Time Series and Longitudinal Analysis

Richard White

2018-11-08

Contents

Preface

When dealing with data measured over time, there are two kinds of analyses that can be performed.

"Time series" analyses generally deal with one variable (the outcome). We can then either:

1. Predict the future only using the previous observations. E.g. predict tomorrow's temperature, using today's and yesterday's temperature as exposures. We will not be focusing on these kinds of analyses.
2. Estimate descriptive statistics about the data. E.g. Today's data is much higher than expected (outbreak?). We will focus on these kinds of analyses.

If we have more than one variable measured over time (e.g. outcome and an exposure) then we can run regression analyses. E.g. seeing how the number of tuberculosis patients (outcome) is affected by the number of immigrants to Norway (exposure) over a 20 year period. We will focus on these kinds of analyses.

It is important to note that if we define our exposure as "time" then we can use the regression framework to estimate descriptive statistics about the data. This means we can use the same regression framework for the two kinds of analyses we will be focusing on.

The "regression framework" is very similar to ordinary regressions that you have been working with for many years. The only difference is that some of the data **may** have more advanced data structures that your normal methods cannot handle.

Chapter 1

Definitions and Scenarios

1.1 Panel Data

Panel data is a set of data with measurements repeated at equally spaced points. For example, number of influenza cases recorded every day, or every week, or every year would be considered panel data. The number of influenza cases on Jan 31, Feb 3, and Nov 21 in 2018 would not be considered panel data.

1.2 Autocorrelation

When you have panel data, autocorrelation is the correlation between subsequent observations. For example, if you have daily observations, then the 1 day autocorrelation is the correlation between observations 1 day apart, and likewise the 2 day autocorrelation is the correlation between observations 2 days apart.

1.3 Scenarios

In this course we will consider two scenarios where we have multiple observations for each geographical area:

- Panel data: One geographical area with/without autocorrelation
- Panel data: Multiple geographical areas without autocorrelation

Note, the following scenario can be covered by standard regression models:

- Multiple geographical areas, one time point/observation per geographical area

1.4 Useful Code

This code is used to calculate prediction intervals. In its most basic form it is:

$$95\% \text{ prediction interval} = \text{sample average} \pm 1.96 \times \text{sample standard deviation} \sqrt{1 + 1/n}$$

However, due to the skewness of the count data, we often choose to use a `2/3s transformation`.

```
sykdomspuls::FarringtonThreshold
```

```
## function (pred, phi, alpha = NULL, z = NULL, skewness.transform = "none")
## {
##     mu0 <- pred$fit
##     tau <- phi + (pred$se.fit^2)/mu0
##     switch(skewness.transform, none = {
##         se <- sqrt(mu0 * tau)
##         exponent <- 1
##     }, `1/2` = {
##         se <- sqrt(1/4 * tau)
##         exponent <- 1/2
##     }, `2/3` = {
##         se <- sqrt(4/9 * mu0^(1/3) * tau)
##         exponent <- 2/3
##     }, {
##         stop("No proper exponent in algo.farrington.threshold.")
##     })
##     if (is.null(z))
##         z <- qnorm(1 - alpha/2)
##     lu <- (mu0^exponent + z * se)^(1/exponent)
##     return(lu)
## }
## <bytecode: 0x4080170>
## <environment: namespace:sykdomspuls>
```

Please note that a prediction interval is not the same as a confidence interval!

Chapter 2

Panel Data - One Area

2.1 Aim

We are given a dataset containing daily counts of diseases from one geographical area. We want to identify:

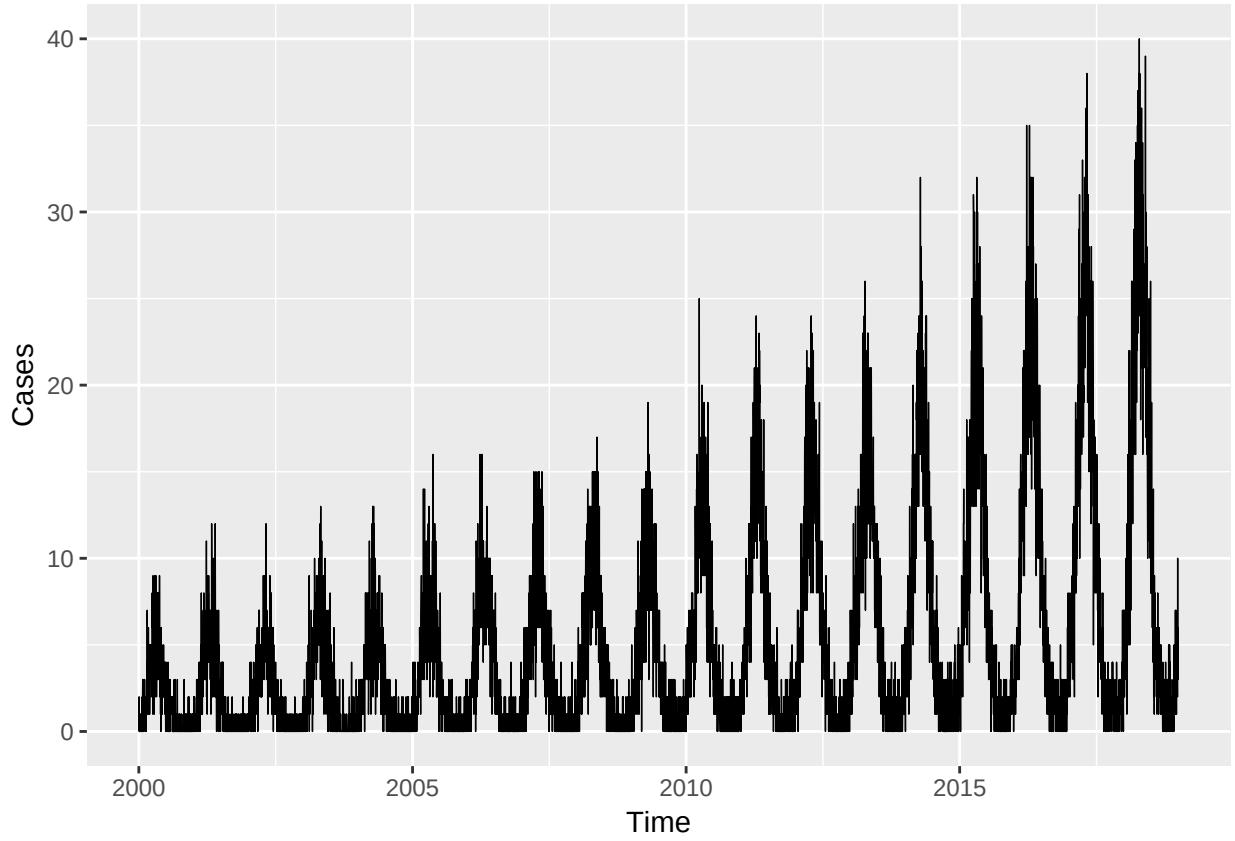
1. Does seasonality exist?
2. If seasonality exists, when are the high/low seasons?
3. Is there a general yearly trend (i.e. increasing or decreasing from year to year?)
4. Is daily rainfall associated with the number of cases?
5. When are there outbreaks?

```
library(data.table)
library(ggplot2)
set.seed(4)

AMPLITUDE <- 1.5
SEASONAL_HORIZONTAL_SHIFT <- 20

d <- data.table(date = seq.Date(
  from = as.Date("2000-01-01"),
  to = as.Date("2018-12-31"),
  by = 1
))
d[, date := as.Date(date, origin = "1970-01-1")]
d[, year := as.numeric(format.Date(date, "%G"))]
d[, week := as.numeric(format.Date(date, "%V"))]
d[, month := as.numeric(format.Date(date, "%m"))]
d[, yearMinus2000 := year - 2000]
d[, dailyrainfall := runif(.N, min = 0, max = 10)]

d[, dayOfYear := as.numeric(format.Date(date, "%j"))]
d[, seasonalEffect := sin(2 * pi * (dayOfYear - SEASONAL_HORIZONTAL_SHIFT) / 365)]
d[, mu := exp(0.1 + yearMinus2000 * 0.1 + seasonalEffect * AMPLITUDE)]
d[, y := rpois(.N, mu)]
```

2.2 Data

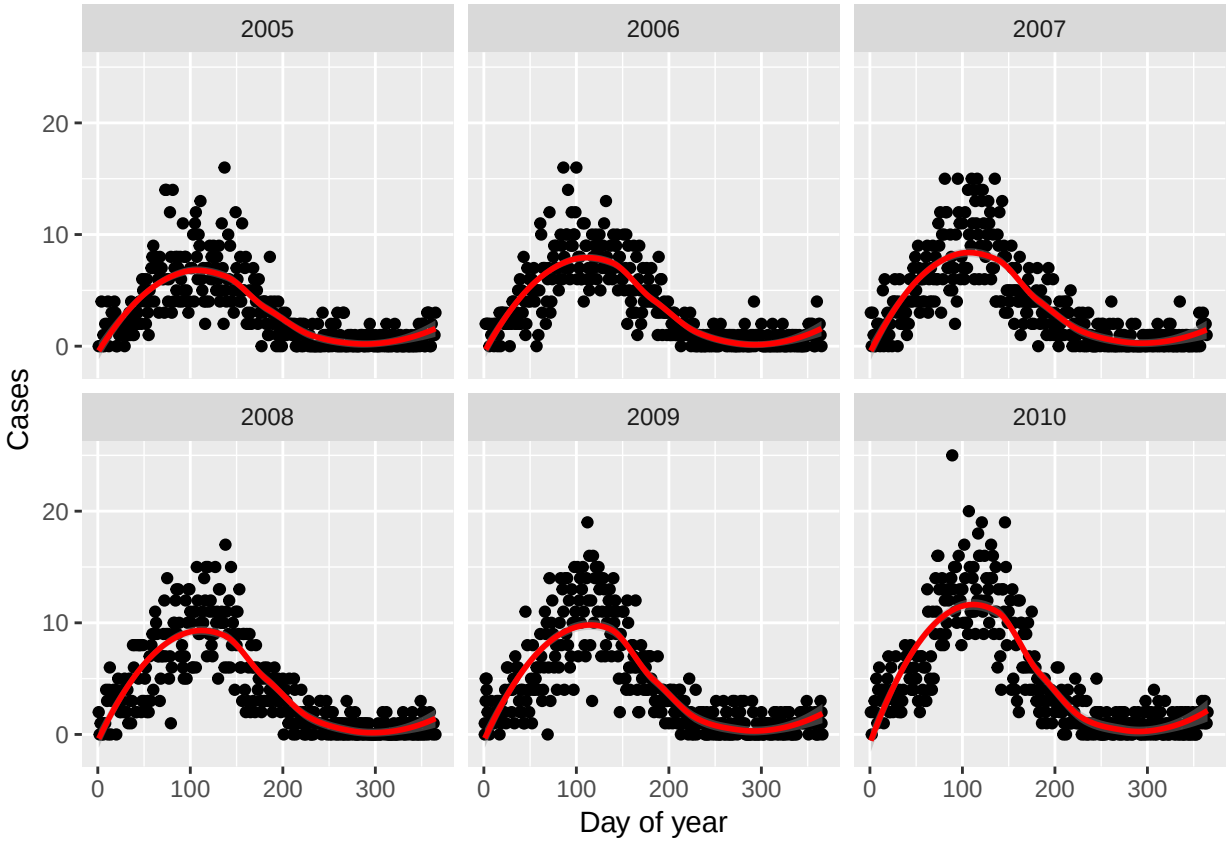
Here we show the true data, and note that there is an increasing annual trend (the data gets higher as time goes on) and there is a seasonal pattern (one peak/trough per year)

```
q <- ggplot(d, aes(x = date, y = y))
q <- q + geom_line(lwd = 0.25)
q <- q + scale_x_date("Time")
q <- q + scale_y_continuous("Cases")
q
```



We split out the data for a few years and see a clear seasonal trend:

```
q <- ggplot(d[year %in% c(2005:2010)], aes(x = dayOfYear, y = y))
q <- q + facet_wrap(~year)
q <- q + geom_point()
q <- q + stat_smooth(colour = "red")
q <- q + scale_x_continuous("Day of year")
q <- q + scale_y_continuous("Cases")
q
```



2.3 Model With Non-Parametric Seasonality

If we want to investigate the seasonality of our data, and identify when are the peaks and troughs, we can use non-parametric approaches. They are flexible and easy to implement, but they can lack power and be hard to interpret:

- Create a categorical variable for the seasons (e.g. `spring`, `summer`, `autumn`, `winter`) and include this in the regression model
- Create a categorical variable for the months (e.g. `Jan`, `Feb`, ..., `Dec`) and include this in the regression model

```
nfit0 <- glm(y ~ yearMinus2000 + dailyrainfall, data = d, family = poisson())
nfit1 <- glm(y ~ yearMinus2000 + dailyrainfall + as.factor(month), data = d, family = poisson())
```

2.3.1 Seasonality

We can test the `month` categorical variable using a likelihood ratio test:

```
lmtest::lrtest(nfit0, nfit1)
```

```
## Likelihood ratio test
##
## Model 1: y ~ yearMinus2000 + dailyrainfall
## Model 2: y ~ yearMinus2000 + dailyrainfall + as.factor(month)
##   #Df LogLik Df Chisq Pr(>Chisq)
```

```
## 1   3 -26904
## 2  14 -13251 11 27307  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 1: Does seasonality exist?

And then we can look at the output of our regression:

```
summary(nfit1)
```

```
##
## Call:
## glm(formula = y ~ yearMinus2000 + dailyrainfall + as.factor(month),
##     family = poisson(), data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2874  -0.9578  -0.1498   0.5894   3.9130
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.003849   0.028882  -0.133    0.894
## yearMinus2000       0.101654   0.001053  96.578   <2e-16 ***
## dailyrainfall       0.000442   0.001852   0.239    0.811
## as.factor(month)2   0.751048   0.029854  25.157   <2e-16 ***
## as.factor(month)3   1.303525   0.027328  47.700   <2e-16 ***
## as.factor(month)4   1.543098   0.026781  57.619   <2e-16 ***
## as.factor(month)5   1.425207   0.026992  52.801   <2e-16 ***
## as.factor(month)6   0.955465   0.028647  33.354   <2e-16 ***
## as.factor(month)7   0.286169   0.032060   8.926   <2e-16 ***
## as.factor(month)8  -0.541443   0.039932 -13.559   <2e-16 ***
## as.factor(month)9  -1.114005   0.049322 -22.586   <2e-16 ***
## as.factor(month)10 -1.350683   0.053389 -25.299   <2e-16 ***
## as.factor(month)11 -1.235671   0.051682 -23.909   <2e-16 ***
## as.factor(month)12 -0.754107   0.042777 -17.629   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 45536.8  on 6939  degrees of freedom
## Residual deviance:  8045.6  on 6926  degrees of freedom
## AIC: 26529
##
## Number of Fisher Scoring iterations: 5
```

NOTE: See that this is basically the same as a normal regression.

Question 2: If seasonality exists, when are the high/low seasons?

2.3.2 Yearly trend

Question 3: Is there a general yearly trend (i.e. increasing or decreasing from year to year?)

2.3.3 Association With Rainfall

Question 4: Is daily rainfall associated with the number of cases?

2.3.4 Outbreaks

If we want to identify outbreaks, then we need to use the standard prediction interval formula:

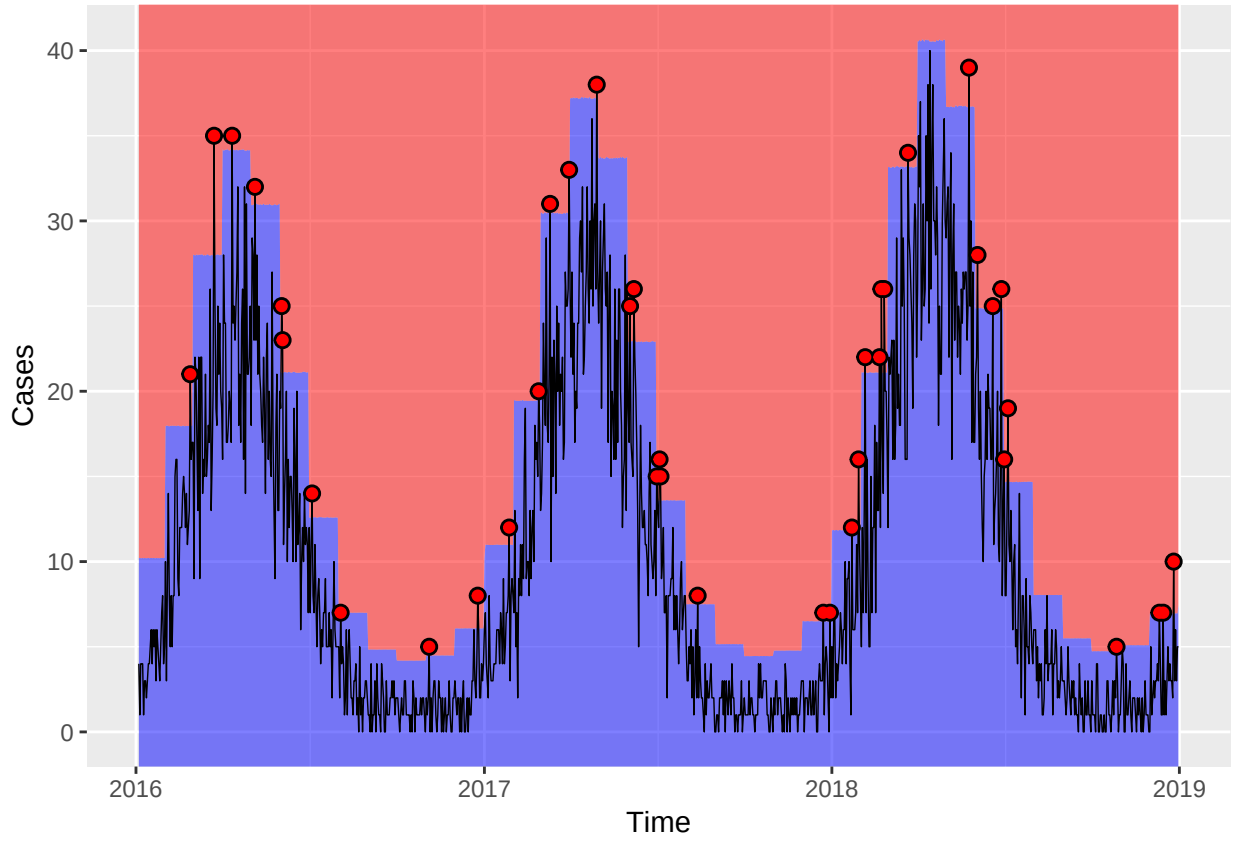
$$95\% \text{ prediction interval} = \text{sample average} \pm 1.96 \times \text{sample standard deviation} \sqrt{1 + 1/n}$$

This allows us to identify what the expected thresholds are:

```
pred <- predict(nfit1, type = "response", se.fit = T, newdata = d)
d[, threshold0 := pred$fit]
d[, threshold2 := sykdomspuls::FarringtonThreshold(pred, phi = 1, z = 2, skewness.transform = "2/3")]
```

Question 5: When are there outbreaks?

```
q <- ggplot(d[year > 2015], aes(x = date, y = y))
q <- q + geom_ribbon(mapping = aes(ymin = -Inf, ymax = threshold2), fill = "blue", alpha = 0.5)
q <- q + geom_ribbon(mapping = aes(ymin = threshold2, ymax = Inf), fill = "red", alpha = 0.5)
q <- q + geom_line(lwd = 0.25)
q <- q + geom_point(data = d[year > 2015 & y > threshold2], colour = "black", size = 2.5)
q <- q + geom_point(data = d[year > 2015 & y > threshold2], colour = "red", size = 1.5)
q <- q + scale_x_date("Time")
q <- q + scale_y_continuous("Cases")
q
```



2.4 Model With Parametric Seasonality

Parametric approaches are more powerful but require more effort:

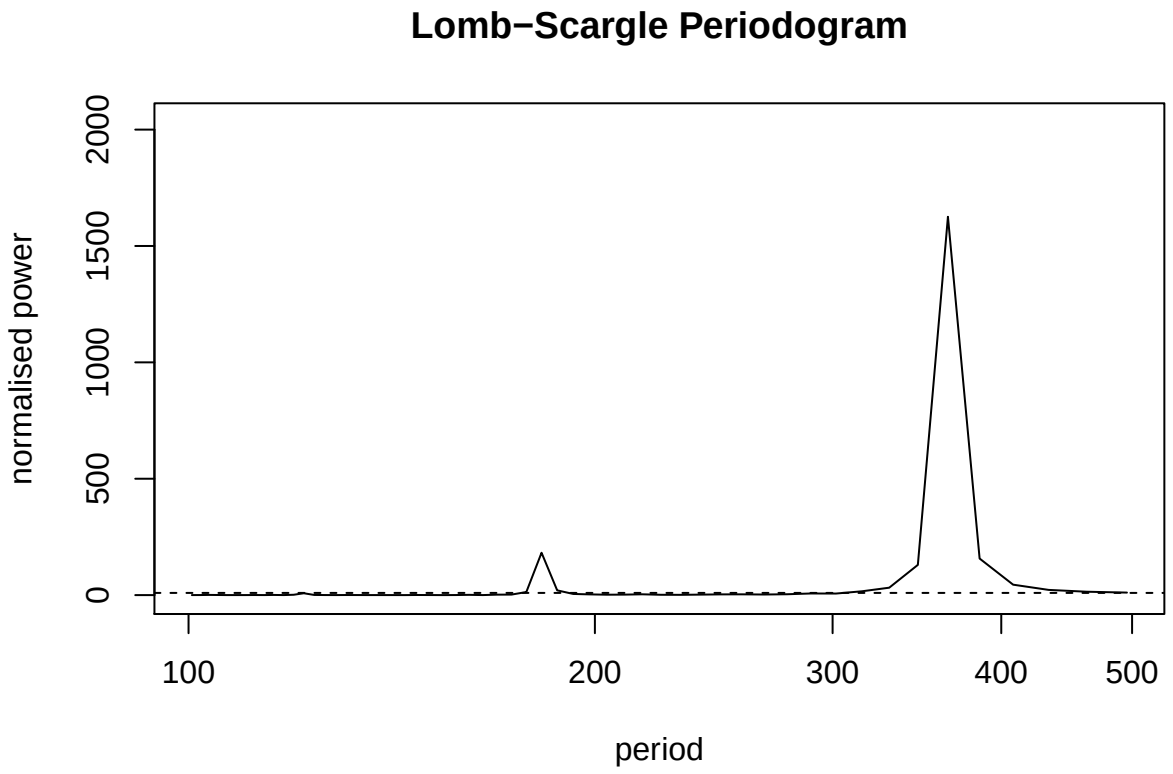
- Identify the periodicity of the seasonality (how many days between peaks?)
- Using trigonometry, transform `day of year` into variables that appropriately model the observed periodicity
- Obtain coefficient estimates
- Back-transform these estimates into human-understandable values (day of peak, day of trough)

NOTE: You don't always have to investigate seasonality! It depends entirely on what the purpose of your analysis is!

2.4.1 Seasonality

The Lomb-Scargle Periodogram shows a clear seasonality with a period of 365 days.

```
# R CODE
lomb::lsp(d$y, from = 100, to = 500, ofac = 1, type = "period")
```



We then generate two new variables `cos365` and `sin365` and perform a likelihood ratio test to see if they are significant or not. This is done with two simple poisson regressions.

```
# R CODE
d[, cos365 := cos(dayOfYear * 2 * pi / 365)]
d[, sin365 := sin(dayOfYear * 2 * pi / 365)]
```

```
pfit0 <- glm(y ~ yearMinus2000 + dailyrainfall, data = d, family = poisson())
pfit1 <- glm(y ~ yearMinus2000 + dailyrainfall + sin365 + cos365, data = d, family = poisson())
```

We can test the seasonality using a likelihood ratio test (which we already strongly suspected due to the periodogram):

```
lmtest::lrtest(pfit0, pfit1)
```

```
## Likelihood ratio test
##
## Model 1: y ~ yearMinus2000 + dailyrainfall
## Model 2: y ~ yearMinus2000 + dailyrainfall + sin365 + cos365
##   #Df LogLik Df Chisq Pr(>Chisq)
## 1   3 -26904
## 2   5 -12892  2 28024  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 1: Does seasonality exist?

And then we can look at the output of our regression:

```
summary(pfit1)
```

```
##
## Call:
## glm(formula = y ~ yearMinus2000 + dailyrainfall + sin365 + cos365,
##     family = poisson(), data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0676  -0.9229  -0.1170   0.5861   3.4103
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.0887436  0.0176742   5.021 5.14e-07 ***
## yearMinus2000  0.1016117  0.0010525  96.539  < 2e-16 ***
## dailyrainfall  0.0002287  0.0018476   0.124    0.901
## sin365         1.3972586  0.0103200 135.393  < 2e-16 ***
## cos365        -0.5035265  0.0086308 -58.341  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 45536.8  on 6939  degrees of freedom
## Residual deviance:  7328.5  on 6935  degrees of freedom
## AIC: 25794
##
## Number of Fisher Scoring iterations: 5
```

We also see that the (significant!) coefficient for `year` is `0.1` which means that for each additional year, the outcome increases by `exp(0.1)=1.11`. We also see that the coefficient for `dailyrainfall` was not significant, which means that we did not find a significant association between the outcome and `dailyrainfall`.

NOTE: See that this is basically the same as a normal regression.

Through the likelihood ratio test we saw a clear significant seasonal effect. We can now use trigonometry to back-calculate the amplitude and location of peak/troughs from the `cos365` and `sin365` estimates:

```
RAWmisc::TransformCosSinToAmplitudePeakTrough
```

```
## function (cos_b, sin_b)
## {
##     b1 <- sin_b
##     b2 <- cos_b
##     amplitude <- sqrt(b1^2 + b2^2)
##     p <- atan(b1/b2) * 366/2/pi
##     if (p > 0) {
##         peak <- p
##         trough <- p + 366/2
##     }
##     else {
##         peak <- p + 366/2
##         trough <- p + 366
##     }
##     if (b1 < 0) {
##         g <- peak
##         peak <- trough
##         trough <- g
##     }
##     return(list(amplitude = amplitude, peak = peak, trough = trough))
## }
## <bytecode: 0x47ea6b8>
## <environment: namespace:RAWmisc>
```

```
retval <- RAWmisc::TransformCosSinToAmplitudePeakTrough(
  cos_b = -0.512912, # cos coefficient
  sin_b = 1.428417 # sin coefficient
)

print(sprintf("amplitude is estimated as %s", round(retval$amplitude, 2)))
```

```
## [1] "amplitude is estimated as 1.52"
```

```
print(sprintf("peak is estimated as %s", round(retval$peak)))
```

```
## [1] "peak is estimated as 112"
```

```
print(sprintf("trough is estimated as %s", round(retval$trough)))
```

```
## [1] "trough is estimated as 295"
```

```
print(sprintf("true amplitude is %s", round(AMPLITUDE, 2)))
```

```
## [1] "true amplitude is 1.5"
```

```
print(sprintf("true peak is %s", round(365 / 4 + SEASONAL_HORIZONTAL_SHIFT)))
```

```
## [1] "true peak is 111"
```

```
print(sprintf("true trough is %s", round(3 * 365 / 4 + SEASONAL_HORIZONTAL_SHIFT)))
```

```
## [1] "true trough is 294"
```

NOTE: An amplitude of 1.5 means that when comparing the average time of year to the peak, the peak is expected to be `exp(1.5)=4.5` times higher than average. We take the exponential because we have run a

poisson regression (so think incident rate ratio).

Question 2: If seasonality exists, when are the high/low seasons?

2.4.2 Yearly trend

Question 3: Is there a general yearly trend (i.e. increasing or decreasing from year to year?)

2.4.3 Association With Rainfall

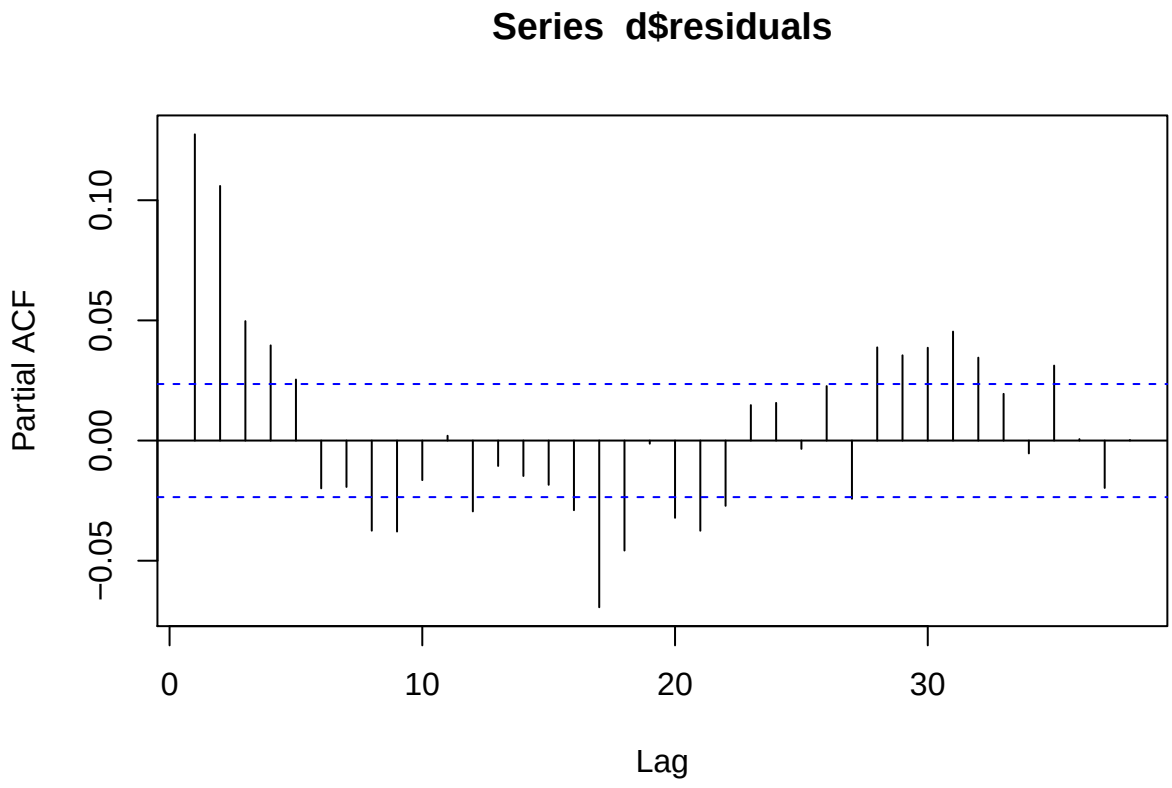
Question 4: Is daily rainfall associated with the number of cases?

2.5 Autocorrelation

We check the `pacf` of the residuals to ensure that there is no autocorrelation. If we observe autocorrelation in our residuals, then we need to use a `robust` variance estimator (i.e. it makes our estimated variances bigger to account for our poor model fitting).

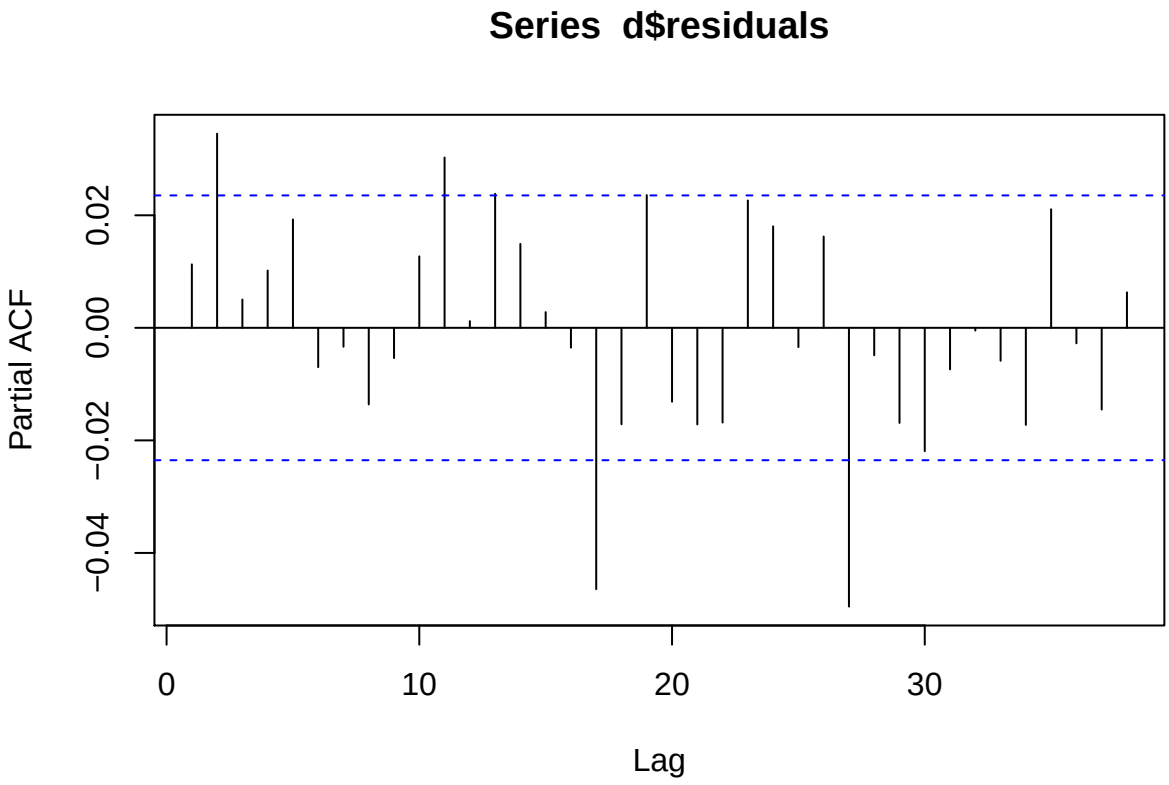
Here we see that our non-parametric seasonality model has not accounted for all of the associations in the data, so there is some autocorrelation in the residuals:

```
d[, residuals := residuals(nfit1, type = "response")]
d[, predicted := predict(nfit1, type = "response")]
pacf(d$residuals)
```



Here we see that our parametric seasonality model has accounted for all of the associations in the data, so there is no autocorrelation in the residuals:

```
d[, residuals := residuals(pfit1, type = "response")]
d[, predicted := predict(pfit1, type = "response")]
pacf(d$residuals)
```



2.6 Hints For Future Analyses

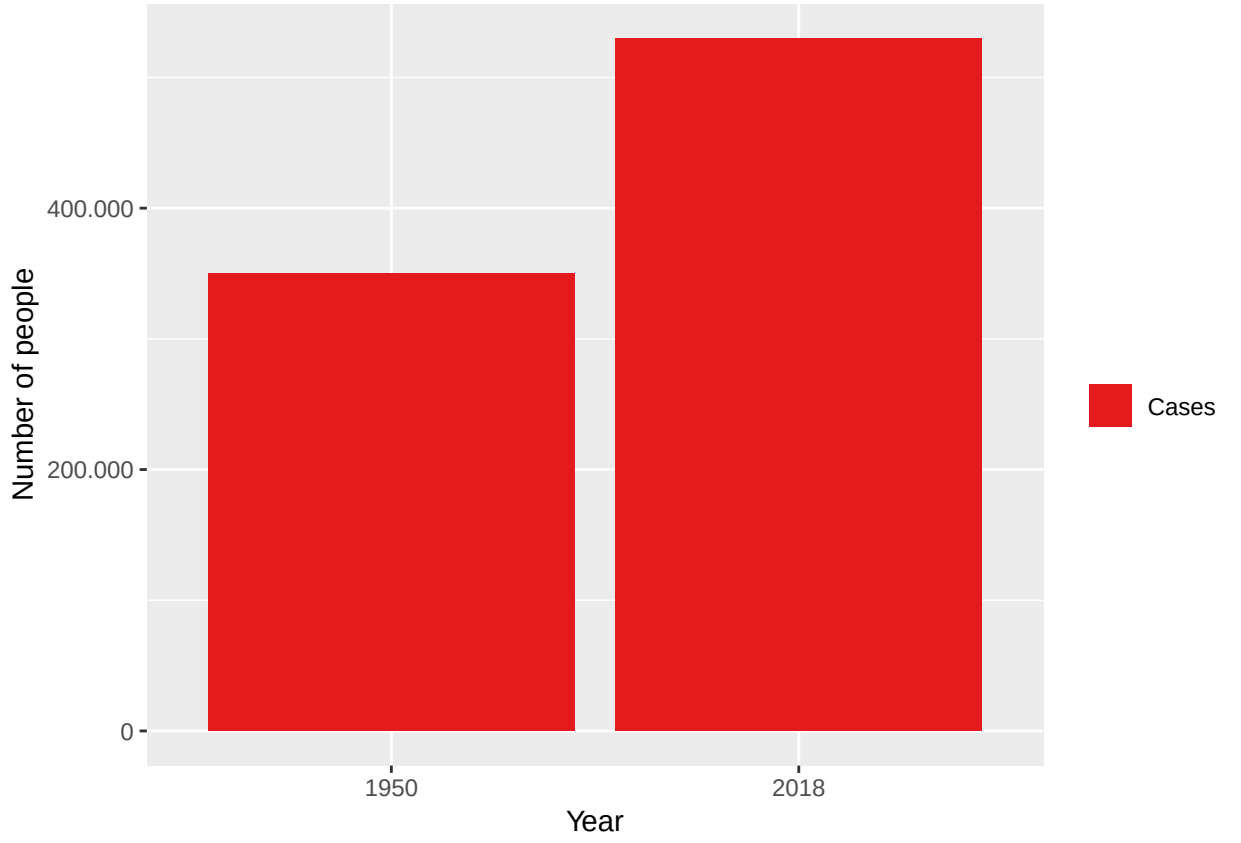
2.6.1 Always Use A Denominator

```
d <- data.table(
  year = c("1950", "2018"),
  Cases = c(350000, 530000),
  Population = c(3500000, 5300000)
)
d[, `Cases per\n100.000 Pop` := Cases / Population * 100000]
d <- melt.data.table(d, id.vars = "year")
```

We start out by considering the number of cases of a disease in 1950 and 2018. We see that the number of cases has increased dramatically over this time period!

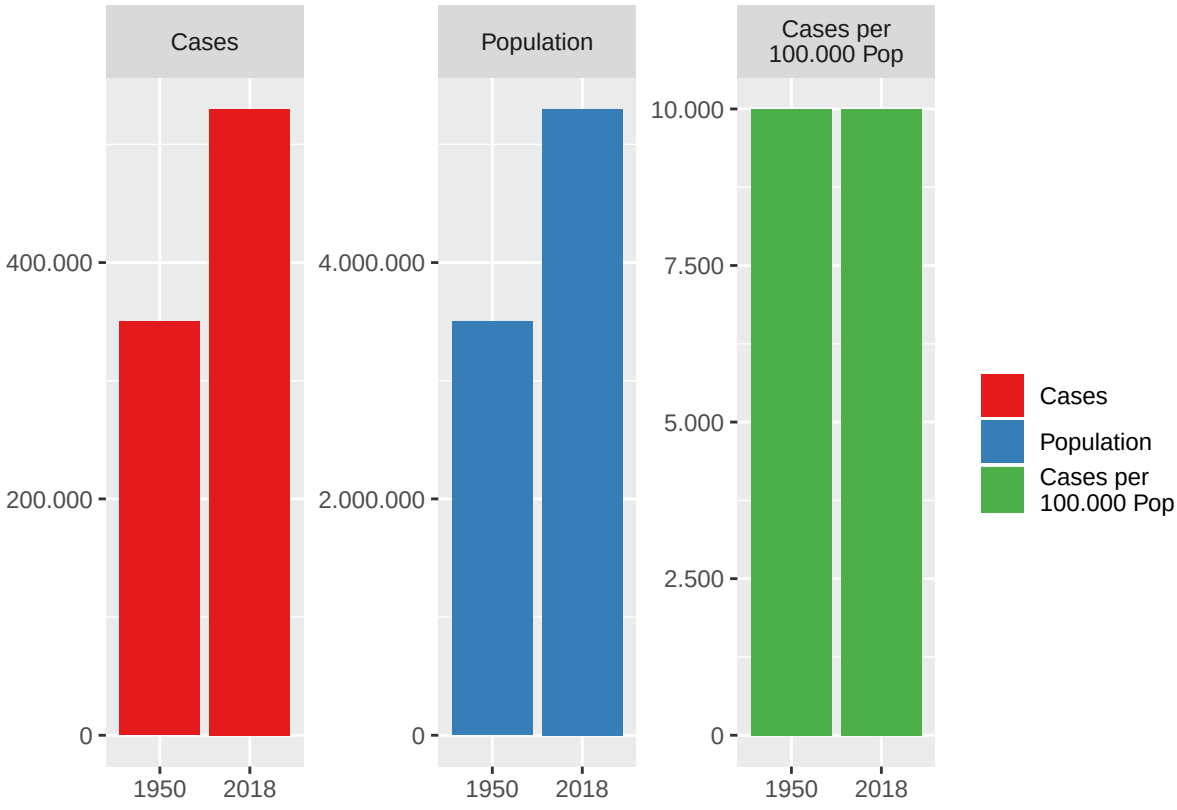
```
q <- ggplot(d[variable == "Cases"], aes(x = year, y = value, fill = variable))
q <- q + geom_col()
q <- q + scale_x_discrete("Year")
```

```
q <- q + scale_y_continuous("Number of people", labels = scales::format_format(
  big.mark = ".",
  decimal.mark = ",",
  scientific = FALSE
))
q <- q + scale_fill_brewer("", palette = "Set1")
q
```



However, upon taking the denominator (i.e. population) into consideration, we can see that the rate is consistent over time.

```
q <- ggplot(d, aes(x = year, y = value, fill = variable))
q <- q + geom_col(position = "dodge")
q <- q + scale_x_discrete("")
q <- q + scale_y_continuous("", labels = scales::format_format(
  big.mark = ".",
  decimal.mark = ",",
  scientific = FALSE
))
q <- q + scale_fill_brewer("", palette = "Set1")
q <- q + facet_wrap(~variable, scales = "free")
q
```



2.6.2 Negative Binomial Is Generally Better Than Poisson

Let us consider linear regression.

A linear regression model takes the form:

$$y_i = \beta_0 + \beta_1 \times x_i + \text{error}_i$$

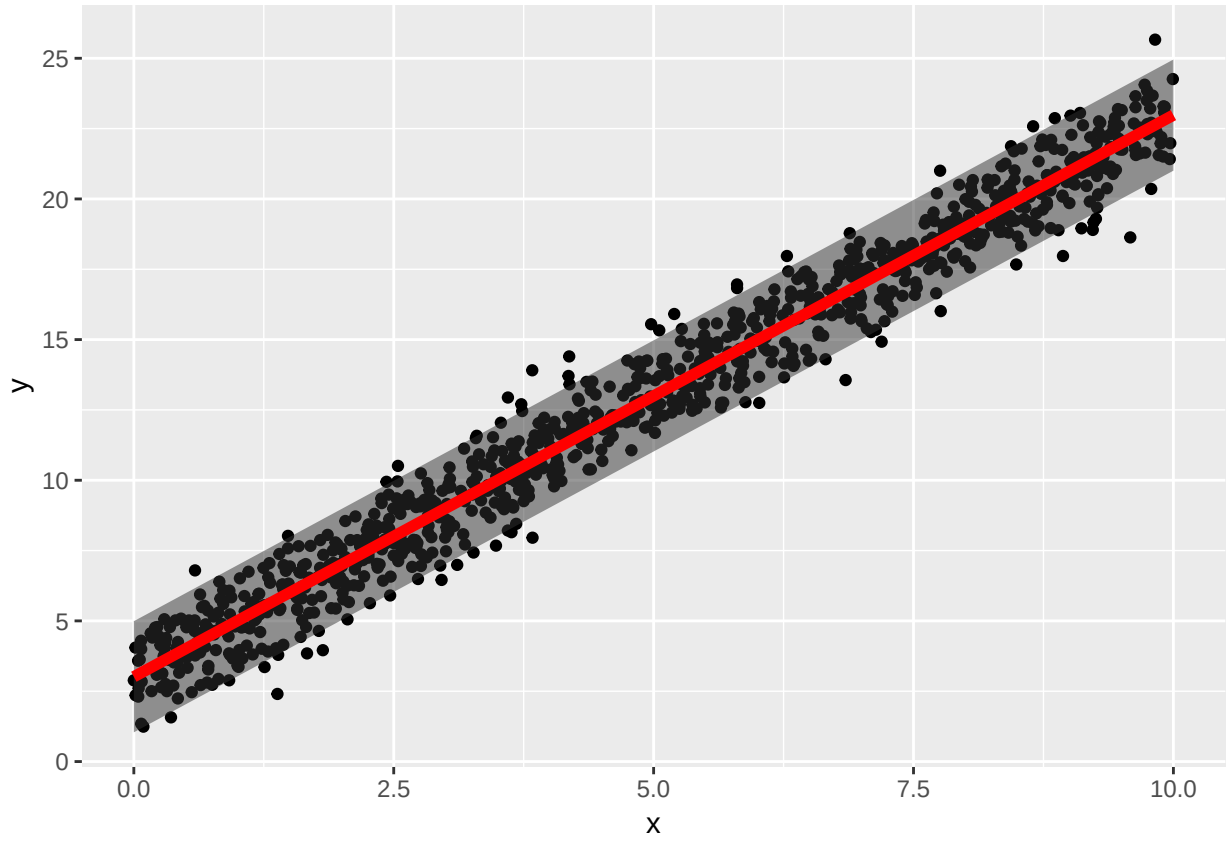
Where

$$\text{error}_i \sim N(0, \sigma)$$

So basically, we have a straight line, and then the data is expected to be in a parallel band surrounding it:

```
d <- data.table(x = runif(1000) * 10)
d[, y := 3 + 2 * x + rnorm(.N)]

fit <- lm(y ~ x, data = d)
resid_sd <- sd(fit$residuals)

thresholds <- data.table(x = c(0:10))
thresholds[, pred := predict(fit, newdata = thresholds)]
thresholds[, pred_l95 := pred - 1.96 * resid_sd]
thresholds[, pred_u95 := pred + 1.96 * resid_sd]
```

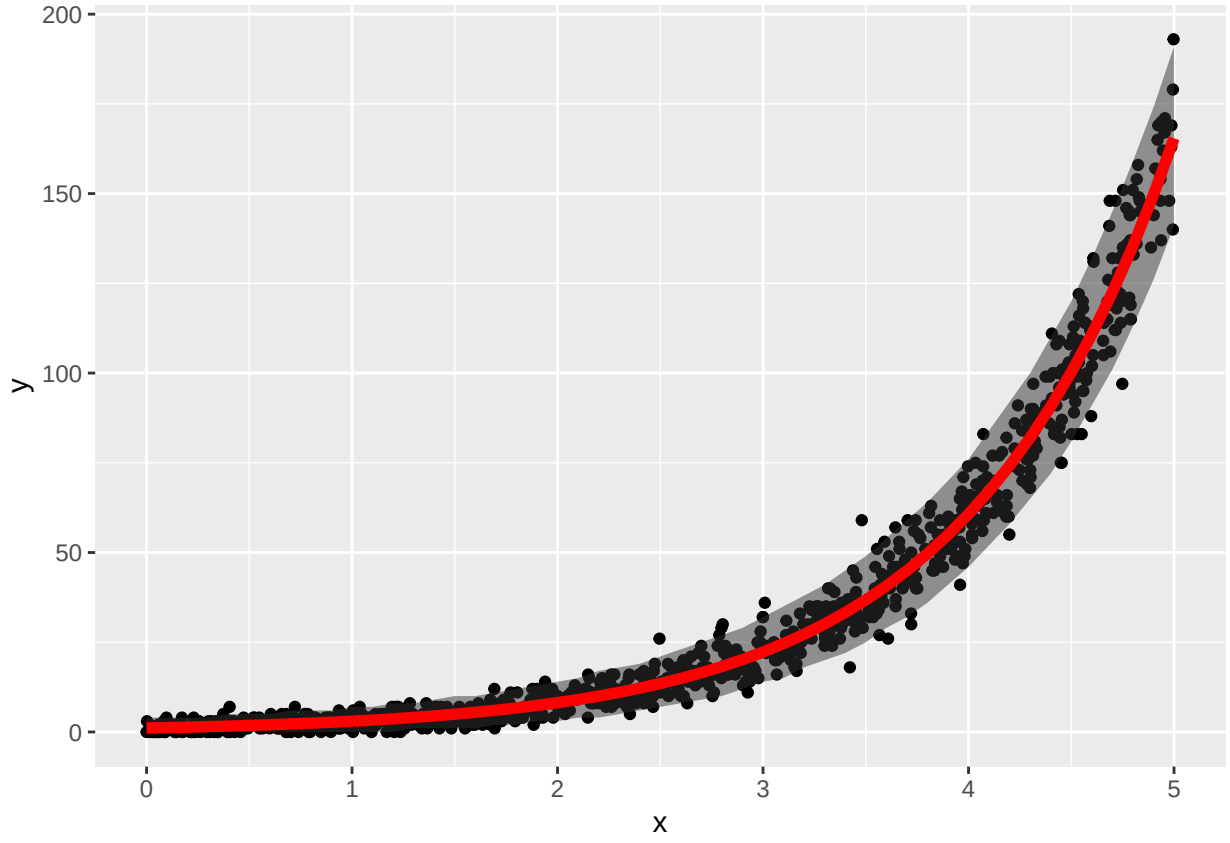
```
q <- ggplot(d, aes(x = x))
q <- q + geom_point(mapping = aes(y = y))
q <- q + geom_ribbon(data = thresholds, mapping = aes(ymin = pred_l95, ymax = pred_u95), alpha = 0.5)
q <- q + geom_line(data = thresholds, mapping = aes(y = pred), colour = "red", lwd = 2)
q
```



Poisson regression operates under the strong assumption that `mean=variance`. This means that when the mean is 150 (e.g. 150 cases per day), then the variance is also 150 (e.g. we expect between 126 and 174 cases each day). When the mean is 5 (e.g. 5 cases per day), then the variance is also 5 (e.g. we expect between 1 and 10 cases each day).

```
d <- data.table(x = runif(1000) * 5)
d[, mu := exp(0.1 + x)]
d[, y := rpois(.N, mu)]

fit <- glm(y ~ x, data = d, family = "poisson")
resid_sd <- sd(fit$residuals)

thresholds <- data.table(x = c(0:50) / 10)
thresholds[, pred := predict(fit, newdata = thresholds, type = "response")]
thresholds[, pred_l95 := qpois(0.025, pred)]
thresholds[, pred_u95 := qpois(0.975, pred)]

q <- ggplot(d, aes(x = x))
q <- q + geom_point(mapping = aes(y = y))
q <- q + geom_ribbon(data = thresholds, mapping = aes(ymin = pred_l95, ymax = pred_u95), alpha = 0.5)
```

```
q <- q + geom_line(data = thresholds, mapping = aes(y = pred), colour = "red", lwd = 2)
q
```



This assumption may be true, or it may not be true. However, the main point is that it is a strong assumption, which makes the poisson regression less flexible.

A more flexible regression model for count data is the negative binomial model. Here, the "dispersion" (i.e. variance of the data) is estimated separately from the mean. You can think of it as being similar to a linear regression.

The benefits of the negative binomial regression model is that it is more flexible and is more likely to fit your data.

The downside of the negative binomial regression model is that it needs more data and may not converge. It is recommended to try and run a negative binomial regression model first, and then if it fails to converge, then run a poisson regression.

2.6.3 Zeroes In Your Individial -> Aggregated Dataset

If your data is at the individual level (i.e. one row per case), then you will need to aggregate it to daily/weekly/monthly levels. If your data source is a registry, and you assume that your dataset contains all of the reported cases, then this means `"no reports"="no cases"`. This means that after aggregating, **you need to make sure that your collapsed/aggregated dataset has zeroes in it!!**

In the following dataset, we have one case per day from `2000-01-01` until `2000-01-29` and then again one case per day from `2000-08-08` until `2000-12-31`.

```
d <- data.table(date = seq.Date(
  from = as.Date("2000-01-01"),
  to = as.Date("2000-12-31"),
  by = 1
))
d <- d[-c(30:220)]
d[, id := 1:.N]
d[, month := as.numeric(format.Date(date, "%m"))]

print(d)
```

```
##            date  id month
##   1: 2000-01-01   1     1
##   2: 2000-01-02   2     1
##   3: 2000-01-03   3     1
##   4: 2000-01-04   4     1
##   5: 2000-01-05   5     1
##  ---
## 171: 2000-12-27 171    12
## 172: 2000-12-28 172    12
## 173: 2000-12-29 173    12
## 174: 2000-12-30 174    12
## 175: 2000-12-31 175    12
```

If we collapse the data into months:

```
collapsed <- d[, .(
  n = .N
), keyby = .(
  month
)]

print(collapsed)
```

```
##    month  n
## 1:     1 29
## 2:     8 24
## 3:     9 30
## 4:    10 31
## 5:    11 30
## 6:    12 31
```

You see that we are missing months `1` through to `6`. We cannot analyse this data, because **we do not have any zeros**.

How do we fix this? We create a `skeleton` of our results:

```
skeleton <- data.table(month = 1:12)
print(skeleton)
```

```
##     month
##  1:     1
##  2:     2
##  3:     3
##  4:     4
##  5:     5
##  6:     6
```

```
##  7:     7
##  8:     8
##  9:     9
## 10:    10
## 11:    11
## 12:    12
```

We then merge our collapsed data with the skeleton:

```
final <- merge(skeleton, collapsed, by = "month", all.x = TRUE)
print(final)
```

```
##     month  n
##  1:     1 29
##  2:     2 NA
##  3:     3 NA
##  4:     4 NA
##  5:     5 NA
##  6:     6 NA
##  7:     7 NA
##  8:     8 24
##  9:     9 30
## 10:    10 31
## 11:    11 30
## 12:    12 31
```

We then set all of the "missing" to 0:

```
final[is.na(n), n := 0]
print(final)
```

```
##     month  n
##  1:     1 29
##  2:     2  0
##  3:     3  0
##  4:     4  0
##  5:     5  0
##  6:     6  0
##  7:     7  0
##  8:     8 24
##  9:     9 30
## 10:    10 31
## 11:    11 30
## 12:    12 31
```

Now we can analyze our data!

2.6.4 Lagging Of Exposures

If you are interested in seeing how exposures affect your outcome (e.g. `rainfall`) then you might want to consider lagging your exposure. This will show you `how did the rainfall from last week affect the number of cases this week?`

Chapter 3

Panel data - Multiple Areas

3.1 Aim

We are given a dataset containing yearly counts of diseases from multiple geographical areas.

We will explore how this is different from the one area case.

For this section, we will use linear regression to make the calculations easier to work with, but the principle is the same as with poisson and negative binomial regression.

3.2 Creating the data

```
library(data.table)
library(lme4)
```

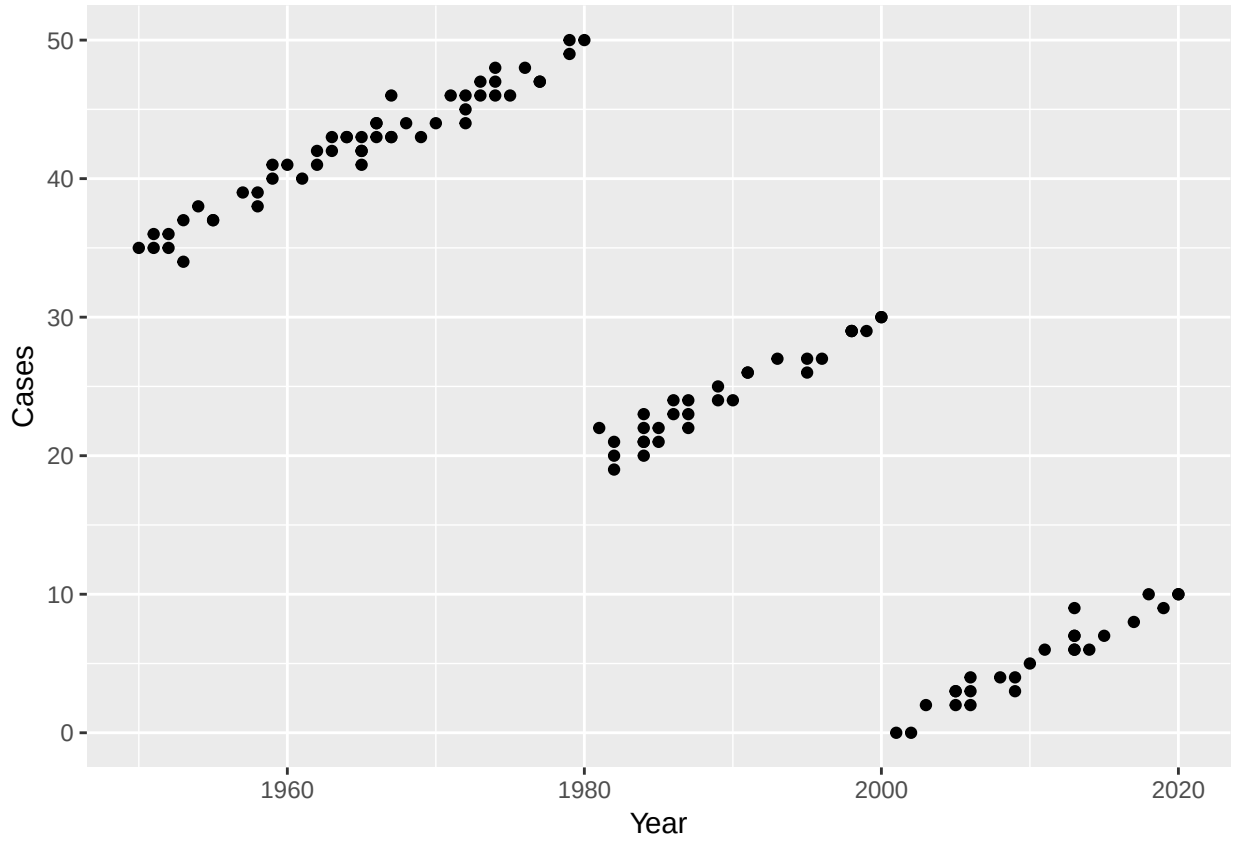
```
## Loading required package: Matrix
```

```
library(ggplot2)
set.seed(4)

fylkeIntercepts <- data.table(fylke = 1:3, fylkeIntercepts = c(30, 0, -30))

d <- data.table(fylke = rep(1:3, each = 100))
d <- merge(d, fylkeIntercepts, by = "fylke")
d[, mainIntercept := 5]
d[, x := runif(.N)]
d[, year := sample(c(1950:2020), .N, replace = T)]
d[, yearMinus1950 := year - 1950]
d[, mu := mainIntercept + fylkeIntercepts + 0.5 * yearMinus1950]
d[, y := round(rnorm(.N, mu, sd = 1))]
# d[,y:=rpois(.N,mu)]

d[fylke == 1 & !year %in% c(1950:1980), y := NA]
d[fylke == 2 & !year %in% c(1980:2000), y := NA]
d[fylke == 3 & !year %in% c(2000:2020), y := NA]

d <- na.omit(d)
```

3.3 Investigating the data

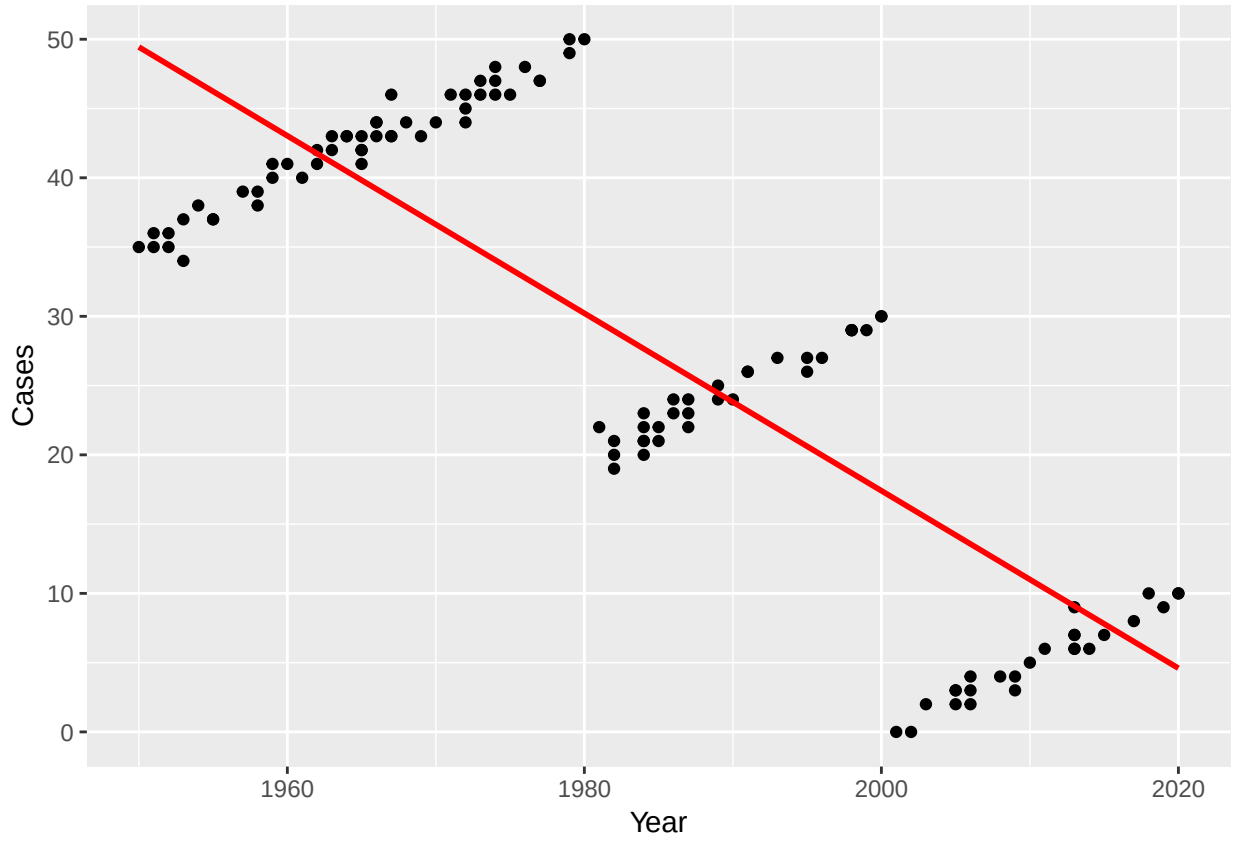
We begin by blindly looking at the data.

```
q <- ggplot(d, aes(x = year, y = y))
q <- q + geom_point()
q <- q + scale_x_continuous("Year")
q <- q + scale_y_continuous("Cases")
q
```



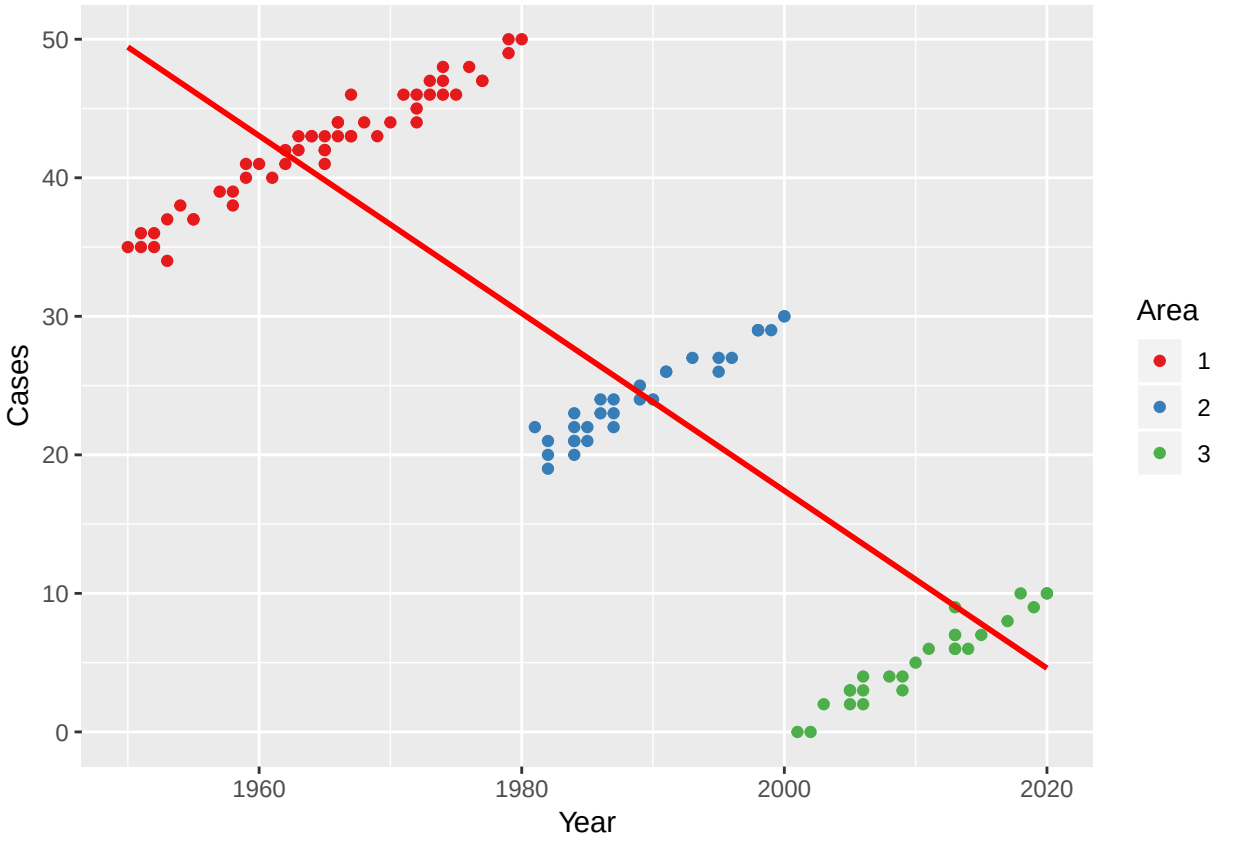
We then blindly run a linear regression and we see that the number of cases is decreasing over time.

```
q <- ggplot(d, aes(x = year, y = y))
q <- q + geom_point()
q <- q + stat_smooth(method = "lm", se = F, colour = "red")
q <- q + scale_x_continuous("Year")
q <- q + scale_y_continuous("Cases")
q
```



However, once we reveal the geographical clustering in the data, we can see that our regression model is extremely wrong:

```
q <- ggplot(d, aes(x = year, y = y))
q <- q + geom_point(mapping = aes(colour = as.factor(fylke)))
q <- q + stat_smooth(method = "lm", se = F, colour = "red")
q <- q + scale_x_continuous("Year")
q <- q + scale_y_continuous("Cases")
q <- q + scale_colour_brewer("Area", palette = "Set1")
q
```

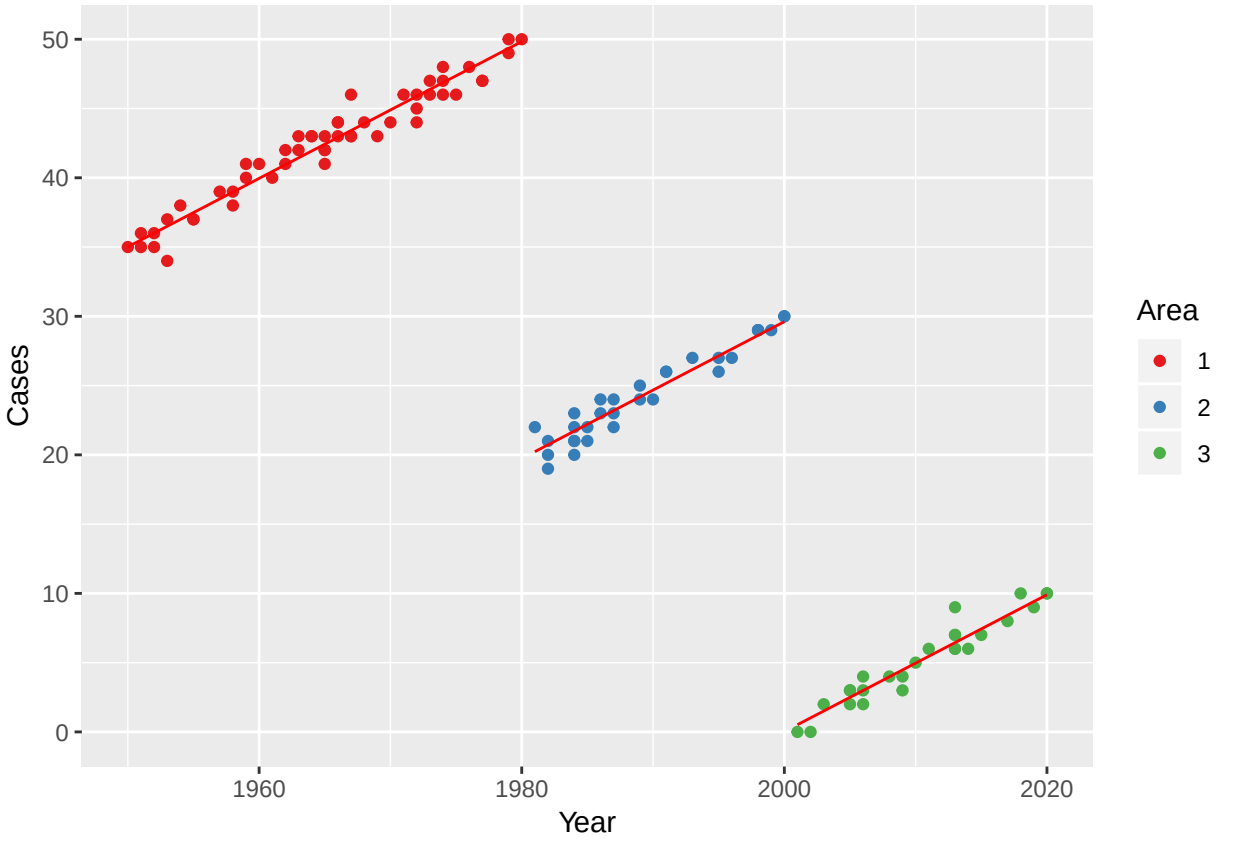


Note: The differing `number of cases` in each geographical area could also be due to differing `populations`. Always remember to `use a denominator`!!!

3.4 Time Trend Within Geographical Areas

What we actually want to do, is observe the time trends `within` each geographical area:

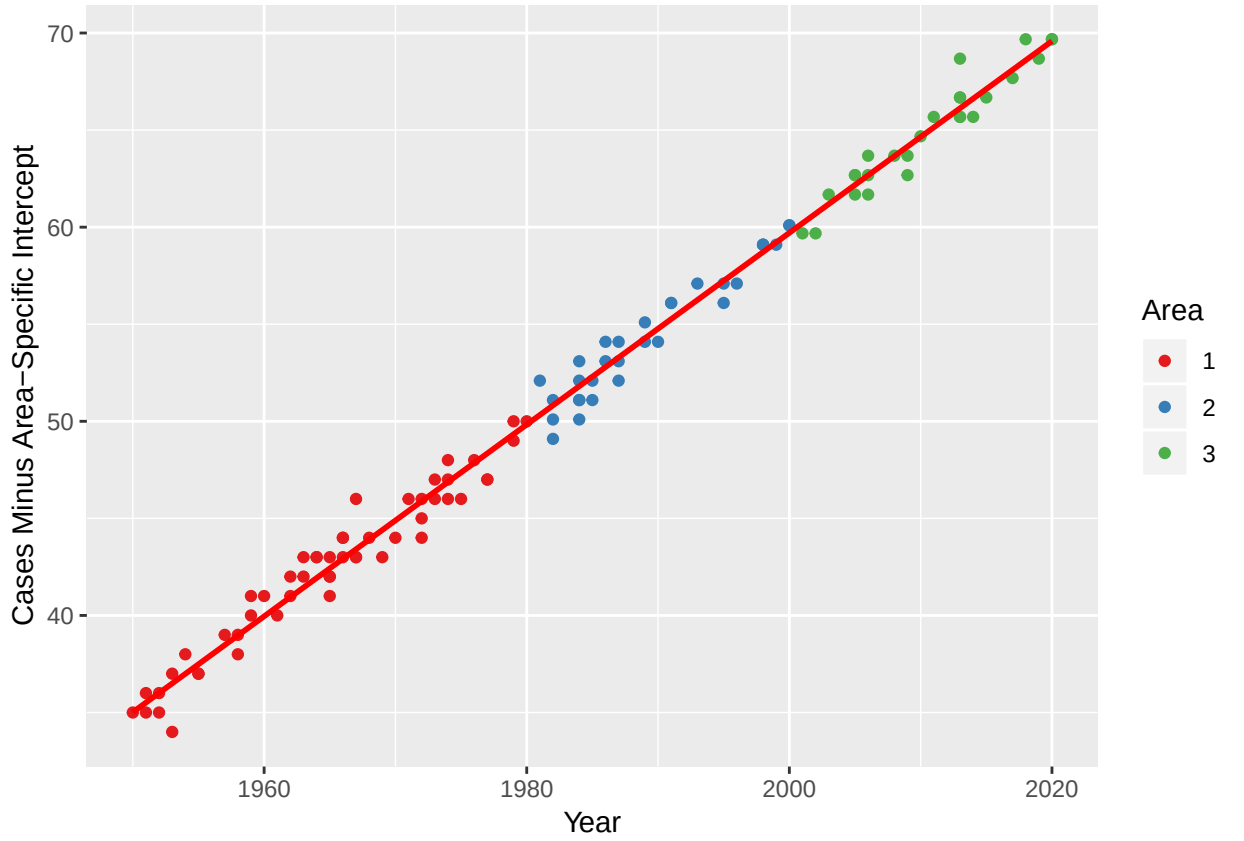
```
fit <- lm(y ~ as.factor(fylke) + yearMinus1950, data = d)
d[, p := predict(fit, newdata = d)]

q <- ggplot(d, aes(x = year, y = y))
q <- q + geom_point(mapping = aes(colour = as.factor(fylke)))
q <- q + geom_line(mapping = aes(y = p, group = as.factor(fylke)), colour = "red")
q <- q + scale_x_continuous("Year")
q <- q + scale_y_continuous("Cases")
q <- q + scale_colour_brewer("Area", palette = "Set1")
q
```



So how do we achieve this? Basically, by identifying how the areas differ from each other, and then raising/lowering each area as appropriate. The amount that each area differs from the global average is called the `area-specific intercept`.

Once we have removed the `area-specific intercept` from each geographical area, we can (basically) just run a normal regression to find the overall time trend.

```
d[, yMinusIntercept := y]
d[fylke == 2, yMinusIntercept := y - -30.0944]
d[fylke == 3, yMinusIntercept := y - -59.6795]

q <- ggplot(d, aes(x = year, y = yMinusIntercept))
q <- q + geom_point(mapping = aes(colour = as.factor(fylke)))
q <- q + stat_smooth(method = "lm", se = F, colour = "red")
q <- q + scale_x_continuous("Year")
q <- q + scale_y_continuous("Cases Minus Area-Specific Intercept")
q <- q + scale_colour_brewer("Area", palette = "Set1")
q
```



3.5 Applying The Regression Models

So how do you obtain `area-specific intercepts`? If you have a large amount of datapoints for each area, and not many areas, you can just include `area` as a categorical variable in your regression. These `area-specific intercepts` are called `fixed effects`.

```
fit <- lm(y ~ as.factor(fylke) + yearMinus1950, data = d)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ as.factor(fylke) + yearMinus1950, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.49741 -0.50980  0.03513  0.55371  2.58931
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         35.01599    0.22624  154.78   <2e-16 ***
## as.factor(fylke)2 -30.09439    0.36891  -81.58   <2e-16 ***
## as.factor(fylke)3 -59.67945    0.60794  -98.17   <2e-16 ***
## yearMinus1950       0.49381    0.01243   39.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.9244 on 105 degrees of freedom
## Multiple R-squared:  0.9966, Adjusted R-squared:  0.9965
## F-statistic: 1.016e+04 on 3 and 105 DF,  p-value: < 2.2e-16
```

If you have few datapoints for each area, and a lot of areas, you will need to use a `mixed effects` regression model, including `random intercepts` for each `area`.

```
fit <- lme4::lmer(y ~ yearMinus1950 + (1 | fylke), data = d)
summary(fit)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ yearMinus1950 + (1 | fylke)
##    Data: d
##
## REML criterion at convergence: 321.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.70424 -0.55449  0.03668  0.59846  2.80231
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  fylke    (Intercept) 890.1680 29.8357
##  Residual               0.8544  0.9244
## Number of obs: 109, groups:  fylke, 3
##
## Fixed effects:
##               Estimate Std. Error t value
## (Intercept)    5.10060   17.23247   0.296
## yearMinus1950  0.49357    0.01243  39.718
##
## Correlation of Fixed Effects:
##             (Intr)
## yearMns1950 -0.028
```

```
lme4::ranef(fit)
```

```
## $fylke
##   (Intercept)
## 1  29.9183802
## 2  -0.1696969
## 3 -29.7486833
```

3.6 Hints For Future Analyses

In general:

$$\text{number of observations} = \text{number of time points} \times \text{number of areas}$$

So 5 years of data for aggregated Norway is worse than 5 years of data for every municipality in Norway.