workshop03 (1)

February 3, 2019

In [1]:
```
library(plotly)
library(tidyverse)
library(GGally)
library(gcookbook)

# Plot size deppening on your screen resolution to 5 x 3
options(repr.plot.width=5, repr.plot.height=3)
```

```
Loading required package: ggplot2

Attaching package: plotly

The following object is masked from package:ggplot2:

    last_plot

The following object is masked from package:stats:

    filter

The following object is masked from package:graphics:

    layout

 Attaching packages  tidyverse 1.2.1
 tibble  1.4.2      purrr   0.2.5
 tidyr   0.8.1      dplyr   0.7.5
 readr   1.1.1      stringr 1.3.1
 tibble  1.4.2      forcats 0.3.0
 Conflicts  tidyverse_conflicts()
 dplyr::filter() masks plotly::filter(), stats::filter()
 dplyr::lag()    masks stats::lag()

Attaching package: GGally

The following object is masked from package:dplyr:
```

```
    nasa


Attaching package: gcookbook

The following object is masked from package:plotly:

    wind
```

1 Welcome to Workshop 3

1.0.1 Exercise 1

Aims: * Learn to create scatterplot matrices, incorporating correlations and distribution density plots * Using graphs for initial analysis of a dataset

One handy function we have not mentioned before is `str` (standing for "structure") which when applied to a data frame gives you necessary information about its variables. For example, consider the data file countries in the package gcookbook.


```
In [2]: str(countries)
```

```
'data.frame':       11016 obs. of  7 variables:
 $ Name         : Factor w/ 216 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Code         : Factor w/ 216 levels "ABW","AFG","AGO",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ Year         : int  1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 ...
 $ GDP          : num  55.6 55.7 54.4 73.2 76.4 ...
 $ laborrate    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ healthexp    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ infmortality: num  NA NA NA NA NA NA NA NA NA NA ...
```

The Name and Code are listed as factors while Year is an integer variable, and the rest are numeric. It is also helpful to see a few values for each variable. However, there are many missing values. At first, you might think something had gone wrong. But considering that we are dealing with the data from many countries since 1960, there may be a lot of measures that are unavailable. Let's tell R to omit those cases that include at least one NA.

```
In [3]: str(na.omit(countries))
```

```
'data.frame':       2548 obs. of  7 variables:
 $ Name         : Factor w/ 216 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 2 2 ...
 $ Code         : Factor w/ 216 levels "ABW","AFG","AGO",..: 2 2 2 2 2 2 2 2 4 4 ...
 $ Year         : int  2001 2002 2003 2004 2005 2006 2007 2008 1995 1996 ...
 $ GDP          : num  92.2 158 168.7 196.2 227.9 ...
 $ laborrate    : num  58.9 59 59 59.1 59.3 59.4 59.6 59.7 61.8 61.9 ...
 $ healthexp    : num  11.1 21.8 25.5 29.6 32.5 ...
 $ infmortality: num  104 103 104 104 104 ...
```

```
 - attr(*, "na.action")=Class 'omit'  Named int [1:8468] 1 2 3 4 5 6 7 8 9 10 ...
  .. ..- attr(*, "names")= chr [1:8468] "1" "2" "3" "4" ...
```

We have less than one-quarter of the original observations. Of course, we still have four numeric variables. We will graph these in a scatterplot matrix.

Scatterplot matrices (aka splom) allow us to visualise the pairwise relationships between a collection of variables. Therefore they are an important tool for building up a picture of what is happening in a dataset with many variables

From the Help File:

Health and economic data about countries around the world from 1960-2010

Description:

Health and economic data about countries around the world from 1960-2010, from the World Bank.

Variables

- Name: Name of country
- Code: Short country code
- Year
- GDP: Per capita Gross Domestic Product, in adjusted 2011 U.S. Dollars
- laborrate: Labor rate.
- healthexp: Health expenditures in U.S. Dollars.
- infmortality: Infant mortality per 1000 live births.

Source

World Bank: http://data.worldbank.org/

We're going to restrict attention to just one year so that we can do without the variables Code and Year. As you see in the commands below, we choose the value of a particular variable (Year) and decide which variables we want, using select.

```
In [18]: countries %>%
             filter(Year==2009) %>%
             select(Name, GDP, laborrate, healthexp, infmortality) %>%
         head()
```

Name	GDP	laborrate	healthexp	infmortality
Afghanistan	NA	59.8	50.88597	103.2
Albania	3772.605	59.5	264.60406	17.2
Algeria	4022.199	58.5	267.94653	32.0
American Samoa	NA	NA	NA	NA
Andorra	NA	NA	3089.63589	3.1
Angola	4068.576	81.3	203.80787	99.9

We notice in print out that we still have partially missing data, let's add a filter by piping everything through `na.omit()`:

```
In [19]: # your code here
         na.omit(countries)%>%
             filter(Year==2009)%>%
             select(Name, GDP, laborrate, healthexp, infmortality)%>%
         head()
```

Name	GDP	laborrate	healthexp	infmortality
Albania	3772.605	59.5	264.6041	17.2
Algeria	4022.199	58.5	267.9465	32.0
Angola	4068.576	81.3	203.8079	99.9
Argentina	7665.073	65.0	730.1730	12.7
Armenia	2768.613	66.3	128.8576	18.4
Australia	42130.820	65.2	3867.4285	4.2

We could also have multiple filter criteria, let's add a filter for GDP being less than 1000. For this add in the filter command `& GDP < 1000`

```
In [20]: # your code here
         na.omit(countries)%>%
             filter(Year==2009&GDP<1000)%>%
             select(Name, GDP, laborrate, healthexp, infmortality)%>%
         head()
```


Name	GDP	laborrate	healthexp	infmortality
Bangladesh	607.7649	70.7	18.43125	40.0
Benin	771.7088	72.7	31.92885	74.7
Burkina Faso	509.2978	84.4	38.07546	93.3
Burundi	162.8704	89.3	19.78891	88.5
Cambodia	748.1512	79.3	42.10635	45.6
Central African Republic	458.5672	79.0	19.33792	106.8

This simple filter and the first few rows give us a hint. The values of `infmortality` have increased. Let's investigate this further, let's look at countries with low GDP per capita `or` an infant mortality per 1000 live births above 30. Hint: `|`

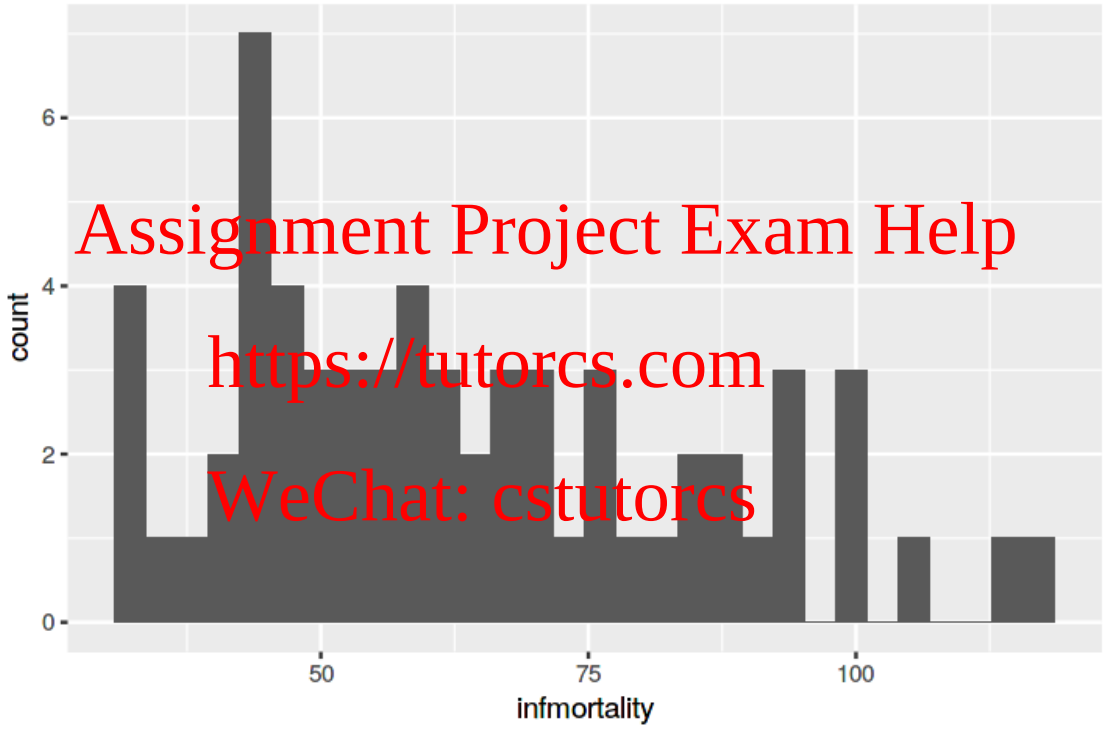
```
In [21]: # your code here
         na.omit(countries) %>%
             filter(Year==2009 & (GDP<1000|infmortality>30))%>%
             select(Name, GDP, laborrate, healthexp, infmortality) %>%
         head()
```

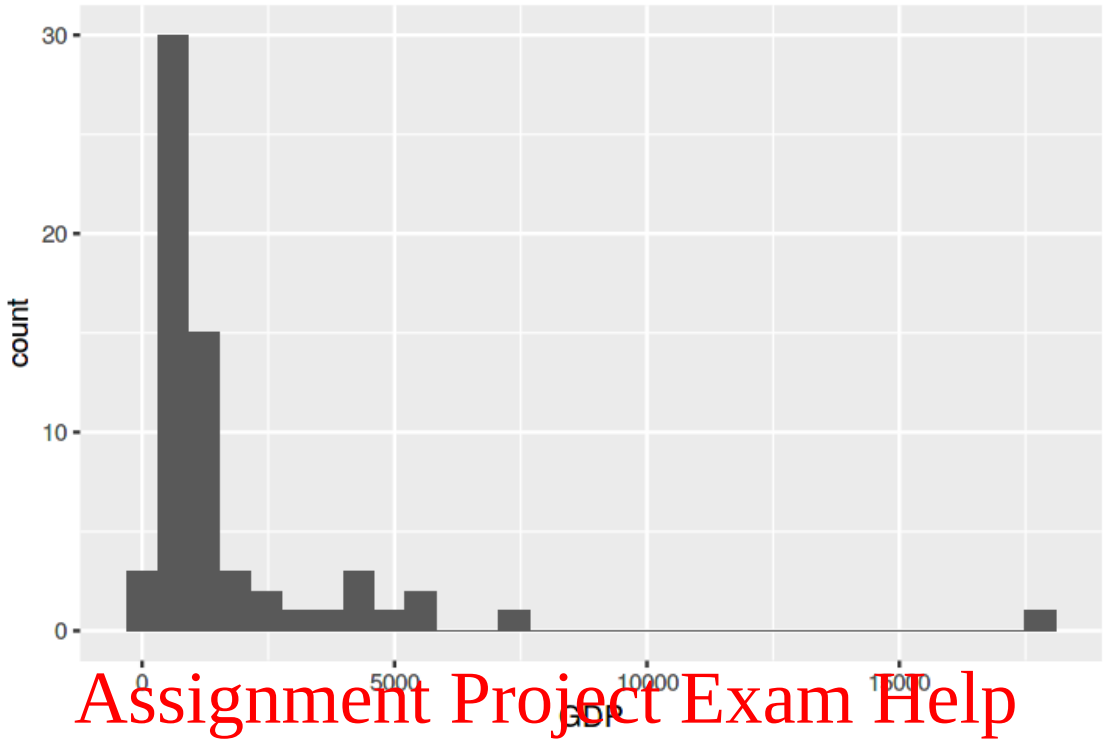
Name	GDP	laborrate	healthexp	infmortality
Algeria	4022.1989	58.5	267.94653	32.0
Angola	4068.5757	81.3	203.80787	99.9
Azerbaijan	4808.1688	63.0	284.72528	41.1
Bangladesh	607.7649	70.7	18.43125	40.0
Benin	771.7088	72.7	31.92885	74.7
Bhutan	1772.2838	62.7	97.98912	45.6

Lets now visualise the variables of interest, `infmortality` and `GDP` in separate plots. Use the same filter as before and create a histogram. The ggplot command for it is `+ geom_histogram(bins=30)` Note: It is a one-dimensional plot, only one `aes` variable is needed.

In [22]:
```
# your code here
na.omit(countries) %>%
    filter(Year==2009 & (GDP<1000|infmortality>30)) %>%
ggplot(data=.,
       aes(x=infmortality))+
geom_histogram(bins=30)

na.omit(countries) %>%
    filter(Year==2009 & (GDP<1000|infmortality>30)) %>%
ggplot(data=.,
       aes(x=GDP))+
geom_histogram(bins=30)
```





If we take a moment to compare the graphs, the choice of bins might be fitting for one variable but too crude for the other. Also, we are only looking at one dimension at a time. To look at pairwise correlation, there is the command `ggcsatmat`, from the package `GGally`, which creates a matrix of plots. First, subset the data as before with the same filters and selecting all numerical columns. Store the results in `subset.df`:


```
In [23]: # your code here
         subset.df<-na.omit(countries) %>%
             filter(Year==2009 & (GDP<1000|infmortality>30)) %>%
             select(Name,GDP, laborrate, healthexp, infmortality)
         head(subset.df)
```

Name	GDP	laborrate	healthexp	infmortality
Algeria	4022.1989	58.5	267.94653	32.0
Angola	4068.5757	81.3	203.80787	99.9
Azerbaijan	4808.1688	63.0	284.72528	41.1
Bangladesh	607.7649	70.7	18.43125	40.0
Benin	771.7088	72.7	31.92885	74.7
Bhutan	1772.2838	62.7	97.98912	45.6

When everything was correctly completed, the next line plots the matrix:

```
In [24]: # we temporarly increase the image size for all plots
         options(repr.plot.width=6, repr.plot.height=4)

         ggscatmat(data=subset.df,
```

```
                columns = 2:ncol(subsetName.df))

        # and reset it again
        options(repr.plot.width=5, repr.plot.height=3)
```

```
    Error in ncol(subsetName.df): object 'subsetName.df' not found
Traceback:


    1. ggscatmat(data = subset.df, columns = 2:ncol(subsetName.df))

    2. data[columns]

    3. `[.data.frame`(data, columns)

    4. ncol(subsetName.df)
```


For only looking at the correlation, there is a package called ggcorrplot. To install it into your local directory, run `install.packages('ggcorrplot',lib='.', verbose=TRUE)`. (Note: as this is a shared environment, an installation in the global environment would cause an error message)

In [29]:
```
# your code here
install.packages('ggcorrplot',lib='.', verbose=TRUE)
```



```
system (cmd0): /usr/lib/R/bin/R CMD INSTALL
foundpkgs: ggcorrplot, /tmp/RtmprE6byt/downloaded_packages/ggcorrplot_0.1.2.tar.gz
files: /tmp/RtmprE6byt/downloaded_packages/ggcorrplot_0.1.2.tar.gz
1): succeeded '/usr/lib/R/bin/R CMD INSTALL -l '/srv/home/xsun0006' /tmp/RtmprE6byt/downloaded_p
```

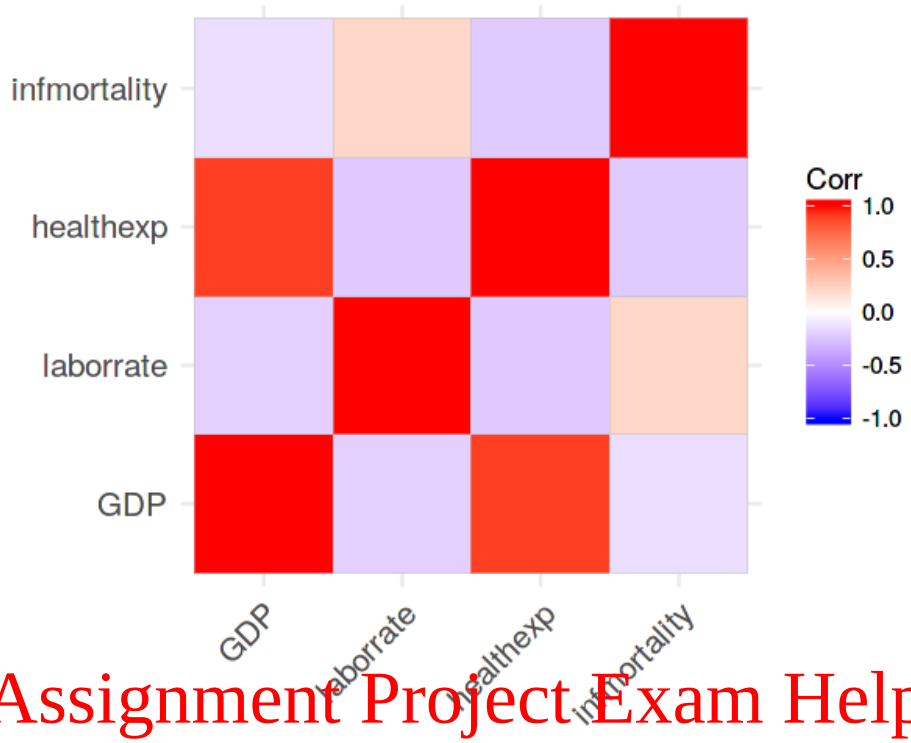
Load the package with `library(ggcorrplot,lib.loc='.')`

In [30]:
```
# your code here
library(ggcorrplot,lib.loc='.')
```

For plotting the correlation matrix use the function `cor` on the subset and pass the results to the plotting function `ggcorrplot`

In [31]:
```
# your code here


corsubset<-cor(subset.df[c(-1)])
ggcorrplot(corsubset)
```

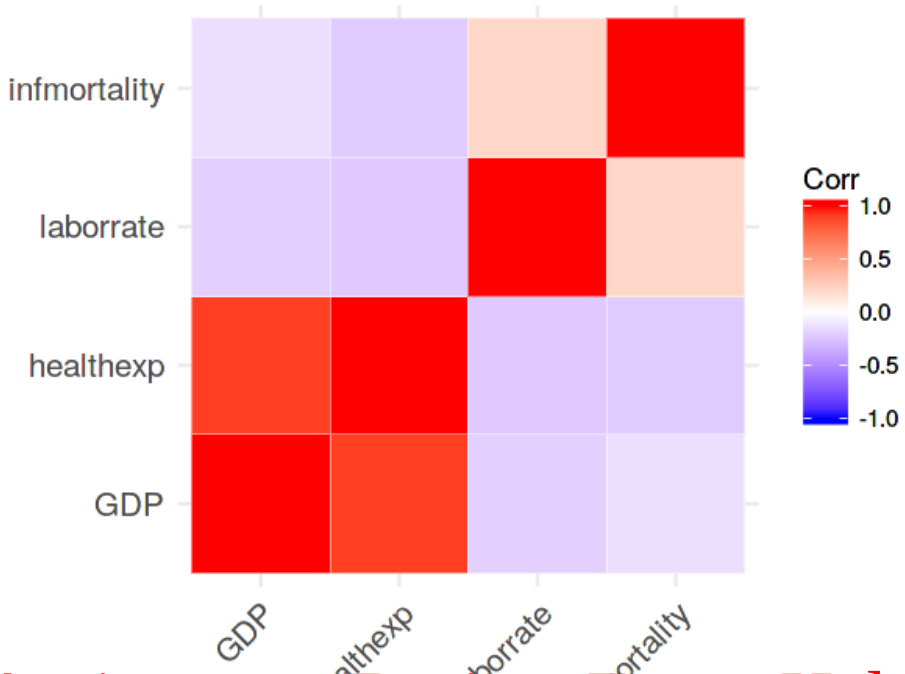


For the ease of use, it is often a good idea to group variables with similar correlation next to each other. You can achieve this by the option `hc.order = TRUE`. To highlight the borders better, specify `outline.col = "white"`.

In [32]:
```
# your code here

```

```
# your code here
corsubset<-cor(subset.df[c(-1)])
ggcorrplot(corsubset,
           hc.order = TRUE,
           outline.col = "white")
```



1.0.2 Exercise 2: Seperating by country group

Currently, we have analysed the correlation across the whole data set. The overall correlation though might be different across regions. To get a more balanced picture, as the overall effect may be driven by a few outlier countries, calculate the correlations across region group. Load the additional csv `CountryMatching.csv` into the variable `countryMatches`:

```
In [37]: # your code here
         countryMatches<- read.table(file = "CountryMatching.csv",
                                     header = TRUE,
                                     sep = ",")
         countryMatches%>%head()
```

Name	RegionNumber	RegionName
Algeria	1	Africa
Angola	1	Africa
Benin	1	Africa
Botswana	1	Africa
Burkina Faso	1	Africa
Burundi	1	Africa

Combining two data-frames is called a join operation. To do this, add the line `inner_join(countryMatches, by='Name') %>%` to your select, filter by year 2009 and omit all rows with missing data. An inner join combines two data frames based on the common column name specified by `by=`. To reduce the columns use `select(RegionName, GDP, laborrate, healthexp, infmortality)`. Run the command line by line to explore the results.

In [38]:
```
# your code here
subsetName.df<-countries%>%
      filter(Year==2009)%>%
      na.omit()%>%
      select(GDP, laborrate, healthexp, infmortality,Name)%>%
      inner_join(countryMatches,by='Name')%>%
      filter(RegionName=='Asia'|RegionName=='Americas')%>%
      select(RegionName, GDP, laborrate, healthexp, infmortality)
head(subsetName.df)
```

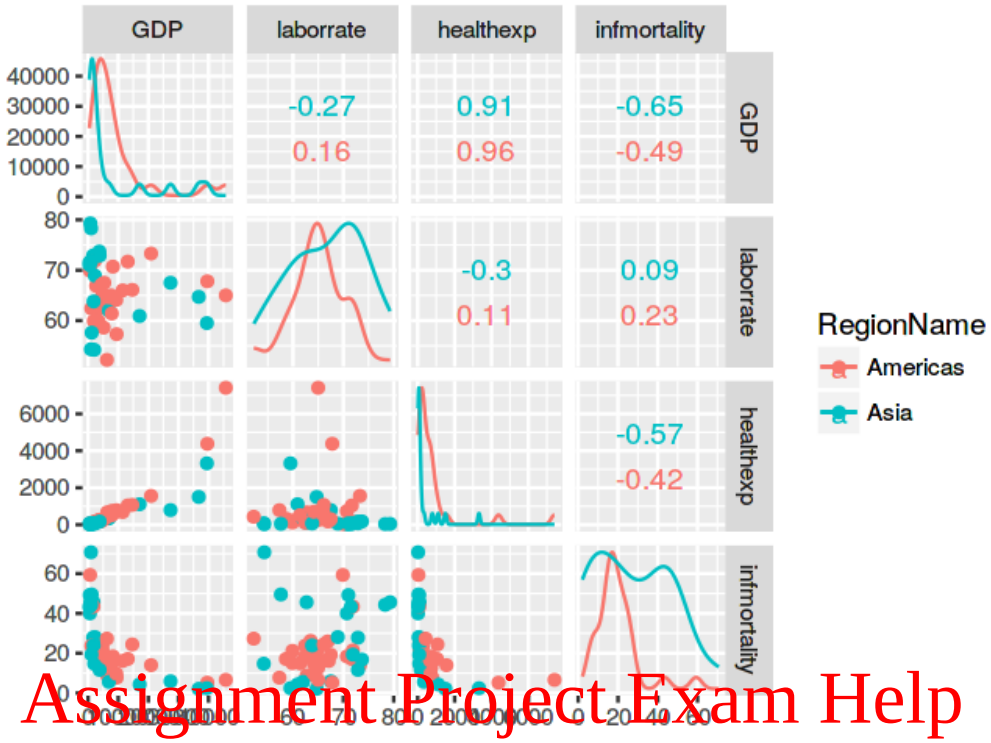
```
Warning message:
Column `Name` joining factors with different levels, coercing to character vector
```

RegionName	GDP	laborrate	healthexp	infmortality
Americas	7665.0734	65.0	730.17301	12.7
Americas	20916.2842	73.3	1557.65512	14.0
Asia	607.7649	70.7	18.43125	40.0
Americas	13181.3416	71.7	1041.44095	17.1
Americas	4056.1224	64.1	217.15214	14.9
Asia	1772.2828	62.7	97.98912	45.6

For creating the scatter plot matrix by region, add the options `color = "RegionName"`, as well as select all numerical columns in the data frame. e.g.: `columns = 2:ncol(subsetName.df)`

In [39]:
```
# your code here
options(repr.plot.width=6, repr.plot.height=4)

ggscatmat(data=subsetName.df,
          color = "RegionName",
          columns = 2:ncol(subsetName.df))
```



What do you see now? Discuss it with your table

1.0.3 Exercise 3: Subsetting data-sets

Use the mtcars dataset again for this excercise. Select subset and filter the rows, viszulize it with ggscatmat. What do you learn ?

```
In [ ]: # your code here
        fail() # No Answer - remove if you provide an answer
```