640a18d851c3a8d03ac09f4

kosmos

(https://auth0.com/docs/connections/social/github)

**https://github.com/r-lib/httr/blob/master/demo/oauth2-github.r**

|  |
| --- |
| myapp <- oauth\_app("github", |
|  | key = "c5b8d1b6921f50e98414", |
|  | secret = "595248eae6590aee0b19bbf1a84a248cac023390") |

dir<-"https://d396qusza40orc.cloudfront.net/getdata%2Fwksst8110.for"

example<-read.fwf(dir,widths=c(10,9,4,9,4,9,4,9,4),skip=4)

> col4<-as.numeric(example[,4])

> sum(col4)

[1] 32426.7

**ongratulations! You passed!**

Next Item

Question 1

Correct

1 / 1

point

## 1. Question 1

Register an application with the Github API here <https://github.com/settings/applications>. Access the API to get information on your instructors repositories (hint: this is the url you want "https://api.github.com/users/jtleek/repos"). Use this data to find the time that the datasharing repo was created. What time was it created?

This tutorial may be useful (<https://github.com/hadley/httr/blob/master/demo/oauth2-github.r>). You may also need to run the code in the base R package and not R studio.



2014-03-05T16:11:46Z



2012-06-20T18:39:06Z



2013-11-07T13:25:07Z

**Correct**



2013-08-28T18:18:50Z

Question 2

Incorrect

0 / 1

point

## 2. Question 2

The sqldf package allows for execution of SQL commands on R data frames. We will use the sqldf package to practice the queries we might send with the dbSendQuery command in RMySQL.

Download the American Community Survey data and load it into an R object called



1

acs

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06pid.csv>

Which of the following commands will select only the data for the probability weights pwgtp1 with ages less than 50?



sqldf("select pwgtp1 from acs")

**This should not be selected**



sqldf("select pwgtp1 from acs where AGEP \lt< 50")



sqldf("select \* from acs")



sqldf("select \* from acs where AGEP \lt< 50 and pwgtp1")

Question 3

Correct

1 / 1

point

## 3. Question 3

Using the same data frame you created in the previous problem, what is the equivalent function to unique(acs$AGEP)



sqldf("select unique AGEP from acs")



sqldf("select distinct AGEP from acs")

**Correct**



sqldf("select AGEP where unique from acs")



sqldf("select distinct pwgtp1 from acs")

Question 4

Correct

1 / 1

point

## 4. Question 4

How many characters are in the 10th, 20th, 30th and 100th lines of HTML from this page:

http://biostat.jhsph.edu/~jleek/contact.html

(Hint: the nchar() function in R may be helpful)



45 92 7 2



45 31 7 31



43 99 8 6



43 99 7 25



45 31 2 25



45 0 2 2



45 31 7 25

**Correct**

Question 5

Correct

1 / 1

point

## 5. Question 5

Read this data set into R and report the sum of the numbers in the fourth of the nine columns.

<https://d396qusza40orc.cloudfront.net/getdata%2Fwksst8110.for>

Original source of the data: <http://www.cpc.ncep.noaa.gov/data/indices/wksst8110.for>

(Hint this is a fixed width file format)



101.83



35824.9



222243.1



32426.7

**Correct**



28893.3



36.5

quantile(pack\_sum$unique,probs=0.99)

99%

465

filter(pack\_sum,unique>465)

by\_package <- group\_by(cran, package)

pack\_sum <- summarize(by\_package,

count = n(),

unique = n\_distinct(ip\_id),

countries = n\_distinct(country),

avg\_bytes = mean(size))

# Here's the new bit, but using the same approach we've

# been using this whole time.

top\_countries <- filter(pack\_sum, countries > 60)

result1 <- arrange(top\_countries, desc(countries), avg\_bytes)

# Print the results to the console.

print(result1)

Is equivalent to…

result2 <-

arrange(

filter(

summarize(

group\_by(cran,

package

),

count = n(),

unique = n\_distinct(ip\_id),

countries = n\_distinct(country),

avg\_bytes = mean(size)

),

countries > 60

),

desc(countries),

avg\_bytes

)

print(result2)

equivalent to

result3 <-

cran %>%

group\_by(package) %>%

summarize(count = n(),

unique = n\_distinct(ip\_id),

countries = n\_distinct(country),

avg\_bytes = mean(size)

) %>%

filter(countries > 60) %>%

arrange(desc(countries), avg\_bytes)

# Print result to console

print(result3)

library("dplyr")

#>

#> Attaching package: 'dplyr'

#> The following objects are masked from 'package:stats':

#>

#> filter, lag

#> The following objects are masked from 'package:base':

#>

#> intersect, setdiff, setequal, union

db <- src\_sqlite(tempfile(), create = TRUE)

iris2 <- copy\_to(db, iris)

**vec <- pull(iris2, Species)**

**head(vec)**

#> [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"

collect(select(iris2, Species))[[1]]

library(magrittr)

library(dplyr)

iris2 %>%

select(Species) %>%

extract2(1)

rlang::last\_error()

## Steve C Walker

# Remove (or replace) everything before or after a specified character in R strings

FEBRUARY 13, 2013

No time to explain this one, but here’s an example:

> x <- 'aabb.ccdd'

> sub('.\*', '', x)

[1] ""

> sub('bb.\*', '', x)

[1] "aa"

> sub('.\*bb', '', x)

[1] ".ccdd"

> sub('\\..\*', '', x)

[1] "aabb"

> sub('.\*\\.', '', x)

[1] "ccdd"

No need for [substring](http://help.r-enthusiasts.com/library/base/html/substr.html), just use [gsub](http://help.r-enthusiasts.com/library/base/html/grep.html):

gsub( " .\*$", "", dob )

# [1] "9/9/43" "9/17/88" "11/21/48"

A space (), then any character (.) any number of times (\*) until the end of the string ($). See [?regex](http://help.r-enthusiasts.com/library/base/html/regex.html) to learn regular expressions.

[share](https://stackoverflow.com/a/15895183)[improve this answer](https://stackoverflow.com/posts/15895183/edit)

answered Apr 9 '13 at 6:51

[Romain Francois](https://stackoverflow.com/users/499163/romain-francois)

**14.5k**23769

* 2

The only advice I can share here is that a sub is enough since there is only one string end position. – [Wiktor Stribiżew](https://stackoverflow.com/users/3832970/wiktor-stribi%c5%bcew) [Oct 12 '16 at 21:50](https://stackoverflow.com/questions/15895050/using-gsub-to-extract-character-string-before-white-space-in-r#comment67296027_15895183)

**> b<-substr(as.character(chi\_yp[7,33]),1,nchar(as.character(chi\_yp[7,33])))**

**> b**

**[1] "6,267.00"**

**> c<-as.numeric(sub(",.\*$","",b))\*1000+as.numeric(sub('.\*\\,', '', b))**

**> c**

**[1] 6267**

sat %>% select(-contains(total))

parse\_number(class\_5)

update(this\_moment, hours = 8, minutes = 34, seconds = 55)

vs

this\_moment<-update(this\_moment, hours = 8, minutes = 34, seconds = 55)

http://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones

pull fuciona con factors si es un data.frame, no si es un tibble!!

library(tidyr)

library(readr) ## parse\_number… incluso de factors

library(stringr)

library(varhandle)

library(assertthat)

library(dplyr)

library(lubridate)

##help(package = lubridate)

library(quantmod)

library(reshape2)

library(magrittr)

Quiz week 3 getting and cleaning data

setwd("C:/Users/javier.lores/OneDrive - EVANS FOOD PRODUCTS/Documents/R")

> acs\_id<-read.csv("acs\_id.csv")

> agricultureLogical<-as.logical(acs\_id$ACR==3&acs\_id$AGS==6)

> length(agricultureLogical)

[1] 6496

> which(agricultureLogical)

[1] **125 238 262** 470 555 568 608 643 787 808 824 849 952 955 1033

[16] 1265 1275 1315 1388 1607 1629 1651 1856 1919 2101 2194 2403 2443 2539 2580

[31] 2655 2680 2740 2838 2965 3131 3133 3163 3291 3370 3402 3585 3652 3852 3862

[46] 3912 4023 4045 4107 4113 4117 4185 4198 4310 4343 4354 4448 4453 4461 4718

[61] 4817 4835 4910 5140 5199 5236 5326 5417 5531 5574 5894 6033 6044 6089 6275

[76] 6376 6420

foto<-readJPEG("Fjeff.jpg",native=TRUE) ##after downloading the pic in my R folder

quantile(foto,probs=.30)

30%

-15259150

> quantile(foto,probs=.80)

80%

-10575416

edu<-read.csv("edu.csv")

gdp\_c<-read.csv("gdp.csv",skip=4,nrows=190)

todo<-merge(gdp\_c,edu,by.x="X",by.y="CountryCode",all.x=FALSE,no.dups=TRUE)

dim(todo)

[1] 189 40

todo\_sor<-arrange(todo,desc(X.1))

todo\_sor[13,4]

[1] St. Kitts and Nevis

190 Levels: Afghanistan Albania Algeria Angola Antigua and Barbuda ... Zimbabwe

todo\_g<-group\_by(todo,Income.Group)

View(todo\_g)

todo\_g\_sum<-summarize(todo\_g,mean=mean(X.1))

View(todo\_g\_sum)

todo\_q<-select(filter(todo,Income.Group=="Lower middle income",X.1<=38),X,X.1,Income.Group)

View(todo\_q)

**Congratulations! You passed!**

Question 1

Correct

1 / 1

point

## 1. Question 1

The American Community Survey distributes downloadable data about United States communities. Download the 2006 microdata survey about housing for the state of Idaho using download.file() from here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06hid.csv>

and load the data into R. The code book, describing the variable names is here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FPUMSDataDict06.pdf>

Create a logical vector that identifies the households on greater than 10 acres who sold more than $10,000 worth of agriculture products. Assign that logical vector to the variable agricultureLogical. Apply the which() function like this to identify the rows of the data frame where the logical vector is TRUE.

which(agricultureLogical)

What are the first 3 values that result?



403, 756, 798



59, 460, 474



25, 36, 45



125, 238,262

**Correct**

Question 2

Correct

1 / 1

point

## 2. Question 2

Using the jpeg package read in the following picture of your instructor into R

<https://d396qusza40orc.cloudfront.net/getdata%2Fjeff.jpg>

Use the parameter native=TRUE. What are the 30th and 80th quantiles of the resulting data? (some Linux systems may produce an answer 638 different for the 30th quantile)



-16776430 -15390165



-15259150 -10575416

**Correct**



10904118 -594524



-10904118 -10575416

Question 3

Correct

1 / 1

point

## 3. Question 3

Load the Gross Domestic Product data for the 190 ranked countries in this data set:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FGDP.csv>

Load the educational data from this data set:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FEDSTATS_Country.csv>

Match the data based on the country shortcode. How many of the IDs match? Sort the data frame in descending order by GDP rank (so United States is last). What is the 13th country in the resulting data frame?

Original data sources:

<http://data.worldbank.org/data-catalog/GDP-ranking-table>

<http://data.worldbank.org/data-catalog/ed-stats>



189 matches, 13th country is St. Kitts and Nevis

**Correct**



190 matches, 13th country is St. Kitts and Nevis



189 matches, 13th country is Spain



190 matches, 13th country is Spain



234 matches, 13th country is Spain



234 matches, 13th country is St. Kitts and Nevis

Question 4

Correct

1 / 1

point

## 4. Question 4

What is the average GDP ranking for the "High income: OECD" and "High income: nonOECD" group?



23, 45



133.72973, 32.96667



32.96667, 91.91304

**Correct**



23, 30



30, 37



23.966667, 30.91304

Question 5

Correct

1 / 1

point

## 5. Question 5

Cut the GDP ranking into 5 separate quantile groups. Make a table versus Income.Group. How many countries

are Lower middle income but among the 38 nations with highest GDP?



0



12



5

**Correct**



13

flr\_date<-filter(flr,!grepl('tes',type,ignore.case=TRUE))

count(flr\_date,type)

View(count(flr\_date,type))

grep("stringofinterest",names(dataframeofinterest),ignore.case=TRUE,value=TRUE)

without the argument value=TRUE you will only get a vector of index positions where the match occurred.

With the stringr package, you can modify the pattern with one of the built in modifier functions (see `?modifiers). For example since we are matching a fixed string (no special regular expression characters) but want to ignore case, we can do

str\_detect(colnames(iris), fixed("species", ignore\_case=TRUE))

Or you can use the (?i) case insensitive [modifier](https://www.regular-expressions.info/modifiers.html)

str\_detect(colnames(iris), "(?i)species")

QUIZ: Cleaning data, course 3, week 4,

##Selecting only the observations that contain numbers in the "Gross…" column using the parse\_number function and filter out for NA

gdp<-read.csv("gdp.csv")

View(gdp)

class(gdp[,2])

[1] "factor"

class(gdp[8,2])

[1] "factor"

gdp\_s<-arrange(gdp,gdp[,2])

View(gdp\_s)

gdp\_r<-mutate(gdp,rank=parse\_number(Gross.domestic.product.2012))

Warning: 5 parsing failures.

row # A tibble: 5 x 4 col row col expected actual expected <int> <int> <chr> <chr> actual 1 3 NA a number Ranking row 2 237 NA a number . col 3 238 NA a number . expected 4 239 NA a number . actual 5 240 NA a number .

View(gdp\_r)

gdp\_rf<-filter(gdp\_r,is.na(rank)==FALSE)

View(gdp\_rf)

##To calculate the mean of the GDP after removing commas

mean(parse\_number(gdp\_rf$X.3))

[1] 377652.4

or

mean(as.numeric(as.character(gsub(",","",gdp\_rf$X.3))))

[1] 377652.4

##to count the number of countries whose name starts with United

grep("^United",gdp\_rf$X.2)

[1] 1 6 32

length(grep("^United",gdp\_rf$X.2))

[1] 3

##to merge data frames and count how many countrys' fiscal year ends in June

todo<-merge(edu,gdp\_rf,by.x="CountryCode",by.y="X",all.x=FALSE,no.dups=TRUE)

View(todo)

grep("Fiscal year end(.\*)[Jj]une",todo$Special.Notes)

[1] 9 16 29 51 65 89 96 133 140 152 159 175 189

length(grep("Fiscal year end(.\*)[Jj]une",todo$Special.Notes))

[1] 13

##more resrtrictive

length(grep("Fiscal year end: June",todo$Special.Notes))

[1] 13

##to know how many times AMZN stock value sampled in 2012 and how many on Mondays, after a few tests

dim(sampleTimes)

NULL

View(sampleTimes)

st<-sampleTimes

class(st)

[1] "Date"

st2012<-st[grep("2012",st)]

View(st2012)

length(st2012)

[1] 250

wday(st2012[1])

[1] 3

wday(st2012[1],label=TRUE)

[1] Tue

Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat

length(grep("Mon",wday(st2012,label=TRUE)))

[1] 47

**Congratulations! You passed!**

Next Item

Question 1

Correct

1 / 1

point

## 1. Question 1

The American Community Survey distributes downloadable data about United States communities. Download the 2006 microdata survey about housing for the state of Idaho using download.file() from here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06hid.csv>

and load the data into R. The code book, describing the variable names is here:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FPUMSDataDict06.pdf>

Apply strsplit() to split all the names of the data frame on the characters "wgtp". What is the value of the 123 element of the resulting list?



"wgtp" "15"



"wgtp"



"" "15"

**Correct**



"wgt" "15"

Question 2

Correct

1 / 1

point

## 2. Question 2

Load the Gross Domestic Product data for the 190 ranked countries in this data set:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FGDP.csv>

Remove the commas from the GDP numbers in millions of dollars and average them. What is the average?

Original data sources:

<http://data.worldbank.org/data-catalog/GDP-ranking-table>



387854.4



381615.4



377652.4

**Correct**



381668.9

Question 3

Correct

1 / 1

point

## 3. Question 3

In the data set from Question 2 what is a regular expression that would allow you to count the number of countries whose name begins with "United"? Assume that the variable with the country names in it is named countryNames. How many countries begin with United?



grep("\*United",countryNames), 2



grep("^United",countryNames), 3

**Correct**



grep("United$",countryNames), 3



grep("^United",countryNames), 4

Question 4

Correct

1 / 1

point

## 4. Question 4

Load the Gross Domestic Product data for the 190 ranked countries in this data set:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FGDP.csv>

Load the educational data from this data set:

<https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FEDSTATS_Country.csv>

Match the data based on the country shortcode. Of the countries for which the end of the fiscal year is available, how many end in June?

Original data sources:

<http://data.worldbank.org/data-catalog/GDP-ranking-table>

<http://data.worldbank.org/data-catalog/ed-stats>



13

**Correct**



8



16



7

Question 5

Correct

1 / 1

point

## 5. Question 5

You can use the quantmod (<http://www.quantmod.com/>) package to get historical stock prices for publicly traded companies on the NASDAQ and NYSE. Use the following code to download data on Amazon's stock price and get the times the data was sampled.

library(quantmod)

amzn = getSymbols("AMZN",auto.assign=FALSE)

sampleTimes = index(amzn)

How many values were collected in 2012? How many values were collected on Mondays in 2012?



251,51



365, 52



252, 50



250, 47

**Correct**

read.table(dat, header = FALSE, sep = ",",

col.names = paste0("V",seq\_len(7)), fill = TRUE)

read.table**(**x\_train, header = FALSE, sep = "",

col.names = c**(**acc1,acc2,acc3,w1,w2,w3,paste0("V",seq\_len(561))**))**

y\_train<-scan("y\_train.txt",character(),quote="")

##Read 7352 items

##View(y\_train)

s\_train<-scan("subject\_train.txt",character(),quote="")

##Read 2947 items

xx\_train<-read.table("X\_train.txt", header = FALSE, sep = "", nrows=7352)

##dim(xx\_train)

##[1] 7352 561

##xx\_train[7352,561]

##[1] 0.03669484

##o bien (parece que da mas garantia; he hecho comprobaciones de elementos de ambas y son iguales, and it is safe because it is explained in the README.txt that one row is one set)

## xxbis\_train<-read.table("X\_train.txt", header = FALSE, sep = "", col.names=feat\_ps)

## dim(xxbis\_train)

> ?cbind

> yxx\_train<-cbind(y\_train,xx\_train)

> syxx\_train<-cbind(s\_train,yxx\_train)

> dim(syxx\_train)

[1] 7352 563

> str(xyxx\_train)

'data.frame': 7352 obs. of 563 variables:

$ s\_train: Factor w/ 21 levels "1","11","14",..: 1 1 1 1 1 1 1 1 1 1 ...

$ y\_train: Factor w/ 6 levels "1","2","3","4",..: 5 5 5 5 5 5 5 5 5 5 ...

$ V1 : num 0.289 0.278 0.28 0.279 0.277 ...

$ V2 : num -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...

$ V3 : num -0.133 -0.124 -0.113 -0.123 -0.115 ...

$ V4 : num -0.995 -0.998 -0.995 -0.996 -0.998 ...

…

…

[list output truncated]

> syxx\_train[7352,563]

[1] 0.03669484

> y\_test<-scan("y\_test.txt",character(),quote="")

Read 2947 items

> s\_test<-scan("subject\_test.txt",character(),quote="")

Read 2947 items

> xx\_test<-read.table("X\_test.txt", header = FALSE, sep = "", nrows=2947)

> dim(xx\_test)

[1] 2947 561

> syxx\_test<-cbind(s\_test,y\_test,xx\_test)

> dim(syxx\_test)

[1] 2947 563

##kind of restarting

feat<-read.delim("features.txt",header=FALSE,sep="")

##> class(feat)

##[1] "data.frame"

##> dim(feat)

##[1] 561 3

##> View(feat)

##> feat\_ps<-substr(as.character(feat\_p$V1),3,nchar(as.character(feat\_p$V1))-1)

##> View(feat\_ps)

##> library(stringr)

##> feat\_ps<-str\_trim(substr(as.character(feat\_p$V1),3,nchar(as.character(feat\_p$V1))-1))

##> View(feat\_ps)

##> length(feat\_ps)

##[1] 561

> colnames(syxx\_test)<-c("subject","activity",feat[,2])

> colnames(syxx\_train)<-c("subject","activity",feat[,2])

##I am going to bind the columns of what the authors call the inertial signals, although it seems that it would not really be necessary because there are no mean or sd calculated variables in those files and therefore nothing to extract

sigt1<-read.table("./UCI HAR Dataset/train/Inertial Signals/body\_acc\_x\_train.txt", header = FALSE, sep = "", col.names= paste0("baxt",seq\_len(128)))

> sigt2<-read.table("./UCI HAR Dataset/train/Inertial Signals/body\_acc\_y\_train.txt", header = FALSE, sep = "", col.names= paste0("bayt",seq\_len(128)))

> sigt3<-read.table("./UCI HAR Dataset/train/Inertial Signals/body\_acc\_z\_train.txt", header = FALSE, sep = "", col.names= paste0("bazt",seq\_len(128)))

> sigt4<-read.table("./UCI HAR Dataset/train/Inertial Signals/body\_gyro\_x\_train.txt", header = FALSE, sep = "", col.names= paste0("bgxt",seq\_len(128)))

> sigt5<-read.table("./UCI HAR Dataset/train/Inertial Signals/body\_gyro\_y\_train.txt", header = FALSE, sep = "", col.names= paste0("bgyt",seq\_len(128)))

> sigt6<-read.table("./UCI HAR Dataset/train/Inertial Signals/body\_gyro\_z\_train.txt", header = FALSE, sep = "", col.names= paste0("bgzt",seq\_len(128)))

> sigt7<-read.table("./UCI HAR Dataset/train/Inertial Signals/total\_acc\_x\_train.txt", header = FALSE, sep = "", col.names= paste0("taxt",seq\_len(128)))

> sigt8<-read.table("./UCI HAR Dataset/train/Inertial Signals/total\_acc\_y\_train.txt", header = FALSE, sep = "", col.names= paste0("tayt",seq\_len(128)))

> sigt9<-read.table("./UCI HAR Dataset/train/Inertial Signals/total\_acc\_z\_train.txt", header = FALSE, sep = "", col.names= paste0("tazt",seq\_len(128)))

sigp1<-read.table("./UCI HAR Dataset/test/Inertial Signals/body\_acc\_x\_test.txt", header = FALSE, sep = "", col.names= paste0("baxt",seq\_len(128)))

sigp2<-read.table("./UCI HAR Dataset/test/Inertial Signals/body\_acc\_y\_test.txt", header = FALSE, sep = "", col.names= paste0("bayt",seq\_len(128)))

sigp3<-read.table("./UCI HAR Dataset/test/Inertial Signals/body\_acc\_z\_test.txt", header = FALSE, sep = "", col.names= paste0("bazt",seq\_len(128)))

sigp4<-read.table("./UCI HAR Dataset/test/Inertial Signals/body\_gyro\_x\_test.txt", header = FALSE, sep = "", col.names= paste0("bgxt",seq\_len(128)))

sigp5<-read.table("./UCI HAR Dataset/test/Inertial Signals/body\_gyro\_y\_test.txt", header = FALSE, sep = "", col.names= paste0("bgyt",seq\_len(128)))

sigp6<-read.table("./UCI HAR Dataset/test/Inertial Signals/body\_gyro\_z\_test.txt", header = FALSE, sep = "", col.names= paste0("bgzt",seq\_len(128)))

sigp7<-read.table("./UCI HAR Dataset/test/Inertial Signals/total\_acc\_x\_test.txt", header = FALSE, sep = "", col.names= paste0("taxt",seq\_len(128)))

sigp8<-read.table("./UCI HAR Dataset/test/Inertial Signals/total\_acc\_y\_test.txt", header = FALSE, sep = "", col.names= paste0("tayt",seq\_len(128)))

sigp9<-read.table("./UCI HAR Dataset/test/Inertial Signals/total\_acc\_z\_test.txt", header = FALSE, sep = "", col.names= paste0("tazt",seq\_len(128)))

##tot\_train<-cbind(syxx\_train, sigt1,sigt2,sigt3,sigt4,sigt5,sigt6,sigt7,sigt8,sigt9)

##tot\_test<-cbind(syxx\_test, sigp1,sigp2,sigp3,sigp4,sigp5,sigp6,sigp7,sigp8,sigp9)

##Adding one column with one character value, adding a constant, with the type of subject partition (train/test)

##tot\_test\_t<-cbind(tot\_test, paste0("test"))

##tot\_train\_t<-cbind(tot\_train, paste0("train"))

##binding both data sets

tot\_bin<-rbind(tot\_train\_t,tot\_test\_t)

tot\_bin1<-rbind(sigt1,sigp1)

tot\_bin2<-rbind(sigt2,sigp2)

tot\_bin3<-rbind(sigt3,sigp3)

tot\_bin4<-rbind(sigt4,sigp4)

tot\_bin5<-rbind(sigt5,sigp5)

tot\_bin6<-rbind(sigt6,sigp6)

tot\_bin7<-rbind(sigt7,sigp7)

tot\_bin8<-rbind(sigt8,sigp8)

tot\_bin9<-rbind(sigt9,sigp9)

tot\_bin\_all<-cbind(tot\_bin1,tot\_bin2,tot\_bin3,tot\_bin4,tot\_bin5,tot\_bin6,tot\_bin7,tot\_bin8,tot\_bin9)

tot\_syxx<-rbind(syxx\_train,syxx\_test)

dim(tot\_syxx)

[1] 10299 563

dim(tot\_bin\_all)

[1] 10299 1152

tot\_tot<-cbind(tot\_syxx,tot\_bin\_all)

dim(tot\_tot)

[1] 10299 1715

##showing the names of the activities rather than having just a number

act\_nam<-read.delim("activity\_labels.txt",header=FALSE,sep="")

colnames(act\_nam)<-c("activity","activityname")

data\_for\_analysis<-merge(tot\_tot,act\_nam,all.x=TRUE)

colnames(data\_for\_analysis)[1716]

[1] "activityname"

##I should have been merging all the time, not cbinding!!?? No, because there are no keys

data\_ms<-data\_for\_analysis[,grep("activ|subje|mean|std",colnames(data\_for\_analysis))]

##Eliminate columns data are not calculated means of a variable (meanFreq)

des<-c(grep("freq",colnames(data\_ms),ignore.case=TRUE))

data\_ms\_2<-data\_ms[,-des]

##remove dashes from col names

b<- mapply(gsub,"-","",colnames(data\_ms\_2))

data\_ms\_2\_col<-data\_ms\_2

colnames(data\_ms\_2\_col)<-b

##Just to simplify the name of the dataset for further manipulation

data\_f1<-data\_ms\_2\_col

##Convert values to numeric, because of the error messages I got when I tried to calculate means of columns

for(i in 1:nrow(data\_f1)){

for(j in 3:(ncol(data\_f1)-1)){

data\_f2[i,j]<-as.numeric(data\_f1[i,j])

}}

##Rename columns of data\_f2, back to the cleaned original variable names stored for the first time in b

nombres<-c("activityname","subject",colnames(data\_f1[,3:68]))

colnames(data\_f2)<-nombres



##That is the dataset requested, and I export it as a csv file

write.csv(data\_f2,"data\_f2.csv")

##Now the second part of the exercise…, Group

data\_f3<-group\_by(data\_f2,activityname,subject)

##Summarize all

means\_final<- summarise\_all(data\_f3,funs(mean))

View(means\_final)



##That was the second dataset requested, that I export as csv file

write.csv(means\_final,"means\_final.csv")

##Means grouped by activity, all subjects collapsed

means\_act\_final<-summarize\_all(group\_by(select(data\_f2,-subject),activityname),funs(mean))

View(means\_act\_final)



##Means by subject, all activities collpased

means\_subj\_final<-summarize\_all(group\_by(select(data\_f2,-activityname),subject),funs(mean))

View(means\_subj\_final)



**https://github.com/thecmos/Coursera\_tidying**

## ERROR can't bind data because arguments have the same name: data\_ms<-select(data\_for\_analysis,activity,activityname,subject,as.character(grep("mean|std",as.character(colnames(data\_for\_analysis)),value=TRUE)))

To replace or remove part of a string in all elements of a vector

* **Using sapply**
* data$abstract <- sapply(data$abstract,
* function(x){gsub(pattern = "no abstract available",
* replacement = " ", x)})
* **Using mapply**
* data$abstract <- mapply(gsub, pattern = "no abstract available", replacement = " ", data$abstract)
* **Using the stringr package**
* library(stringr)

data$abstract <- str\_replace(data$abstract, "no abstract available",

" ")

Ideas not to type all different tot\_bins…

The direct solution to your question would be to use get with paste

for(i in 1:10) {

Object = get(paste0("Season", i))

Object[1] = 0

assign(paste0("Season", i), Object)

}

**But don't do this**.

It's a horrible use of R. As suggested in the comments, store results in lists:

Seasons = lapply(rep(10,10), rnorm) #Generate data

Seasons

Then apply functions:

Seasons = lapply(Seasons, replace, list=1, values=0)

**Here's a solution based on @joran's comment**

> set.seed(1) # for reproducibility

> # the following does the same as your `for` loop and returned value is a list

> Season.list <- replicate(10, rnorm(10, 0, 1), simplify=FALSE)

> # giving some names

> names(Season.list) <- paste0("Season", 1:length(Season.list))

> # setting first element to 1

> Season.list <- lapply(Season.list, function(x) {x[1] <- 0; x})

> list2env(Season.list, envir = .GlobalEnv) # will give you each `Season` as you want :D

**Also, another way to do it,**

> set.seed(1)

> Season <- replicate(10, rnorm(10, 0, 1)) # the returned object is a matrix

> colnames(Season) <- paste0("Season", 1:ncol(Season))

> Season[1,] <- 0

If you want to have a vector for each Season then use attach (not a good idea)

> attach(as.data.frame(Season))

> Season1

[1] 0.0000000 0.1836433 -0.8356286 1.5952808 0.3295078 -0.8204684 0.4874291 0.7383247 0.5757814 -0.3053884

> Season2

[1] 0.00000000 0.38984324 -0.62124058 -2.21469989 1.12493092 -0.04493361 -0.01619026 0.94383621 0.82122120

[10] 0.59390132

Or, if you want to keep the last duplicated column, you can do

temp <- temp[, !duplicated(colnames(temp), fromLast = TRUE)]

To group CHI pellet types properly

pellets<-function()**{**

chi\_yp\_prev<-read.csv("TEST Chicago Daily YP.csv",skip=1)

#Only 2018 data

chi\_yp<-filter(chi\_yp\_prev,grepl("2018", chi\_yp\_prev$Yield))

il\_yp<-tbl\_df(chi\_yp)

##View(il\_yp)

##head(chi\_yp)

pel\_typ<-c("S-2","S2","S-3","S3","M-6","M6","M-7","M7","M-8","M8","M-5","ME","L6A","M5B","S3B","Str","CRA")

##pel\_typ<-c("S","M","L","C")

##pel\_cols<-c("X22","X27","X32","X37","X42","X47","X52","X57","X62")

pel\_cols<-c(22,27,32,37,42,47,52,57,62)

fila<-vector()

typ<-vector()

colu<-vector()

flr<-data.frame()

##colu<-NULL

##fila<-NULL

m<-0

n<-0

for(i in 1:length(pel\_typ))**{**

for(j in 1:length(pel\_cols))**{**

for(k in 1:nrow(chi\_yp))**{**

##chi\_yp\_selj<-filter(chi\_yp,pel\_cols[j]==pel\_cols[grep(pel\_typ[i],pel\_cols,ignore.case=TRUE)])

##Comprobar que hay coincidencia

if(length(grep(pel\_typ[i],as.character((chi\_yp[k,pel\_cols[j]])),ignore.case=TRUE))!=0) **{**

##if(length(grep(pel\_typ[i], pull(il\_yp[,pel\_cols[j]])[k]))!=0)**{**

##if(length(grep(pel\_typ[i],(chi\_yp[,pel\_cols[j]])[k]))!=0)**{**

##guardar el numero de la fila y columna en la que hay coincidencia

##en posiciones sucesivas de respectivos vectores

m<-m+1

colu[m]<-pel\_cols[j]

fila[m]<-k

typ[m]<-pel\_typ[i]

**}**

**}**

**}**

**}**

write.csv(colu,"colu.csv")

write.csv(fila,"fila.csv")

##build dataframe with every row in fila[m] and column colu[m] plus the next three

##y<-unique(x)

for(i in 1:m)**{**

#para cada fila seleccionada…

#sacar fecha

flr[i,1]<-as.Date(chi\_yp[fila[i],2], "%m/%d/%Y")

d<-substr(as.character(chi\_yp[fila[i],colu[i]]),1,nchar(as.character(chi\_yp[fila[i],colu[i]])))

flr[i,2]<-d

##flr[i,2]<-typ[i]

**##b<-substr(as.character(**chi\_yp[fila[i],colu[i]+1]**),1,nchar(as.character(**chi\_yp[fila[i],colu[i]+1]**)))**

**##c<-as.numeric(sub(",.\*$","",b))\*1000+as.numeric(sub('.\*\\,', '', b))**

##flr[i,3]<-c

##flr[i,3]<-chi\_yp[fila[i],colu[i]+1]

for(j in 1:3)**{**

#para cada fila seleccionada…

#sacar columna del ítem

#sacar las tres columnas siguientes (las libras)

#por que el as.numeric no funciona?? flr[i,j+1]<-as.numeric(chi\_yp[fila[i],colu[i]+j])

##**b<-substr(as.character(**chi\_yp[fila[i],colu[i]+j]**),1,nchar(as.character(**chi\_yp[fila[i],colu[i]+j]**)))**

**##b<-substr(as.character(**chi\_yp[fila[i],colu[i]+j]**),1,nchar(as.character(**chi\_yp[fila[i],colu[i]+j]**)))**

**##c<-as.numeric(sub(",.\*$","",b))\*1000+as.numeric(sub('.\*\\,', '', b))**

c<-parse\_number( chi\_yp[fila[i],colu[i]+j] )

flr[i,j+2]<-c

##flr[i,j+2]<- chi\_yp[fila[i],colu[i]+j]

**}**

**}**

colnames(flr)<-c("date","type","lbcut","lbcooked","lbobtained")

write.csv(flr,"flr.csv")

flr\_tes<-filter(flr,!grepl('tes',type,ignore.case=TRUE))

flr\_tesdat<-filter(flr\_tes,grepl('201',date))

write.csv(flr\_tesdat,"flr\_tesdat.csv")

##remove NA to clean the original table completely

flr\_tesdat1<-select(flr\_tesdat,-lbcut)

flr\_tdc<-flr\_tesdat1[complete.cases(flr\_tesdat1),]

##step to help build yield\_key file

for\_key<- summarize(group\_by(arrange(flr\_tdc,type),type),sumc=sum(lbcooked,na.rm=TRUE),sumo=sum(lbobtained,na.rm=TRUE),y=(sumo/sumc))

write.csv(for\_key,"for\_key.csv")

##Prepare for merging by selecting only the rows that we are interested in

flr\_key<-select(read.csv("yield\_key.csv"),type,fet)

##Merge, no need to write column names "…"

##The key is written manually on the yield\_key file (at least the first time)

flr\_mrg<-merge(flr\_tdc,flr\_key,all.x=TRUE)

##dim(flr\_mrg)

##[1] 1699 6

flr\_mrgg<-arrange(group\_by(flr\_mrg,fet,date),fet,date)

write.csv(flr\_mrgg,"flr\_mrgg.csv")

##flr\_mrgg\_y<-mutate(group\_by(select(flr\_mrgg,-type),fet),yield=lbobtained/lbcooked)

alfa<-select(flr\_mrgg,-type)

beta<-mutate(alfa,weekc=as.character(ymd(date)+(1-wday(ymd(date)))))

write.csv(beta,"beta.csv")

gamma<-read.csv("beta.csv")[,3:6]

##write.csv(gamma,"gamma.csv")

delta<-group\_by(gamma,fet,weekc)

##write.csv(delta,"delta.csv")

flr\_mrgg\_weekc<-summarize(delta, sumc=sum(lbcooked,na.rm=TRUE),sumo=sum(lbobtained,na.rm=TRUE),yield=(sumo/sumc))

write.csv(flr\_mrgg\_weekc,"flr\_mrgg\_weekc.csv")

##yield debe ser construido a partir de flr\_mrgg para que muestre las categorias refinadas

yield<-arrange(summarize(group\_by(arrange(flr\_mrgg,fet),fet),sumc=sum(lbcooked,na.rm=TRUE),sumo=sum(lbobtained,na.rm=TRUE),y=(sumo/sumc)),desc(sumo))

write.csv(yield,"yield.csv")

yield\_fil<-yield[1:10,]

write.csv(yield\_fil,"yield\_fil.csv")

flr\_weekc\_fil<-merge(flr\_mrgg\_weekc,yield\_fil,by="fet")

write.csv(flr\_weekc\_fil,"flr\_weekc\_fil.csv")

**}**

a<-substr(as.character(chi\_yp[7,23]),1,nchar(as.character(chi\_yp[7,23])))

> a

[1] "43,807"

> class(a)

[1] "character"

as.numeric(substr(a,1,1))

[1] 4

as.numeric(substr(a,1,1))\*3

[1] 12

> b<-sub('\\..\*', '', a)

> b

[1] "43,807"

> gsub(",.\*$","",a)

[1] "43"

**> b<-substr(as.character(chi\_yp[7,33]),1,nchar(as.character(chi\_yp[7,33])))**

**> b**

**[1] "6,267.00"**

**> c<-as.numeric(sub(",.\*$","",b))\*1000+as.numeric(sub('.\*\\,', '', b))**

**> c**

**[1] 6267**

b<-0

for(i 1:nchar(a)){

b<-b+10^(nchar(a)-i) \* ( as.numeric(substr(a,1,)))

chi\_yp\_pels<-filter(chi\_yp,pel\_typ%in%pel\_cols)

https://stats.stackexchange.com/questions/16796/reading-only-two-out-of-three-columns-with-read-csv

read.csv(file = "result1", sep = " ")[ ,c('col1', 'col2')]

subsetting and selecting all in one

ovw\_yn\_ps<-subset(ovw\_yn,subset=grepl("PS",ovw\_yn$loctype),select=!grepl("CR",colnames(ovw\_yn)))

ovw\_yn\_cr<-subset(ovw\_yn,subset=grepl("CR",ovw\_yn$loctype),select=!grepl("PS",colnames(ovw\_yn)))

yield\_factors\_R

dplyr

tidyr

readr

factors<-function(){

##reading tables and obtaining the tidy data set

OvW\_table<-read.csv("z) OvW Plant Wk.csv")

##View(OvW\_table)

y\_by\_wk<-read.csv("Copy Of 3 2) Y by week.csv")

##View(y\_by\_wk)

col\_ovw<-gsub("\\.","",colnames(OvW\_table))

col\_ovw<-gsub("^SumOfSumof","",col\_ovw)

colnames(OvW\_table)<-col\_ovw

col\_y<-gsub("\\.","",colnames(y\_by\_wk))

##View(col\_y)

colnames(y\_by\_wk)<-col\_y

colnames(y\_by\_wk)<-gsub("\_","",colnames(y\_by\_wk))

ovw\_y<-merge(OvW\_table,y\_by\_wk,by.x=c("WeekC","Location"),by.y=c("WeekC","Location"))

##write.csv(ovw\_y,"ovw\_y.csv")

ovw\_y1<-sapply(select(ovw\_y,-1,-2,-3,-7),parse\_number)

ovw\_y2<-as.data.frame(cbind("date"=(as.character(ovw\_y[,1])), "location"=as.character(ovw\_y[,2]), "loctype"=as.character(ovw\_y[,3]),"locnav"= as.character(ovw\_y[,7]),ovw\_y1))

write.csv(ovw\_y2,"ovw\_y2.csv")

## colnames(ovw\_y2)

##[1] "date" "location" "loctype" "locnav"

##[5] "Overwpounds" "Totalpounds" "OvW" "PSWkY"

##[9] "CRWkY" "TOTWkY" "PSFriedLbsTot" "PSLbstot"

##[13] "PSWSeasLbsTot" "CRFriedLbsTot" "CRLbsTot" "AllFGLbsTot"

##[17] "Cheeselbs" "PSWLb" "CRWLb"

##Add the new calculated variables

ovw\_yn<-mutate(ovw\_y2, PS\_FG\_OvW= parse\_number(PSLbstot)\*(1+parse\_number(OvW)/100), PS\_Oil\_to\_Y\_PS=parse\_number(PSLbstot)-parse\_number(PSFriedLbsTot), PS\_OvW\_Oil\_to\_Y= parse\_number(PS\_FG\_OvW)- parse\_number(PSFriedLbsTot)+ parse\_number(PSWLb), CR\_FG\_OvW= parse\_number(CRLbsTot)\*(1+parse\_number(OvW)/100), CR\_Oil\_to\_Y\_CR=parse\_number(CRLbsTot)-parse\_number(CRFriedLbsTot), CR\_OvW\_Oil\_to\_Y= parse\_number(CR\_FG\_OvW)-parse\_number(CRFriedLbsTot)+ parse\_number(CRWLb), OilConFctPS= parse\_number(PSWkY)-1+parse\_number(Overwpounds)/parse\_number(PSFriedLbsTot)+parse\_number(PSWLb)/parse\_number(PSFriedLbsTot), OvWFctPS=-parse\_number(Overwpounds)/parse\_number(PSFriedLbsTot), WFctPS=-parse\_number(PSWLb)/parse\_number(PSFriedLbsTot),OilConFctCR= parse\_number(CRWkY)-1+parse\_number(Overwpounds)/parse\_number(CRFriedLbsTot)+parse\_number(CRWLb)/parse\_number(CRFriedLbsTot), OvWFctCR=-parse\_number(Overwpounds)/parse\_number(CRFriedLbsTot), WFctCR=-parse\_number(CRWLb)/parse\_number(CRFriedLbsTot))

write.csv(ovw\_yn,"ovw\_yn.csv")

##split in two tables, PS and CR

ovw\_yn\_ps<-subset(ovw\_yn,subset=grepl("PS",ovw\_yn$loctype),select=!grepl("CR",colnames(ovw\_yn)))

ovw\_yn\_cr<-subset(ovw\_yn,subset=grepl("CR",ovw\_yn$loctype),select=!grepl("PS",colnames(ovw\_yn)))

##a<-subset(ovw\_yn\_ps,as.character(ovw\_yn\_ps$date)=="02-Sep-18")

## convert to date and sort

aa<-arrange(mutate(ovw\_yn\_ps,date1=dmy(date)),desc(date1))

bb<-arrange(mutate(ovw\_yn\_cr,date1=dmy(date)),desc(date1))

write.csv(aa,"aa.csv")

write.csv(bb,"bb.csv")

##plot charts

}

##ovw\_yn1<-ovw\_yn[,ovw\_yn$loctype %in% row.names(ovw\_yn)]

##d1[row.names(d1) %in% row.names(d2),]

<http://stanford.edu/~jgrimmer/RDataManagement.pdf>

25 paginitas

<https://www.stat.berkeley.edu/~spector/Rcourse.pdf>

<https://stackoverflow.com/questions/2628621/how-do-you-use-scoping-assignment-in-r>

<https://www.r-bloggers.com/data-caching/>

<file:///C:/Users/javier.lores/Downloads/v40i01.pdf>

<https://www2.uned.es/pea-metodos-estadisticos-aplicados/>

Graphs

<http://varianceexplained.org/r/teach_ggplot2_to_beginners>

<http://www.ling.upenn.edu/~joseff/rstudy/week4.html>

Sorting data frames plus creating them with col names

<https://www.r-bloggers.com/r-sorting-a-data-frame-by-the-contents-of-a-column/>

<https://stackoverflow.com/questions/43813249/r-round-down-dates-to-first-day-of-the-week>

<http://shop.oreilly.com/product/0636920034407.do>

<http://garrettgman.github.io/tidying/>

Tidy Data <Wickham, H. (2014). Tidy Data. Journal of Statistical Software, 59(10), 1 - 23. doi:[http://dx.doi.org/10.18637/jss.v059.i10>](http://dx.doi.org/10.18637/jss.v059.i10%3E)

<https://stackoverflow.com/questions/1299871/how-to-join-merge-data-frames-inner-outer-left-right>

<https://uc-r.github.io/dplyr>

<https://chartio.com/resources/tutorials/how-to-sort-a-data-frame-by-multiple-columns-in-r/>

<http://vita.had.co.nz/papers/tidy-data.pdf>

Using Census ACS 2012-2016 data, we took the percent of adults in each state that have at least a masters and ranked them from 1-50 with 1 being the most highly educated.

<http://research.collegeboard.org/programs/sat/data/cb-seniors-2013>

On replacing/extracting charactersbefore or after a given one…

<https://stevencarlislewalker.wordpress.com/2013/02/13/remove-or-replace-everything-before-or-after-a-specified-character-in-r-strings/>

<https://stackoverflow.com/questions/15895050/using-gsub-to-extract-character-string-before-white-space-in-r>

<https://www.r-bloggers.com/r-tip-use-istrue/>

<https://blog.exploratory.io/filter-data-with-dplyr-76cf5f1a258e>

dplyr removing na:

<http://kbroman.org/datacarpentry_R_2016-06-01/03-dplyr.html>

Filtering row based on content (grepl):

<https://stackoverflow.com/questions/22850026/filtering-row-which-contains-a-certain-string-using-dplyr>

2011 Journal of Statistical Software paper titled 'Dates and Times Made Easy with lubridate'.

<https://github.com/swirldev/swirl_courses>

<http://data.worldbank.org/data-catalog/GDP-ranking-table>

<http://data.worldbank.org/data-catalog/GDP-ranking-table>

<http://data.worldbank.org/data-catalog/ed-stats>

<http://www.quantmod.com/>  ##for stock historic data

Wearables (x2):

<http://www.insideactivitytracking.com/data-science-activity-tracking-and-the-battle-for-the-worlds-top-sports-brand/>

<http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones>

How to use regular expressions in R:

<https://www.dummies.com/programming/r/how-to-use-regular-expressions-in-r/>

Regular expressions in R:

<http://stat545.com/block022_regular-expression.html>

<https://www.r-bloggers.com/the-complete-catalog-of-argument-variations-of-select-in-dplyr/>

Summarize multiple columns with dplyr:

<https://stackoverflow.com/questions/21644848/summarizing-multiple-columns-with-dplyr>

[Artem Klevtsov](https://stackoverflow.com/users/1863950/artem-klevtsov)

Principles\_of\_Analytic\_Graphs. (Slides for this and other Data Science

| courses may be found at github

| https://github.com/DataScienceSpecialization/courses/. If you care to use

| them, they must be downloaded as a zip file and viewed locally. This lesson

| corresponds to 04\_ExploratoryAnalysis/Principles\_of\_Analytic\_Graphics.)

In fact, this is an example of Simpson's paradox, or the

| Yule–Simpson effect. Wikipedia

| (http://en.wikipedia.org/wiki/Simpson%27s\_paradox) tells us that this "is a

| paradox in probability and statistics, in which a trend that appears in

| different groups of data disappears when these groups are combined."

Congrats! You've concluded exploring this lesson on principles of graphing.

| We hope you found it principally principled.

| Although ggplot2 bears a superficial similarity to lattice, it's generally

| easier and more intuitive to use. Its default mode makes many choices for you

| but you can still customize a lot. The package is based on a "grammar of

| graphics" (hence the gg in the name), so you can control the aesthetics of

| your plots. For instance, you can plot **conditioning graphs and panel plots** as

| we did in the lattice example.

| Congrats! You've concluded this plotting lesson. We hope you didn't find it

| plodding.