Mathematics 4MB3/6MB3 Mathematical Biology

http://www.math.mcmaster.ca/earn/4MB3

2019 ASSIGNMENT 2 SOLUTIONS

This assignment was due in class on **4 February 2019 at 9:30am**.

1 Plot P&I mortality in Philadelphia in 1918

(a) Confirm that you have received this data file by e-mail:

`pim_us_phila_city_1918_dy.csv`

This plain text comma-separated-value file can be examined (if you wish) using any plain text editor, such as `Emacs`.

Remark. Note that if you open the data file in Microsoft Excel, Excel will misinterpret the meaning of the date column and convert it to a list of incorrect dates. A plain text editor such as `Emacs` will not alter the file without telling you. □

(b) Read the data into a data frame in R, using the `read.csv()` function. For example, the following chunk of R code should work:

```
datafile <- "pim_us_phila_city_1918_dy.csv"
philadata <- read.csv(datafile)
philadata$date <- as.Date(philadata$date)
```

The purpose of the last line of code above is to ensure that R encodes character strings such as `"1918-10-15"` as dates.

Solution. In the following code, we read the data, convert the date to an R `Date` object, and add a column with the day number (starting on the first day).

```
datafile <- "pim_us_phila_city_1918_dy.csv"
philadata <- read.csv(datafile)
philadata$date <- as.Date(philadata$date)
philadata$daynum <- as.numeric(philadata$date - philadata$date[1] + 1)
```

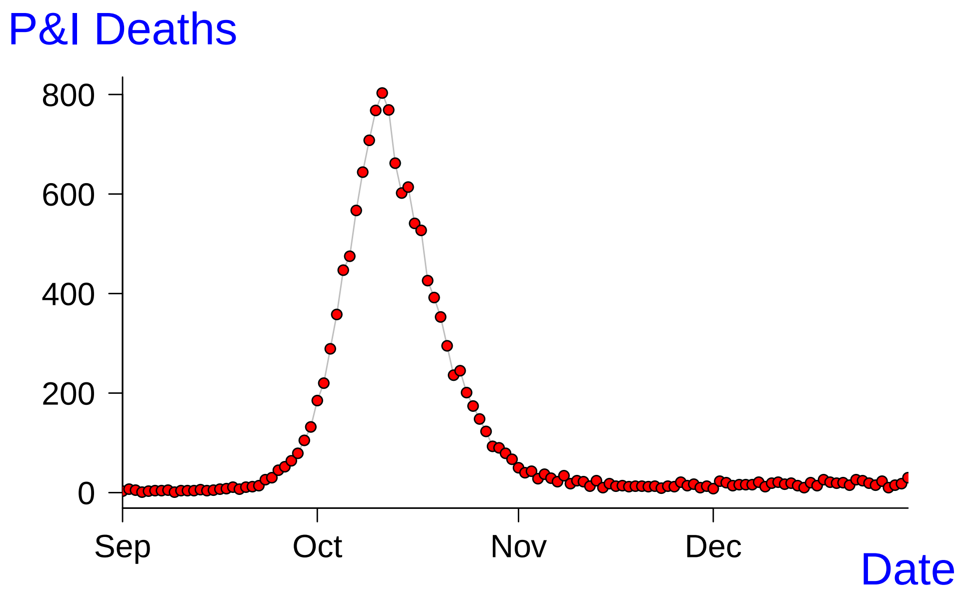
It's a good idea to check that our data frame really does contain what we think, so let's look at the first few rows:

```
head(philadata)
```

```
##          date pim daynum
## 1 1918-09-01   3      1
## 2 1918-09-02   7      2
## 3 1918-09-03   5      3
## 4 1918-09-04   1      4
## 5 1918-09-05   3      5
## 6 1918-09-06   4      6
```

□

(c) Reproduce the Philadelphia 1918 P&I plot:



You'll need to use functions such as `plot()`, `points()` and `lines()`. For a comprehensive list of graphics parameters accepted by these functions, enter `?par` into the Console pane in `RStudio`. There are multiple ways to produce a graph exactly like the above, but the following steps work:

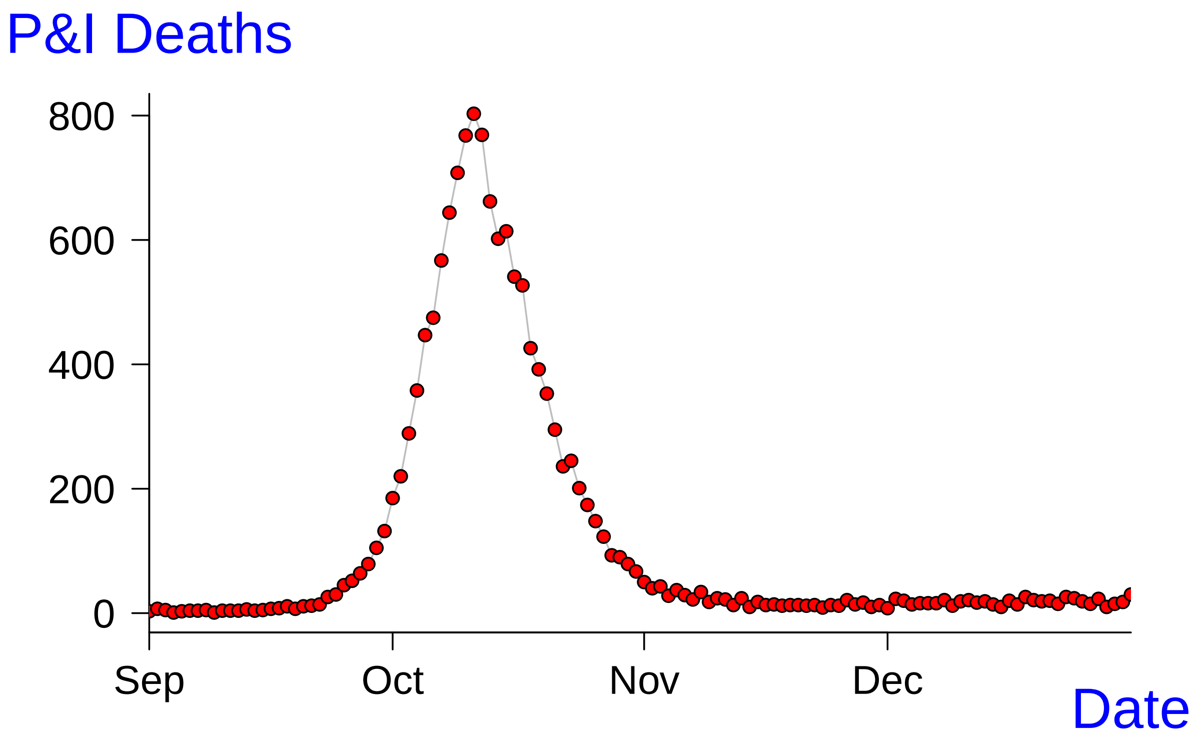
- Use `plot()` to draw the box and basic annotation and the grey line. Suppress labels when doing this (*e.g.*, `xlab=""`). The box type is controlled by the `bty` option and the orientation of annotation is controlled by the `las` option.
- Use `points()` to draw the heavy red dots with black borders. The most elegant way to do this is to set the point character type to 21 (`pch=21`) and the point background colour to red (`bg="red"`). Alternatively, you can use `points()` twice (first to draw the red dots and then to draw the black circles around them).
- Use `mtext()` to add the x and y axis labels in the margins of the plot.

Solution. Since we will need to keep plotting this in various ways in the remainder of the assignment, let's create a function that makes this plot. Later, it will be convenient to be able to plot as a function of day number (since 1 Sep 1918) rather than date, so let's include an option to the plotting function that allows us to use the day number on the x axis. Note that the first argument of the function is the data frame that contains the data of interest. Later it will be convenient to pass a different data frame that contains only part of the full time range.

```r
plot.philadata <- function(df=philadata,use.daynum=FALSE, ...) {
  with(df,{
    ## set the x coordinate (date by default):
    x <- if (use.daynum) daynum else date
    ## draw the graph with a grey line:
    plot(x,pim,
         typ="l",      # draw lines rather than points
         col="grey",   # grey lines
         ann=FALSE,    # suppress axis labels
         las=1,        # horizontal axis labels
         cex.axis=1.4,# larger font for axis labels
         bty="L",      # axes on bottom and left only
         xaxs="i",     # do NOT extend x axis on either side of data
         ...           # other arguments may be passed
         )
    ## add red points with black borders at each data point:
    points(x,pim,pch=21,bg="red")
    ## label the y axis:
    mtext("P&I Deaths",side=3,at=min(x),line=1,col="blue",cex=2)
    ## label the x axis:
    if (use.daynum) {
      title(xlab="Days since 1 Sep 1918",col.lab="blue",cex.lab=1.5)
    } else {
      mtext("Date",side=1,at=max(x),line=2,col="blue",cex=2)
    }
  })
}
```

With no arguments, our function plots the graph in the desired format:

```r
plot.philadata()
```



There are, of course, other ways to create the identical plot. □

2 Estimate $\mathcal{R}_0$ from the Philadelphia P&I time series

(a) The observed mortality time series $M(t)$ is certainly not equal to the prevalence $I(t)$ that appears in the SIR model. Suppose, however, that $I(t) = \eta M(t-\tau)$ for all time (where η and τ are constants), *i.e.,* that the mortality curve is exactly a scaled and translated version of the prevalence curve. Prove that if both I and M are growing exactly exponentially over some time period then their exponential rates are identical. Thus, if we compare them during the "exponential phase" on a logarithmic scale, then both curves will be perfectly straight with exactly the same slope.

Proof. During the time period over which both I and M are growing exponentially, there exist positive constants a and b such that

$$I(t) = I_0 e^{at} \quad \text{and} \quad M(t) = M_0 e^{bt} .$$

But $I(t) = \eta M(t-\tau)$, so

$$I_0 e^{at} = \eta M(t-\tau) = \eta M_0 e^{b(t-\tau)} ,$$

and hence

$$\log I_0 + at = \log(\eta M_0) - b\tau + bt .$$

Equivalently,

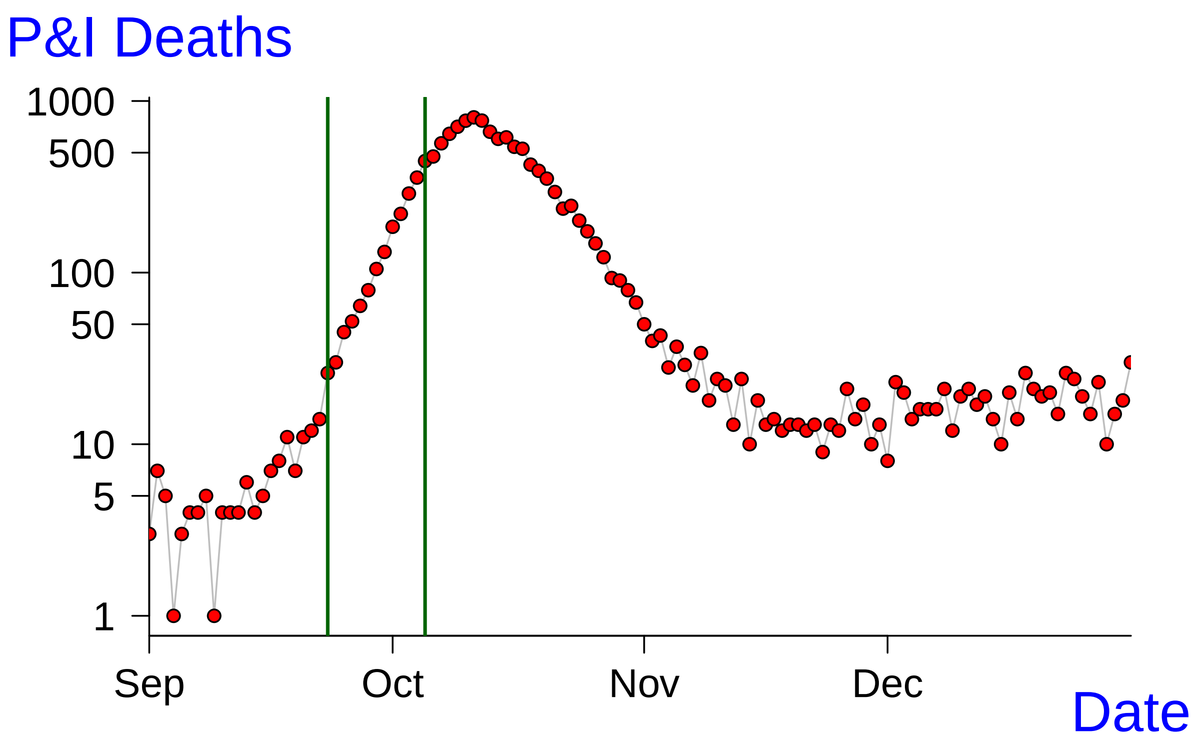
$$(a-b)t + \log I_0 b/(\eta M_0) = 0\,,$$

which implies, in particular, that $a = b$. □

(b) Fit a straight line to the part of the Philadelphia 1918 mortality time series that looks straight on a logarithmic scale (and show your result in a plot). Once you get the hang of it, the easiest way to do this is to use the `lm()` function in R (lm stands for linear model). Note that the simplest way to draw a straight line with given slope and intercept is with the `abline()` function. If you find `lm()` counter-intutive to understand then experiment with `abline()` until your eyes tell you that you have discovered a line that provides a good fit.

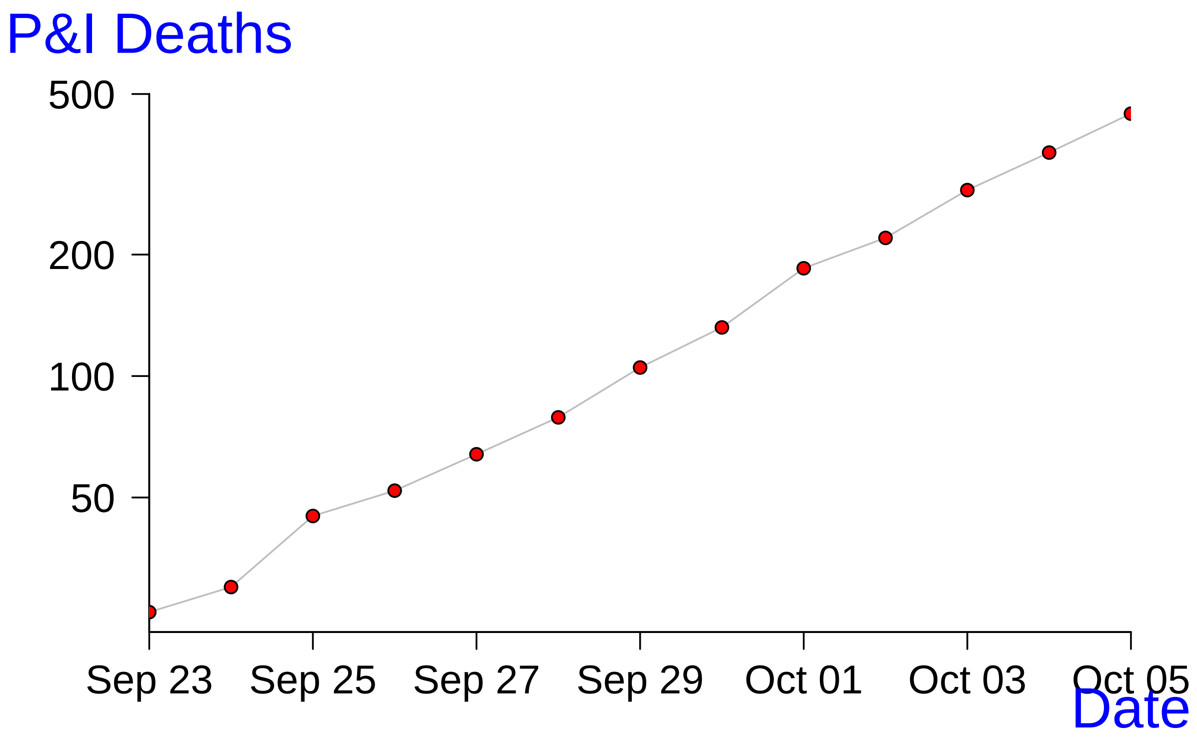
Solution. We first plot the data on a logarithmic scale, and (with a bit of trial and error) locate starting and ending dates of what appears to be the exponentially growing phase of the epidemic (which we identify in the plot below with vertical green lines).

```
exp.start <- as.Date("1918-09-23")
exp.end <- as.Date("1918-10-05")
plot.philadata(log="y")
abline(v=c(exp.start, exp.end), col="darkgreen", lwd=2)
```



We now create a new data frame (`exp.phase.data`) that contains only the exponentially growing phase, and plot just that.

```
exp.phase.data <- subset(philadata, date >= exp.start
                         & date <= exp.end)
plot.philadata(df=exp.phase.data, log="y")
```



We can now use `abline` to draw a line with y intercept a and slope b. But this requires some care. Although R has stored the x axis values in its `Date` format, from the point of view of plotting functions the x axis is the real line (in units of days). So, what date corresponds to the origin? The answer is completely unobvious. We can figure it out by considering the numeric value of any date we like, for example:

```
## extra brackets cause result to be printed:
(exp.start.numeric <- as.numeric(exp.start))

## [1] -18728
```

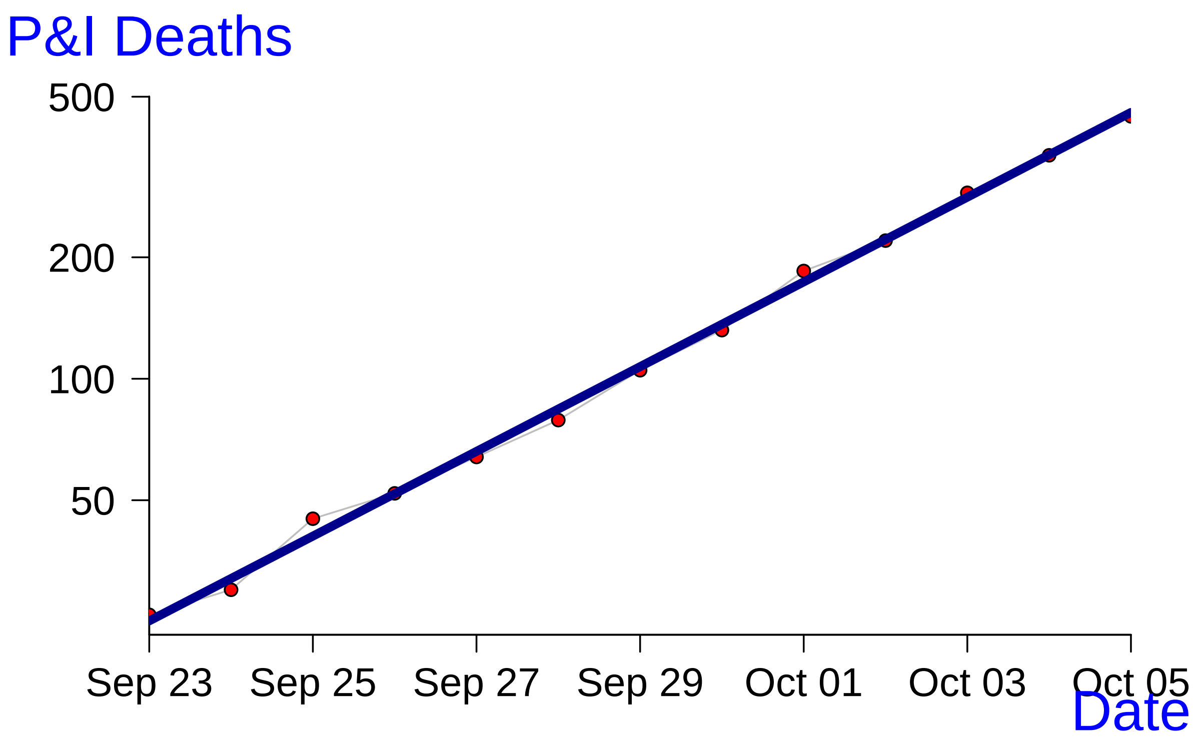
Thus, 1918-09-23 is 18728 days before R's time origin. A bit of arithmetic will tell you that the time origin is 1 January 1970, which we can easily verify:

```
as.numeric(as.Date("1970-01-01"))

## [1] 0
```

Looking at the plot above, and attempting to guess the best fit line, it will be easier to set the y intercept based on an x origin at `exp.start`, *i.e.*, 1918-09-23. We'll call this `myintercept` (note that R stores the y axis as $\log_{10} y$ not y). And we'll call our guessed slope `myslope`. A bit of trial and error with intercepts and slopes yields:

```
plot.philadata(df=exp.phase.data,log="y")
myintercept <- 1.4 # log10 of value marked on plotted axis
myslope <- 0.105
abline(a = myintercept - myslope*exp.start.numeric,
       b = myslope,
       col = "darkblue", lwd=5)
```



Of course, R is a statistical programming language, so it has built-in function for linear regression and countless other statistical tools. The `lm()` function fits a "linear model", *i.e.*, a best fit line, for a given data frame. This is much better than fitting by eye, and the standard statistical method that is used yields an error estimate (the standard error) for the fitted slope and intercept, and provides a measure of statistical significance (p value) for each estimated quantity. This won't make a lot of sense to you unless you've taken a basic statistics course, but here's how to construct the fit and look at the results:

```
fit.lm.log10 <- lm( log10(pim) ~ date, data=exp.phase.data )
coef(summary(fit.lm.log10))

##                 Estimate   Std. Error  t value     Pr(>|t|)
## (Intercept) 1961.7437545 29.675611319 66.10626 1.177158e-15
## date           0.1046744  0.001585066 66.03788 1.190605e-15
```

In the table above, the first line (labelled `"(Intercept)"`) tells us the estimated intercept (at the time origin, not 1918-09-23), the standard error in this estimate, the value of a test statistic (t value) and the measure of statistical significance (p value) obtained from the test ($p < 0.05$ is considered significant, so the fit is highly significant). The second line in the table (labelled `"date"`) gives the estimate of the slope, which is close to the estimate obtained by trial and error above. To extract the estimated slope and its standard error from the table above, we need to refer to the appropriate entry in the table, which is (somewhat counter-intuitively) stored as a vector rather than a matrix:

```
(slope.lm.log10 <- coef(summary(fit.lm.log10))[2])

## [1] 0.1046744

(slope.lm.log10.err <- coef(summary(fit.lm.log10))[4])

## [1] 0.001585066
```

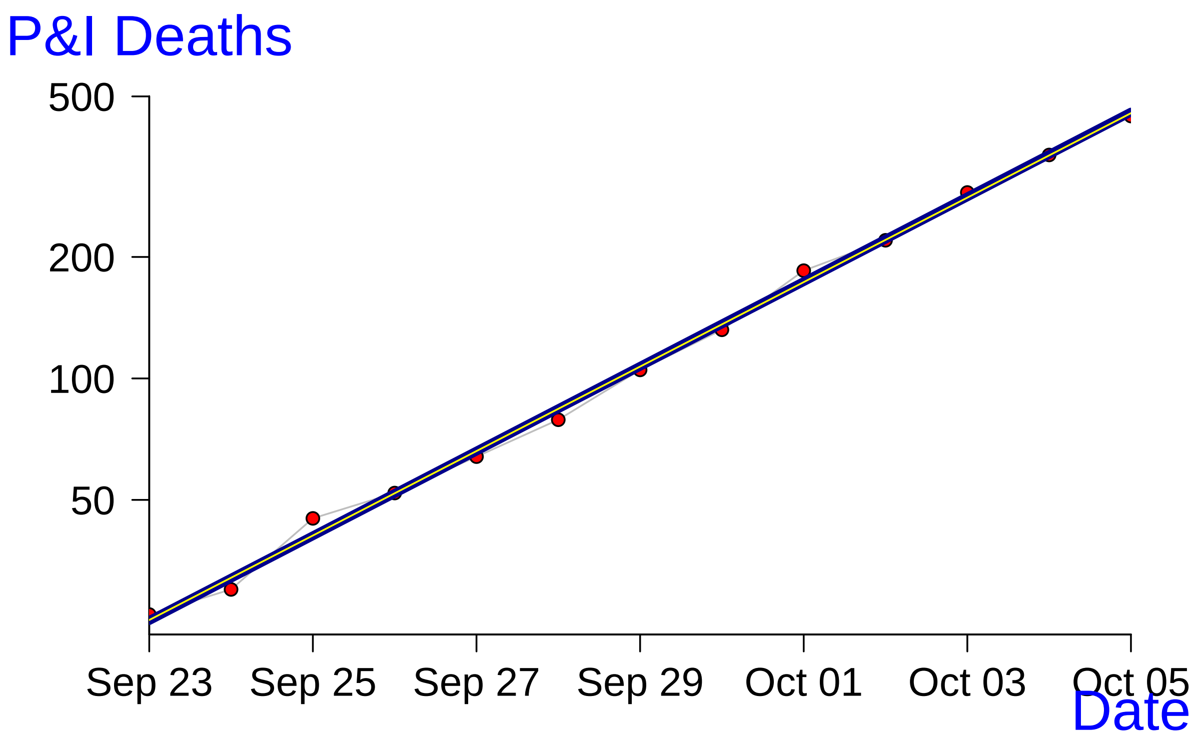
We can now quote our slope estimate together with the error estimate obtained from this simple linear (statistical) model:

$$b = 0.1046744 \pm 0.0015851 \qquad \big(\text{slope of } \log_{10} I \text{ vs } t \text{ [days]}\big). \tag{1}$$

This error estimate for b would be extremely important in a real situation because it could be used to contribute to an error estimate for $\mathcal{R}_0$.

Finally, it is worth noting that if you construct a linear model as above using `lm()` then you can simply pass the result to `abline()` which knows what to do with a linear model. Below, we repeat the plot above and then draw a yellow line by passing our fitted linear model `lm.fit` to `abline()`. The dark blue and yellow lines are identical, but constructing the yellow line using `lm()` is much easier. This is a simple illustration of the object-oriented nature of the R language (`abline()` is a function that acts differently depending on what type of object we pass to it; the same is true of `plot()`).

```
plot.philadata(df=exp.phase.data,log="y")
abline(a = myintercept - myslope*exp.start.numeric,
       b = myslope,
       col = "darkblue", lwd=5)
abline(fit.lm.log10, col="yellow", lwd=1)
```



As emphasized in Equation (1), and highlighted by the fact that we named our fit `fit.lm.log10`, the slope b quoted in Equation (1) is based on taking our vertical axis to be $\log_{10} I(t)$ rather than $\log I(t)$. This allowed us to compare the slope estimated by `lm()` with the slope we got by trial and error. However, below it will be more convenient to have the slope of $\log I(t)$. Noting that

$$\log_{10} x = a + bt \qquad \Longrightarrow \qquad \log x = a\log(10) + b\log(10)\,t,$$

we have

```
slope.lm <- slope.lm.log10 * log(10)
slope.lm.err <- slope.lm.log10.err * log(10)
```

$$b = 0.2410217 \pm 0.0036497 \qquad \big(\text{slope of } \log I \text{ vs } t \text{ [days]}\big). \tag{2}$$

Of course, the same result could be obtained via

```
fit.lm <- lm( log(pim) ~ date, data=exp.phase.data )
```

□

(c) How is the slope of your fitted line related to the parameters of the SIR model? (*Hint:* When I is small, $S \simeq 1$.) Why do you need an independent measure of the mean infectious period to estimate $\mathcal{R}_0$? If the mean infectious period is 4 days, what is your estimate of $\mathcal{R}_0$?

Solution. The basic SIR model (*cf.* equations (2) of Assignment 1) can be written

$$\frac{dS}{dt} = -\mathcal{R}_0 SI \tag{3a}$$

$$\frac{dI}{dt} = \mathcal{R}_0 SI - I \tag{3b}$$

Initially, when I is small and $S \simeq 1$, equations (3) can be approximated by

$$\frac{dS}{dt} = -\mathcal{R}_0 I\,, \tag{4a}$$

$$\frac{dI}{dt} = \mathcal{R}_0 I - I\,. \tag{4b}$$

The second equation is decoupled from the first and can be solved immediately to find

$$\log I(t) = \log I(0) + (\mathcal{R}_0 - 1)t\,, \tag{5}$$

from which it appears that we can infer that the slope of the prevalence curve on a log scale is $\mathcal{R}_0 - 1$. Not quite. We have ignored the fact that equations (3) were written with the time unit chosen to be the mean infectious period ($1/\gamma$) or equivalently with the assumption that $\gamma = 1$. We need the solution expressed without a hidden assumption about the mean infectious period. We can either realize that making the time unit explicit means replace t with $t/(1/\gamma)$ or by sticking γ back in explicitly in equations (3). Either way, we have

$$\log I(t) = \log I(0) + (\mathcal{R}_0 - 1)\gamma t\,, \tag{6}$$

and we can now infer correctly that the initial growth rate of the epidemic (which is the slope of our fitted line) is $(\mathcal{R}_0 - 1)\gamma$. Since the slope contains a factor of γ, if we have no information about the mean infectious period then our slope cannot be used to estimate $\mathcal{R}_0$. However, if we know that the mean infectious period is $1/\gamma = 4$ days then from Equation (2) we can estimate

$$(\mathcal{R}_0 - 1)\frac{1}{4\ \text{day}} \simeq \frac{0.2410217}{\text{day}}\,, \tag{7}$$

i.e.,

$$\mathcal{R}_0 \simeq 1.964087\,. \tag{8}$$

Note that when presenting our results to others, it would be better to say $\mathcal{R}_0 \simeq 2$, since we have no error estimate and 7 decimal precision suggests more confidence than is reasonable based on what we have done. You might like to think about how we could obtain a formal error estimate on $\mathcal{R}_0$ from an error estimate for the mean infectious period together with the standard error in the slope obtained from `lm()` above.

Note that equations (4) correspond exactly to the linearization of equations (3) about the disease free equilibrium (DFE) at $(S, I) = (1, 0)$. Rewriting equations (4) in matrix form we have

$$\begin{pmatrix} \dot{S} \\ \dot{I} \end{pmatrix} = \begin{pmatrix} 0 & \mathcal{R}_0 \\ 0 & \mathcal{R}_0 - 1 \end{pmatrix} \begin{pmatrix} S \\ I \end{pmatrix} . \tag{9}$$

From this you could infer correctly that the dominant eigenvalue of the linearization about the DFE is $\mathcal{R}_0 - 1$ and that that corresponds to the initial growth rate (in units in which the mean infectious period is unity). But beware: the equilibrium is non-hyperbolic so we cannot infer that the phase portrait of the full nonlinear system is topologically conjugate to the phase portrait of the linearization in the vicinity of the focal equilibrium. Moreover, the linearization would be irrelevant if it were zero. Still, even in that case, looking at the behaviour for small I and $S \simeq 1$ would allow us to determine the character of initial epidemic growth.

Estimating initial growth rates (with meaningful error bars) from epidemic curves is actually quite a tricky business. If you are seriously interested in this problem, see the following paper (which can be obtained from my web site):

Ma J, Dushoff J, Bolker BM, Earn DJD, 2013. "Estimating initial epidemic growth rates", *Bulletin of Mathematical Biology*, **76**, 245–260
DOI 10.1007/s11538-013-9918-2 □

3 Fit the basic SIR model to the Philadelphia P&I time series

(a) Install the "deSolve" package. This is done by typing the following command in the Console pane of RStudio:

```
install.packages("deSolve")
```

You will then be prompted to choose a mirror site from which to download the package. It doesn't matter which mirror you choose, but choosing a site in Ontario might save a fraction of a second. *Note:* This is a one-time operation. You do not want an `install.packages()` command inside your solutions code.

(b) Write an R function that plots the solution $I(t)$ of the SIR model for given parameter values ($\mathcal{R}_0$ and $1/\gamma$) and given initial conditions (S_0, I_0). Use the `ode()` function in the `deSolve` package.

Solution. We begin by loading the `deSolve` package:

```
library("deSolve")
```

Now define the vector field for our differential equation.

```r
## Vector Field for SIR model
SIR.vector.field <- function(t, vars, parms=c(beta=2,gamma=1)) {
  with(as.list(c(parms, vars)), {
    dx <- -beta*x*y           # dS/dt
    dy <- beta*x*y - gamma*y  # dI/dt
    res <- c(dx=dx, dy=dy)
    list(res)
  })
}
```

We then use this function inside another function that plots the solution of the associated differential equation.

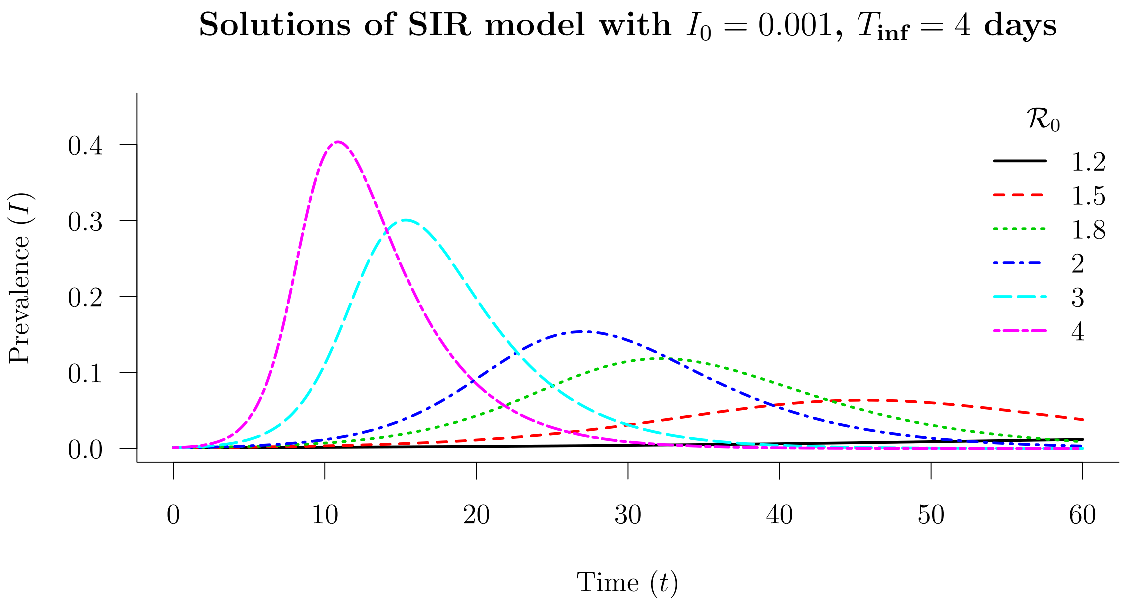
```r
## Draw solution
draw.soln <- function(ic=c(x=1,y=0), tmax=1,
                      times=seq(0,tmax,by=tmax/500),
                      func, parms, scale=1, ... ) {
  soln <- ode(ic, times, func, parms)
  lines(times, soln[,"y"], lwd=3, ... )
}
```

$\square$

(c) For $I_0 = 10^{-3}$ and $S_0 = 1 - I_0$, plot the solutions of the SIR model assuming $1/\gamma = 4$ days and $\mathcal{R}_0 \in \{1.2, 1.5, 1.8, 2, 3, 4\}$. Use the `legend()` command to make a legend on the plot that shows which curves correspond to which values of $\mathcal{R}_0$.

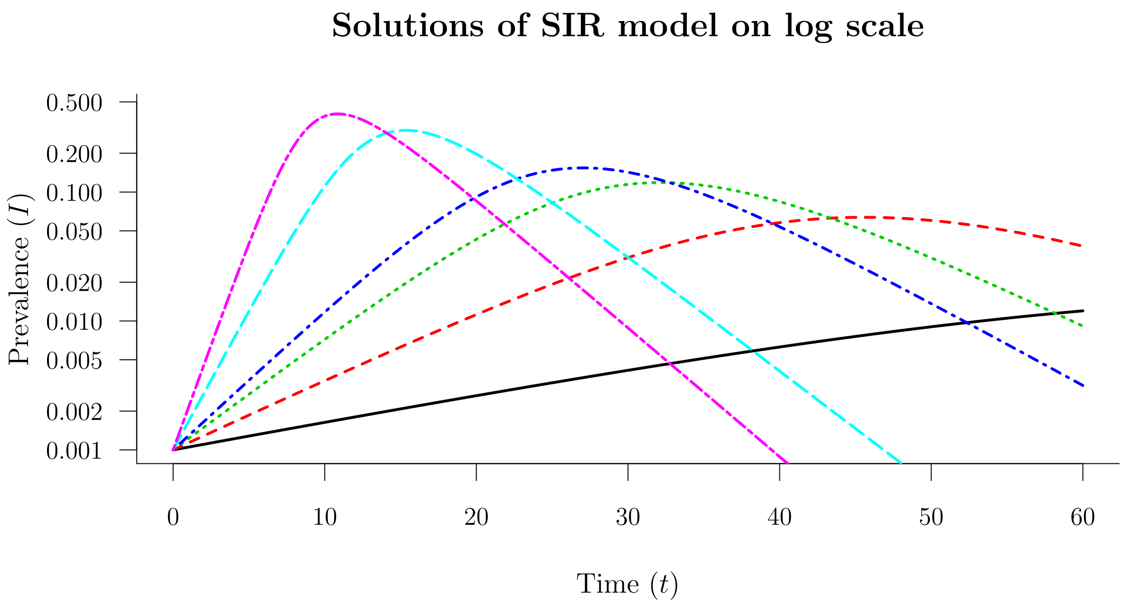
Solution.

```r
## Plot solutions of the SIR model
tmax <- 60 # end time for numerical integration of the ODE
## draw box for plot:
plot(NA,NA,xlim=c(0,tmax),ylim=c(0,0.45),
     type="n",xlab="Time ($t$)",ylab="Prevalence ($I$)",las=1,
     bty="L")
## initial conditions:
I0 <- 0.001
S0 <- 1 - I0
## draw solutions for several values of parameter beta:
R0vals <- c(1.2,1.5,1.8,2,3,4)
gammaval <- 1/4
n <- length(R0vals)
for (i in 1:n) {
  draw.soln(ic=c(x=S0,y=I0), tmax=tmax,
            func=SIR.vector.field,
            parms=c(beta=R0vals[i]*gammaval,gamma=gammaval),
            lty=i, # use a different line style for each solution
            col=i  # use a different colour for each solution
            )
}
legend("topright", legend=R0vals, col=1:n, lty=1:n, lwd=3, bty="n",
       title="$\\R_0$")
## Add a title to the graph:
title.text <- "Solutions of SIR model with $I_0=0.001$, $\\Tinf=4$ days"
title(main=title.text)
```



It is instructive to repeat this plot on a logarithmic scale:

```
## draw box for plot:
plot(NA,NA,xlim=c(0,tmax),ylim=c(0.001,0.45), cex.axis=0.9, log="y",
     type="n",xlab="Time ($t$)",ylab="Prevalence ($I$)",las=1,
     bty="L")
## draw solutions for several values of parameter beta:
for (i in 1:length(R0vals)) {
  draw.soln(ic=c(x=S0,y=I0), tmax=tmax,
            func=SIR.vector.field,
            parms=c(beta=R0vals[i]*gammaval,gamma=gammaval),
            lty=i, # use a different line style for each solution
            col=i  # use a different colour for each solution
            )
}
## Add a title to the graph:
title.text <- "Solutions of SIR model on log scale"
title(main=title.text)
```

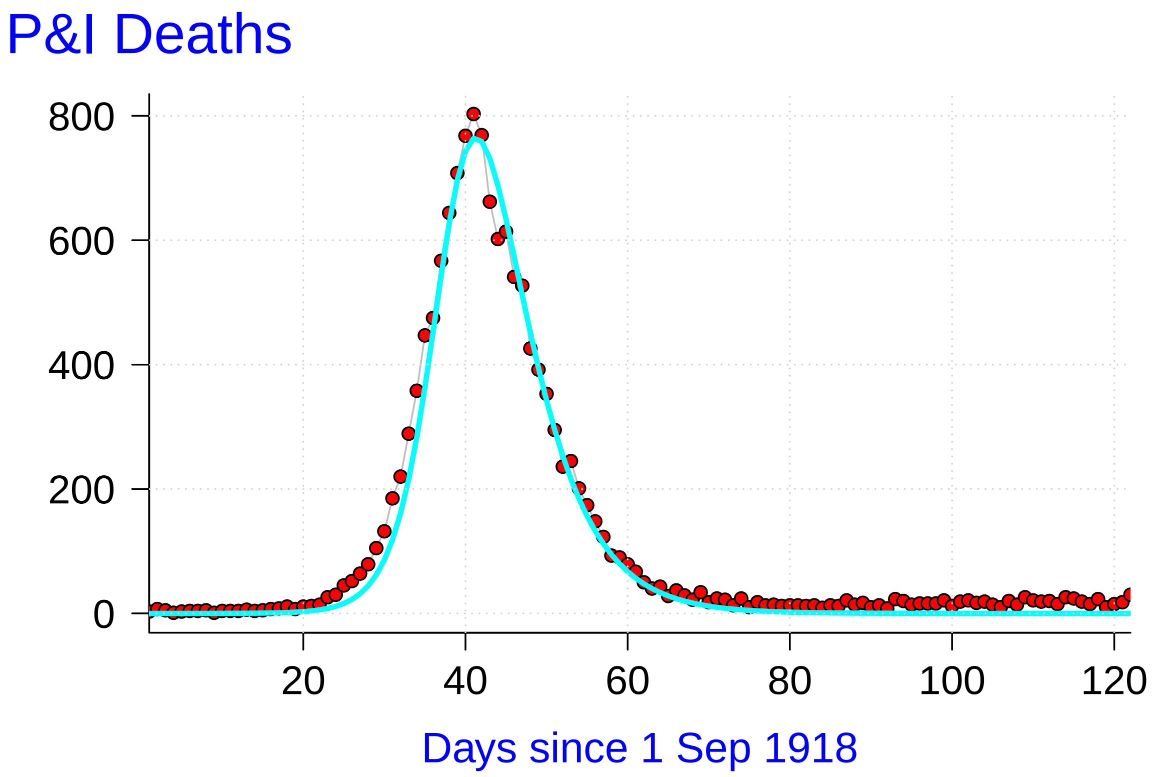


□

(d) By trial and error, find values of $\mathcal{R}_0$ and γ that yield a solution of the SIR model that fits the Philadelphia P&I times series reasonably well. You can assess the quality of fit using the Euclidean distance between the model solution and the data. (*Note:* The trial and error approach is a valuable exercise, but not a suggestion of a method you would really use in practice. We'll discuss better methods for fitting ODE models to data later.)

Solution. In addition to playing with $\mathcal{R}_0$ and γ, in order to produce a simulated epidemic that approximately matches the observed P&I time series, we will need to find an appropriate set of initial conditions. Given the biological context (invasion of a new pathogen) we will assume $S_0 = 1 - I_0$. Note that adjusting I_0 changes the proportion of the population infected at the start of the observed time series, which will have the effect of translating the position of the epidemic peak. It is reasonable to start with the guess that $1/\gamma = 4$ days, and adjust that later if it seems necessary. But we will certainly need to scale the simulated time series, since it produces a proportion of the population infected, whereas the observed data contain the number of deaths on each day. For simplicity, let's ignore the delay between infection and death (assuming a fixed delay, as in the previous question, would affect our estimated I_0 but not the parameters of genuine interest). We are left with three quantities to vary to find a best fit: I_0, $\mathcal{R}_0$ and the scale factor. A few minutes of experimentation led to the values used here:

```r
## set initial conditions:
I0 <- 1e-06
S0 <- 1-I0
## set parameter values:
R0 <- 2.4
gamma <- 0.25
## set factor by which we will scale the simulated epidemic:
scale.factor <- 3500
## set max time for simulation to be the length of observed data:
tmax <- nrow(philadata)
## plot the observed data as a function of day number:
plot.philadata(use.daynum=TRUE)
## compute the simulated epidemic, saving results at observed data times
soln <- ode(y=c(x=S0,y=I0),
            times=1:tmax,
            func=SIR.vector.field,
            parms=c(beta=R0*gamma,gamma=gamma)
           )
I.fit <- scale.factor*soln[,"y"]
lines(soln[,"time"], I.fit, col="cyan", lwd=3)
grid()
```



To test the goodness of fit via the "method of least squares", let's define a function that computes the Euclidean distance between an `observed` time series and a `predicted` time series. Writing this as a function allows us to illustrate a good programming practice, namely checking for nonsense and returning an error if nonsense is detected (instead returning a garbage result).

```
Euclid <- function(observed,    # the data
                   predicted) { # the simulation
  ## stop with an error if our simulated time series
  ## is not the same length as the observed time series
  ## to which we are attempting to fit:
  stopifnot( length(observed) == length(predicted) )
  ## given that all's well, return the Euclidean distance:
  return(sqrt(sum((observed-predicted)^2)))
}
```

We can now apply this function to the fitted solution we plotted above.

```
Euclid( observed=philadata[,"pim"], predicted=I.fit)

## [1] 265.6863
```

Since we identified a good fit by eye, we might well wonder if we can do better. Finding the "best fit" is a very tricky problem in general, but let's at least try to see whether we have found the best estimate of $\mathcal{R}_0$ if we assume that the I_0 and the `scale.factor` are well chosen. To that end, let's first define a function that returns the scaled, simulated $I(t)$:

```
get.I.fit <- function(R0, gamma=0.25, I0=1e-6, S0=1-I0,
                      scale.factor=3500) {
  soln <- ode(y=c(x=S0,y=I0),
              times=1:tmax, # save results at observed data times
              func=SIR.vector.field,
              parms=c(beta=R0*gamma,gamma=gamma)
              )
  I.fit <- scale.factor*soln[,"y"]
  return(I.fit)
}
```

Let's now check we get the answer we got above when we apply this function to $\mathcal{R}_0 = 2.4$:

```
I.fit <- get.I.fit(R0=2.4)
Euclid( observed=philadata[,"pim"], predicted=I.fit)
```

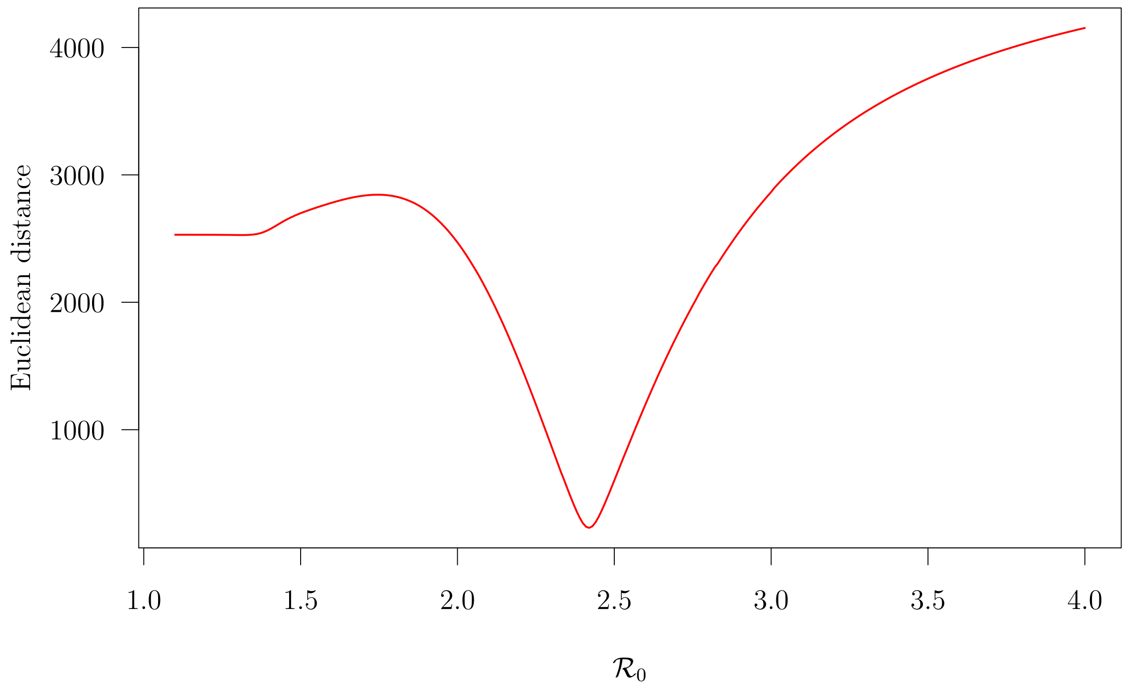
```
## [1] 265.6863
```

Excellent. The function works. Let's now compute the distance for a range of plausible $\mathcal{R}_0$ values:

```
## set the sequence of R0 values we want to check
R0vals <- seq(1.1,4,by=0.01)
## create an empty vector in which we'll store the distances
dist.vec <- c()
## loop over the R0 values and compute the distance for each value,
## saving each computed distance in dist.vec as we go along:
for (R0 in R0vals) {
  dist.vec <- c(dist.vec,
                Euclid( observed=philadata[,"pim"],
                        predicted=get.I.fit(R0)))
}
```

Finally, let's plot the distance as function of $\mathcal{R}_0$:

```
plot(R0vals,dist.vec, typ="l", lwd=2, col="red",
     xlab="$\\R_0$", ylab="Euclidean distance", las=1)
```



It would appear that 2.4 is pretty close to the best value of $\mathcal{R}_0$. To find the very best value, we can ask R to tell us which of the `R0vals` corresponds to the minimum distance:

```
index.of.best.value <- which(dist.vec == min(dist.vec))
best.R0 <- R0vals[ index.of.best.value ]
least.distance <- dist.vec[ index.of.best.value ]
cat("Best estimate: R0 =", best.R0, ",  distance =", least.distance, "\n")

## Best estimate: R0 = 2.42 ,  distance = 229.2044
```

Of course, we could get more precise by using a denser set of `R0vals`. More importantly, we could find the minimum in the higher dimensional parameter space that includes the mean infectious period, `I0` and `scale.factor` as well. That becomes a much more involved and computationally demanding task. □

4 Executive summary for the Public Health Agency

The Public Health Agency of Canada (PHAC) is revising their pandemic plan and has asked your group to summarize what you learned from analyzing the 1918 Philadelphia P&I time series. Besides explaining what inferences you feel you can make from your analysis so far, PHAC wants to know what you would investigate if they were to fund you to continue your work full time for a month. They want a maximum of one page from your group.

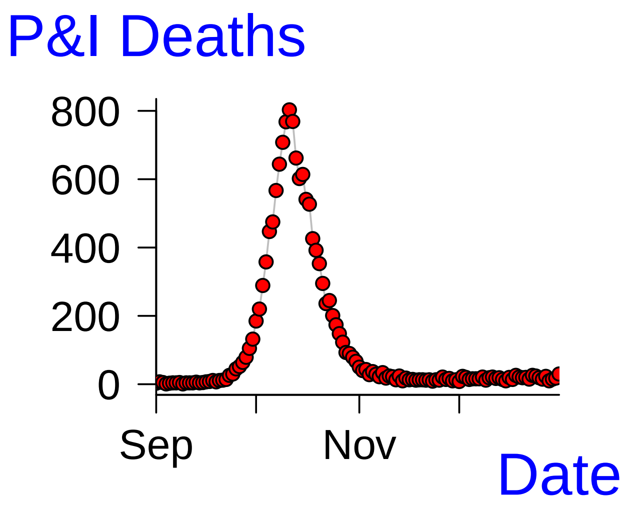
Incidentally, you might be interested to know that rumour has it that all of the members of the pandemic planning committee took Math 2C03 at McMaster University between 1980 and 2003, but they all failed. Also, when the chair of the committee was recently asked "What is a differential equation?" he apparently bent over and vomited (it is hard to know quite what to make of this given that PHAC was investigating a norovirus outbreak at the time).

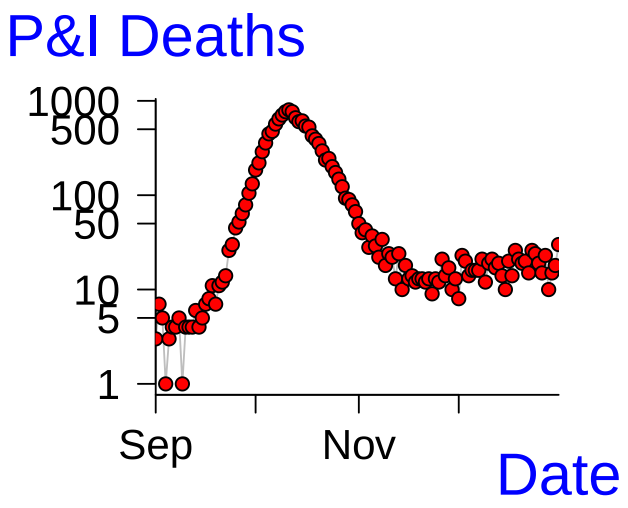
Note: When submitting your assignment solution, it is imperative that the one-page executive summary be printed on its own page. To start a new page in LaTeX, use the `\newpage` *command. Also, as usual, your summary should be in 12 point font. Don't try to cram in as much as possible. Make that page as clear and concise as you can, so that a public health planner can absorb its content quickly and easily.*

See next page for one-page Executive Summary. Note that this summary does not contain any details of the modelling. It summarizes results that are relevant to a public health official needing to make policy decisions.

Executive Summary

We were provided with the daily counts of deaths from pneumonia and influenza (P&I) during the 1918 influenza pandemic in Philadelphia. We found that the temporal pattern of the epidemic had the classic shape that is expected for an invading pathogen (left panel below), though the epidemic curve on a logarithmic scale (right panel below) reveals a rise that can probably be attributed to seasonal pneumonia (unrelated to influenza).





Our primary aim was to estimate the basic reproduction number $\mathcal{R}_0$ (the average number of secondary cases caused by a primary case at the beginning of the epidemic) in Philadelphia in 1918. Given an estimate of $\mathcal{R}_0$, well-established theory allows us to estimate the peak prevalence (percentage of the population infected at the epidemic peak) and the final size (cumulative percentage of the population infected during the entire epidemic). We estimated $\mathcal{R}_0$ in two ways: (i) based strictly on the rate of exponential growth early in the epidemic, which yielded $\mathcal{R}_0$ approximately 2 (implying peak prevalence approximately 15% and final size approximately 80%), and (ii) by fitting an epidemic model to the entire time series, which yielded $\mathcal{R}_0$ approximately 2.4 (implying peak prevalence approximately 22% and final size approximately 88%). While our two point estimates for $\mathcal{R}_0$ differ substantially, we note that both are within the plausible range of $\mathcal{R}_0$ for pandemic influenza (based on other studies in the literature).

If given the opportunity to continue our research on the 1918 influenza pandemic in Philadelphia, we would conduct much more extensive and sophisticated mathematical and statistical analyses that would allow us to estimate $\mathcal{R}_0$ more precisely and to estimate confidence intervals for the results we present. We believe this work would be of long term value for PHAC not only for the purpose of describing the character of the 1918 pandemic, but in order to understand the limitations of parameter estimates based on fitting mechanistic models to epidemic time series. In addition to travel expenses associated with presentations to the pandemic planning committee, the total cost to PHAC would be $40,000 to cover salary and benefits (and coffee) for our group for the month.

— END OF ASSIGNMENT —

Compile time for this document: January 20, 2019 @ 10:33