Analysing the effect of vaccine intervention on IPD incidence in Scotland, 2000-2017

# Copyright and License

**Source:**

This case study was first designed by Lukas Richter and Alexander Spina in 2018 using *R*. It is based on surveillance data from an Scotland.

**Revisions:**

*If you modify this case study, please indicate below your name and changes you made*

**You are free:**

* **to Share** — to copy, distribute and transmit the work
* **to Remix** — to adapt the work

**Under the following conditions:**

* **Attribution** — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work). The best way to do this is to keep as it is the list of contributors: sources, authors and reviewers.
* **Share Alike** — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one. Your changes must be documented. Under that condition, you are allowed to add your name to the list of contributors.
* You cannot sell this work alone but you can use it as part of a teaching.

**With the understanding that:**

* **Waiver** — Any of the above conditions can be waived if you get permission from the copyright holder.
* **Public Domain** — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
* **Other Rights** — In no way are any of the following rights affected by the license:
  + Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
  + The author’s moral rights;
  + Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
* **Notice** — For any reuse or distribution, you must make clear to others the license terms of this work by keeping together this work and the current license.

This licence is based on <http://creativecommons.org/licenses/by-sa/3.0/>

## Objectives of the tutorial

At the end of the tutorial participants should be able to

* create a plan of analysis for simple segmented regression
* use R for exploratory analysis including installing the software, importing a dataset, setting time units for aggregation, data checking, investigating trend and seasonality
* understand how to run a simple negative binomial segmented regression

## Background to the tutorial

The objective of the tutorial is to carry out an analysis on the effect of a vaccine intervention on the incidence of invasive pneumococcal disease (IPD).

There are two available vaccines for IPD, one which covers seven serotypes and one which covers 13 serotypes, called PCV7 and PCV13 respectively.

The dataset is Scottish surveillance data from 2000 to 2017. This includes several years of pre-vaccination data (2000-2006), 3 years of PCV7 years (2007-2009) and 8 years of PCV13 years (2010-2017).

The dataset is case-based (every row is a case) and provides the year and month of notification, the age group, and the serotype.

In addition, there are population counts for Scotland between 2000 and 2017 overall and broken down by age group.

# Question 1

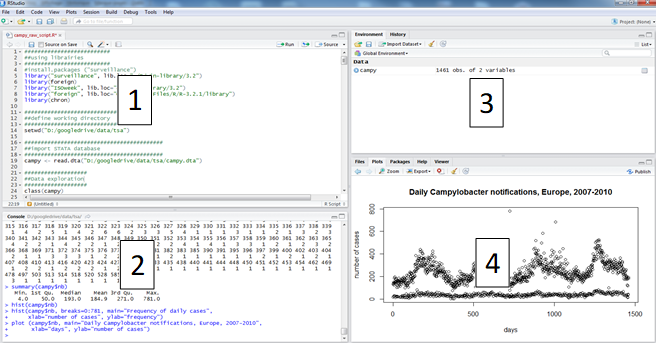
Open the datasets (*ipd\_cases.csv* and *ipd\_population.csv*) in R and briefly explore them (make sure your working directory is set in R). Import the necessary packages and install the corresponding packages. With the assumption that the reporting in the dataset here is stable over time, create an outline of a mini-protocol for the analysis, including hypotheses to be tested, a plan of analysis, with specifications for the dataset, inclusion/exclusion criteria, etc.

# Help Question 1

## R-Studio environment

*R-Studio* is an integrated development environment (IDE) for *R*. It includes a console, syntax highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. You can download it freely from the [*R-Studio* website](https://www.rstudio.com/).

The *RStudio* window comprises the following sections:

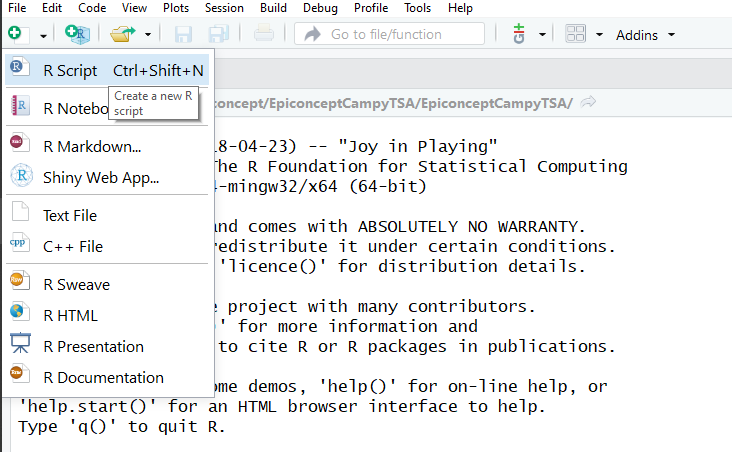


1. A script which is equivalent to a do file in STATA or pgn in EpiInfo.
2. A console where you can write commands and execute them, and where results other than graphs are shown.
3. A “Environment” and “History” windows where you can find the list of R objects being stored and the log of your session, respectively.
4. The “Plots”, “Packages” and “Help” windows, for graphic visualisation, selection of installed packages and help documentation, respectively.

## R-scripts

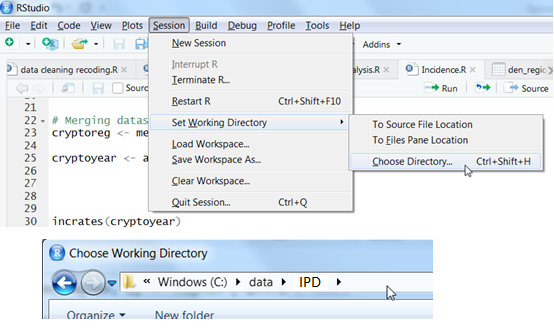
Throughout this case study we strongly encourage you to use *R-scripts* to keep track of the commands you are using to follow your plan of analysis. It will contain data management steps as well as descriptive and statistical analysis.

Open R studio and open a new *R-script*. To open a new *R-script*, click on the plus button on the top left (below File), then select *R-script* from the drop-down menu.



## Working directory

Make sure you have created a working directory on your computer. This is where you store your data and your R scripts and outputs. An example of working directory could be “C:/data/crypto” if you are using a PC. To set your working directory you can use the menu function by clicking “Session” > “Set Working Directory” > “Choose Directory…” and then you can select the appropriate folder.



You could alternatively do this by adding code to your *R-script* (or the command line).

You can check the path for your current working directory using the *getwd* function.

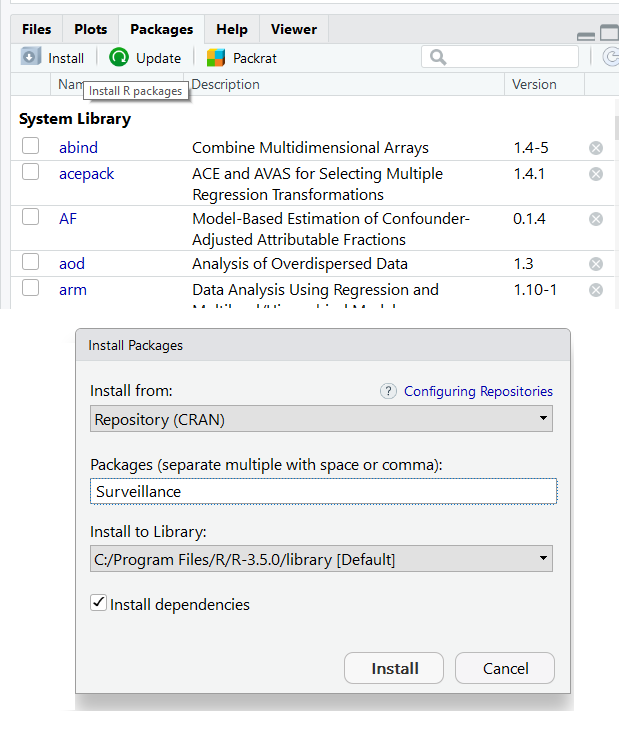
#Check your current working directory  
getwd()

To set your working directory you can use the *setwd* function.

#set working directory   
 #note the forward rather than backwards slashes  
setwd("C:/data/IPD")

## Packages

R comes with a set of default functions. However one of the strengths of R is that many user-written functions exist to help you do your work. To use them, you first need to install the package they belong to. To do this you can click on “Install” on the bottom right in the “Packages” window.



We will mainly use packages which come ready installed with R (base code), but where it makes things easier we will use add-on packages. All the R packages you need for the exercises can be installed over the Internet.

You will need to install these before starting. We will be using the following packages.

* *zoo*: for producing rolling statistics and year-months
* *MASS*: for performing negative binomial regression
* *broom*: for tidying and formatting model outputs

You can also install packages using the *install.packages* function, as below. Do not try to understand this code.

#choose packages required   
required\_packages <- c("zoo", "MASS", "broom")

# Installing required packages for this case study  
for (pkg in required\_packages) {  
 if (!pkg %in% rownames(installed.packages())) {  
 install.packages(pkg, dependencies = TRUE)  
 }  
}

You only need to install packages once. However you will need to load them each time you start a new session using the *library* function.

library(MASS)

Here too, you can load multiple packages at the same time within a loop; again do not try to understand this just yet, just appreciate it is possible.

# Loading required packages for this case study  
for (i in seq(along = required\_packages)) {  
 library(required\_packages[i], character.only = TRUE)  
}

## Reading in data

Import the dataset from a comma seperated file (.csv) using the read.csv function, storing it as a dataframe within *R* called *ipd\_cases*. Also import population data using the same approach (*population*). For a CSV file the separator is normally a comma (“,”), however depending on the language of your operating system this can also be other values, for example some non-english speaking European countries use a semi-colon (“;”) as standard. Here we also specify that we do not want to format strings as factors. When reading population data we additionally define the option check.names = FALSE which makes sure that variable names are kept as defined in the CSV. By default (check.names = TRUE) *R* would automatically change column names to follow certain rules (e.g. no numeric as first character).

# read in your cases dataset  
ipd\_cases <- read.csv("data/ipd\_cases.csv", sep = ",", stringsAsFactors = FALSE)  
  
# read in your population dataset  
population <- read.csv("data/ipd\_population.csv", sep = ",",   
 stringsAsFactors = FALSE, check.names = FALSE)

## Browse data

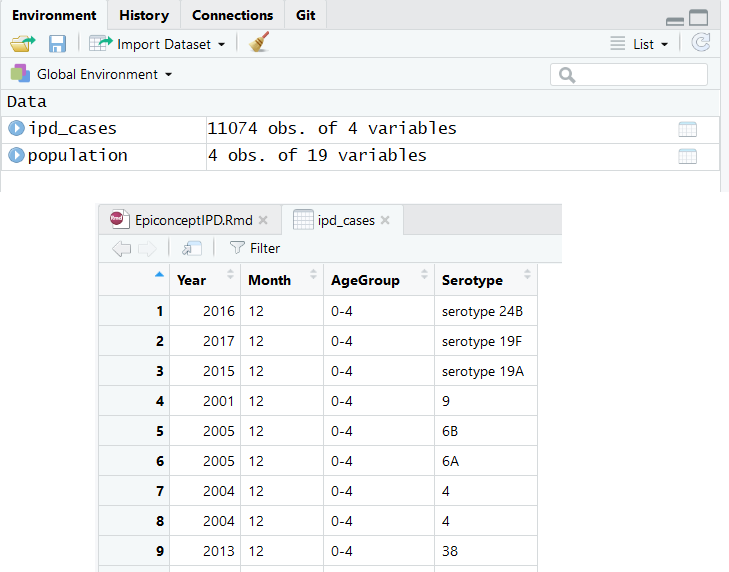
Take a look at the first 10 rows of the data or browse the dataframe in the RStudio preview.

# view the first ten rows of data  
head(ipd\_cases, n = 10)  
  
# view your whole dataset   
View(ipd\_cases)

Remember that View needs to be written with a capital *V*.

|  |  |  |  |
| --- | --- | --- | --- |
| Year | Month | AgeGroup | Serotype |
| 2016 | 12 | 0-4 | serotype 24B |
| 2017 | 12 | 0-4 | serotype 19F |
| 2015 | 12 | 0-4 | serotype 19A |
| 2001 | 12 | 0-4 | 9 |
| 2005 | 12 | 0-4 | 6B |
| 2005 | 12 | 0-4 | 6A |
| 2004 | 12 | 0-4 | 4 |
| 2004 | 12 | 0-4 | 4 |
| 2013 | 12 | 0-4 | 38 |
| 2002 | 12 | 0-4 | 33F |

Alternatively you can view the data by clicking on the data by clicking on the data icon to the right of the view.



Another option is to use *R*’s str function to display the structure of an *R* object compactly.

str(ipd\_cases)

## 'data.frame': 11074 obs. of 4 variables:  
## $ Year : int 2016 2017 2015 2001 2005 2005 2004 2004 2013 2002 ...  
## $ Month : chr "12" "12" "12" "12" ...  
## $ AgeGroup: chr "0-4" "0-4" "0-4" "0-4" ...  
## $ Serotype: chr "serotype 24B" "serotype 19F" "serotype 19A" "9" ...

You get a short overview of the dimensions of the dataset (number of observations, number of variables) and additionally the class of each variable including the first couple of observations.

It is often useful to inspect character and factor variables separately to get to know the used values and labels and find out about potential spelling mistakes and different styles. Use the unique function on each of the character variables to get a list of all unique values in each variable.

# see unique values in your variables  
unique(ipd\_cases$Month)  
unique(ipd\_cases$AgeGroup)  
unique(ipd\_cases$Serotype)

**Examples of hypotheses to be tested could be:**

* Is there a trend or seasonality in the IPD reporting?
* Are there sufficient case counts for monthly regression?
* Has the introduction of thirteen valent vaccine impacted incidence?

**Inclusion/exclusion criteria:** Only records with year of notification will be included (drop records with missing notification year)

**Time unit for analysis:** Months or Years

**Draft plan of analysis:**

**Data cleaning**

* Plot original data
* Data description (frequency distributions, summary tables) of all variables
* Checks for missing data
* Checks for numeric attraction in reporting due to reporting bias, including statistical tests

**Data description**

* To better understand the nature of the data
  + Plot the data
  + Calculate statistical descriptors
  + Moving average (exploring trend and seasonality)
* Decomposition of the dataset
  + Checks for trend (check R2)
  + Checks for seasonality (taking trend into account)
    - Spectral analysis, Periodogram
    - Check R2 of models
  + Test for white noise

**Segmented regression** - Decide whether there are sufficient case counts for segmented regression based on monthly data - Define pre- and post-intervention periods - Carry out negative binomial regression

# Question 2

Make any changes necessary to the dataset. Carry out data checking, checks for missing data. Create new variables that may be necessary.

* Make sure that ipd\_cases$Month only contains values between 1-12 (Jan-Dec)
  + Check if there are cases missing month
    - If there are, make sure these are labelled correctly
* Make sure that ipd\_cases$Serotype categories are uniform
* Define two variables in our data to indicate if cases correspond to PCV7 and/or PCV13 serotypes
* Define a variable which combines year and month

# Help Question 2

Remember in the last section that we looked at unique values in each of our variables. This is a quick and easy way to check the contents of variables. Particularly in our case as we only have a few variables.

# see unique values in your variables  
unique(ipd\_cases$Year)  
unique(ipd\_cases$Month)  
unique(ipd\_cases$AgeGroup)  
unique(ipd\_cases$Serotype)

From this we can see that the years all appear to be in the time frame that we expected and age groups appear to be uniformly reported. However the month variable appears to have missings which have been entered as “#NULL!” and the serotype variable does not appear to be uniformly reported with some cases having “serotype” written out in front of the number.

## Data cleaning

### correcting month variable

Here we use the square brackets to subset rows where month is equal to “#NULL!” and replace the content of the month variable with NA.

# correct the labelling of missing values  
ipd\_cases$Month[ipd\_cases$Month == "#NULL!"] <- NA

The next thing that we might realise is that this is not a numeric variable.

# check the class of month  
class(ipd\_cases$Month)

## [1] "character"

We may want to change this from a character to a numeric variable.

# overwrite month as a numeric variable  
ipd\_cases$Month <- as.numeric(ipd\_cases$Month)

### Correcting serotype variable

In order to get rid of the “serotype” text before the value we can use the *gsub* function. This will search for a specific string of characters in a given variable and then replace that with another string. So here we can search for “serotype” (note the space at the end!) in ipd\_cases$Serotype and replace that with nothing (""), thereby deleting it.

# replace "serotype " with nothing in the serotype variable  
ipd\_cases$Serotype <- gsub(pattern = "serotype ", replacement = "",   
 x = ipd\_cases$Serotype, fixed = TRUE)

## Variable creation

### Assigning cases based on serotype

The PCV7 covers IPD serotypes 4, 6B, 9V, 14, 18C, 19F and 23F. The PCV13 additionally covers serotypes 1, 5, 7F, 3, 6A and 19A. First define which serotypes are in each of the two vaccines. We will use these character strings later to identify cases with either PCV7 or PCV13 serotypes.

# define strings of each vaccine type  
pcv7 <- c("4", "6B", "9V", "14", "18C", "19F", "23F")  
pcv13 <- c("4", "6B", "9V", "14", "18C", "19F", "23F", "1", "5", "7F", "3", "6A", "19A")

To do this we will create two variables, one for each vaccine type. So if the identified serotype is one of the PCV7 serotypes, then that case will be tagged as a PCV7 case.

Define two columns in our data to indicate if the case corresponds to PCV7 and/or PCV13 serotypes, respectively. Values in these columns are either TRUE or FALSE; for most purposes *R* will interpret these variables as 1/0.

In plain english, the code below reads create a new “PCV7” variable and where the serotype is in the “pcv7” string (defined above), return TRUE for that row otherwise return FALSE.

# create a new variable defining cases by serotypes  
ipd\_cases$PCV7 <- ipd\_cases$Serotype %in% pcv7  
ipd\_cases$PCV13 <- ipd\_cases$Serotype %in% pcv13

Also create a third variable which defines serotypes covered by neither vaccine. To do this we use the exclamation mark to designate “not” and the ampersand to combined arguments with an “and”. So this reads, make the new variable NonVacc true where serotype is not in pcv7 and serotype is not in pcv13.

# create new var with other serotypes not covered   
ipd\_cases$NonVacc <- !ipd\_cases$Serotype %in% pcv7 & !ipd\_cases$Serotype %in% pcv13

### Creating a year/month combined variable

It may be useful to combine the year and month variable for later investigation. To do this use the *paste0* function, which simply combines multiple inputs in to one character string. In this case we will be pasting the year, followed by an dash and then the month (e.g. 2011-12). We could then use the *as.yearmon* function from the *zoo* package to order this character variable correctly.

# create a new variable with combined year and month   
ipd\_cases$Year\_Month <- paste0(ipd\_cases$Year, "-", ipd\_cases$Month)  
  
# change this variable in to a year month variable  
ipd\_cases$Year\_Month <- as.yearmon(ipd\_cases$Year\_Month)

# Question 3

Describe the data. Check how many missing values there are. Plot cases by different time units of interest (e.g. by month, quarter and year). Use the “moving average” to explore possible trends and seasonality within your data.

# Help Question 2

## Looking at missings

You will want to start by seeing what missing values there are.

You can see this by combining the *is.na* function with the *table* function.

#check missing values for serotype  
table(is.na(ipd\_cases$Serotype))

##   
## FALSE TRUE   
## 9626 1448

# check missing values for month  
table(is.na(ipd\_cases$Month))

##   
## FALSE TRUE   
## 10229 845

We can see that there are several missing values for both serogroup and month. This is important to keep in mind for later when we are deciding on the time unit of interest.

## Basic descriptives

You may want to start by just looking at simple counts and proportions of your variables using the *table* and *prop.table* functions.

# case counts by age group  
agegrp <- table(ipd\_cases$AgeGroup)  
  
# case counts by year   
counts <- table(ipd\_cases$Year)

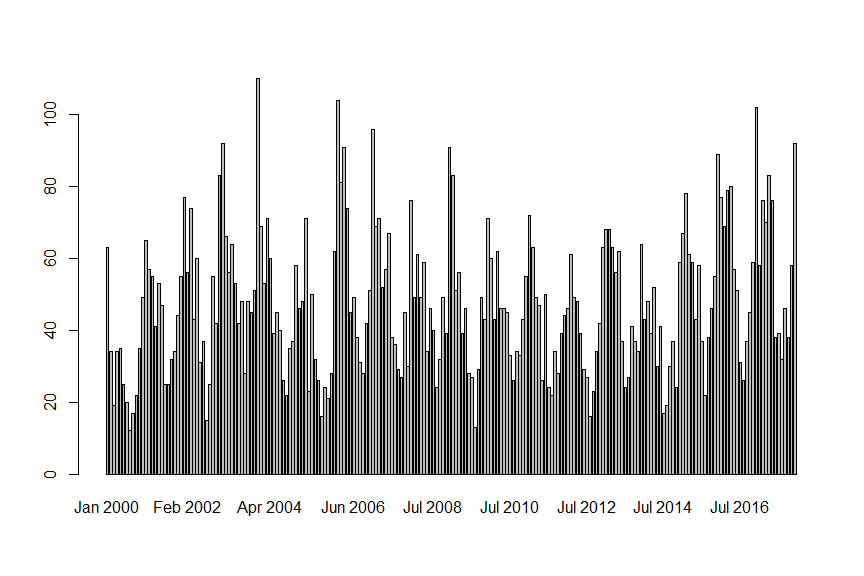
Primarily the older age groups appear to be affected. Annual cases seem to fluctuate somewhat by year though this all appears relatively constant. You could then also break down counts by age group and year.

# counts by age group and year   
table(ipd\_cases$Year, ipd\_cases$AgeGroup)

##   
## 0-4 5-64 65+  
## 2000 57 218 278  
## 2001 70 216 311  
## 2002 60 256 282  
## 2003 88 333 368  
## 2004 91 295 241  
## 2005 98 363 257  
## 2006 67 382 313  
## 2007 26 320 283  
## 2008 34 322 250  
## 2009 39 320 220  
## 2010 28 256 256  
## 2011 36 246 217  
## 2012 35 208 239  
## 2013 36 263 285  
## 2014 26 192 221  
## 2015 53 282 318  
## 2016 46 333 334  
## 2017 42 294 370

When looking at the number of cases per month, a table might be too large to get an oversight. So you may want to plot your cases by month, combining *table* and *barplot*.

# get counts by year and month  
monthcounts <- table(ipd\_cases$Year\_Month)  
  
# make a barplot   
barplot(monthcounts)



This plot highlights two things. First is that there appears to be some seasonality, which is worth investigating. Second is that there are consistently less than 100 cases per month, which will become important later when considering time units for regression.

# Seasonality and moving averages

Start by creating a timeseries object from your monthly variable. To do this we first need to use our monthly counts from above and the *ts* function. We can then define starts and ends to our series as well as the frequency we are interested in (in our case 12 is monthly).

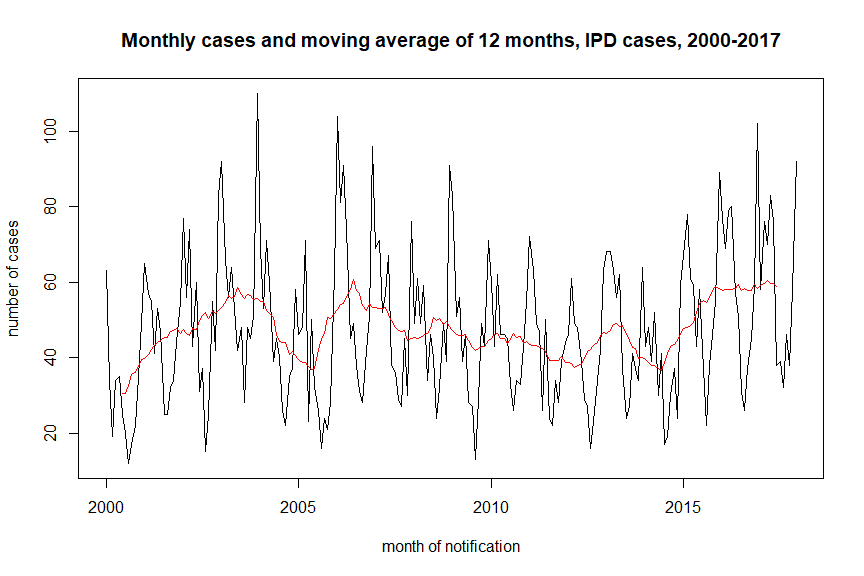
ipd\_ts <- ts(monthcounts, start = c(2000,1), end = c(2017, 12), frequency = 12)

We now want to use moving averages to investigate possible trends and seasonality. We can calculate these using the *rollapply* function. We will calculate the centred moving average for several different time windows. Try for example to do one each for 12 months, 6 months and 3 months.

# calculate annual moving average  
ma12 <- rollapply(ipd\_ts, width = 12, FUN = mean)

We can then plot this using the *ts.plot* function.

# plot your timeseries with annual moving average  
ts.plot(ipd\_ts, ma12,   
 main = "Monthly cases and moving average of 12 months, IPD cases, 2000-2017",   
 ylab = "number of cases",   
 xlab = "month of notification", col = c("black", "red"))

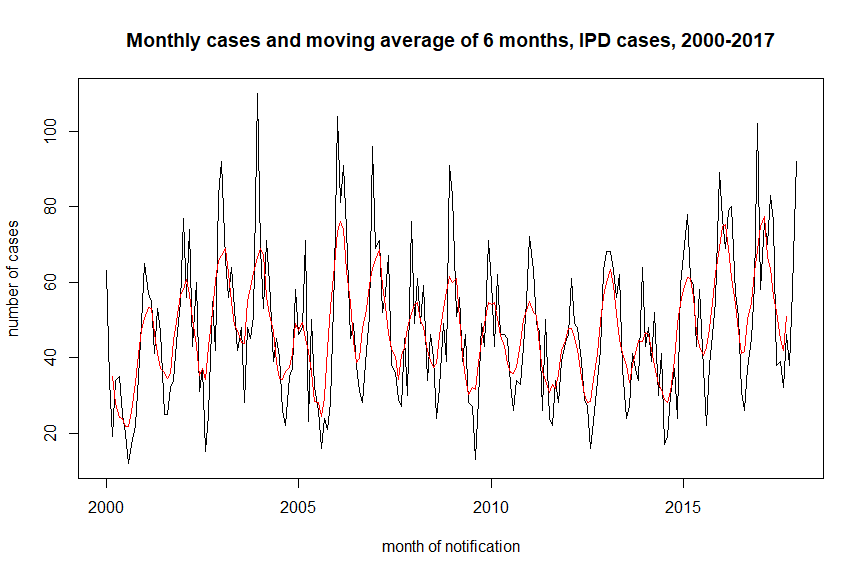


Now try with a 6 month moving window.

# calculate annual moving average  
ma6 <- rollapply(ipd\_ts, width = 6, FUN = mean)

We can again plot this using the *ts.plot* function.

# plot your timeseries with annual moving average  
ts.plot(ipd\_ts, ma6,   
 main = "Monthly cases and moving average of 6 months, IPD cases, 2000-2017",   
 ylab = "number of cases",   
 xlab = "month of notification", col = c("black", "red"))

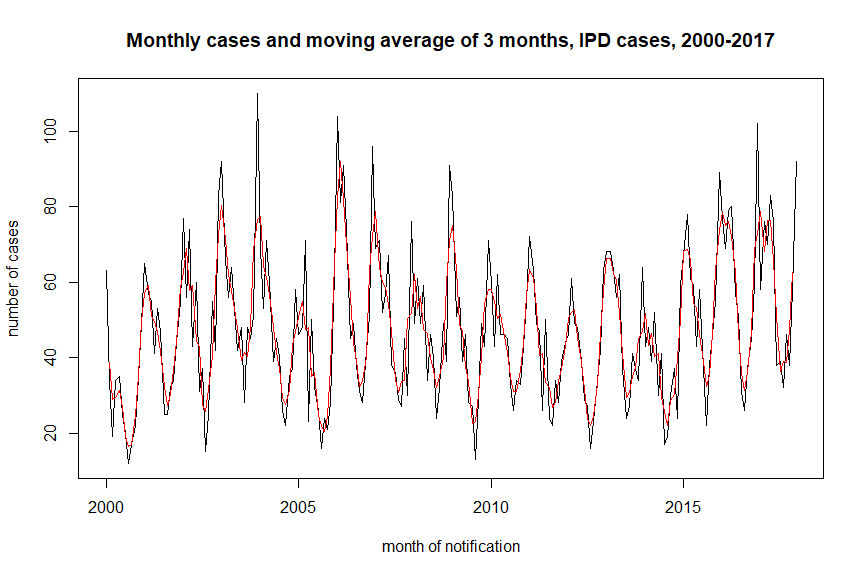


Now try with a 3 month moving window.

# calculate annual moving average  
ma3 <- rollapply(ipd\_ts, width = 3, FUN = mean)

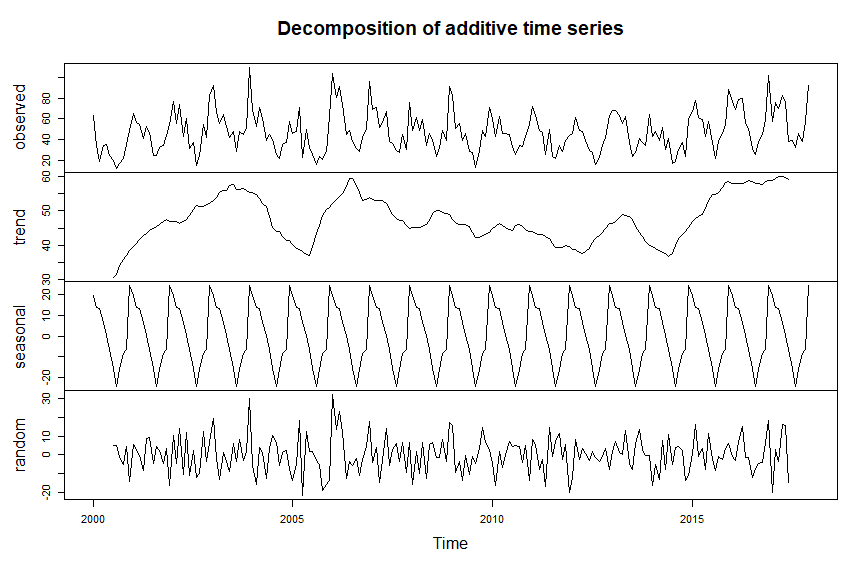
We can again plot this using the *ts.plot* function.

# plot your timeseries with annual moving average  
ts.plot(ipd\_ts, ma3,   
 main = "Monthly cases and moving average of 3 months, IPD cases, 2000-2017",   
 ylab = "number of cases",   
 xlab = "month of notification", col = c("black", "red"))



It is also possible to decompose a time series automatically into seasonal, trend and irregular components using moving averages with the *decompose* function.

plot(decompose(ipd\_ts))



A clear trend cannot be observed. It appears to fluctuate a bit, however there does appear to be a decrease in cases after the vaccine introduction in 2007 but this seems to pick up again after 2013. A strong seasonal trend can be observed in the winter months, which is consistent with published findings of IPD seasonality.

It may make things clearer to first look at regressions with annual counts. This is for a number of reasons for this; no clear trend, seasonality stable and low monthly case numbers (as well as missing notification months) means that looking at annual counts without correcting for seasonality may give more robust results. If absolutely necessary it may be worth considering moving on to quarterly counts.

# Question 3

Before moving on to segmented regression, consider looking at counts and plots based on the vaccine serogrouping variables we created previously.

# Help question 3

## Producing counts by age group and year

First get counts for the overall group by year and age group. Remember that in a previous question we produced this using the *table* function. The *table* function, however, does not provide column or row sums. In order to do this, we can use the *addmargins* function in combination with *table*. We add our table of counts, then specify that we are interested in row sums (margin = 2),

# tabulate cases by year and age-group.  
annual\_cases <- addmargins(table(ipd\_cases$Year, ipd\_cases$AgeGroup),   
 margin = 2, FUN = sum)  
  
# rename the sum column   
 # colnames where colname is "sum" change to "Total"  
colnames(annual\_cases)[colnames(annual\_cases) == "sum"] <- "Total"

An alternative to having to rename the “sum” column above, would be to instead name the column immediately when calling the function by typing: FUN = list(Total = sum)

# view your counts table  
annual\_cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0-4 | 5-64 | 65+ | Total |
| 2000 | 57 | 218 | 278 | 553 |
| 2001 | 70 | 216 | 311 | 597 |
| 2002 | 60 | 256 | 282 | 598 |
| 2003 | 88 | 333 | 368 | 789 |
| 2004 | 91 | 295 | 241 | 627 |
| 2005 | 98 | 363 | 257 | 718 |
| 2006 | 67 | 382 | 313 | 762 |
| 2007 | 26 | 320 | 283 | 629 |
| 2008 | 34 | 322 | 250 | 606 |
| 2009 | 39 | 320 | 220 | 579 |
| 2010 | 28 | 256 | 256 | 540 |
| 2011 | 36 | 246 | 217 | 499 |
| 2012 | 35 | 208 | 239 | 482 |
| 2013 | 36 | 263 | 285 | 584 |
| 2014 | 26 | 192 | 221 | 439 |
| 2015 | 53 | 282 | 318 | 653 |
| 2016 | 46 | 333 | 334 | 713 |
| 2017 | 42 | 294 | 370 | 706 |

## Producing counts by serotype groupings, age group and year

We can do that same calculation as above, however subsetting our counts for the different serotype groupings of interest. Remember that we created variables for PCV7, PCV13 and NonVacc serotypes, which were all TRUE/FALSE variables.

We subset using the square brackets, and only need to specify the variable of interest - as only those rows with TRUE will be counted.

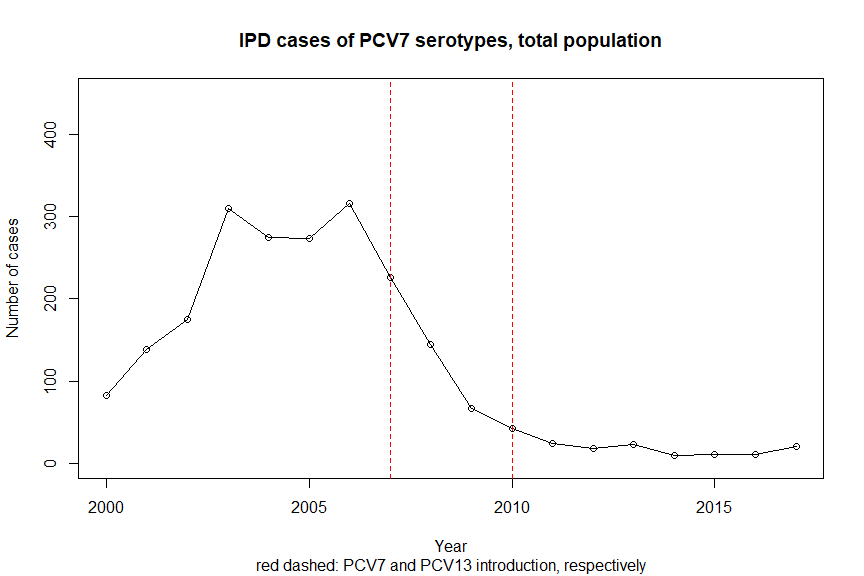
# create annual counts of PCV7 serotype cases  
annual\_pcv7 <- addmargins(table(ipd\_cases$Year[ipd\_cases$PCV7],   
 ipd\_cases$AgeGroup[ipd\_cases$PCV7]),   
 margin = 2, FUN = list(Total = sum))  
  
# create annual counts of PCV13 serotype cases  
annual\_pcv13 <- addmargins(table(ipd\_cases$Year[ipd\_cases$PCV13],   
 ipd\_cases$AgeGroup[ipd\_cases$PCV13]),   
 margin = 2, FUN = list(Total = sum))  
  
# create annual counts of non vaccination serotype cases  
annual\_nonvacc <- addmargins(table(ipd\_cases$Year[ipd\_cases$NonVacc],   
 ipd\_cases$AgeGroup[ipd\_cases$NonVacc]),   
 margin = 2, FUN = list(Total = sum))

## Plot counts by vaccine serotype group

We can then use these annual counts to plot the various serotype groupings. We can also highlight years with PCV7/13 introduction (as red dashed vertical lines). Remember, PCV7 was introduced in 2007, and PCV13 in 2010. In our annual counts tables, the years are in the row names (i.e. not a column within the table). So within plot we specify that our x-variable is years (i.e. the rownames of our counts table). We then specify that our y-variable should be the “Total” column from our counts table. We specify that we want a line graph and put our various axis, main and sub labels as well as set the y-axis minimum and maximum (ylim). We could then add *points* to our graph specifying the x and y variables in the same way as above. Finally, we can add vertical lines by specifying where we would like them on the x axis (2007 and 2010), the colour and the line type we would like.

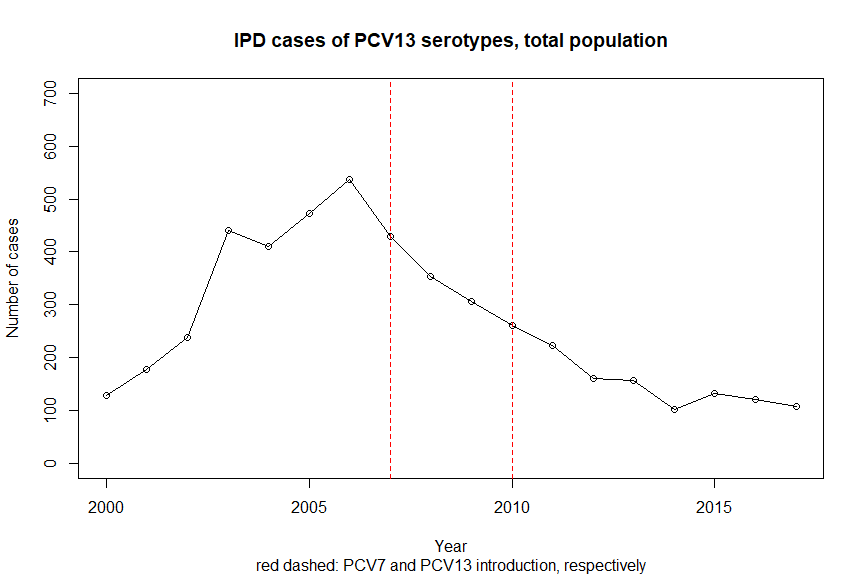
**PCV7:**

# plot your annual counts as a line graph  
plot(rownames(annual\_pcv7), annual\_pcv7[, "Total"],   
 type = "l",  
 xlab = "Year", ylab = "Number of cases",  
 main = "IPD cases of PCV7 serotypes, total population",  
 sub = "red dashed: PCV7 and PCV13 introduction, respectively",  
 ylim = c(0, 450))  
  
# add points to your line graph   
points(rownames(annual\_pcv7), annual\_pcv7[, "Total"])  
  
# add vertical lines to designate vaccine introduction  
abline(v = c(2007, 2010), col = "red", lty = "dashed")



**PCV13:** We can then do the same thing for PCV13 serotypes, adjusting the y axis accordingly.

# plot your annual counts as a line graph  
plot(rownames(annual\_pcv13), annual\_pcv13[, "Total"],   
 type = "l",  
 xlab = "Year", ylab = "Number of cases",  
 main = "IPD cases of PCV13 serotypes, total population",  
 sub = "red dashed: PCV7 and PCV13 introduction, respectively",  
 ylim = c(0, 700))  
  
# add points to your line graph   
points(rownames(annual\_pcv13), annual\_pcv13[, "Total"])  
  
# add vertical lines to designate vaccine introduction  
abline(v = c(2007, 2010), col = "red", lty = "dashed")

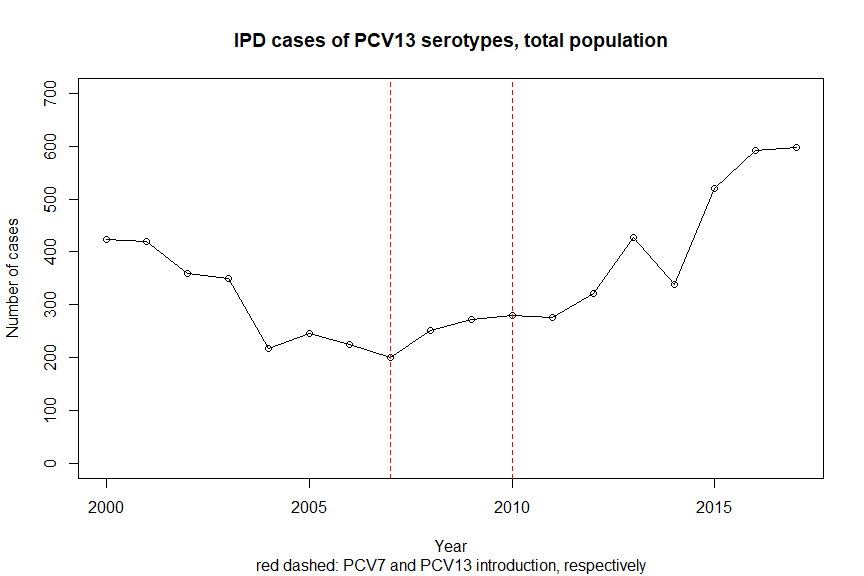


The two graphs above show that there was a strong reduction after in the number of annual cases with the serotypes covered by the vaccines.

**NonVacc:**

We can then repeat for non vaccination serotypes.

# plot your annual counts as a line graph  
plot(rownames(annual\_nonvacc), annual\_nonvacc[, "Total"],   
 type = "l",  
 xlab = "Year", ylab = "Number of cases",  
 main = "IPD cases of PCV13 serotypes, total population",  
 sub = "red dashed: PCV7 and PCV13 introduction, respectively",  
 ylim = c(0, 700))  
  
# add points to your line graph   
points(rownames(annual\_nonvacc), annual\_nonvacc[, "Total"])  
  
# add vertical lines to designate vaccine introduction  
abline(v = c(2007, 2010), col = "red", lty = "dashed")

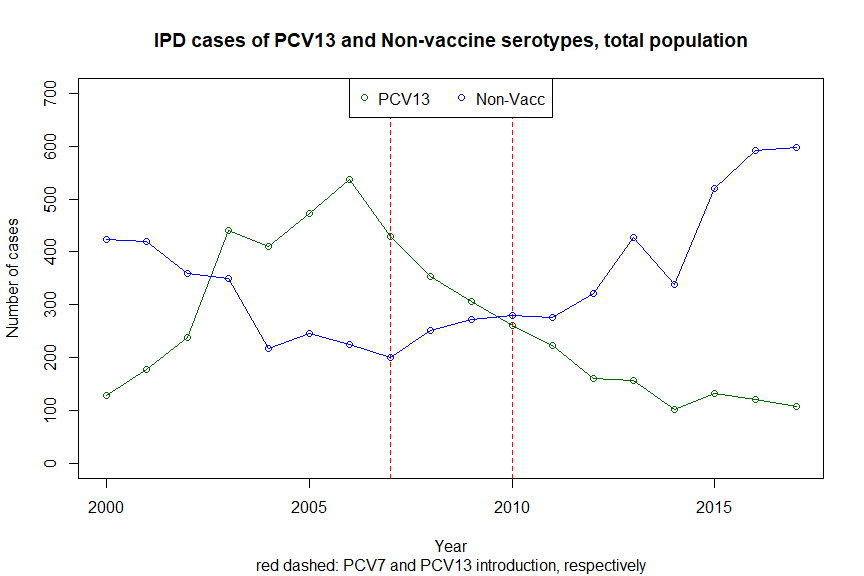


In contrast to the previous plots, this shows us that the annual cases with serotypes not covered by the vaccine acctually increased after their introduction.

## Combining plots for comparison

We can combine the plots for PCV13 and non vaccine types in order to get a clearer understanding of the relation. To do this plot your PCV13 plot as above, then add in lines and points for non-vaccination serotypes using the *lines* and *points* functions. You can then also add a legend using *legend*.

# plot PCV13 annual counts as a line graph  
plot(rownames(annual\_pcv13), annual\_pcv13[, "Total"],  
 type = "l",   
 xlab = "Year", ylab = "Number of cases",  
 main = "IPD cases of PCV13 and Non-vaccine serotypes, total population",  
 sub = "red dashed: PCV7 and PCV13 introduction, respectively",  
 ylim = c(0, 700), col = "dark green")  
  
# add points for pcv13  
points(rownames(annual\_pcv13), annual\_pcv13[, "Total"], col = "dark green")  
  
# add vertical lines for intervention   
abline(v = c(2007, 2010), col = "red", lty = "dashed")  
  
  
# add line to graph for non vaccine serotypes  
lines(rownames(annual\_nonvacc), annual\_nonvacc[, "Total"], col = "blue")  
# add points to graph for non vaccine serotypes  
points(rownames(annual\_nonvacc), annual\_nonvacc[, "Total"], col = "blue")  
  
# add a horizontal legend  
 # pch specifies that you want coloured dots  
legend("top", horiz = TRUE,   
 legend = c("PCV13", "Non-Vacc"),   
 col = c("dark green", "blue"), pch = 1)



The above figure quite nicely shows that non vaccination serotypes increased while the PCV13 serotypes decreased concurrently after the introduction of vaccine in 2007. This is quite a neat demonstration of serotype replacement, which has been widely documented in literature. This may be something that should be investigated in regression analysis.

# Question 5

Run a segmented negative binomial regression model investigating the effect of vaccination on the incidence rates of cases with PCV13 serotypes.

# Help question 5

## Segmented regression for PCV13 serotypes

We can use segmented regression to assess the impact of a certain interventions, such as the introduction of a vaccine programme. You can think of it as the data being devided into two segments (or periods), before and after the intervention, where regression models are estimated for both time periods. To do this, you could split the time series in two segments and run regression models for each. However, the advantage of segmented regression is that estimates for both segments come from the same model and therefore it is easy to calculate the effect of the intervention (which can loosely be defined as the difference of the two segments) and confidence intervals directly from the model.

We will need the glm.nb function from the *MASS* package to perform negative binomial regression. Additionally we will use the tidy function from the *broom* package to create nice model outputs.

In general, the negative binomial regression model can be written as follows.

Here, is the number of IPD cases observed in year ; is the population size in 100,000s in year .  
 is the last year (7th year) of the pre-period (including the transition time if there is any), is the indicator function (it is 0 if and 1 if ), is the cut off operator (it is if and 0 otherwise), and denotes the residual. This means, that is the additional generalised linear part of the post-period, which is zero in the pre-period. The estimates of and will denote the effects of the intervention. Of course more periods and terms could be added, if needed, in order to control for seasonality.

In the following we will analyse annual incidence of PCV13 serotypes in the overall population (*total*). We will define 2007 as the time of intervention, this is when PCV7 vaccination began. We choose this because PCV7 serotypes are also covered by PCV13 and we already see an impact on PCV13 serotypes from 2007 onwards. Analoguously, sub-populations and other intervention definitions can be analysed. However, our observed number of cases is not high enough for this type of stratification. To obtain more robust results a certain number of time points per period and a certain number of observations per time point is required. It is desirable to have 12 time points per period (e.g. 12 years pre- and post-intervation; not fulfilled in our example) and *at least* 100 observations (cases) per time point. It must be said however that these numbers are rough suggestions and not derived power estimates.

First, transform the table (*annual\_pcv13*) into a data.frame. Use the *as.data.frame.matrix* function to do so. Check the class of *annual\_pcv13*:

# check class of object   
class(annual\_pcv13)

## [1] "table" "matrix"

In our case, the year is stored as row name and not as a column itself. Therefore transform the data into a data.frame as explained above and add the year as seperate column. Make sure that the year column is of class numeric.

# make counts table in to a data.frame  
pcv13\_aggr <- as.data.frame.matrix(annual\_pcv13)  
  
# make years in to a variable   
 # take the rownames and make them numeric  
pcv13\_aggr$year <- as.numeric(rownames(pcv13\_aggr))

We then need to merge our population data to be able to calculate incidences. To do this we use the *merge* function. We use “year” as our unique identifier in both datasets, and specify that pcv13\_aggr is our main dataset so only the rows of that dataset should be considered (no new additions from population rows; equivalent to a left-join).

Once merged we can subset and only keep the columns of interest to us. Using square brackets.

# merge counts with population   
pcv13\_df <- merge(pcv13\_aggr, population, by = "year", all.x = TRUE)  
  
# only keep relevant columns  
pcv13\_df <- pcv13\_df[, c("year", "Total", "Total\_pop")]

We then want to define the parameters and columns needed for segmented regression.

# year when vaccination with PCV7 started  
intervention\_year <- 2007  
  
# corresponds to t in the formula  
 # count of years  
pcv13\_df$linear <- 1:nrow(pcv13\_df)   
  
# corresponds to delta(t-t0) in the formula  
 # pre or post vaccination period  
pcv13\_df$const\_vacc <- as.numeric(pcv13\_df$year >= intervention\_year)   
  
# corresponds to (t-t0)^+ in the formula  
 # count of years post intervention  
pcv13\_df$lin\_vacc <- pmax(0, pcv13\_df$year - intervention\_year + 1)   
  
# population in 100,000s  
pcv13\_df$pop100 <- pcv13\_df$Total\_pop / 100000

We can then pass these parameters through the *glm.nb* function of the *MASS* package. We specify our dataset, as well as the “Total” variable (with annual counts) and our parameters from above. This gives us a segmented negative binomial regression with population per 100,000 as offset.

# negative binomial segmented regression  
nb <- glm.nb(Total ~ linear + const\_vacc + lin\_vacc + offset(log(pop100)), data = pcv13\_df)

## Interpretation of intervention effects

Create table with results. Use the tidy function from *broom* package to simplify the regression output.

# create a table with results  
nb\_tidy <- tidy(nb, exponentiate = FALSE, conf.int = TRUE)

Take a look at your results.

# view output  
nb\_tidy

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| (Intercept) | 0.8718519 | 0.1256272 | 6.939992 | 0.0000000 | 0.6180475 | 1.1304593 |
| linear | 0.2353922 | 0.0272885 | 8.626063 | 0.0000000 | 0.1792278 | 0.2918999 |
| const\_vacc | -0.321926 | 0.1331916 | -2.417014 | 0.0156484 | -0.5865459 | -0.0590232 |
| lin\_vacc | -0.386883 | 0.0308973 | -12.521584 | 0.0000000 | -0.4498154 | -0.3242605 |

We then need to exponentiate the estimates to make them interpretable as Incidence Rate Ratios. This makes the intercept equal to the estimate of baseline incidence per 100,000.

# exponentiate results for IRRs  
nb\_tidy\_exp <- tidy(nb, exponentiate = TRUE, conf.int = TRUE)

# view results  
nb\_tidy\_exp

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| (Intercept) | 2.3913352 | 0.1256272 | 6.939992 | 0.0000000 | 1.8553019 | 3.0970787 |
| linear | 1.2654050 | 0.0272885 | 8.626063 | 0.0000000 | 1.1962932 | 1.3389690 |
| const\_vacc | 0.7247518 | 0.1331916 | -2.417014 | 0.0156484 | 0.5562453 | 0.9426849 |
| lin\_vacc | 0.6791706 | 0.0308973 | -12.521584 | 0.0000000 | 0.6377459 | 0.7230619 |

The percentage change of the anual incidence by year can be computed using the formula .

Therefore, after the intervention, the anual incidence changed by , after controlling for the baseline trend.

## Plot results

We may then want to visualse our results. To do this we first need to calculate model predictions of incidence and confidence intervals using the *predict* function.

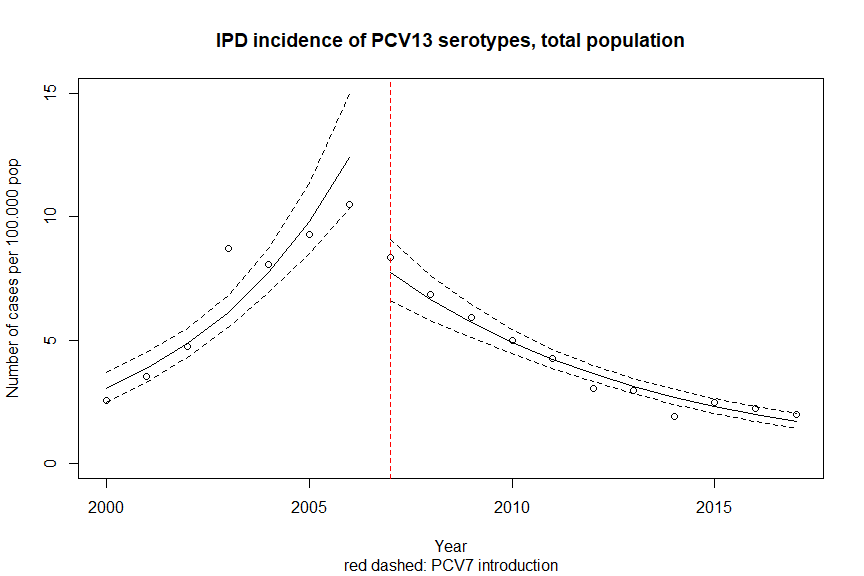
# store new dataset for fit estimation   
nb\_predict <- pcv13\_df  
nb\_predict$pop100 <- 1  
  
# pass fitted model and baseline through predict to calculate standard error  
pred.ci <- predict(nb, nb\_predict, type = "link", se.fit = TRUE)  
  
# use standard errors of model fit to calculate CIs  
pred.ci <- within(pred.ci, {  
 inci <- exp(fit)  
 LL <- exp(fit - 1.96 \* se.fit)  
 UL <- exp(fit + 1.96 \* se.fit)  
})  
  
# create a variable for observed incidence rate  
nb\_predict$inci\_observed <- nb\_predict$Total / nb\_predict$Total\_pop \* 100000  
  
# create variables for modelled incidence with CIs  
nb\_predict$inci\_modelled <- pred.ci$inci  
nb\_predict$lowerCI <- pred.ci$LL  
nb\_predict$upperCI <- pred.ci$UL

Once we have calculated these values and incorporated them in to our data frame, we want to split this in two. One data.frame for the pre vaccine period and one for the post vaccine period.

# split into 2 data.frames for plotting (pre/post)  
pre <- nb\_predict[nb\_predict$year < intervention\_year, ]  
post <- nb\_predict[nb\_predict$year >= intervention\_year, ]

We can then pass all these values through the *plot* function. We start by plotting our observed incidence by year as points. Then we add lines for the modelled incidence in the pre and post vaccine periods with their respective confidence intervals.

# plot observed incidence by year as points  
plot(nb\_predict$year, nb\_predict$inci\_observed, type = "p",  
 xlab = "Year", ylab = "Number of cases per 100.000 pop",  
 main = "IPD incidence of PCV13 serotypes, total population",  
 sub = "red dashed: PCV7 introduction",  
 xlim = range(nb\_predict$year),  
 ylim = c(0, 15))  
  
# plot lines for modelled incidence (with CIs) in pre period  
lines(pre$year, pre$inci\_modelled, type = "l")  
lines(pre$year, pre$lowerCI, type = "l", lty = "dashed")  
lines(pre$year, pre$upperCI, type = "l", lty = "dashed")  
  
# plot lines for modelled incidence (with CIs) in post period  
lines(post$year, post$inci\_modelled, type = "l")  
lines(post$year, post$lowerCI, type = "l", lty = "dashed")  
lines(post$year, post$upperCI, type = "l", lty = "dashed")  
  
# add a vertical line to denote vaccine intervention year  
abline(v = c(2007), col = "red", lty = "dashed")



# Question 6

Based on the findings from descriptive analysis, it may be interesting to look at serotype replacement. Run a segmented negative binomial regression model investigating the effect of vaccination on the incidence rates of all IPD cases.

# Help question 6

## Segmented regression for all serotypes

In the following we will analyse annual incidence of all serotypes in the overall population (*total*). We will define 2007 as the time of intervention.

First, transform the table (*annual\_cases*) into a data.frame. Use the *as.data.frame.matrix* function to do so.

In our case, the year is stored as row name and not as a column itself. Therefore transform the data into a data.frame as explained above and add the year as seperate column. Make sure that the year column is of class numeric.

# make counts table in to a data.frame  
cases\_aggr <- as.data.frame.matrix(annual\_cases)  
  
# make years in to a variable   
 # take the rownames and make them numeric  
cases\_aggr$year <- as.numeric(rownames(cases\_aggr))

We then need to merge our population data to be able to calculate incidences. To do this we use the *merge* function. We use “year” as our unique identifier in both datasets, and specify that cases\_aggr is our main dataset so only the rows of that dataset should be considered (no new additions from population rows; equivalent to a left-join).

Once merged we can subset and only keep the columns of interest to us. Using square brackets.

# merge counts with population   
cases\_df <- merge(cases\_aggr, population, by = "year", all.x = TRUE)  
  
# only keep relevant columns  
cases\_df <- cases\_df[, c("year", "Total", "Total\_pop")]

We then want to define the parameters and columns needed for segmented regression.

# year when vaccination with PCV7 started  
intervention\_year <- 2007  
  
# corresponds to t in the formula  
 # count of years  
cases\_df$linear <- 1:nrow(cases\_df)   
  
# corresponds to delta(t-t0) in the formula  
 # pre or post vaccination period  
cases\_df$const\_vacc <- as.numeric(cases\_df$year >= intervention\_year)   
  
# corresponds to (t-t0)^+ in the formula  
 # count of years post intervention  
cases\_df$lin\_vacc <- pmax(0, cases\_df$year - intervention\_year + 1)   
  
# population in 100,000s  
cases\_df$pop100 <- cases\_df$Total\_pop / 100000

We can then pass these parameters through the *glm.nb* function of the *MASS* package. We specify our dataset, as well as the “Total” variable (with annual counts) and our parameters from above. This gives us a segmented negative binomial regression with population per 100,000 as offset.

# negative binomial segmented regression  
nb\_total <- glm.nb(Total ~ linear + const\_vacc + lin\_vacc + offset(log(pop100)), data = cases\_df)

## Interpretation of intervention effects

Create table with results. Use the tidy function from *broom* package to simplify the regression output and exponentiated the estimates to make them interpretable as Incidence Rate Ratios. This makes the intercept equal to the estimate of baseline incidence per 100,000.

# exponentiate results for IRRs  
nb\_total\_tidy\_exp <- tidy(nb\_total, exponentiate = TRUE, conf.int = TRUE)

# view results  
nb\_total\_tidy\_exp

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| (Intercept) | 10.759791 | 0.1028748 | 23.094249 | 0.0000000 | 8.8051967 | 13.2191878 |
| linear | 1.0486175 | 0.0229109 | 2.072050 | 0.0382608 | 1.0021630 | 1.0972686 |
| const\_vacc | 0.7032642 | 0.1140654 | -3.086148 | 0.0020277 | 0.5634297 | 0.8766768 |
| lin\_vacc | 0.9609599 | 0.0256958 | -1.549769 | 0.1211969 | 0.9137743 | 1.0105509 |

The percentage change of the anual incidence by year can be computed using the formula .

Therefore, after the intervention, the anual incidence changed by , after controlling for the baseline trend.

This means that there was no significant reduction in the overall incidence of IPD as a result of vaccine introduction.

## Plot results

Just to confirm, take a look at the visualisation of results. Calculate model predictions of incidence and confidence intervals using the *predict* function.

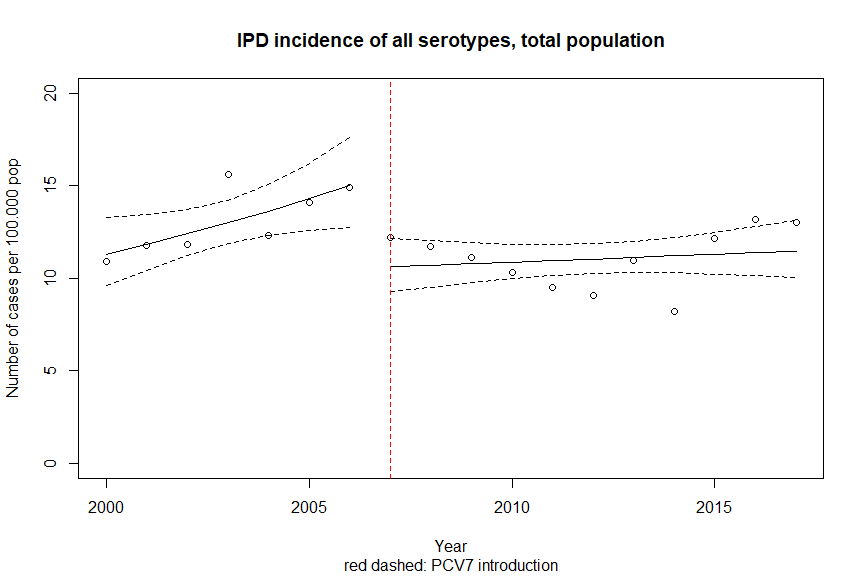
# store new dataset for fit estimation   
nb\_total\_predict <- cases\_df  
nb\_total\_predict$pop100 <- 1  
  
# pass fitted model and baseline through predict to calculate standard error  
pred.ci.total <- predict(nb\_total, nb\_total\_predict, type = "link", se.fit = TRUE)  
  
# use standard errors of model fit to calculate CIs  
pred.ci.total <- within(pred.ci.total, {  
 inci <- exp(fit)  
 LL <- exp(fit - 1.96 \* se.fit)  
 UL <- exp(fit + 1.96 \* se.fit)  
})  
  
# create a variable for observed incidence rate  
nb\_total\_predict$inci\_observed <- nb\_total\_predict$Total / nb\_total\_predict$Total\_pop \* 100000  
  
# create variables for modelled incidence with CIs  
nb\_total\_predict$inci\_modelled <- pred.ci.total$inci  
nb\_total\_predict$lowerCI <- pred.ci.total$LL  
nb\_total\_predict$upperCI <- pred.ci.total$UL

Split your data.frame in two; on for pre vaccine period and one for post vaccine period.

# split into 2 data.frames for plotting (pre/post)  
pre\_total <- nb\_total\_predict[nb\_total\_predict$year < intervention\_year, ]  
post\_total <- nb\_total\_predict[nb\_total\_predict$year >= intervention\_year, ]

We can then pass all these values through the *plot* function. We start by plotting our observed incidence by year as points. Then we add lines for the modelled incidence in the pre and post vaccine periods with their respective confidence intervals.

# plot observed incidence by year as points  
plot(nb\_total\_predict$year, nb\_total\_predict$inci\_observed, type = "p",  
 xlab = "Year", ylab = "Number of cases per 100.000 pop",  
 main = "IPD incidence of all serotypes, total population",  
 sub = "red dashed: PCV7 introduction",  
 xlim = range(nb\_total\_predict$year),  
 ylim = c(0, 20))  
  
# plot lines for modelled incidence (with CIs) in pre period  
lines(pre\_total$year, pre\_total$inci\_modelled, type = "l")  
lines(pre\_total$year, pre\_total$lowerCI, type = "l", lty = "dashed")  
lines(pre\_total$year, pre\_total$upperCI, type = "l", lty = "dashed")  
  
# plot lines for modelled incidence (with CIs) in post period  
lines(post\_total$year, post\_total$inci\_modelled, type = "l")  
lines(post\_total$year, post\_total$lowerCI, type = "l", lty = "dashed")  
lines(post\_total$year, post\_total$upperCI, type = "l", lty = "dashed")  
  
# add a vertical line to denote vaccine intervention year  
abline(v = c(2007), col = "red", lty = "dashed")



# Going further

Based on these findings it might be interesting to consider what further analyses would be informative. One example would be to look at the impact in different age groups. Unfortunately the number of cases is too small for this kind of stratified analysis, and you would face issues such as the model not converging and needing to increase the number of iterations. Can you think of other options?