

Day 3 : Data Visualization Fundamentals in R



Ambu Vijayan

Bioinformatician

BioLit, Thiruvananthapuram

Data Manipulation in R

Consider the code for calculating mean of birthweight earlier.

```
mean(birthdata$birthweight)
```

Now Observe this command,

```
mean(birthdata$birthweight[birthdata$geriatric.pregnancy])
```

What do you think this code does?

```
mean(birthdata$birthweight[birthdata$geriatric.pregnancy])
```

The R code calculates the mean (average) birthweight of babies in birthdata dataset based on a condition specified in the `geriatric.pregnancy` column.

Let me break it down step by step:

```
birthdata$geriatric.pregnancy
```

This part extracts the values in the `geriatric.pregnancy` column of the `birthdata` dataset.

This column contains binary values (TRUE or FALSE) indicating whether a pregnancy is classified as a geriatric pregnancy.

```
birthdata$birthweight[birthdata$geriatric.pregnancy]
```

This part subsets the `birthdata$birthweight` column based on the condition in `birthdata$geriatric.pregnancy`.

In other words, it extracts the birthweights of babies where `geriatric.pregnancy` is TRUE.

```
mean(...)
```

It calculates the mean (average) of the birthweights obtained in the previous step.

This gives you the average birthweight of babies born to mothers with geriatric pregnancies.

Now run the code :

```
mean(birthdata$birthweight[birthdata$geriatric.pregnancy])
```

How to handle missing data

Run this code :

```
mean(birthdata$paternal.age)
```

You will get an output *NA*

NA stands for *Not Available*

Go through your dataset to check if you have NA.

Installing an R package : fixr

For finding which all columns have how many of the NAs in our database, Lets use a package called **fixr**

CRAN is the repository where we download ll packages in R.

The link for fixr is : <https://cran.r-project.org/web/packages/fixr/index.html>

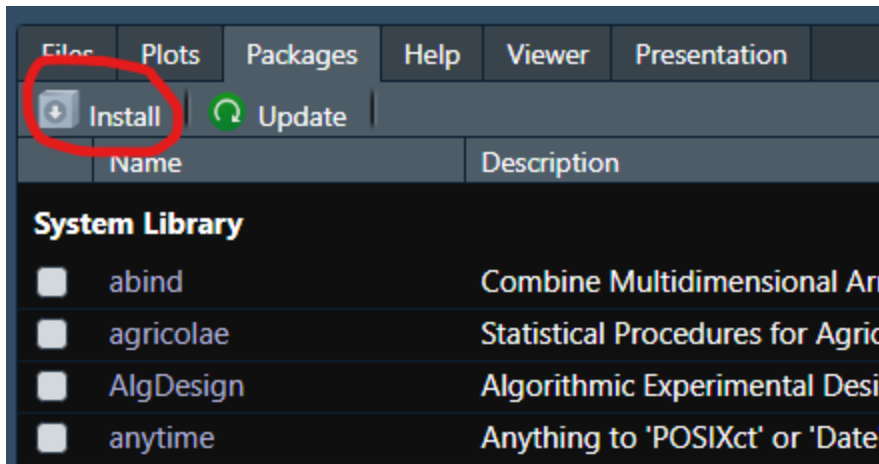
Reference manual of fixr : <https://cran.r-project.org/web/packages/fixr/fixr.pdf>

We will use a function called `check_missing_values` from this package.

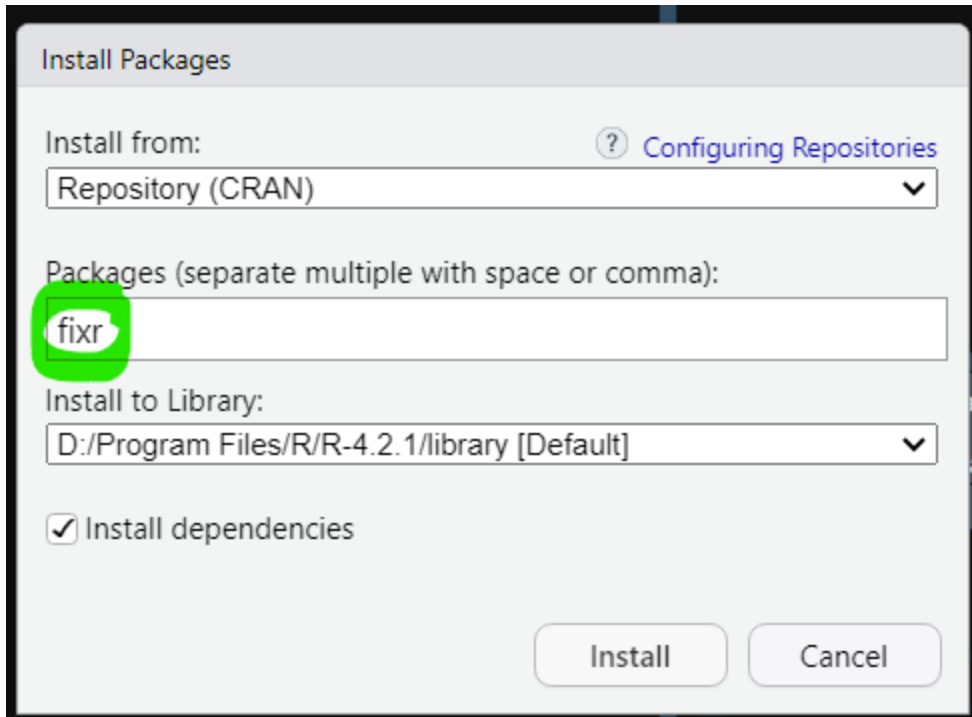
There are two ways: By code or By IDE

Using IDE

Use the install option from packages tab



Search for fixr package



Install Packages

Install from: [? Configuring Repositories](#)

Repository (CRAN) ▼

Packages (separate multiple with space or comma):

fixr

Install to Library:

D:/Program Files/R/R-4.2.1/library [Default] ▼

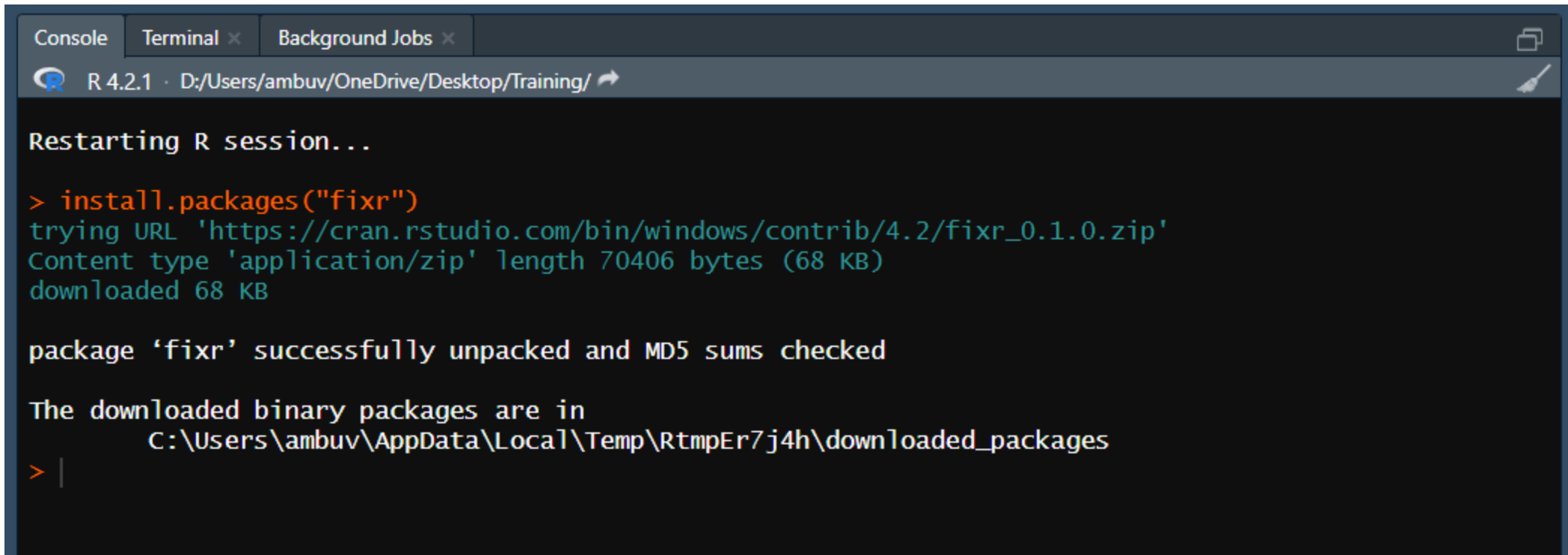
☒ Install dependencies

Install Cancel

Click Install.

Installation by code

```
install.packages("fixr")
```



The screenshot shows the RStudio interface with the Console tab active. The title bar indicates 'R 4.2.1' and the current working directory is 'D:/Users/ambuv/OneDrive/Desktop/Training/'. The console output shows the process of installing the 'fixr' package from CRAN, including downloading the zip file and unpacking it.

```
Restarting R session...

> install.packages("fixr")
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/fixr_0.1.0.zip'
Content type 'application/zip' length 70406 bytes (68 KB)
downloaded 68 KB

package 'fixr' successfully unpacked and MD5 sums checked

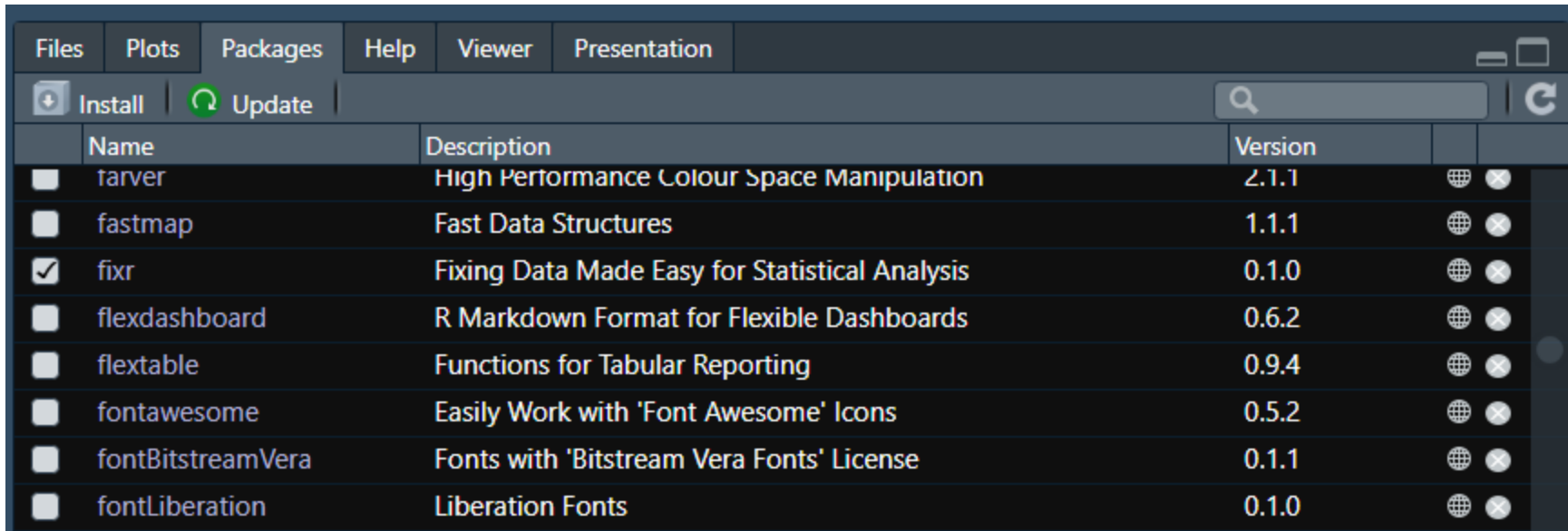
The downloaded binary packages are in
  C:\Users\ambuv\AppData\Local\Temp\RtmpEr7j4h\downloaded_packages
> |
```

Activating a package

There are two ways: By code or By IDE

Using IDE

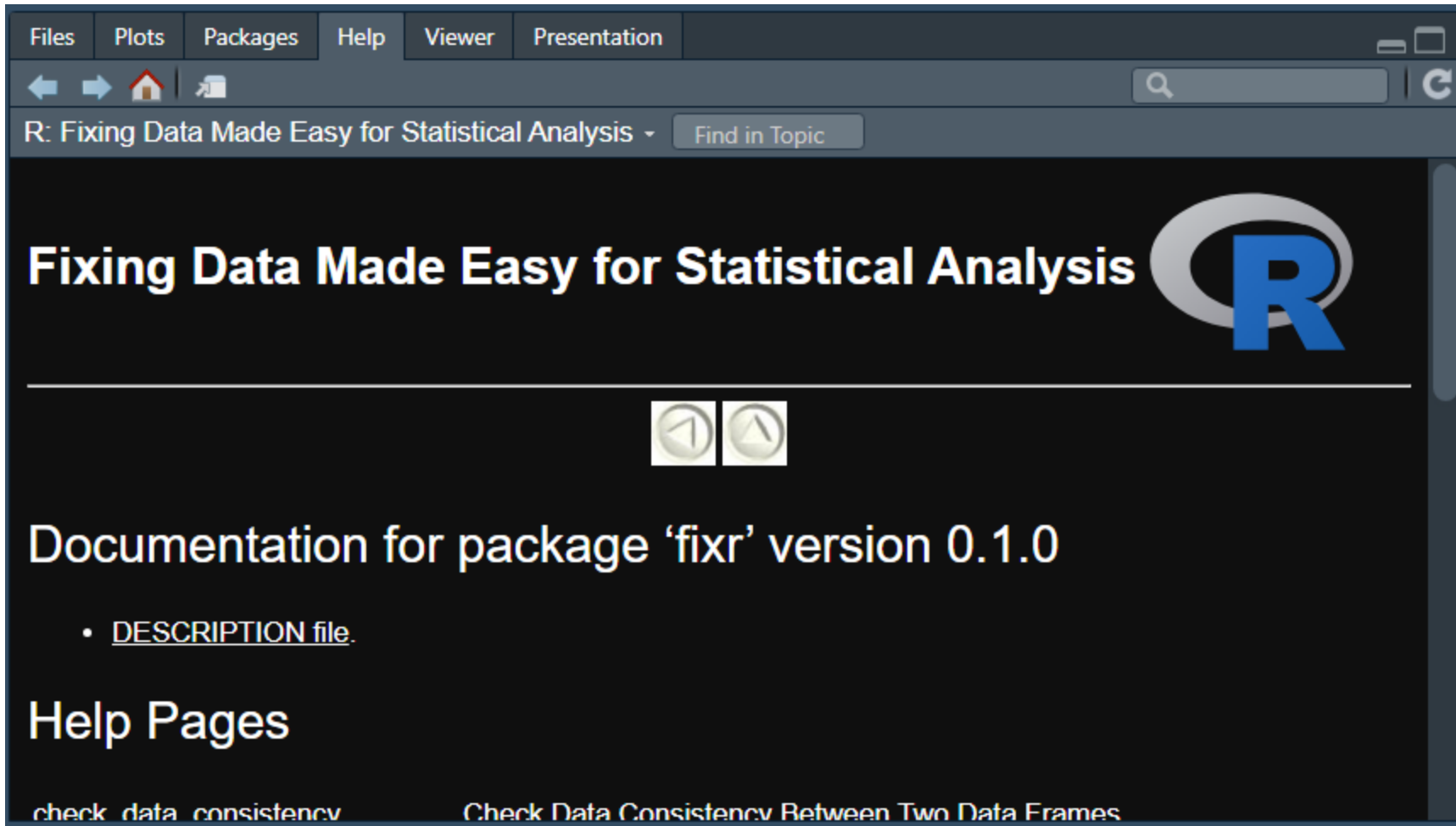
Check the tick box near fixr in packages tab



Using Code

```
library(fixr)
```

Use `help(package = "fixr")` to learn more about this package



Finding NAs in a dataset

Lets use `check_missing_values` function in `fixr` package.

```
check_missing_values(birthdata)
```

```
> check_missing_values(birthdata)
Missing values found in the following columns:
paternal.age: 4
paternal.education: 5
paternal.cigarettes: 5
paternal.height: 5
> |
```

Outputs the name of columns and number of NAs in them.

Now we know that `paternal.age` has 4 NAs.

How to handle NAs

Lets go back to :

```
mean(birthdata$paternal.age)
```

We need to specify mean function not to use the NAs or to omit the rows with NAs.

For that we use : `na.rm = TRUE` along with our mean command.

Below command will omit rows with NAs

```
mean(birthdata$paternal.age, na.rm = TRUE)
```

Splitting a data into meaning full data

Consider column `birth.date`

```
birthdata$birth.date
```

```
> birthdata$birth.date
[1] "1/25/1967" "2/6/1967"  "2/14/1967" "3/9/1967"  "3/13/1967" "3/23/1967" "4/23/1967"
[8] "5/5/1967"  "6/4/1967"  "6/7/1967"  "6/14/1967" "6/20/1967" "6/25/1967" "7/12/1967"
[15] "7/13/1967" "9/7/1967"  "10/7/1967" "10/19/1967" "11/1/1967"  "12/7/1967"  "12/14/1967"
[22] "1/8/1968"  "1/10/1968" "1/21/1968" "2/2/1968"   "2/16/1968"  "2/22/1968"  "4/2/1968"
[29] "4/24/1968" "4/25/1968" "6/19/1968" "7/18/1968"  "7/24/1968"  "8/12/1968"  "8/17/1968"
[36] "9/7/1968"  "9/16/1968" "9/27/1968" "10/9/1968"  "10/25/1968" "12/11/1968" "12/19/1968"
```

We cant use this data to calculate any meaning full result.

Lets split this data into Day, Month and Year

strsplit() function

Lets use this function to split the date, We specify dates are separated using "/" symbol.

```
strsplit(birthdata$birth.date, split = "/")
```

Lets assign this to an object called dates.

```
dates_list <- strsplit(birthdata$birth.date, split = "/")
```

Check the class of dates : `class(dates_list)`

The output of strsplit() is a list containing 42 vectors of length 3, while the columns of birthweight are vectors of length 42.

The apply() family of functions

apply takes a matrix, applies a function either by row or by column, and returns a vector.

IMPORTANT there are 2 axis for a matrix

Axis 1 is row

Axis 2 is column

So command for apply is :

```
apply(dataset[rows,columns], axis1/axis2, operation)
```

Lets add total of "maternal.cigarettes" and "paternal.cigarettes" in all rows.

```
apply(birthdata[,c("maternal.cigarettes", "paternal.cigarettes")], 1, sum)
```

Which gives total of maternal and paternal cigarettes.

```
apply(birthdata[,c("maternal.cigarettes", "paternal.cigarettes")], 2, sum)
```

Which gives total of maternal cigarettes column and paternal cigarettes column individually.

maternal.cigarettes : 396

paternal.cigarettes : NA (as data has missing values)

How to omit NAs and get a meaning full result here?

```
apply(birthdata[,c("maternal.cigarettes", "paternal.cigarettes")], 2, sum,  
na.rm=T)
```

mapply

mapply takes a function and applies it to the elements of one or more vectors.

```
mapply(sum, birthdata$maternal.cigarettes, birthdata$paternal.cigarettes)
```

Result is same as using axis 1 in apply function.

tapply

tapply takes two vectors, applies a function to the subsets of the first based on the categories in the second vector, and returns a table.

```
tapply(birthdata$birthweight, birthdata$smoker, mean)
```

No : 3.509500

Yes : 3.134091

lapply

lapply takes a list, applies a function to each element, and returns a list.

For Month

```
lapply(strsplit(birthdata$birth.date, split = "/"), '[', 1)
```

For Date

```
lapply(strsplit(birthdata$birth.date, split = "/"), '[', 2)
```

For Year

```
lapply(strsplit(birthdata$birth.date, split = "/"), '[', 3)
```

Difference of [and [[

[(Single Square Bracket):

[is used for subsetting objects, such as vectors, lists, and data frames.

[[(Double Square Bracket):

[[is used for extracting a single element from a list or data frame. It is specifically designed for accessing elements inside lists and data frame

```
my_list <- list(a = 1, b = 2, c = 3)
```

```
my_list[2]
```

```
my_list[[2]]
```

do.call

The `do.call()` function in R is used to apply a function to a list of arguments. It takes two main arguments:

1. `what`: This argument specifies the function to be called.
2. `args`: This argument is a list of arguments to pass to the function.

```
do.call(what, args)
```

cbind

cbind (Column Bind):

The `cbind()` function is used to combine two or more objects (vectors, matrices, or data frames) by column. It effectively stacks the objects side by side, creating a new data structure.

```
coldata <- cbind(birthdata$location, birthdata$length)
```

```
View(coldata)
```


rbind

rbind (Row Bind):

The `rbind()` function is used to combine two or more objects (vectors, matrices, or data frames) by row. It effectively stacks the objects on top of each other, creating a new data structure.

```
rowdata <- rbind(birthdata$location, birthdata$length)
```

```
View(rowdata)
```

Combining what we learned for splitting dates

Splitting dates into a list

```
dates_list <- strsplit(birthdata$birth.date, split = "/")
```

```
class(dates_list)
```

Combining list into a matrix

```
dates_matrix <- do.call(rbind, dates_list)
```

```
class(dates_matrix)
```

Converting matrix into data frame

```
dates_df <- data.frame(dates_matrix)
```

```
class(dates_df)
```

Naming the column as month, day and year

```
names(dates_df) = c("month", "day", "year")
```

Add the new columns to the birthdata data frame

```
birthdata <- cbind(birthdata, date_df)
```

```
View(birthdata)
```

Using apply functions for splitting dates

Split the dates into month, day, and year using sapply

```
dates <- sapply(strsplit(birthdata$birth.date, split = "/"), as.integer)
```

Transpose the dates matrix

```
dates <- t(dates)
```

```
View(dates)
```

Create a data frame from the transposed matrix and

```
date_df <- data.frame(dates)
```

```
View(date_df)
```

Give it column names

```
colnames(date_df) <- c("month", "day", "year")
```

Add the new columns to the birthdata data frame

```
birthdata <- cbind(birthdata, date_df)
```

```
View(birthdata)
```

Exercise 3: summarizing the data

Answer the following questions to answer, or come up with one of your own. Work together.

Once you've answered a question in one way, can you come up with alternate code that generates the same answer?

1. Are preterm babies more likely to have low birth weight?
2. What is the ratio of maternal cigarettes to paternal cigarettes for births at each of the hospitals?
3. Do taller mothers have taller partners? Do they have longer babies?