Most of the R u need 4 this course

Ralf Martin

08/10/2021

In this document I am going through a couple of commands and use cases using R. This follows more or less Exercise 2.2.

In terms of R, this covers most of what you need for the graded part of this course. It also largely covers what would be needed for the group coursework (although you are of course welcome to go beyond).

I hope you will appreciate that these are actually only a few commands and concepts. R can be confusing because it can do so many things. But think of it like learning a language, which means:

* It takes a while to get used to
* You have to practice regularly
* You cannot expect to read and write abstract poetry or a novel that will be nominated for the Nobel Prize after a week
* However, you should be able to use a couple of words and expressions right away and go and use this as much as possible while discovering new words and expressions here and there.
* Also: you will be able to use those expressions successfully without necessarily understanding the grammatical logic or etymology behind them. But as you keep on at you might find out these things at some point.

What I am trying to do in this document and in the course overall is to provide you a couple of R words and expressions for you to play around with. Playing means:

* you try the same commands with different data (e.g. the data for your group project)
* you try change things here and there (change how a figure looks slighlty, change how a variable is defined)
* you think of something you might want to do (construct a new variable based on exisitng ones, create a figure, combine and clean some unusual data). You see if you can do it by consulting the help functions and the internet. If you don’t manage you can ask your teachers (us) for help.

So what are the basic expressions in R that you want to start your journey to Rland with; i.e. the equivalent of ordering a coffee and a croissant?

# Getting data

You will want to start with getting data. Let’s look at the data from Exercise 2.2 however it will be very similar if you were to use some other data.

covid=read.csv("https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv")

i.e. you use the read.csv command to load the dataset into the memory as a data frame

# Data manipulation and cleaning

Data comes rarely in a way such that you can start analysis right away. So you need to know some commands that help you with things like removing some observations or creating new variables.

## Create new variables from existing and filter data

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

# keep the data point for the end of July 2020  
covid\_july=covid %>%   
 filter(date=="2020-07-31")   
  
  
# create various ratios  
covid\_july = covid\_july %>%  
 mutate(  
 deathsOcases=deaths/cases,  
 deaths\_sh=deaths/sum(deaths),  
 meandeaths=mean(deaths),  
 cases\_sh=cases/sum(cases))

Part of data manipulation is that you look at your data. We already saw the summary command

covid\_july %>% summary()

## date state fips cases   
## Length:55 Length:55 Min. : 1.00 Min. : 42   
## Class :character Class :character 1st Qu.:17.50 1st Qu.: 13457   
## Mode :character Mode :character Median :31.00 Median : 46948   
## Mean :32.04 Mean : 83121   
## 3rd Qu.:45.50 3rd Qu.: 90956   
## Max. :78.00 Max. :502273   
## deaths deathsOcases deaths\_sh meandeaths   
## Min. : 2.0 Min. :0.004528 Min. :0.000013 Min. :2797   
## 1st Qu.: 263.5 1st Qu.:0.014963 1st Qu.:0.001713 1st Qu.:2797   
## Median : 943.0 Median :0.021222 Median :0.006129 Median :2797   
## Mean : 2797.5 Mean :0.029697 Mean :0.018182 Mean :2797   
## 3rd Qu.: 3491.0 3rd Qu.:0.039424 3rd Qu.:0.022689 3rd Qu.:2797   
## Max. :32372.0 Max. :0.088978 Max. :0.210396 Max. :2797   
## cases\_sh   
## Min. :9.190e-06   
## 1st Qu.:2.944e-03   
## Median :1.027e-02   
## Mean :1.818e-02   
## 3rd Qu.:1.990e-02   
## Max. :1.099e-01

With that we see that the maximum death rate is nearly 9%. But which state was that in? You can use the filter command again to work it out:

covid\_july %>% filter(deathsOcases>0.088)

## date state fips cases deaths deathsOcases deaths\_sh meandeaths  
## 1 2020-07-31 Connecticut 9 49810 4432 0.08897812 0.02880503 2797.491  
## cases\_sh  
## 1 0.01089536

covid\_july=covid\_july %>% arrange(deathsOcases)  
covid\_july=covid\_july %>% arrange(-deathsOcases)

## Compute summary statistics or aggregate data

For instance work out the total death and case figures from state level ones

# now we aggregate to daily level  
usdaily=covid %>% group\_by(date) %>%   
 summarise(deaths=sum(deaths),   
 cases=sum(cases))  
   
head(usdaily)

## # A tibble: 6 x 3  
## date deaths cases  
## <chr> <int> <int>  
## 1 2020-01-21 0 1  
## 2 2020-01-22 0 1  
## 3 2020-01-23 0 1  
## 4 2020-01-24 0 2  
## 5 2020-01-25 0 3  
## 6 2020-01-26 0 5

#### make lag and lead example

What if you want something other than the sum of a variable? There loads of other functions. You can start by using help function (by typing ?summarise() in the console) or read about the summarise command online (just google dplyr summarise which would lead you for instance to [this](https://dplyr.tidyverse.org/reference/summarise.html))

## Combining data

First you need another dataframe to combine

pop=read.csv("https://mondpanther.github.io/datastorieshub/data/populationdata.csv")

You can use the merge command which is an older command. A newer version of the same idea are the join commands (As I mentioned: the difficulty with R is that there are often several different commands that do more or less the same)

covid\_julyb=covid\_july %>% left\_join(pop,by="state")

## Calculations that involve values from different rows

The mutate command examples we have seen so far create new variables by combining values from existing variables that are in the same row of a dataframe (e.g. when computing deathsOcases). We also have seen examples where we use values that refer to the same line as well as calculations that apply to the full dataframe columns (e.g. when computing the share variables such as deaths\_sh=deaths/sum(deaths)). Sometimes we need to compute new values that are based on values in other rows but not necessarily all values as in sum(deaths). A good example and use case of that emerges from the covid figures used earlier. Note that the dataset we are using report cumulative figures of deaths and cases for the whole pandemic. What if instead we wantd to examine the daily figures instead? For that we would have to look at the change in cumulative numbers for a given day (i.e. the cumulative figure today minus that of yesterday for all days)

To calculate that on the usdaily dataframe we can use the lag() function within the mutate() function:

usdaily=usdaily %>% mutate(Dcases=cases-lag(cases,1),  
 Ddeaths=deaths-lag(deaths,1))

i.e. we subtract the value of the cases and death variables in a given row with the value in the previous row (make sure the table is sorted in the right way; i.e. in this case by date). Also note that you that the second parameter controls the lag; i.e. if you wanted to subtract the value 2 days back you would have to put a 2 instead of a 1. There is also a function that allows you to refer to values in rows further down in the dataframe; it’s called lead().

Note that you can combine that with the group\_by command. For instance, in the original covid dataframe you might want to get daily changes by US state:

covid=covid %>% arrange(state,date) %>%   
 group\_by(state) %>%   
 mutate(Dcases=cases-lag(cases,1),  
 Ddeaths=deaths-lag(deaths,1))

Compare this with using lag command without group\_by:

xcovid=covid %>% arrange(state,date) %>%   
 group\_by(state) %>%   
 mutate(Dcases=cases-lag(cases,1),  
 Ddeaths=deaths-lag(deaths,1))

Can you see the difference covid and xcovid? Can you explain what is wrong in the xcovid way of computing daily changes?

# Data visualisation

Let’s run through the ggplot command once more…..as I said: it’s a bit of swiss army knife of plotting

## Scatter plot

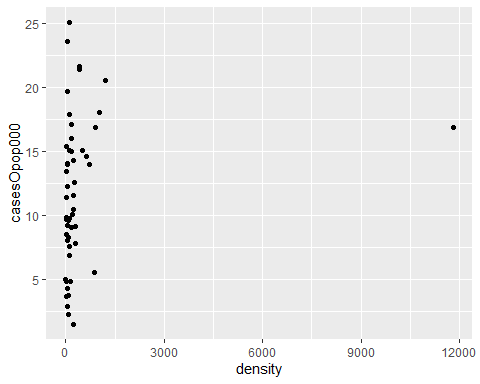
However first: with more data comes more data manipulation; e.g. cases per capita….

covid\_julyb=covid\_julyb %>%   
 mutate(casesOpop000=cases/pop\*1000)

Now how about a scatter plot of cases per capita and population density (measured in people per square mile)?

library(ggplot2)  
  
  
p=ggplot(covid\_julyb,   
 aes(x=density, y=casesOpop000 )) +   
 geom\_point()   
p

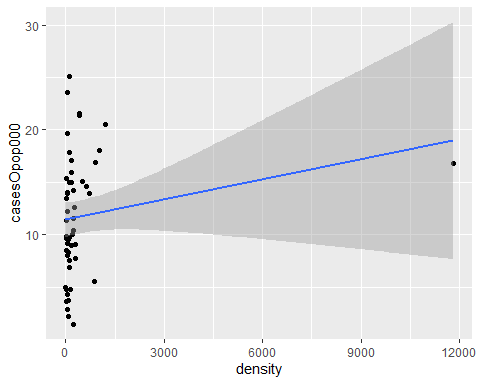
## Warning: Removed 3 rows containing missing values (geom\_point).



p + geom\_smooth(method=lm)

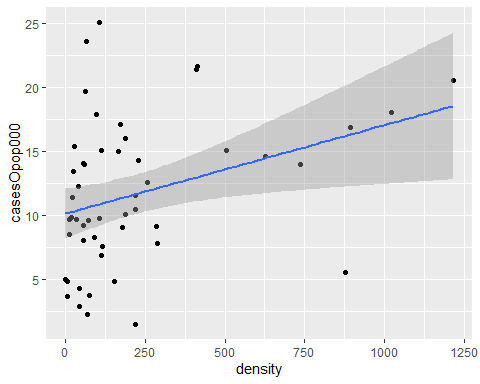
## `geom\_smooth()` using formula 'y ~ x'

## Warning: Removed 3 rows containing non-finite values (stat\_smooth).  
  
## Warning: Removed 3 rows containing missing values (geom\_point).



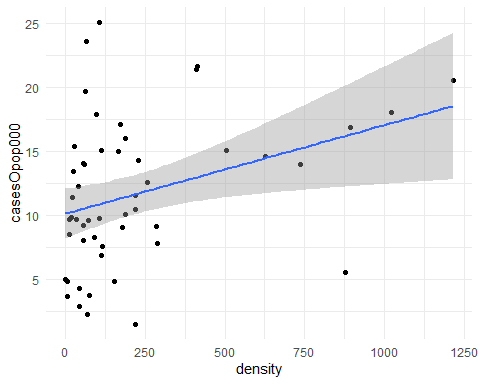
# Dropping an outlier  
  
ggplot(covid\_julyb %>% filter(density<6000), aes(x=density, y=casesOpop000 ))+   
 geom\_point() +   
 geom\_smooth(method=lm)

## `geom\_smooth()` using formula 'y ~ x'



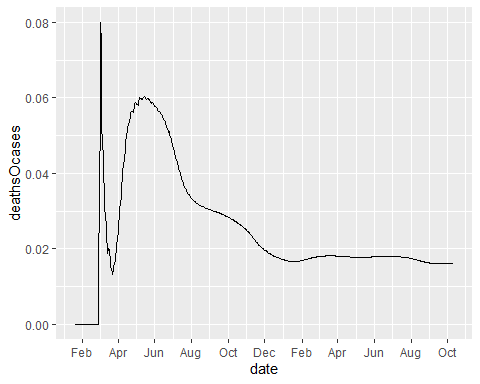
# Less clutter  
  
#newfr=covid\_julyb %>% filter(density<6000)  
  
ggplot(covid\_julyb %>% filter(density<6000), aes(x=density, y=casesOpop000 ))+   
 geom\_point() +   
 geom\_smooth(method=lm)+theme\_minimal()

## `geom\_smooth()` using formula 'y ~ x'



## Time series plot

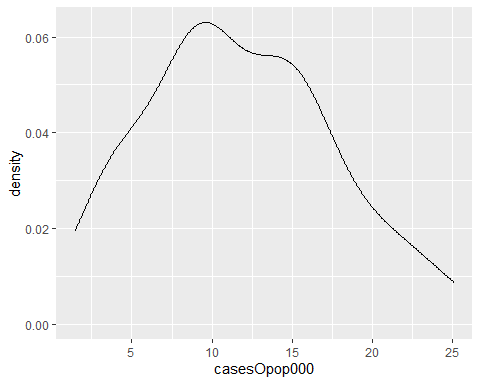
library(ggplot2)  
  
usdailyx=usdaily %>% mutate(date=as.Date(date),  
 deathsOcases=deaths/cases)  
  
ggplot(usdailyx, aes(x=date, y=deathsOcases)) +  
 #geom\_point() +  
 geom\_line() +  
 scale\_x\_date(date\_breaks="2 month",  
 date\_labels = "%b")



## Density plot

covid\_julyb %>% ggplot(aes(x=casesOpop000))+geom\_density()

## Warning: Removed 3 rows containing non-finite values (stat\_density).



Note, that what we put in the aesthetic depends on the plot type. A density plot only needs one variable. # Fitting an econometric model

The simplest case is the linear model which we can fit using the OLS algorithm. The lm() command is your tool for that.

df=covid\_julyb %>% filter(density<6000)   
  
reg=lm(casesOpop000~density,df)  
  
summary(reg)

##   
## Call:  
## lm(formula = casesOpop000 ~ density, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -10.6365 -3.1810 -0.4811 3.3123 14.2090   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 10.103760 0.970901 10.407 5.27e-14 \*\*\*  
## density 0.006921 0.002751 2.516 0.0152 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.466 on 49 degrees of freedom  
## Multiple R-squared: 0.1144, Adjusted R-squared: 0.0963   
## F-statistic: 6.328 on 1 and 49 DF, p-value: 0.01521

reg %>% summary()

##   
## Call:  
## lm(formula = casesOpop000 ~ density, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -10.6365 -3.1810 -0.4811 3.3123 14.2090   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 10.103760 0.970901 10.407 5.27e-14 \*\*\*  
## density 0.006921 0.002751 2.516 0.0152 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.466 on 49 degrees of freedom  
## Multiple R-squared: 0.1144, Adjusted R-squared: 0.0963   
## F-statistic: 6.328 on 1 and 49 DF, p-value: 0.01521