Lab 3: The 2014 Ebola Epidemic, Part 1

### 1. Objectives

* Use functions from the tidyverse of R packages to load, re-organize, and plot data from the World Health Organization (WHO) that measured the 2014 Ebola Epidemic.
* Use linear and nonlinear least-squares regression methods to quantify parameters that describe the exponential model for early epidemic dynamics and the logistic model for long-term epidemic dynamics.

And foreshadowing Objectives for next week:

* Use parameters fit from the WHO data to simulate the dynamics of the 2014 Ebola Epidemic in systems of differential equations.
* Use vector fields to understand the state space for ebola dynamics.

### 2. Introduction

This week, we’ll use functions from the tidyverse package to load, clean, and plot data from the WHO on the 2014 Ebola Epidemic in Guinea, Liberia, and Sierra Leone, which are among the nations most severely impacted by the outbreak. This lab will draw on functions from [R4DS 12.1-12.4](http://r4ds.had.co.nz/tidy-data.html) to “tidy” data. We’ll start by visualizing the data that describe the growth in the total ebola cases and total ebola deaths through time.

After we visualize the data, we’ll fit linear and nonlinear models to quantify parameters that describe the dynamics of exponential growth in ebola infection at the beginning of the epidemic. Next, we’ll fit nonlinear logistic models to describe the long-term dynamics of ebola infection and mortality.

### 3. Formatting dates in the WHO Ebola Data

The first step for our data analysis is to load the tidyverse library. After that, you can read the WHO ebola data with the read\_csv() function.

library(tidyverse)  
  
# Load data into R   
ebola\_raw <- read\_csv("data/ebola.csv")  
  
# Look at some data characteristics  
dim(ebola\_raw)

## [1] 156 7

ebola\_raw

## # A tibble: 156 x 7  
## `WHO report dat… `Total Cases, G… `Total Deaths, … `Total Cases, L…  
## <chr> <dbl> <dbl> <dbl>  
## 1 3/1/14 0 0 0  
## 2 3/2/14 0 0 0  
## 3 3/3/14 0 0 0  
## 4 3/4/14 0 0 0  
## 5 3/5/14 0 0 0  
## 6 3/6/14 0 0 0  
## 7 3/7/14 0 0 0  
## 8 3/8/14 0 0 0  
## 9 3/9/14 0 0 0  
## 10 3/10/14 0 0 0  
## # … with 146 more rows, and 3 more variables: `Total Deaths,  
## # Liberia` <dbl>, `Total Cases, Sierra Leone` <dbl>, `Total Deaths,  
## # Sierra Leone` <dbl>

The first thing that we are going to fix about the raw data is the date column. Currently, the date column within the ebola\_raw data frame is a character class. But we don’t want each date to be a totally unique set of characters, we want R to recognize the dates as sequential. In R, Date is another class of R object (along with numeric, character, and factor classes) that we haven’t yet encountered. Converting our column of dates from a character class to a Date class object will make it easier to deal with time series appropriately within R.

For example, you can imagine that if you’re making a plot with date along the x-axis, we’d want R to appropriately scale the dates so that they display with the proper values at the tick marks along the axis. We wouldn’t want every single date to be its own unique tick mark. If we keep the date column as a character class, each entry looks like an entirely unique object compared against all the other dates, and R wouldn’t know how to connect them all appropriately.

To convert the dates column from a character to a Date class, we’ll use the lubridate library, which is installed with the tidyverse package, but is not automatically loaded, so needs to be loaded on its own.

Once we load lubridate, we look at the format of the date data. When we see that it’s in month/day/year format (the U.S. convention), so we’ll use the mdy() function from lubridate to re-define the character object as a Date class object.

# load lubridate library  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

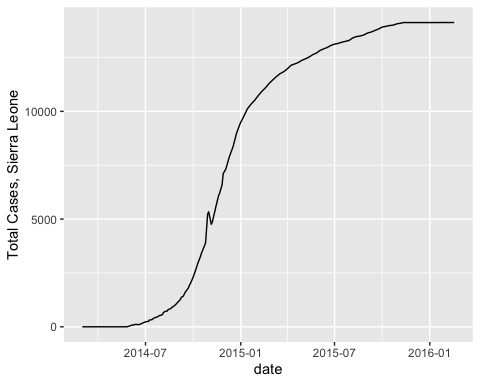
# Look to see what format the dates data are in (first 10 elements in vector)  
ebola\_raw$`WHO report date`[1:10]

## [1] "3/1/14" "3/2/14" "3/3/14" "3/4/14" "3/5/14" "3/6/14" "3/7/14"   
## [8] "3/8/14" "3/9/14" "3/10/14"

# Use the mdy() function to convert the character data to Date data and  
# erase old column called `WHO report date`  
ebola\_date <- mutate(ebola\_raw,   
 date = mdy(`WHO report date`), `WHO report date` = NULL)  
  
# Look to see how the dates have changed (look at first 10 elements in column)  
ebola\_date$date[1:10]

## [1] "2014-03-01" "2014-03-02" "2014-03-03" "2014-03-04" "2014-03-05"  
## [6] "2014-03-06" "2014-03-07" "2014-03-08" "2014-03-09" "2014-03-10"

# Plot time series of total ebola cases in Sierra Leone  
ggplot(ebola\_date) +   
 geom\_line(mapping = aes(x = date, y = `Total Cases, Sierra Leone`))



### 4. Tidying the WHO Ebola Data

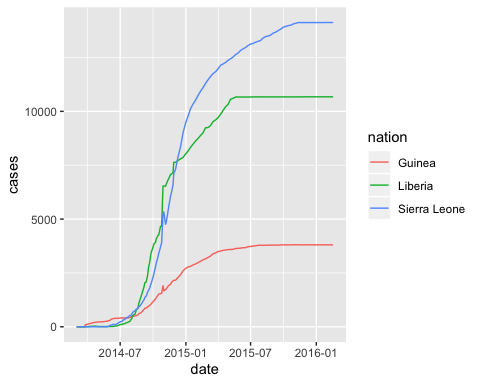
To tidy the WHO data, we need to re-oranize the data frame so that the things in columns are the most basic level of unqiue variables, and each row represents one observation. Right now, variables are mixed up in the column names, with the variables “nation” and “cases” or “deaths” combined into individual columns. This makes it challenging to visualize and analyze the data, because it’s impossible to subset the data by its unique variables (for example all the cases) when there are variables that are stuck together.

1. Examining ebola\_date, draw the data frame that we want to end up with if we’re going to tidy the data to have one one observation “event” per row, with variables as the column names. How would we have to move the columns/variables in ebola\_date to get this ideal tidy data frame?

# Reorganize dataset to be in a tidy & cleaned up format   
  
# Move columns with a mix of nation and count variables to single rows with   
# one variable per row  
ebola\_tidy <- gather(ebola\_date, nation, "count",   
 `Total Cases, Guinea`:`Total Deaths, Sierra Leone`)  
  
# Separate out the count variable name (cases or deaths) from the nation variable  
ebola\_tidy2 <- separate(ebola\_tidy, nation, c("type","nation"), ", ")  
  
# Move the cases & deaths to be two columns per row: date, nation, cases, deaths  
ebola\_tidy3 <- spread(ebola\_tidy2, type, count)  
  
# Rename column names with spaces - clean up data frame  
ebola\_tidy4 <- mutate(ebola\_tidy3,   
 cases = `Total Cases`, deaths = `Total Deaths`,  
 `Total Cases` = NULL, `Total Deaths` = NULL)

We can see above that we took four steps to manipulate and clean the data to get it in a tidy format. In the R tidyverse there is a powerful way to combine these four steps into more fluid code: the pipe, written as %>%. Pipes can connect functions together, by passing the output data frame from one step as the input dataframe to the next step. We can use pipes to combine the four steps from above into one simpler script as:

# Tidy all the WHO ebola data  
ebola\_alltidy <- ebola\_date %>%  
 gather(nation, "count", `Total Cases, Guinea`:`Total Deaths, Sierra Leone`) %>%  
 separate(nation, c("type","nation"), ", ") %>%  
 spread(type, count) %>%  
 mutate(cases = `Total Cases`, deaths = `Total Deaths`,  
 `Total Cases` = NULL, `Total Deaths` = NULL)  
  
# Look at time series of total ebola cases in all countries  
ggplot(ebola\_alltidy) +   
 geom\_line(mapping = aes(x = date, y = cases, color = nation))



1. Describe (in words) how transforming the ebola\_date data frame into a tidy format facilitates the use of ggplot() within the above graph?

The last step that we’ll do at this point is to add a column to the ebola\_date data frame that represents the days since monitoring began. This will translate the date column into a vector that starts at 1 and ends at the last row in the data frame (the last date of monitoring).

# Add column for days since monitoring began: useful to have in addition to the date column for modeling later on (days is easier to model than date)   
ebola\_alltidy <- ebola\_alltidy %>%  
 mutate(monitor\_days = as.numeric(date) - min(as.numeric(date)) + 1)

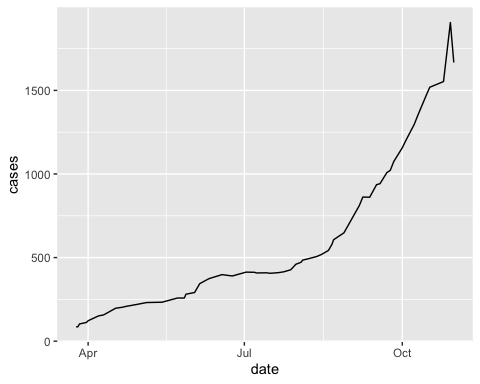
1. Why do you thing that we need as.numeric() around date in the above function? (Hint: what class is date and what class are we trying to make monitoring\_days?)

### 4. Fitting an exponential function to early-epidemic data

We’ll now focus on data from the early days of the ebola epidemic. During this time, epidemiologists were concerned about quantifying the [rate of exponential growth](http://www.npr.org/sections/goatsandsoda/2014/09/18/349341606/why-the-math-of-the-ebola-epidemic-is-so-scary) of ebola cases. This is important for trying to predict the near-term dynamics of ebola infections.

To focus on the early part of the epidemic, we’ll subset the entire ebola dataset to the time period before 11 November 2014 and we’ll focus just on dynamics within the nation of Guinea, using the filter() function. filter() takes only particular rows from a data frame that meet the conditions that you specify. It comes from the tidyverse, so the first argument to filter is the data frame, and the subsequent arguments are the conditions that we want to filter by.

# Filter to only look at early data using filter(): before 2014-11-01, cases only,   
# Guinea only, counts > 0 (to avoid ln(0) errors when fitting exponenetial)  
ebola\_early\_guinea <- filter(ebola\_alltidy, date < "2014-11-01",   
 nation=="Guinea", cases > 0)  
  
# Plot early ebola outbreak in Guinea  
ggplot(ebola\_early\_guinea) +   
 geom\_line(mapping = aes(x = date, y = cases))



We’ll fit this early-epidemic data to an exponential growth model. In this example of exponential growth, we don’t know the parameter values for the model:

.

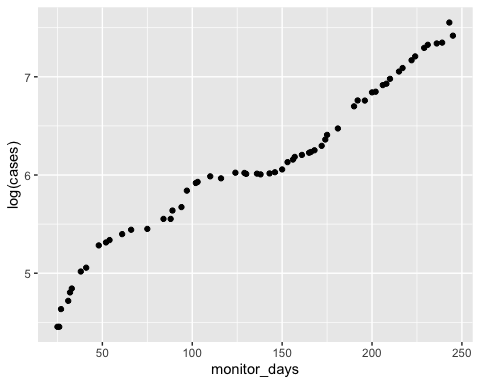
The WHO ebola data describe , and we know the times where data were collected, but we need to fit the exponential model to our data in order to get values for the parameters and . To do this, we’ll first use linear regression (with the lm() function) by log-transforming the count data as the y-variable within the model. Then, we’ll use the nonlinear least-squares method to fit the exponential model directly using the function nls(), and we’ll compare the fit of the nls model to that of the linear model.

1. Explain the biological meaning of the parameters and within the exponential model fit to total ebola cases within a single nation.

Linear regression fits a staight line through a set of data with x and y values by the Least Squares Method, and Least Squares Methods are used to find the best fit line (and the associated parameters) for the most functions that can be integrated numerically, including nonlinear functions like our exponential model. See your lab notes for details on the Least Squares Method, and you can see a proof of the method at [this link](https://web.williams.edu/Mathematics/sjmiller/public_html/BrownClasses/54/handouts/MethodLeastSquares.pdf).

One common technique for fitting data that have an exponential growth relationship is to take the logarithm of the y-variable (i.e., the dependent variable), which transforms the relationship to the x-variable to be linear.

# Plot the data with the log-transformed linear fit  
ggplot(ebola\_early\_guinea) +  
 geom\_point(aes(x = monitor\_days, y = log(cases)))



We can fit a linear model with the lm() function in R, where the variable to the left of the ~ is the response variable (y-variable) and the variables to the right of the ~ are the independent variables (x-variables).

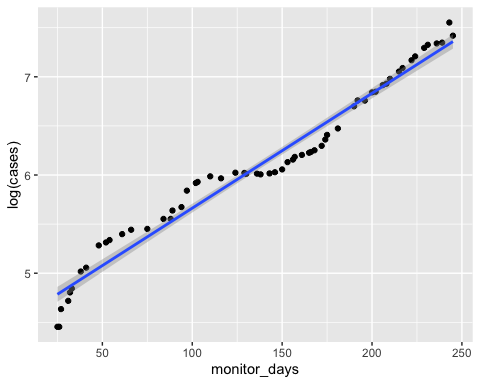
# Take the log of count and fit a linear model with lm()  
early\_linear <- lm(log(cases) ~ monitor\_days, data = ebola\_early\_guinea)  
  
# Look at the summary statistic of the linear regression:   
# focus on the coefficient estimates & Std Error  
summary(early\_linear)

##   
## Call:  
## lm(formula = log(cases) ~ monitor\_days, data = ebola\_early\_guinea)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.34252 -0.13895 0.01983 0.08817 0.23294   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.4927097 0.0443297 101.35 <2e-16 \*\*\*  
## monitor\_days 0.0116985 0.0002876 40.67 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1498 on 58 degrees of freedom  
## Multiple R-squared: 0.9661, Adjusted R-squared: 0.9655   
## F-statistic: 1654 on 1 and 58 DF, p-value: < 2.2e-16

# Look at just the coefficient estimates for r and N0   
coef(early\_linear)

## (Intercept) monitor\_days   
## 4.49270974 0.01169846

# Plot the data with the log-transformed linear fit  
ggplot(ebola\_early\_guinea, aes(x = monitor\_days, y = log(cases))) +  
 geom\_point() +  
 geom\_smooth(method = "lm", se = TRUE)



Rather than transforming the cases data by taking the logarithm and fitting the model with (ordinary least-squares) linear regression, we can also directly fit the exponential with nonlinear least squares regression. To do this, we’ll use the nls() function, where we specify the model and data, and provide starting values for the nls algorithm to find the best fit values for N0 and r

# Fit the non-linear least-squares exponential model: cases = N \* e^(r \* days)  
# nls starting parameters for N and k are required for the optimization  
early\_nls <- nls(cases ~ (N0 \* exp(r\*monitor\_days)),  
 data = ebola\_early\_guinea,   
 start = list(N0 = 10, r = 0.04))  
  
# Look at the summary of the nls fit statistics - focus on Estimated Parameters & Std Error  
summary(early\_nls)

##   
## Formula: cases ~ (N0 \* exp(r \* monitor\_days))  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## N0 6.472e+01 4.694e+00 13.79 <2e-16 \*\*\*  
## r 1.343e-02 3.384e-04 39.70 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 67.7 on 58 degrees of freedom  
##   
## Number of iterations to convergence: 13   
## Achieved convergence tolerance: 8.806e-07

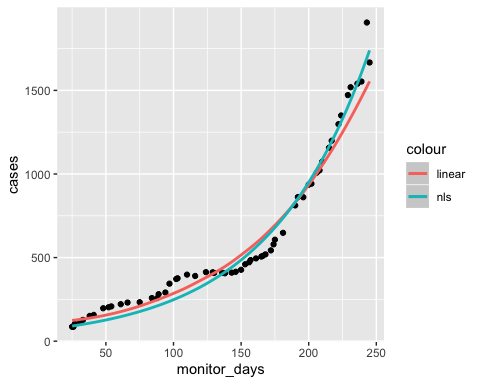
# Look at just the fitted coefficients  
coef(early\_nls)

## N0 r   
## 64.72368375 0.01343319

1. Compare the fits for the linear and nonlinear estimates of and . Recognize that the linear estimate for is captured by the intercept, which need to be “un-logged” (we can do this with the exp() function, which is e raised to the power in the parentheses).

# Plot model output with the Total Cases data for early ebola infection in Sierra Leone  
# 1. Set up main data plotting in the first ggplot function, add geom\_point() to display data  
# 2. Add a geom\_line() to show the fit of the linear model with log-transformed count data  
# 3. Add a geom\_smooth() to show the nonlinear model   
ggplot(ebola\_early\_guinea, aes(x = monitor\_days, y = cases))+   
 geom\_point() +   
 geom\_smooth(aes(x = monitor\_days,   
 y = exp(coef(early\_linear)[1] + monitor\_days\*coef(early\_linear)[2]),  
 color = "linear")) +   
 geom\_smooth(method = "nls",   
 formula = y ~ (N0 \* exp(r\*x)),   
 method.args = list(start=list(N0 = 10, r = 0.04)),  
 se = FALSE, aes(color = "nls")) # se = FALSE needs to be here for nls to work

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



1. For Guinea before 1 November 2014, fit the exponential nls model to total deaths, instead of total cases. How did the value of the parameter change? Is this surprising?

### 5. Fitting a logistic function to the whole epidemic time series

The exponential model is useful for simulating early epidemic dynamics, but its utility begins to fall apart as the infection rate eventually subsides. As we can see in the early ggplot data for the entire dataset, these long-term data look like good candidates for the logistic model. We can also use the nls() function to fit the logistic models for our WHO data.

# Fit logistic model to Sierra Leone Total Cases: cases = K\*N0 / (N0 + (K-N0) \* exp(-r\*t))  
# Use filter() within the data argument of the nls function to constrain the analysis to the  
# nation and data type without having to do this in a separate step.   
SL\_logistic <- nls(cases ~   
 (K\*N0) / (N0 + (K - N0)\*exp(-r\*monitor\_days)),  
 start = list(N0 = 10, r = 0.02, K = 14000),  
 data = filter(ebola\_alltidy, nation=="Sierra Leone"),   
 trace=TRUE)

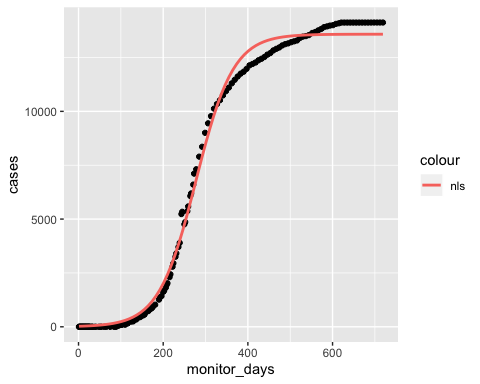
## 685051621 : 1.0e+01 2.0e-02 1.4e+04  
## 654830049 : 1.742843e+01 1.840830e-02 1.403646e+04  
## 642115966 : 3.751943e+01 1.612169e-02 1.408452e+04  
## 549676334 : 9.409270e+01 1.371993e-02 1.408566e+04  
## 127050319 : 1.889385e+02 1.411922e-02 1.363227e+04  
## 98498691 : 1.321885e+02 1.541873e-02 1.360194e+04  
## 83254486 : 6.658435e+01 1.765735e-02 1.358810e+04  
## 31082284 : 4.002914e+01 2.031712e-02 1.363493e+04  
## 22548195 : 3.017882e+01 2.176122e-02 1.362160e+04  
## 21510052 : 2.686619e+01 2.236454e-02 1.359897e+04  
## 21438298 : 2.559711e+01 2.258709e-02 1.358730e+04  
## 21433540 : 2.525193e+01 2.264885e-02 1.358340e+04  
## 21433279 : 2.515629e+01 2.266525e-02 1.358231e+04  
## 21433265 : 2.513399e+01 2.266908e-02 1.358204e+04  
## 21433264 : 2.512878e+01 2.266997e-02 1.358198e+04  
## 21433264 : 2.512759e+01 2.267017e-02 1.358196e+04

# Examine stats summary of the logistic fit  
summary(SL\_logistic)

##   
## Formula: cases ~ (K \* N0)/(N0 + (K - N0) \* exp(-r \* monitor\_days))  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## N0 2.513e+01 3.244e+00 7.745 1.22e-12 \*\*\*  
## r 2.267e-02 4.915e-04 46.126 < 2e-16 \*\*\*  
## K 1.358e+04 5.817e+01 233.474 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 374.3 on 153 degrees of freedom  
##   
## Number of iterations to convergence: 15   
## Achieved convergence tolerance: 8.794e-06

1. Describe the biological meaning of the parameters , , and within the logistic model in this context.

# Plot the logistic model output with the Total Cases data for early ebola infection in Sierra Leone  
ggplot(filter(ebola\_alltidy, nation=="Sierra Leone"),   
 aes(x = monitor\_days, y = cases)) +   
 geom\_point() +   
 geom\_smooth(method = "nls",   
 formula = y ~ (K\*N0) / (N0 + (K - N0)\*exp(-r\*x)),   
 method.args = list(start=list(N0 = 10, r = 0.02, K = 14000)),  
 se = FALSE, aes(color = "nls"))



### Lab Report 3 Questions (due with the set from 3B)

The first Lab Report 3 question will summarize results from this lab that we’ll use to simulate ebola dynamics within Lab 3B:

* Using the nls() function, fit the parameters for the whole epidemic logistic growth model for total ebola cases within each of the three nations. Create a table that organizes the values for the parameters , , and for each nation from the fit to the logistic growth equation.