Analytical Notebook

Stefan Zimmermann

11 7 2020

Install and load R-Packages

```
# install and load packages
loadpackage <- function(x){
  for( i in x ){
    #  require returns TRUE invisibly if it was able to load package
    if( ! require( i , character.only = TRUE ) ){
      #  If package was not able to be loaded then re-install
      install.packages( i , dependencies = TRUE )
    }
    #  Load package (after installing)
    library( i , character.only = TRUE )
  }
}

# load packages
loadpackage( c("readr", "knitr", "dplyr", "tidyr", "sparklyr"))
```

Load Covid-Datasets

The datasets UID_ISO_FIPS_LookUp_Table.csv and time_series_covid19_confirmed_global.csv are loaded.

```
# use url to current dataset
url_data1 <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/UID_IS
url_data2 <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_c

# load datasets
data1 <- read_csv(url_data1)
```

```
## Parsed with column specification:
## cols(
##   UID = col_double(),
##   iso2 = col_character(),
##   iso3 = col_character(),
##   code3 = col_double(),
##   FIPS = col_character(),
##   Admin2 = col_character(),
##   Province_State = col_character(),
##   Country_Region = col_character(),
##   Lat = col_double(),
##   Long_ = col_double(),
##   Combined_Key = col_character(),
##   Population = col_double()
## )
```

```r
data2 <- read_csv(url_data2)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Province/State` = col_character(),
##   `Country/Region` = col_character()
## )

## See spec(...) for full column specifications.
```

Data Cleaning

Here the data sets are prepared for merging and reshaping. The ID variables must be standardized. Since we are only interested in countries and not in regions # we keep only a subset of the dataset without regions. Then we drop countries without information and we keep all important variables. We reshape the dataset to long format

```r
# harmonise ID-Variables in both datasets
names(data1)[names(data1) == "Long_"] <- "Long"
names(data2)[names(data2) == "Country/Region"] <- "Country_Region"
names(data2)[names(data2) == "Province/State"] <- "Province_State"

# Since we are only interested in countries and not in regions
# we keep only a subset of the dataset without regions.
data1 <-subset(data1, is.na(data1$Province_State))
data2 <-subset(data2, is.na(data2$Province_State))

# merge with country name. long and lat differ in both datasets
data_wide <- left_join(data1, data2, c("Country_Region"))

# Check Countries without infos
data_wide$Country_Region[is.na(data_wide$`1/22/20`)]
data_wide$Country_Region[is.na(data_wide$Population)]
# drop countries without info
data_wide <- data_wide %>% drop_na(`1/22/20`)
data_wide <- data_wide %>% drop_na(Population)

# keep only necessary variables
data_wide <-subset(data_wide, select = c("Country_Region", "Population",
                                         names(data_wide)[16:length(data_wide)]))

# Reshape wide to long
data_long <-
  reshape(
    data = as.data.frame(data_wide),
    varying = list(names(data_wide)[3:length(data_wide)]),
    timevar = "day",
    v.names = "count",
    idvar = c("Country_Region"),
    direction = "long",
    times = names(data_wide)[3:length(data_wide)]
  )
```

Setting up Spark

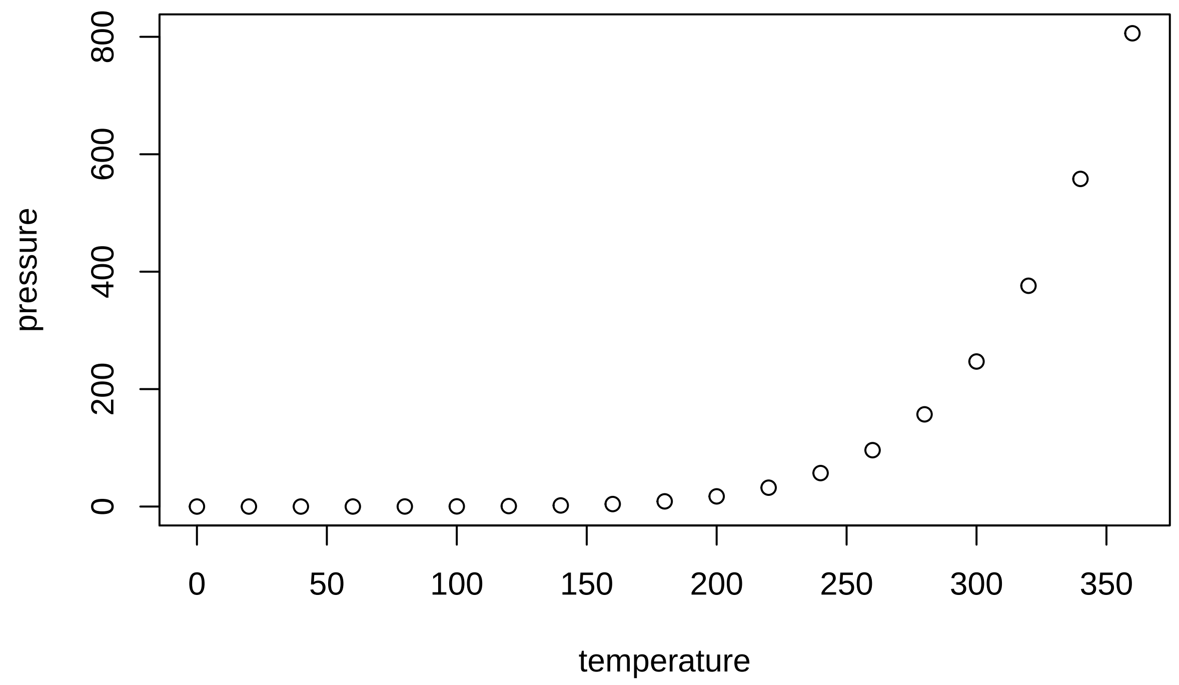
```r
sc <- spark_connect(master = "local",
                    version = "2.4.3")
```

```
## Re-using existing Spark connection to local
```

```r
data1 <- copy_to(sc, data1, overwrite = T)
data2 <- copy_to(sc, data2, overwrite = T)
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.