MTL DATA

Introduction to R Timeseries

June 2, 2016

mr.quan.nguyen@gmail.com
https://github.com/2uanta/introduction-R-timeseries

Canadian Car Sales since 2010

Month	Canadian Total Market Sales 2010	Canadian Total Market Sales 2011	Canadian Total Market Sales 2012	Canadian Total Market Sales 2013	Canadian Total Market Sales 2014
January	81,693	84,599	97,635	95,434	95,796
February	100,352	96,105	106,894	103,330	105,927
March	145,539	153,700	157,962	156,918	157,373
April	149,757	160,089	157,847	171,965	179,044
May	155,008	149,203	175,525	185,332	195,905
June	154,656	165,001	169,708	171,825	175,809
July	148,865	141,641	148,350	159,186	177,231
August	136,173	140,474	149,463	159,059	171,837
September	135,232	134,708	143,363	149,287	168,224
October	123,317	125,835	135,696	145,657	155,216
November	116,100	121,306	125,912	134,052	138,886
December	111,787	114,696	109,096	113,201	131,520

http://www.goodcarbadcar.net/2012/10/canada-overall-auto-industry-sales-figures.html

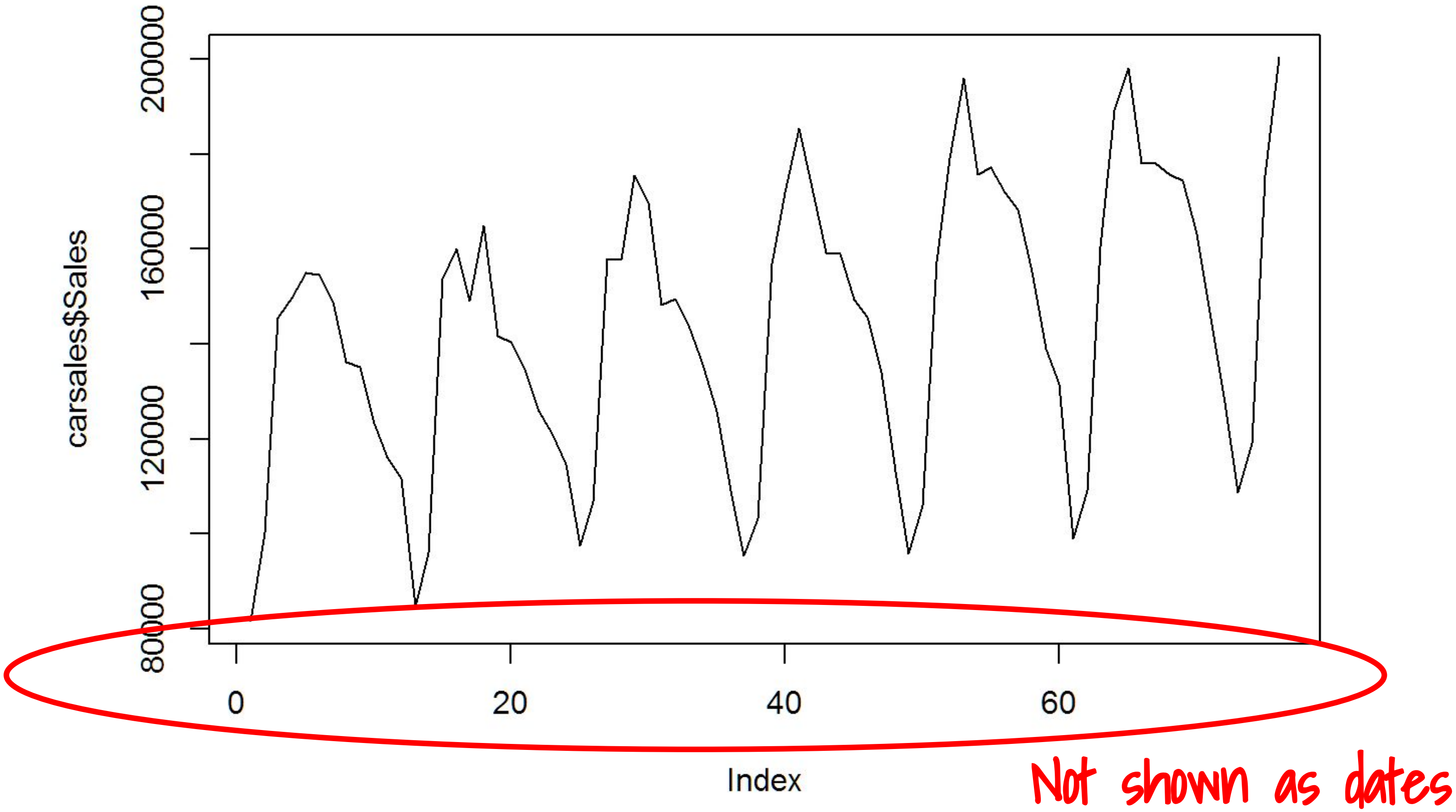
Import from CSV into a data frame

```
> carsales
        Month Year   Sales
1     January 2010  81,693
2    February 2010 100,352
3      March  2010 145,539
4      April  2010 149,757
5        May  2010 155,008
6        June 2010 154,656
7       July  2010 148,865
8     August  2010 136,173
9   September 2010 135,232
10    October 2010 123,317
11   November 2010 116,100
12   December 2010 111,787
13    January 2011  84,599
```

```
> str(carsales)
'data.frame':  76 obs. of  3
variables:
 $ Month: chr  "January" "February"
"March " "April " ...
 $ Year : int  2010 2010 2010 2010
2010 2010 2010 2010 2010 2010 ...
 $ Sales: chr  "81,693" "100,352"
"145,539" "149,757" …
```

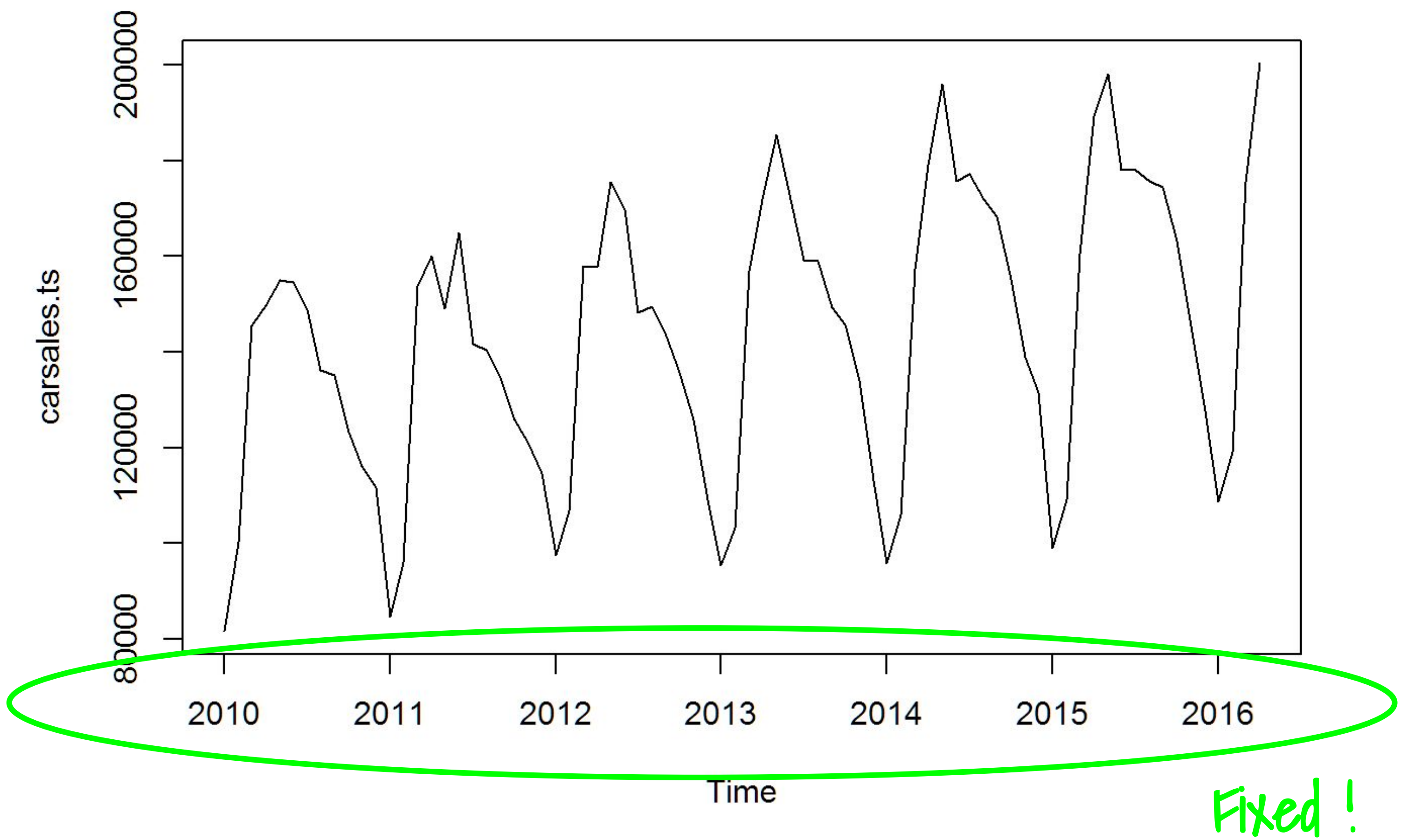
```
carsales$Sales = as.numeric(
  gsub(",", "", carsales$Sales)
)
```

Plot sales over time



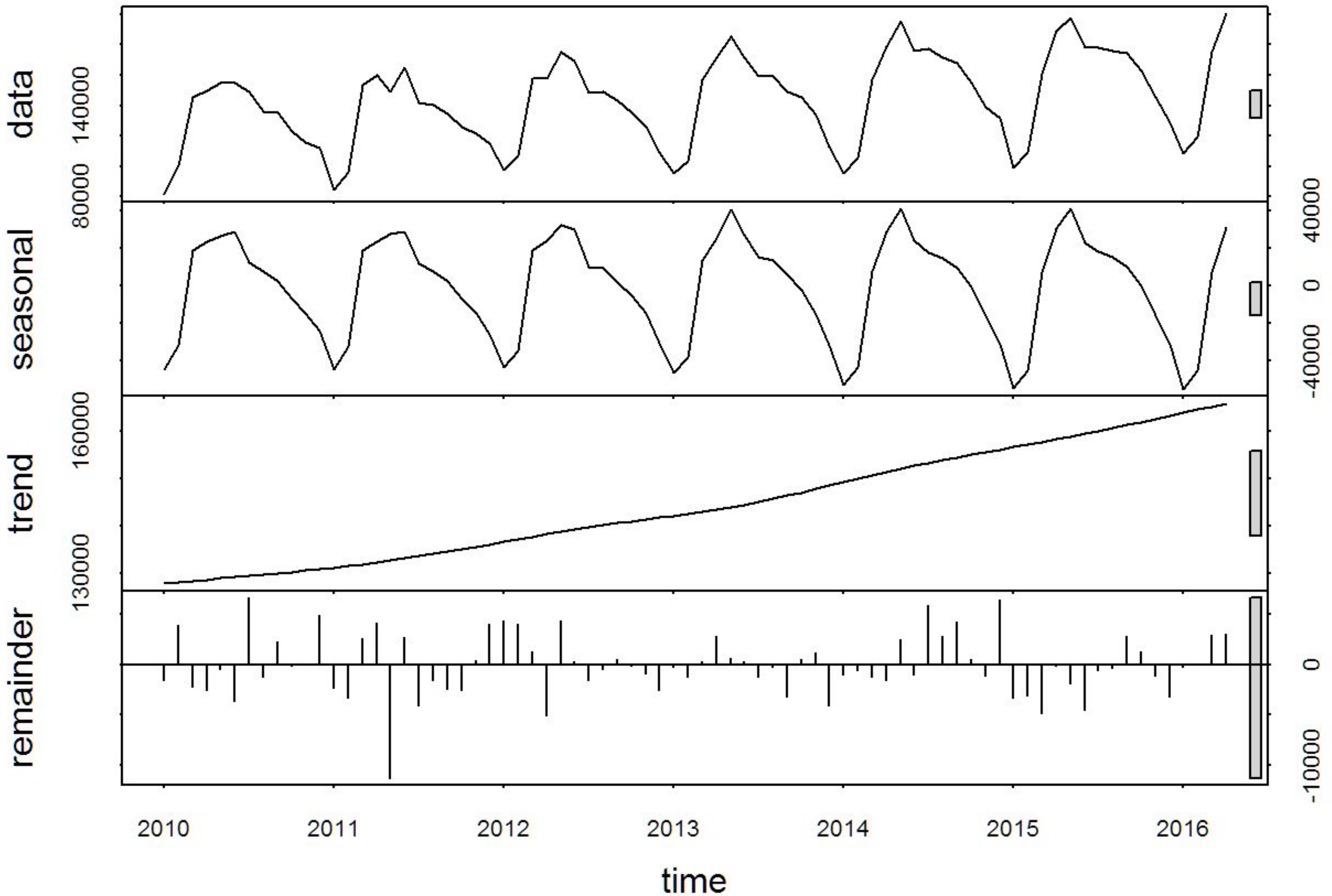

```
plot(carsales$Sales)
```

Let's convert the data frame to a time series



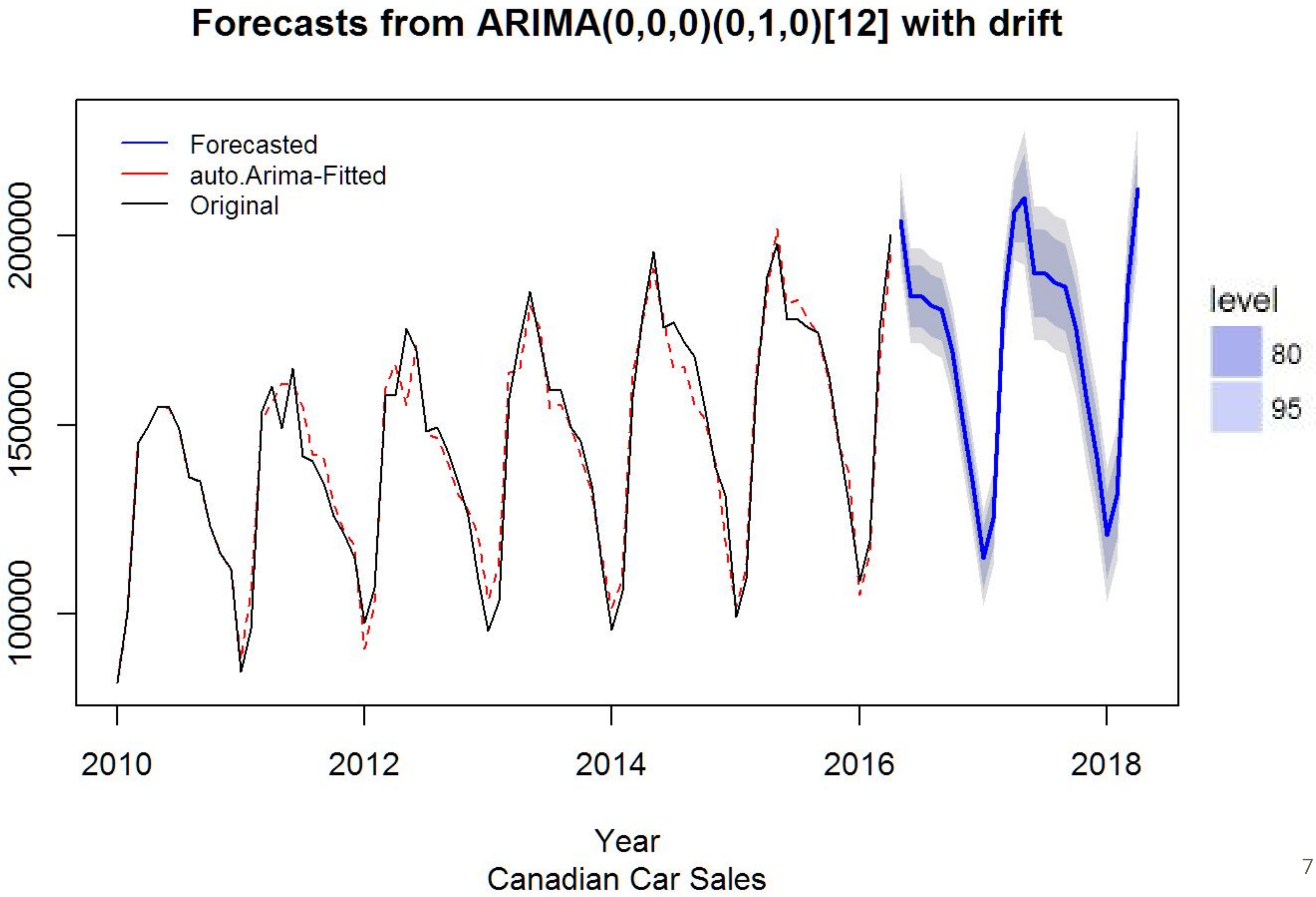

```
carsales.ts = ts(carsales$Sales, frequency=12, start=c(2010,1))
plot(carsales.ts)
```

A time series could be decomposed




```
carsales.stl = stl(carsales.ts, s.window = 4)
plot(carsales.stl)
```

We could predict the future



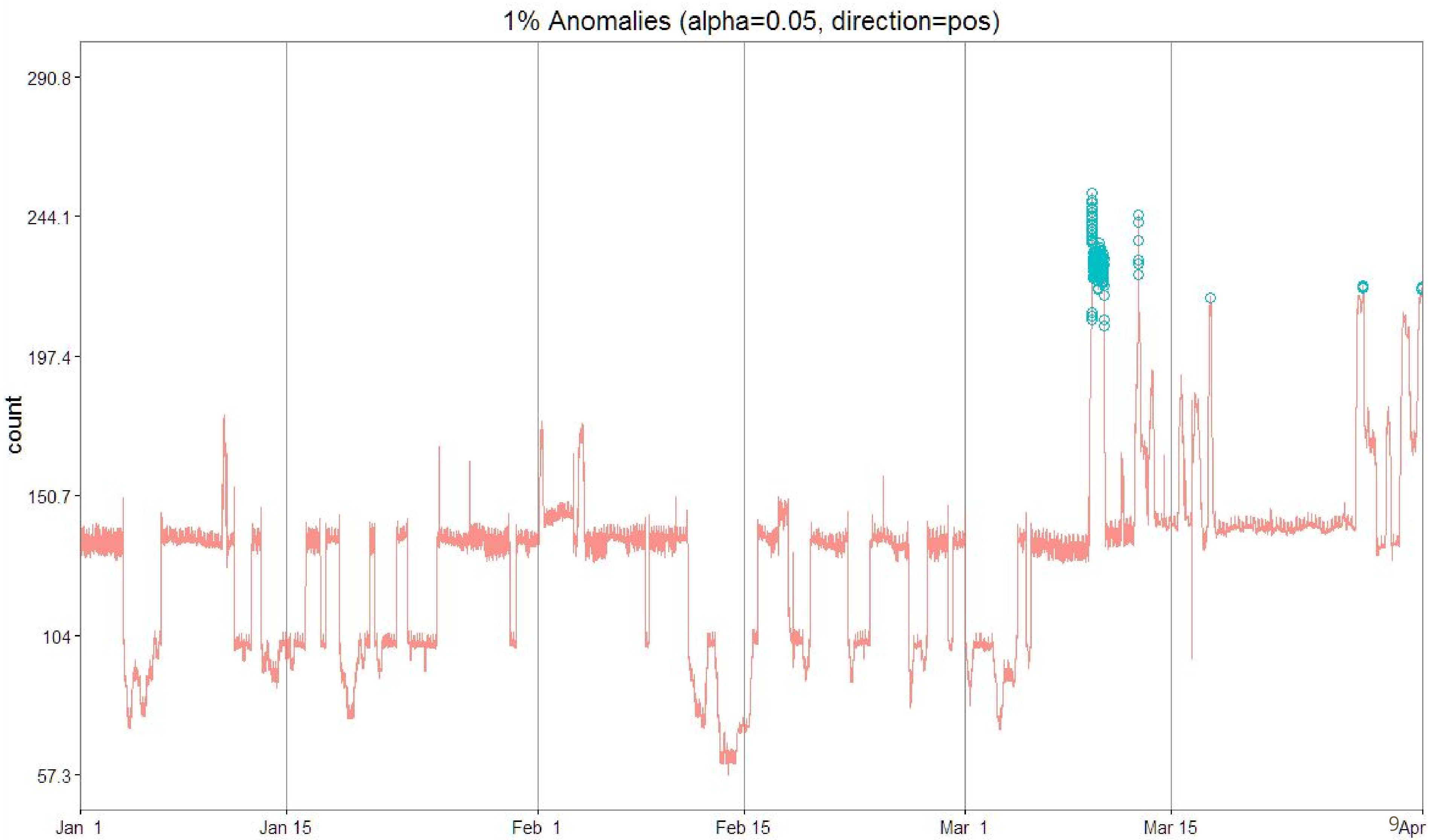
Here is the magic

```
library(forecast)
carsales.forecast = forecast.Arima(auto.arima(carsales.ts))

# Compare actual, auto.arima and forecast

plot.forecast(
  carsales.forecast, col="blue", xlab="Year",
  sub="Canadian Car Sales"
)
lines(carsales.forecast$fitted, col="red", lty=2)
lines(carsales.ts, col="black")
legend(
  'topleft', inset=.02,
  legend=c("Forecasted", "auto.Arima-Fitted", "Original"),
  col=c("blue", "red", "black"),
  lty=1, box.lty=0, cex=0.8
)
```

Anomaly Detection



Sample Anomaly Detection Code

```
library(AnomalyDetection)
library(ggplot2)

# Keep only desired variables
myts = subset(rbind(
  mec_2016_01, mec_2016_02, mec_2016_03),
  select = c("datetime", "kwh"))
attr(myts$datetime, "tzone") = "UTC" # required!

# Plotting data
ggplot(myts, aes(x=datetime, y=kwh, color=kwh)) + geom_line()

## Apply anomaly detection
data_anomaly = AnomalyDetectionTs(myts, max_anoms=0.01,
  direction="pos", plot=TRUE, e_value = T, na.rm = T)

data_anomaly$plot
```

Recap

1. Import data from CSV into a data frame
2. Convert numbers to numeric values
3. Convert data frame into a time series using `ts()`
4. Decompose the time series using `stl()`
5. Forecasting
    1. `auto.arima()`
    2. `forecast.Arima()`
    3. `plot.forecast()`
6. Detect anomalies

But life is not always so easy

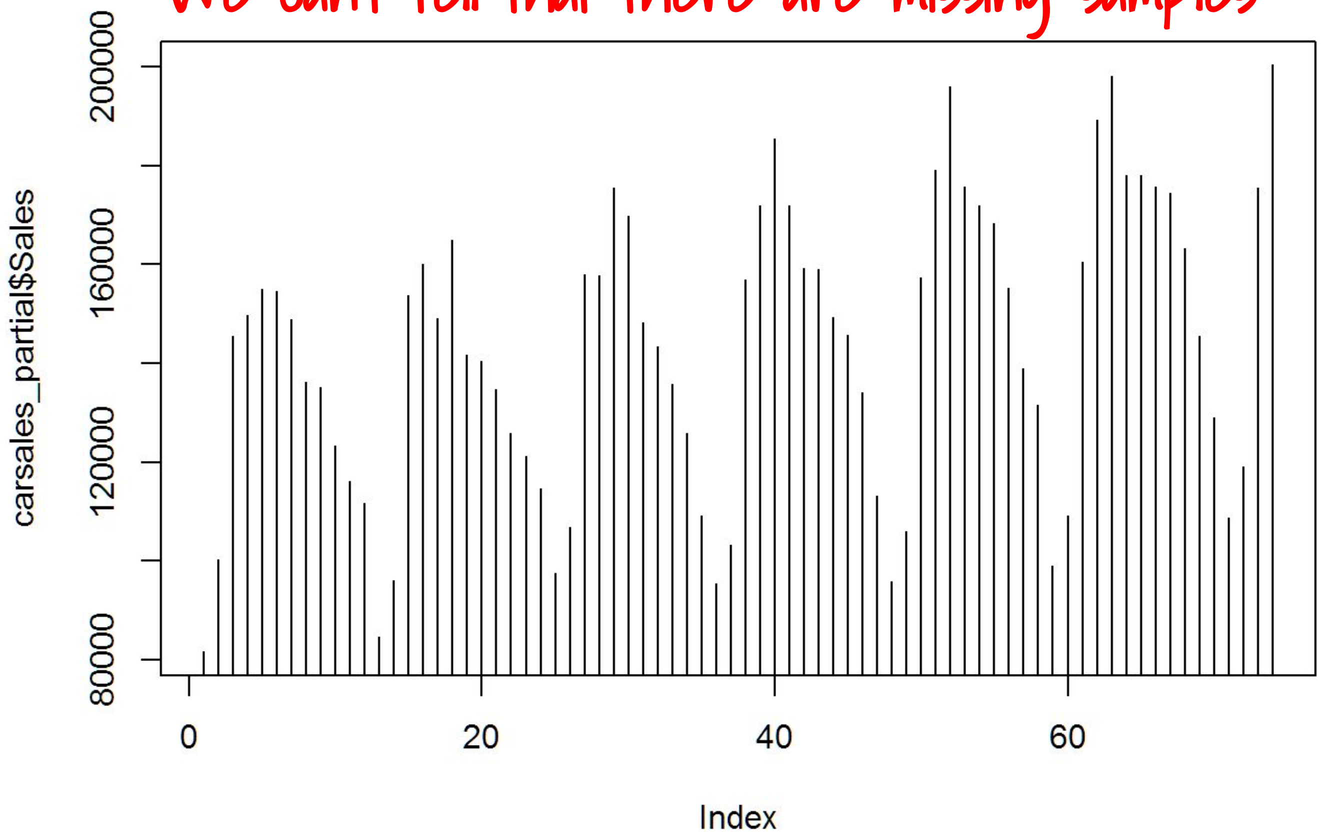
- People have very creative ways to write a date
- Missing samples (happens a lot with sensor data)
  - Sensor breakdown
  - Data collection system breakdown
  - System breakdown
  - Forgot to collect data
- Duplicate samples
- Samples are not collected at regular intervals
  - Timestamps mismatch, clocks not synchronized
  - US GSS survey data yearly then every 2 years
- It looks like a time series but it is not

We don’t have any such problems with the car sales dataset

Let’s create chaos in carsales data frame

```
carsales_partial = carsales[-c(32, 55),]
plot(carsales_partial$Sales, type='h')
```

We can't tell that there are missing samples



We can no longer use ts() to convert to time series

`ts()` expect frequency= and start time. It assumes that there is no gap in the sequence

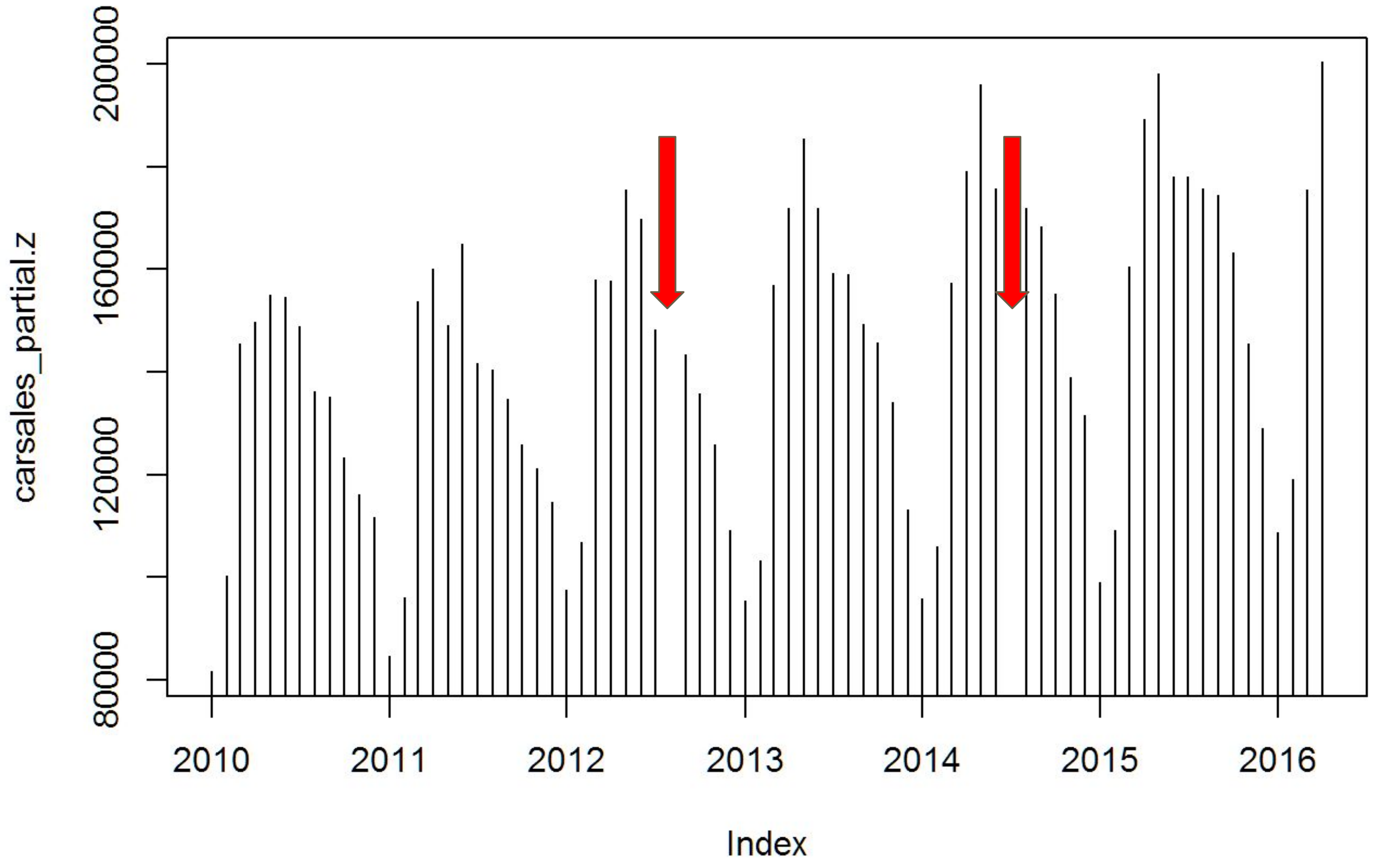
The `zoo()` function will handle gaps by requiring a time index column to go with the data value column

But… many time series functions still expect a perfect timeseries to process. The zoo package provides more functions to regularize a non-regular time series back into a ts series

Convert to a time series using zoo()

```
carsales_partial$date = as.Date(
  paste0(
    carsales_partial$Year,'-',
    trimws(carsales_partial$Month),
    '-01'
  ),
  format="%Y-%B-%d"
)
carsales_partial.z = zoo(
  carsales_partial$Sales,          # value column
  order.by=carsales_partial$date,  # datetime index column
  frequency=12
)
plot(carsales_partial.z, type="h")
```

Time Series with Missing Samples



Dealing with missing samples

It doesn't make sense to create a ts() series with missing samples

zoo() can handle missing samples but not NA’s

Time decomposition with decompose() works with zoo objects with missing samples but not with NA’s

Time decomposition with stl() only works with ts objects

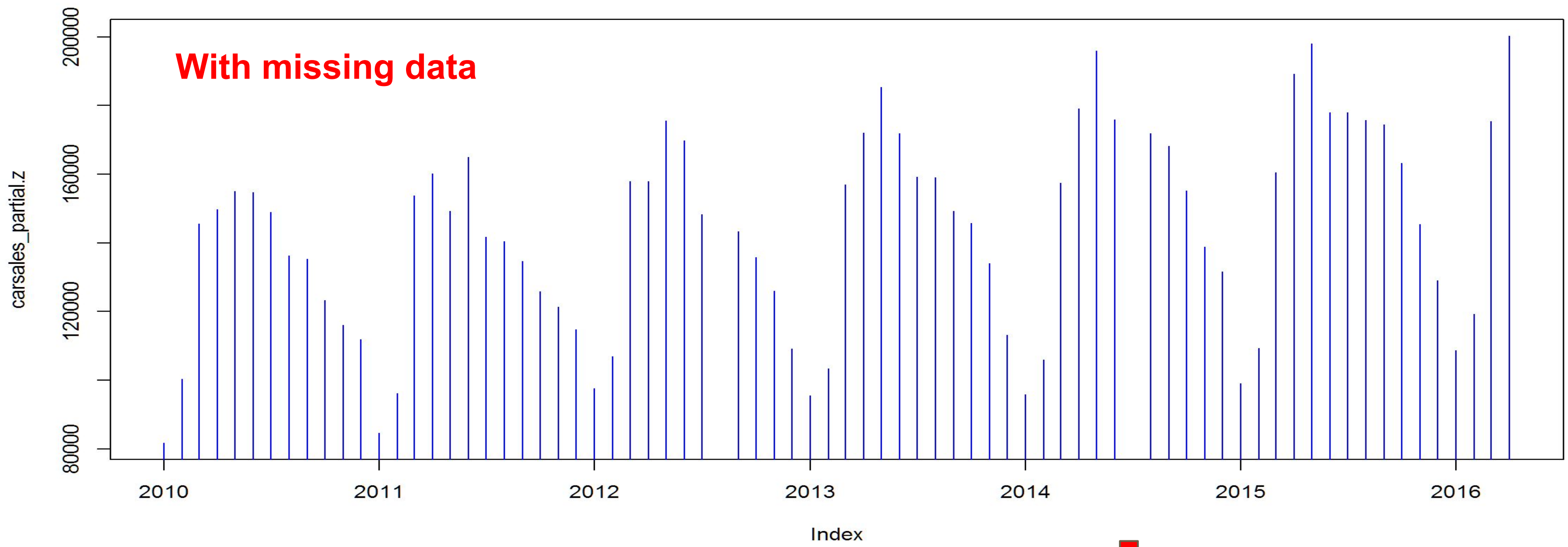
Arima does not check for missing samples so we have to.

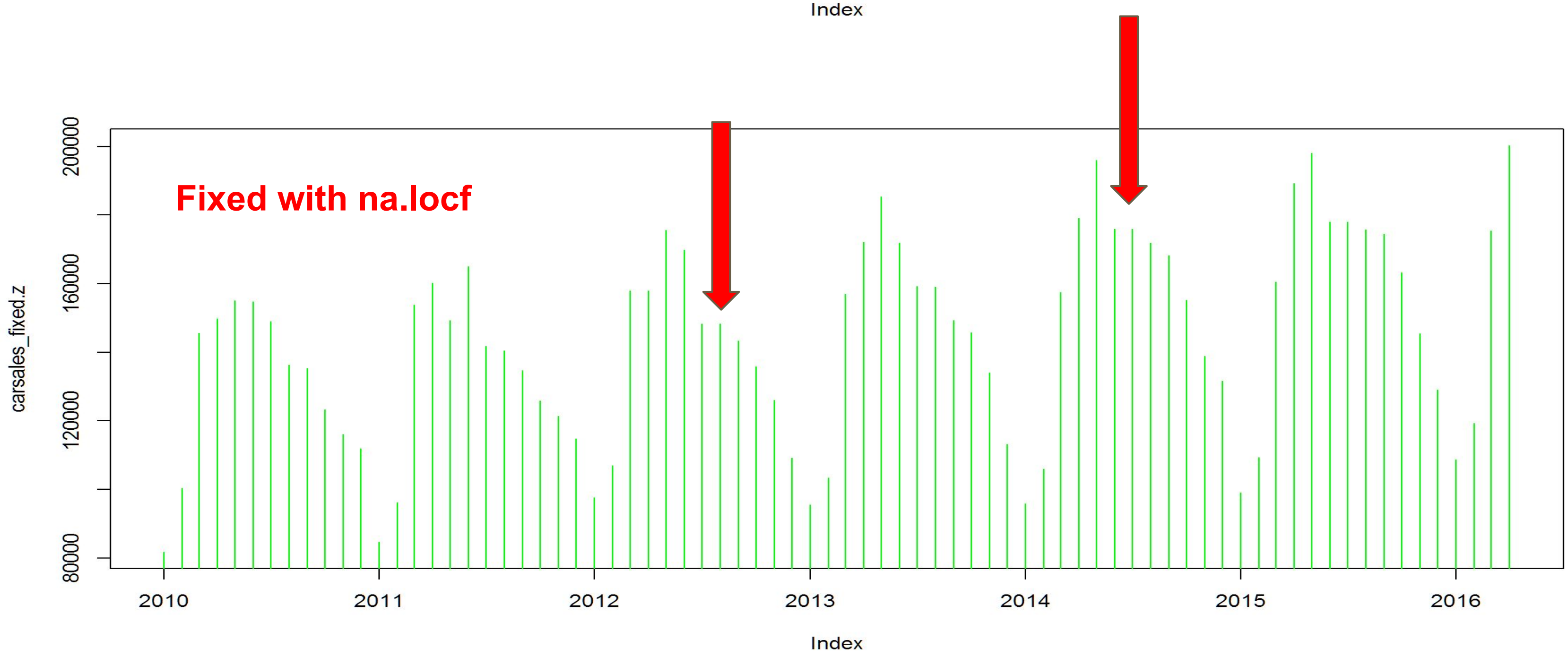
Fix NA’s by imputing

Fix NA's or missing samples

```
# generate a sequence of hourly intervals
g = seq(start(carsales_partial.z), end(carsales_partial.z), by="
month")
# Last Observation Carried Forward
carsales_fixed.z = na.locf(carsales_partial.z, xout=g)

par(mfrow=c(2,1))
plot(carsales_partial.z, col="blue", type='h')
plot(carsales_fixed.z, col="green", type='h')
```


With missing data
carsales_partial.z
80000
120000
160000
200000
2010
2011
2012
2013
2014
2015
2016
Index


Fixed with na.locf
carsales_fixed.z
80000
120000
160000
200000
2010
2011
2012
2013
2014
2015
2016
Index

zoo() decompose

```
carsales_fixed.decompose = decompose(
  ts(
    carsales_fixed.z,
    frequency = 12,
    start=c(2010,1)
  ),
  "additive"
)
#
plot(carsales_fixed.decompose, xlab="Year")
title(sub=paste(sub_text,collapse="  to  "))
```

Arima on a zoo object

```
carsales_fixed.z.forecast =
  forecast.Arima(auto.arima(coredata(carsales_fixed.z)))

# Compare actual, auto.arima and forecast
plot.forecast(carsales_fixed.z.forecast, col="blue")
lines(carsales_fixed.z.forecast$fitted, col="red", lty=2)
lines(carsales.ts, col="black")
legend(
  'topleft', inset=.02,
  legend=c("Forecasted", "auto.Arima-Fitted", "Original"),
  col=c("blue", "red", "black"),
  lty=1, box.lty=0, cex=0.8
)
```

References

https://www.youtube.com/watch?v=gHdYEZA50KE

https://www.youtube.com/watch?v=xBP4cQetoNM

http://rdc.uwo.ca/events/docs/presentation_slides/2012-13/Medovikov-AppliedTime2013.pdf

https://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#arima-models

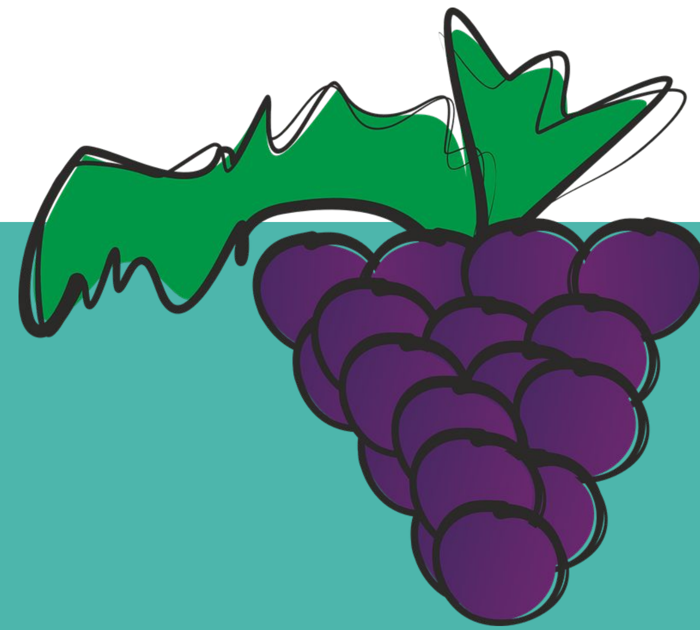
http://www.sapub.org/global/showpaperpdf.aspx?doi=10.5923/j.statistics.20140406.05

http://www.r-bloggers.com/anomaly-detection-in-r/

https://github.com/pablo14/anomaly_detection_post

http://r-exercises.com/2016/05/30/zoo-time-series-exercises/

Bonus Section
(Not covered in the presentation)



Breakins in Montreal since January 2015

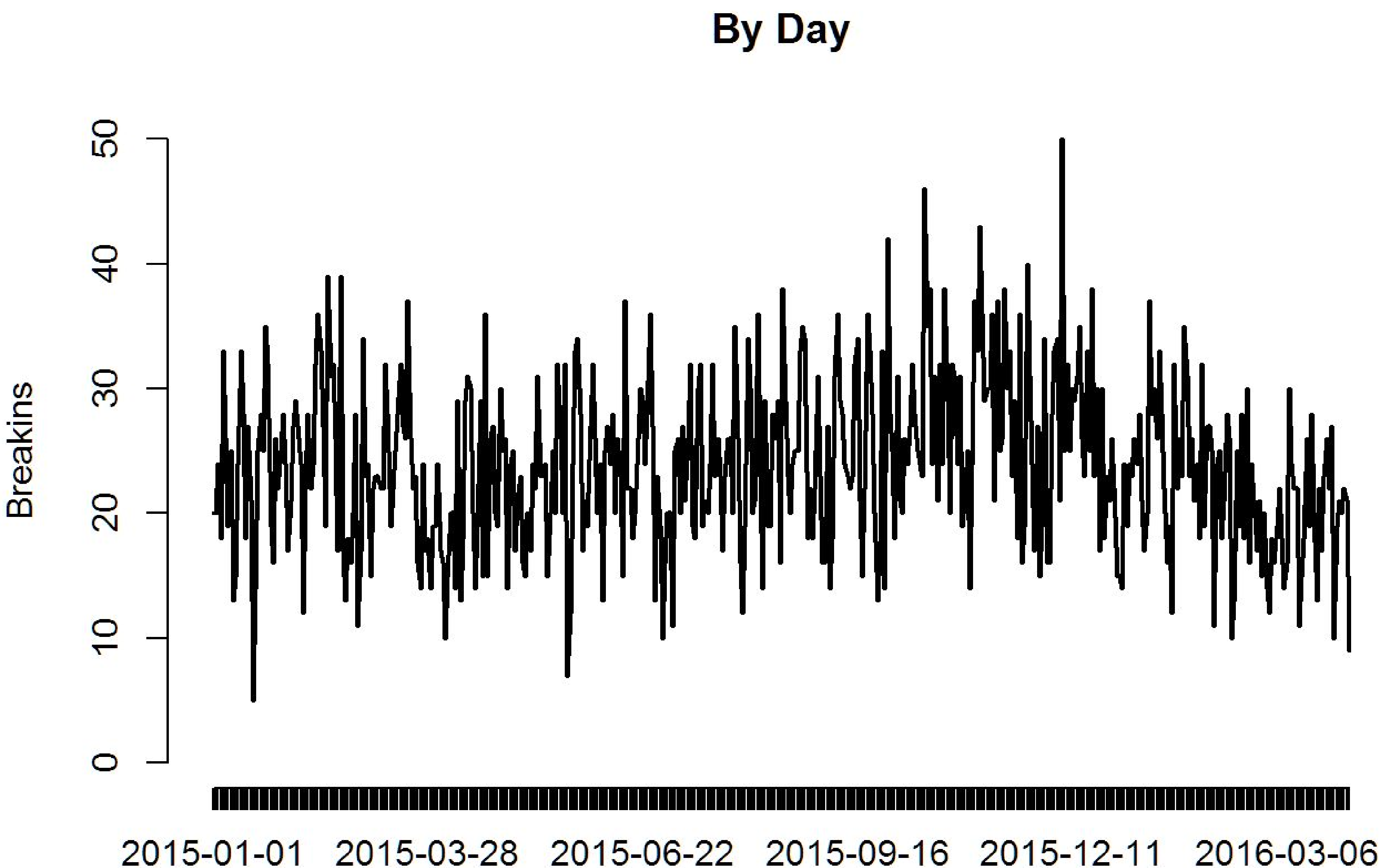
http://donnees.ville.montreal.qc.ca/dataset/actes-criminels

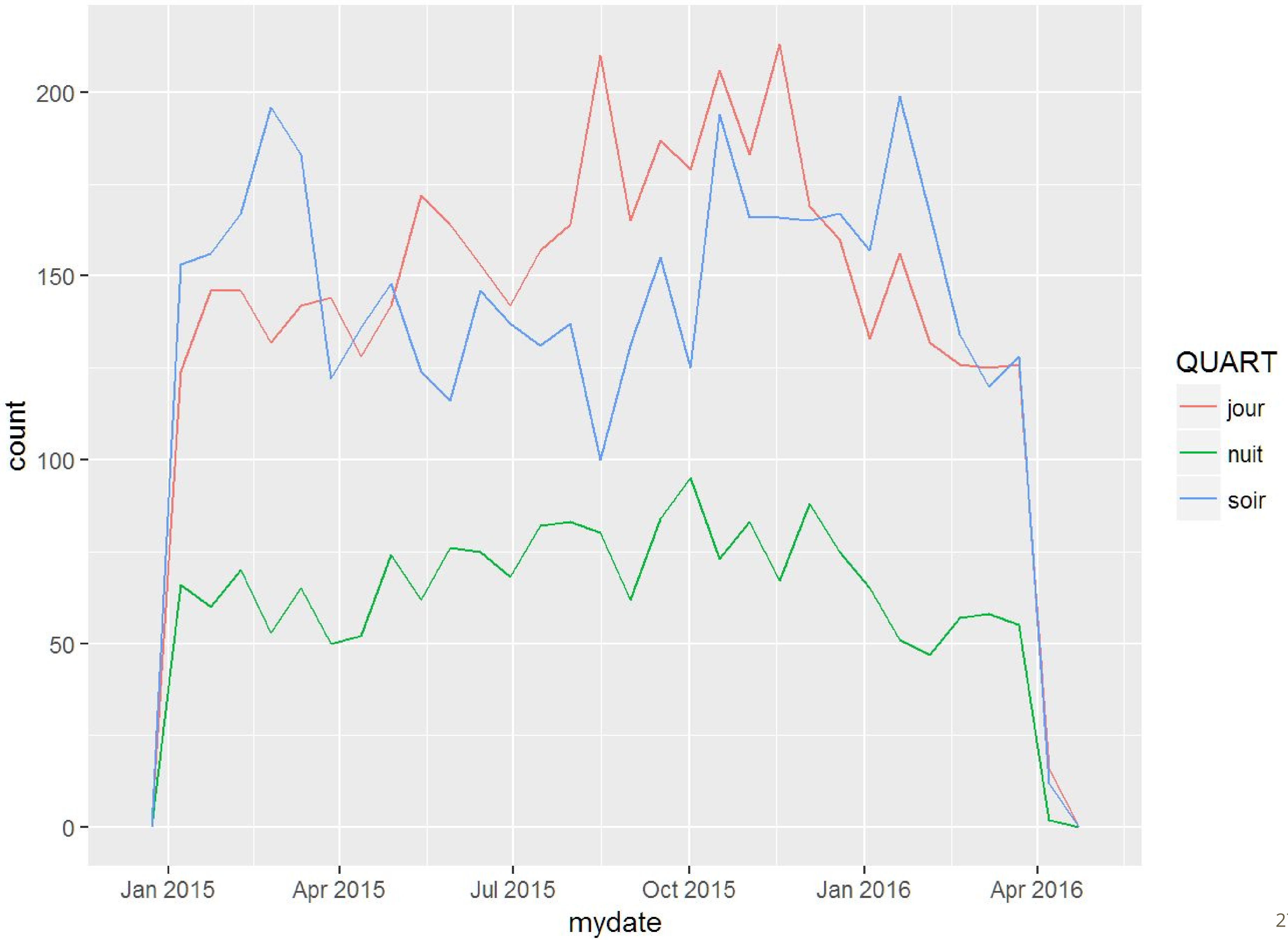
https://github.com/2uanta/mtl_donnees_ouvertes/blob/master/Breakins.Rmd

Input Data

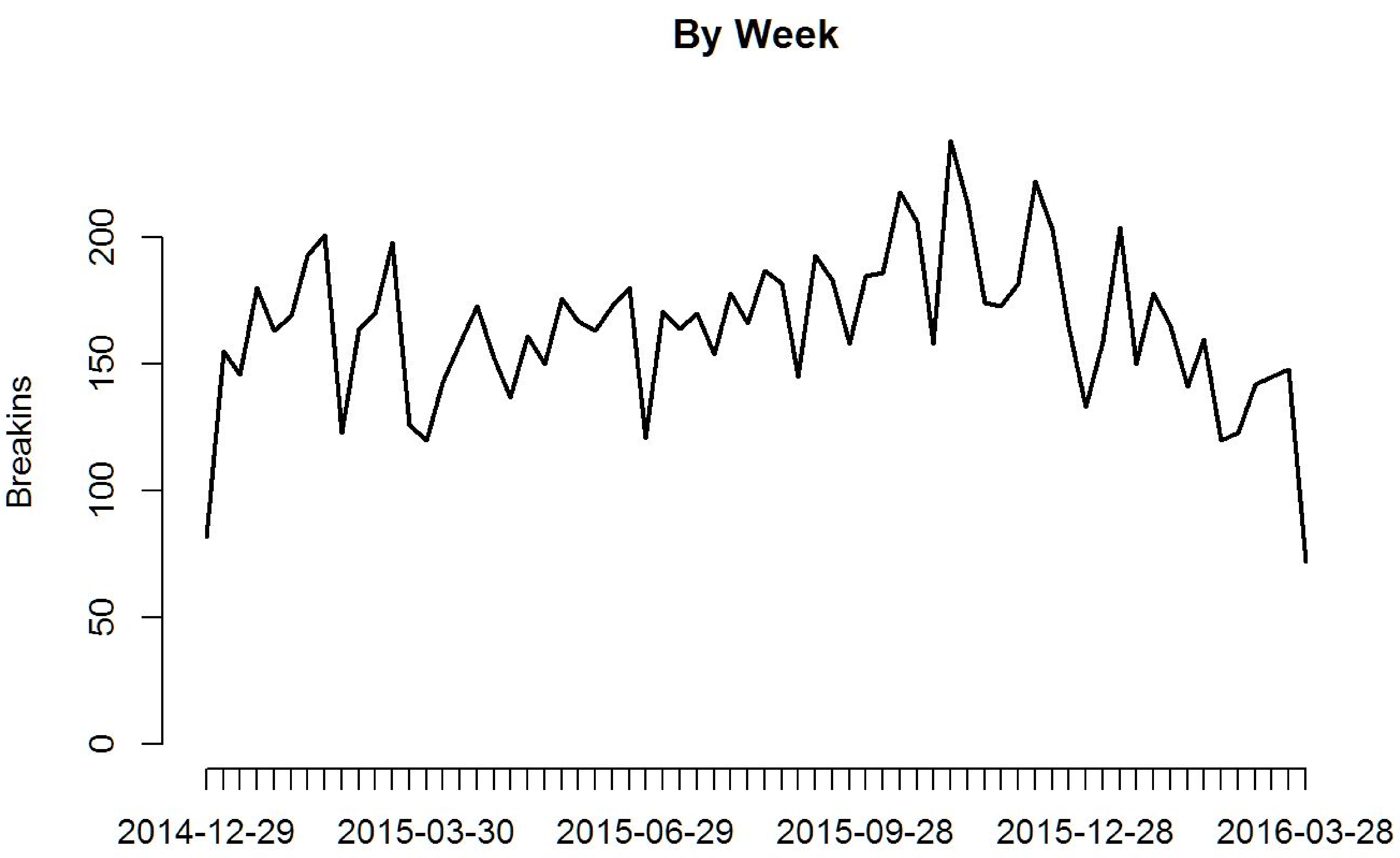
```
DT = fread(input = "donneesouvertes-citoyens.csv", header = T,
sep=",", stringsAsFactors = T)
```

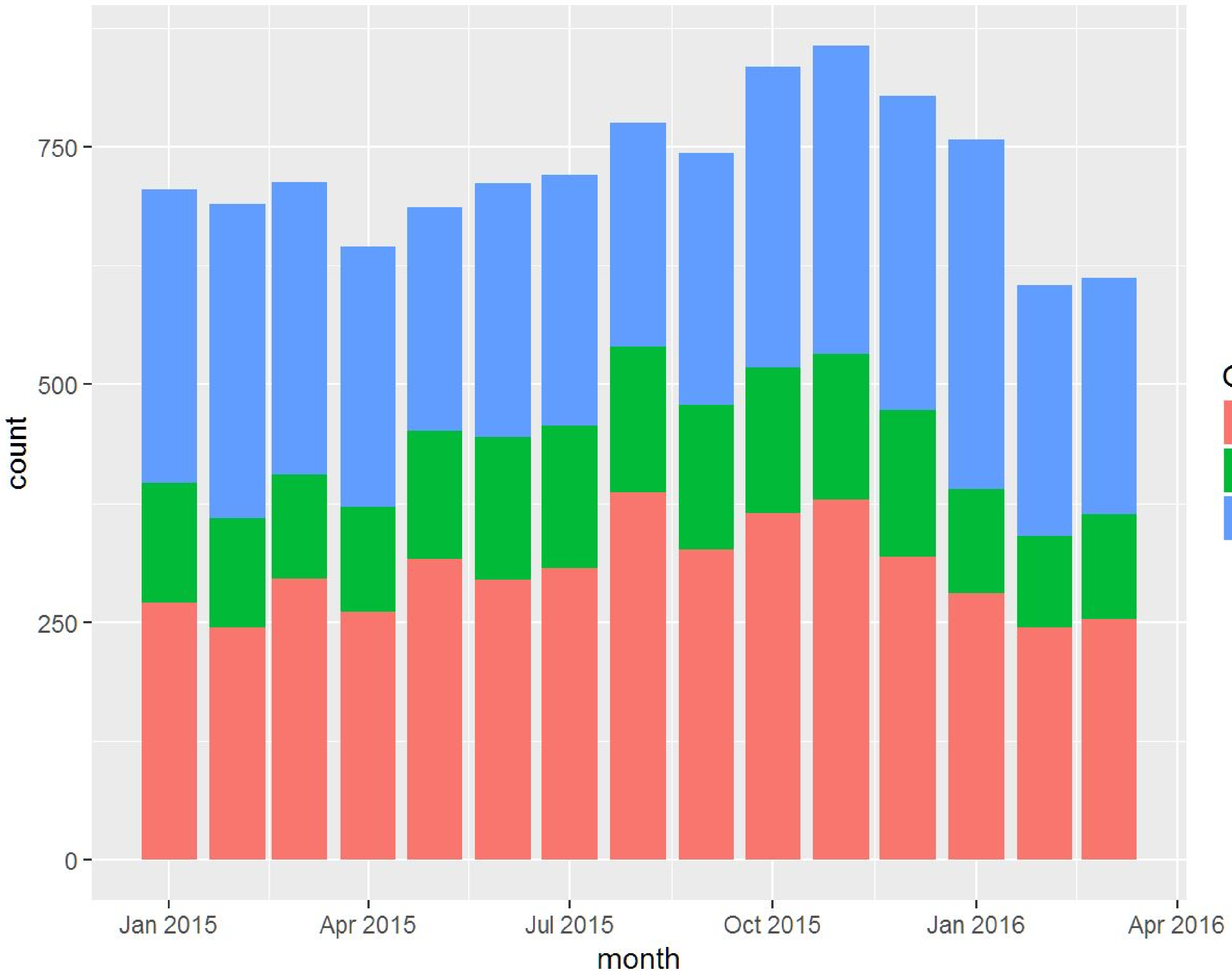
```
##  CATEGORIE   DATE   QUART PDQ        X       Y       LAT     LONG
Introduction 20150101  nuit   8 289215.1 5036423 45.46756 -73.6993
Introduction 20150101  nuit  48 302729.3 5050946 45.59841 -73.5265
##    ---
Introduction 20160331  soir  16 299214.3 5035799 45.46210 -73.5714
Introduction 20151118  jour  26 295080.9 5041034 45.50916 -73.6243
```


By Day
Breakins
0
10
20
30
40
50
2015-01-01
2015-03-28
2015-06-22
2015-09-16
2015-12-11
2016-03-06

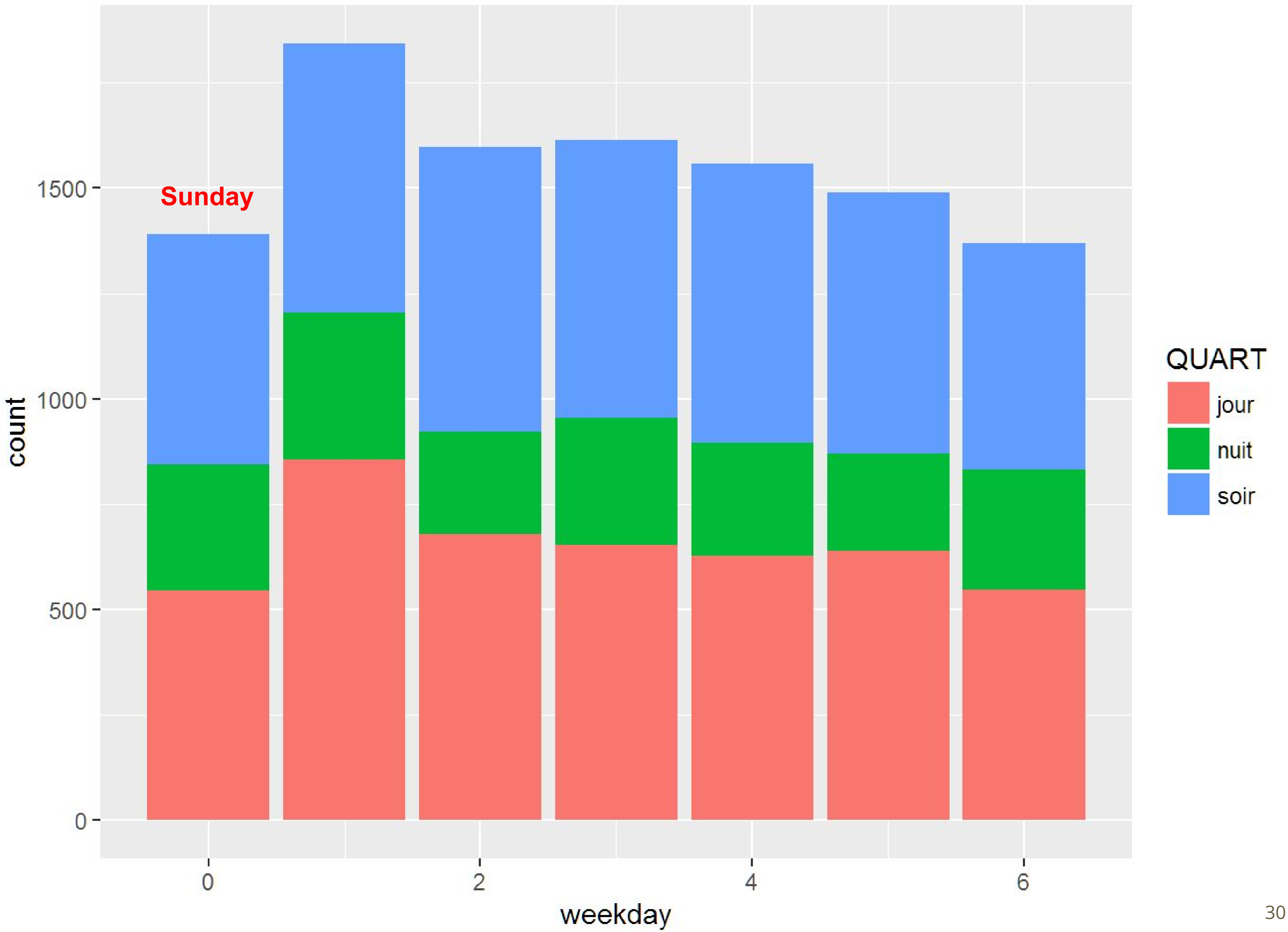


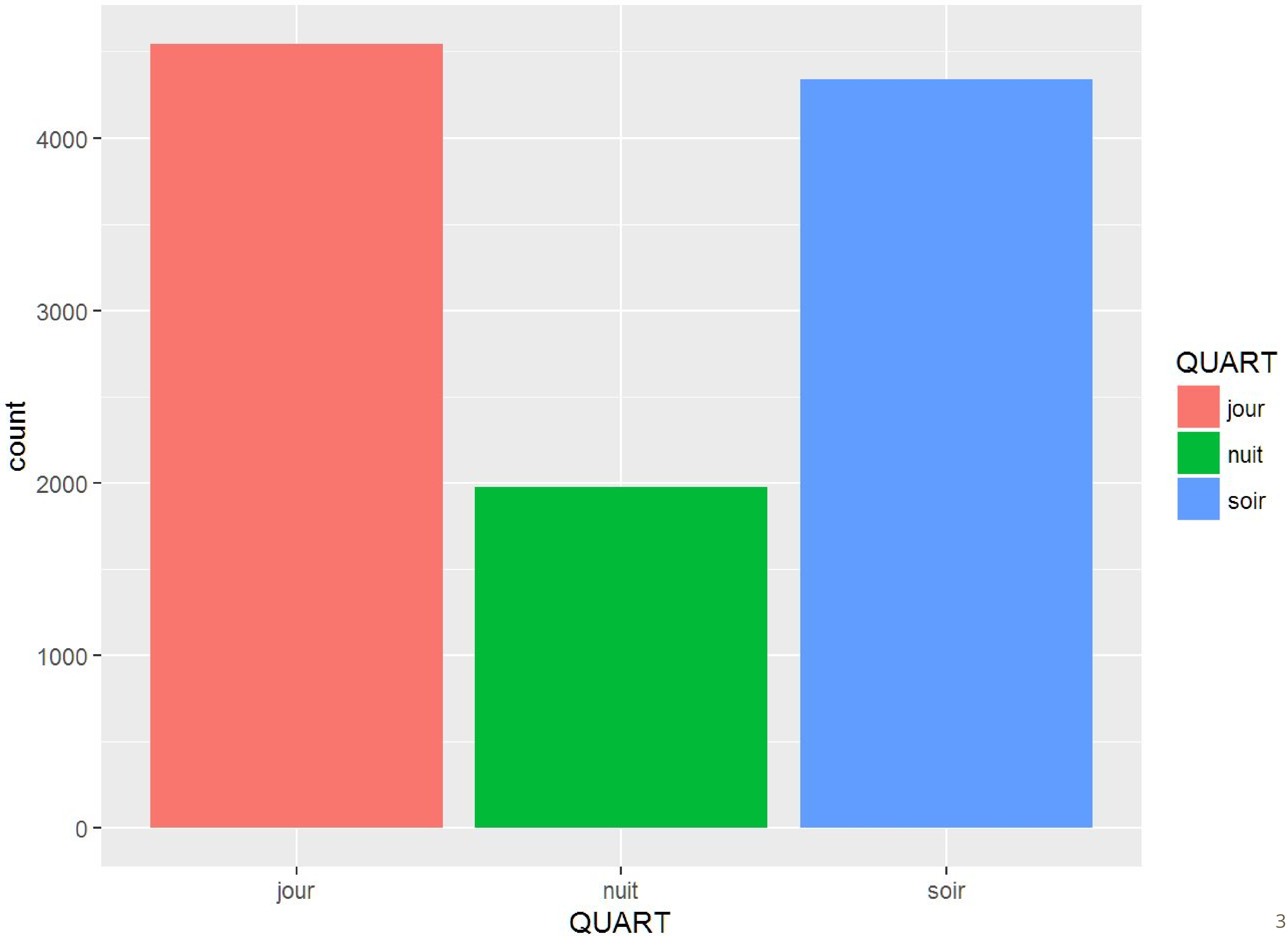
200
150
100
50
0
count
Jan 2015
Apr 2015
Jul 2015
Oct 2015
Jan 2016
Apr 2016
mydate
QUART
jour
nuit
soir

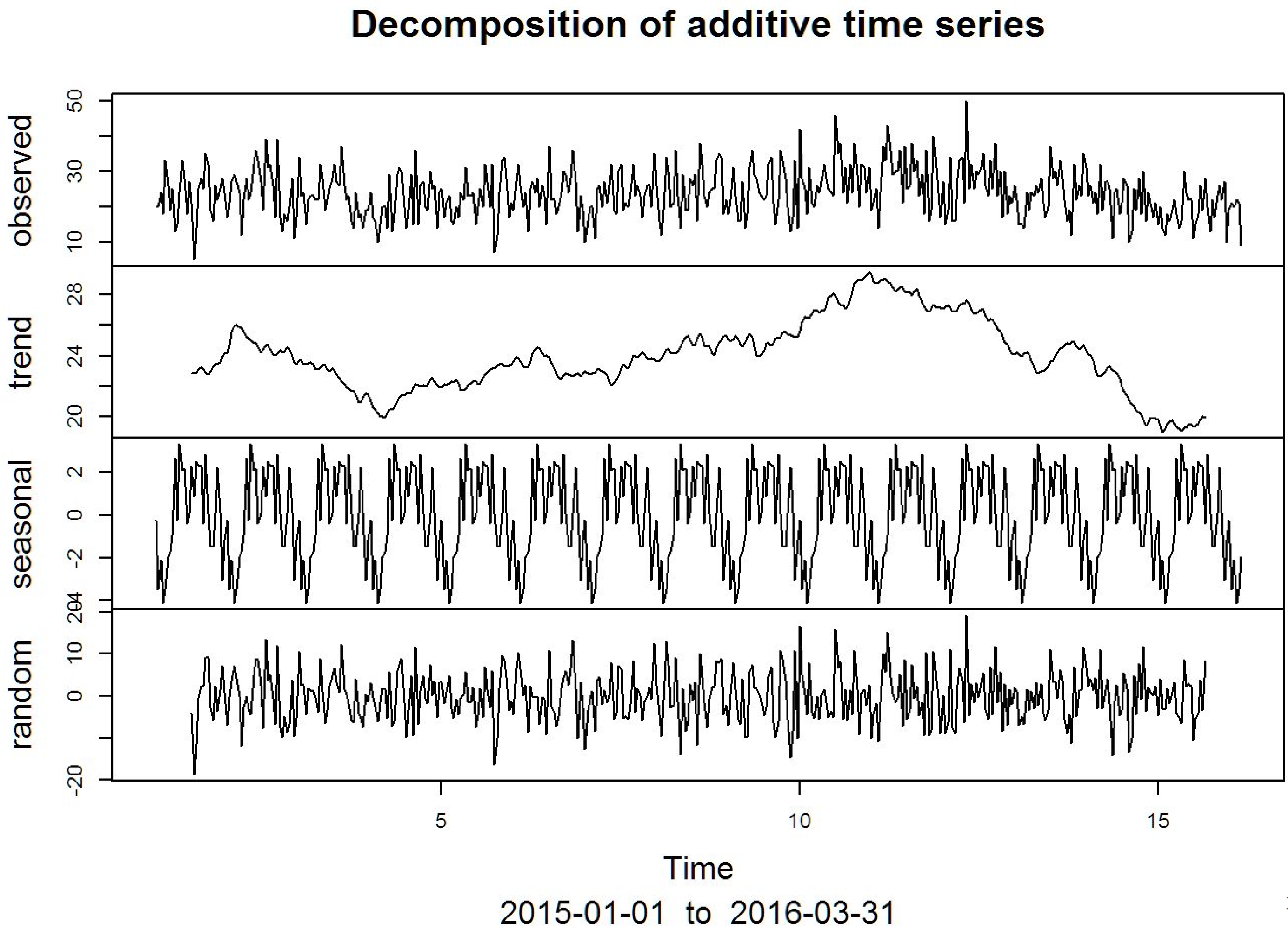

By Week
Breakins
0
50
100
150
200
2014-12-29
2015-03-30
2015-06-29
2015-09-28
2015-12-28
2016-03-28

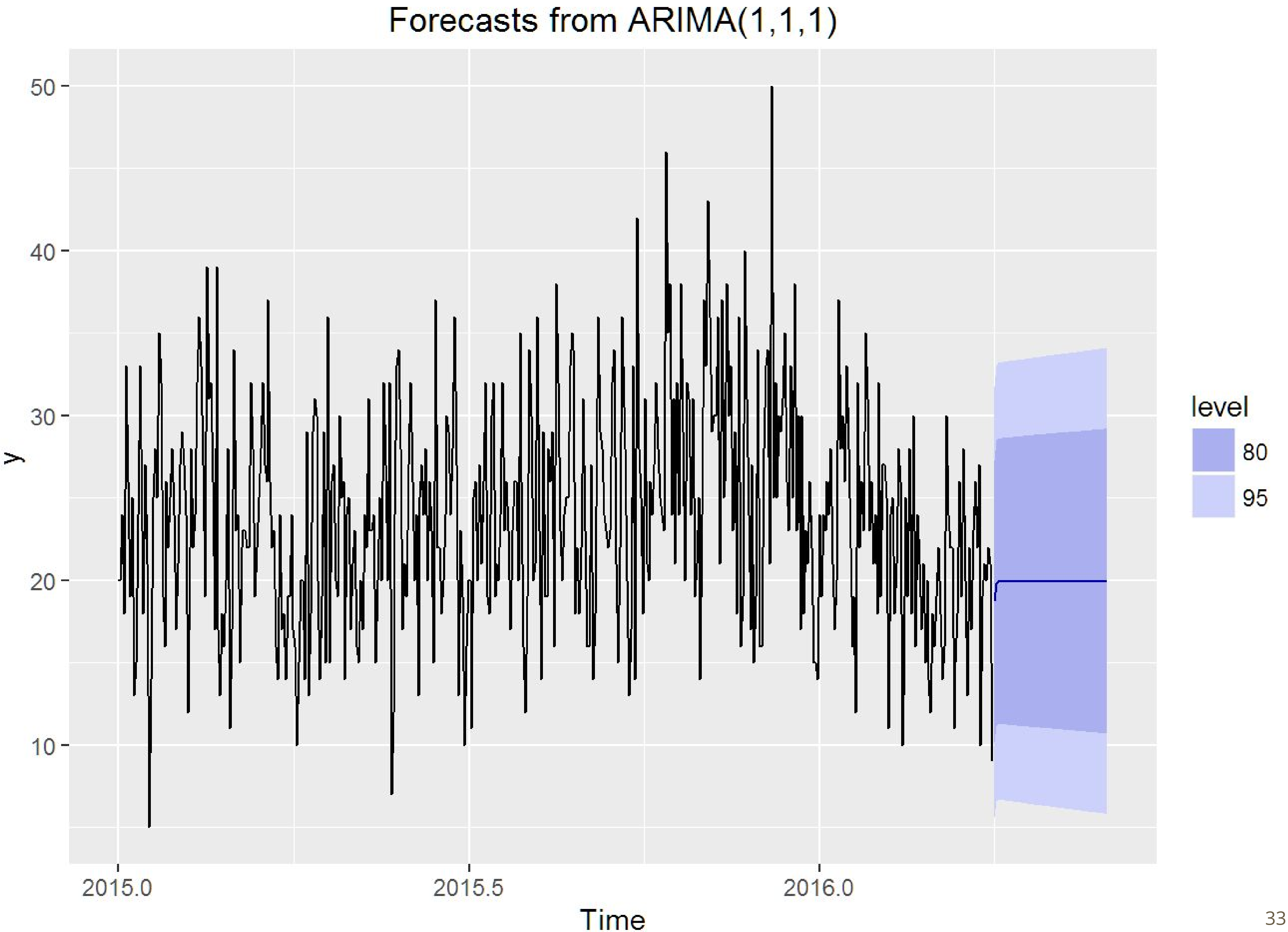


count
750
500
250
0
Jan 2015
Apr 2015
Jul 2015
Oct 2015
Jan 2016
Apr 2016
month
QUART
jour
nuit
soir

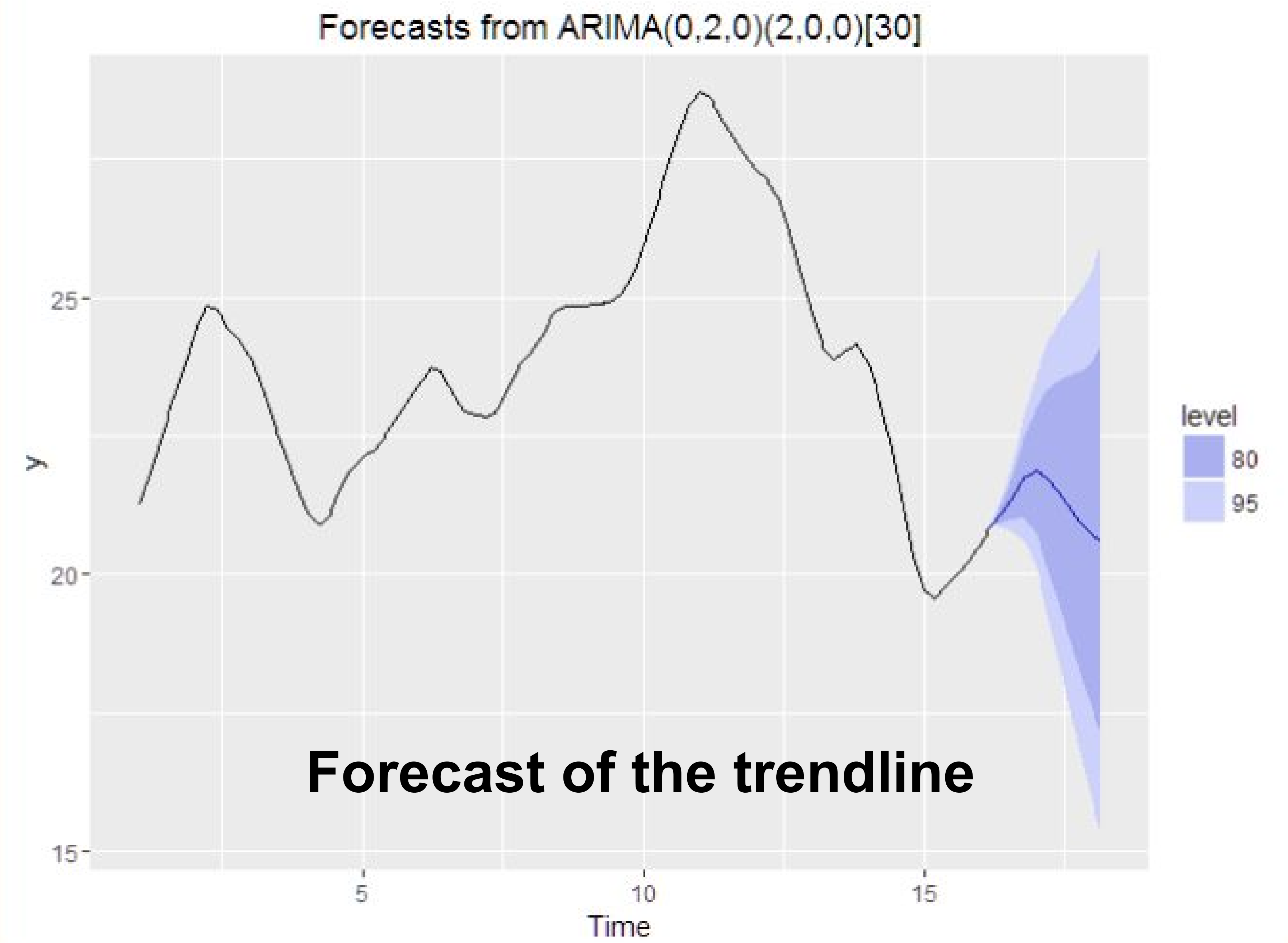

Sunday
1500
1000
500
0
count
0
2
4
6
weekday
QUART
jour
nuit
soir


4000
3000
2000
1000
0
count
jour
nuit
soir
QUART
QUART
jour
nuit
soir


Decomposition of additive time series
observed
trend
seasonal
random
50
30
10
28
24
20
2
0
-2
-4
20
10
0
-20
5
10
15
Time
2015-01-01 to 2016-03-31

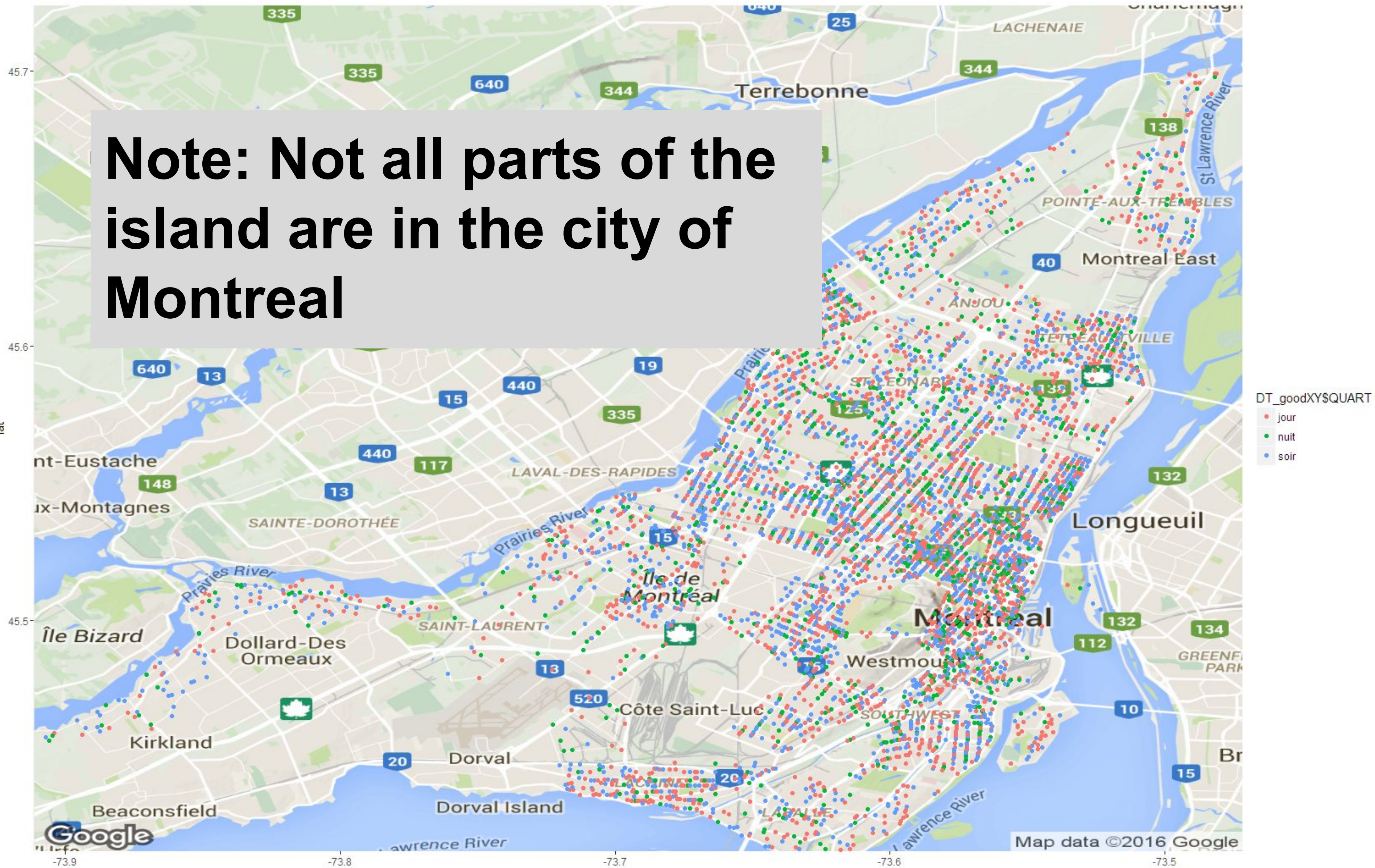


Forecasts from ARIMA(1,1,1)
50
40
30
20
10
y
2015.0
2015.5
2016.0
Time
level
80
95


Forecasts from ARIMA(0,2,0)(2,0,0)[30]
y
25
20
15
5
10
15
Time
level
80
95
Forecast of the trendline

https://2uanta.cartodb.com/viz/802d9bf2-2787-11e6-96d7-0e31c9be1b51/public_map

Note: Not all parts of the island are in the city of Montreal



More code samples

Date and Time Formats

Date and time values are stored in internal format which is the number of days since 1-jan-1970

Use as.Date, as POSIXct, POSIXlt to convert character strings into internal R format that `ts()` or `zoo()` require:

- Use the format option to specify a particular format (similar to strftime function in C)

Read.xlsx with read Excel datetime format into R datetime format directly

```r
my.dates = c("2013-12-19", "2003-12-20")
date1 = as.Date(my.dates)

date2 = as.Date("1/15/1970", format="%m/%d/%Y")
date3 = as.Date("January 10, 1970", format="%B %d, %Y")
date4 = as.Date("01JAN70", format="%d%b%y")

as.Date(32500, origin=as.Date("1900-01-01"))

diff.date = date2 - date3

mydate1 = as.Date("2015-08-10")
mydate2 = as.Date("2015-08-12")
mydate2 - mydate1
  Time difference of 2 days
```

creating a sequence

```r
my.dates = seq(as.Date("1993/3/1"), as.Date("2003/3/1"),
  "2 months")
```

```
myDateTimeStr = "2013-12-19 10:17:07"
myPOSIXct = as.POSIXct(myDateTimeStr)

myDateTimeStr1 = "19-12-2003 10:17:07"
myPOSIXct1 = as.POSIXct(myDateTimeStr1, format="%d-%m-%Y %H:%M:%S")
```

Non-standard datetime format

```
myPOSIXct3 = myPOSIXct2 + 8*60*60


myPOSIXct5 = ISOdatetime(year=2013, month=12, day=19,
  hour = 10, min = 17, sec = 7,   tz = "")


format(myPOSIXct, format="%b %d, %Y")
```

Get year as a number value

```
as.numeric(format(myPOSIXct, format="%Y"))
```

POSIXct is a char string. POSIXlt is a structure.

Also check out `library(lubridate)`

```
data$datetime = as.POSIXlt(data$datetime,tz="EST",format="%Y/%m/%d %H:%M:%S")
data$year = as.Date(cut(as.Date(data$datetime), breaks="year"))
data$quarter = as.Date(cut(as.Date(data$datetime), breaks="quarter"))
data$month = as.Date(cut(as.Date(data$datetime), breaks="month"))
data$week = as.Date(cut(as.Date(data$datetime), breaks="week"))
data$day = as.Date(cut(as.Date(data$datetime), breaks="day"))
data$hour = as.POSIXlt(cut(as.POSIXlt(data$datetime,format="%Y-%m-%d %H:%M:%S"), breaks="hour"), tz="EST", format="%Y-%m-%d %H:%M:%S")
```

Frequency parameter in ts()

Data	**Frequency = number of samples per season**
Annual	1
Quarterly	4
Monthly	12
Weekly	52

Use msts() if you need to specify the period and season

Aggregate in time series

```
> by(data$watt,data$week,summary)
data$week: 2014-10-20
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 367200  543600  553700  537500  562400  588200 
----------------------------------------------------------
data$week: 2014-10-27
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 531500  547400  555000  554900  562500  580100 
----------------------------------------------------------
...
> by(data$watt,data$week,sum)
data$week: 2014-10-20
[1] 70415533
```

ggplot() options

```
ggplot(data=data, aes(datetime, watt)) +
  geom_line() +
  scale_y_continuous(labels = comma) +
  scale_x_datetime(labels = date_format("%b %Y"),
       breaks="1 month") +
  xlab(paste(sub_text,collapse="  to  ")) +
  ylab("bits") +
  ggtitle("Server Power Consumption")
```

Remove Duplicate Observations

```
# fix time :59 import from Excel
data$datetime = round(data$datetime)

# sort it again
data = data[order(data$datetime),]

# anyDuplicated will give the index-1 of the first duplicated
record
anyDuplicated(data["datetime"])

# remove duplicates
data = data[!duplicated(data$datetime),]
```

Deal with NA's and missing samples

```
# convert to zoo format
dt_all = as.POSIXlt(all$datetime)
all.z = zoo(x=all$kwh,order.by=dt_all)
# find NA's
head(subset(all.z, is.na(all.z)))

# Many options to fix
na.locf(newIRTS)                # Last observation is carried forward
na.fill(newIRTS, "extend")      # Linear interpolation
na.approx(newIRTS)              # Linear Interpolation
na.spline(newIRTS)              # Cubic Spline Interpolation
na.StructTS(newIRTS)            # Interpolates using
                                  Seasonal Kalman Filter
```

Review: Conditions for linear regression

I. linearity and additivity of the relationship between dependent and independent variables
II. statistical independence of the errors (in particular, no correlation between consecutive errors in the case of time series data)
III. homoscedasticity (constant variance) of the errors
IV. normality of the error distribution.

Linear Regression applied to timeseries

Timeseries are usually nonlinear and do not satisfy these conditions

There are techniques/models to transform a timeseries into a different series that satisfy the linear regression conditions

Once transformed, one can apply linear regression methodology to do the forecast

ARIMA model

Most popular model

Recognize the fact that timeseries is composed of: Trend, Seasonality, White noise

MA (Moving Average) → Trend (q)

AR (AutoRegressive) → Seasonality (p)

I (Integrated) → White Noise (d)

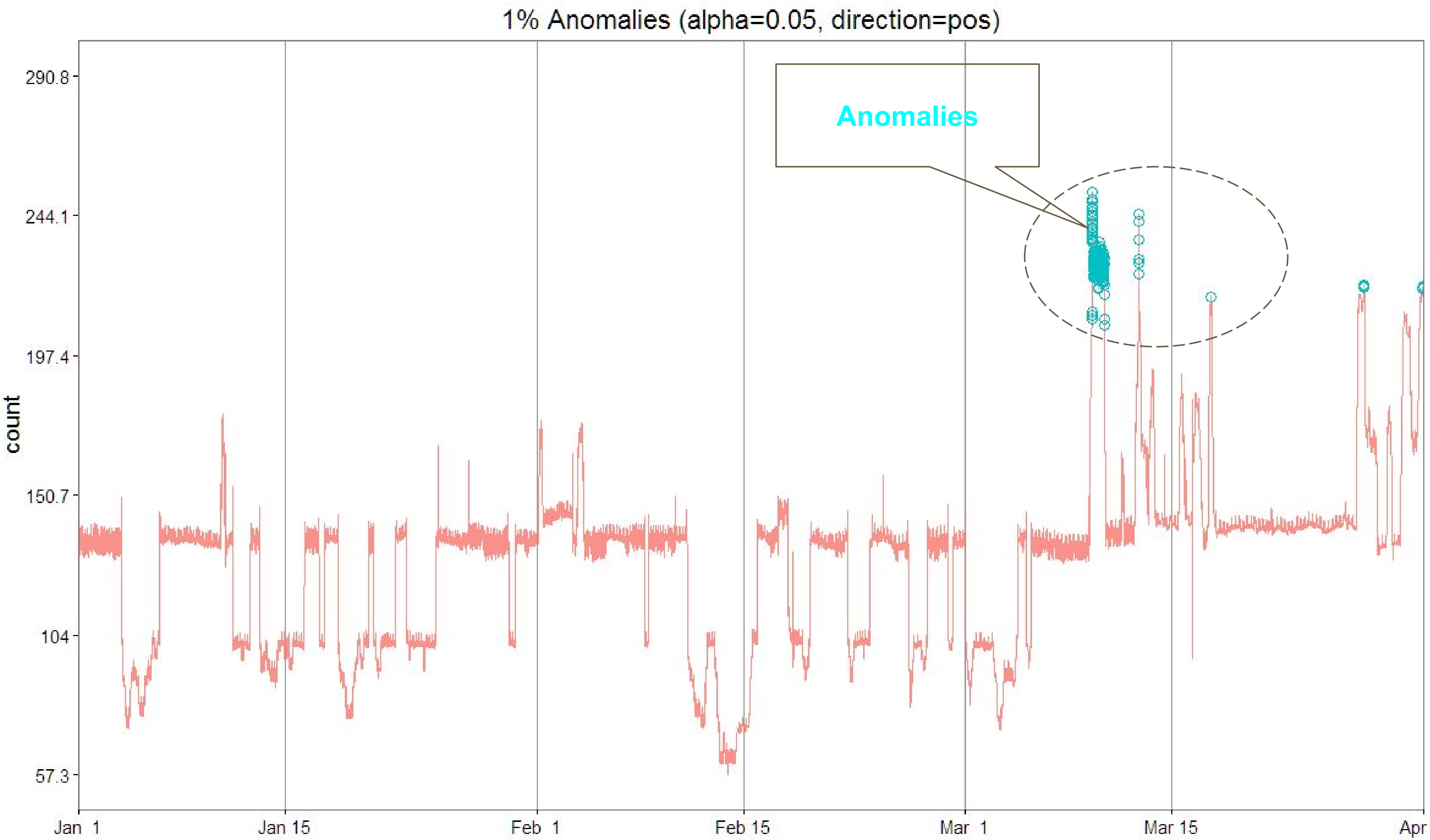
ARIMA(p, d, q)

Conditions for timeseries

```
# Box–Pierce or Ljung–Box test statistic for examining the null
hypothesis of independence in a given time series. Small p-values
(i.e., less than 0.05) suggest that the series is stationary.
Box.test(carsales.ts)
Box.test(carsales.ts, type = "Ljung-Box")
# The Augmented Dickey–Fuller (ADF) t-statistic test: small p-
values suggest the data is stationary and doesn’t need to be
differenced stationarity.
library(tseries)
adf.test(carsales.ts)
# The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test; here
accepting the null hypothesis means that the series is
stationarity, and small p-values suggest that the series is not
stationary and a differencing is required.
kpss.test(carsales.ts)
```

Anomaly Detection

```
library(AnomalyDetection)
myts = as.data.frame(
  cbind(as.POSIXct(index(carsales_fixed.z)),
    coredata(carsales_fixed.z)))
colnames(myts) = c("month", "Sales")
attr(myts$month, "tzone") = "UTC"
ggplot(myts, aes(x=month, y=Sales)) + geom_line()
data_anomaly = AnomalyDetectionTs(
  myts, max_anoms=0.01, direction="pos", plot=F,
  e_value = T, na.rm = T
)
# No anomaly detected as NULL result returned
data_anomaly
data_anomaly$plot
```



1% Anomalies (alpha=0.05, direction=pos)
Anomalies
count
290.8
244.1
197.4
150.7
104
57.3
Jan 1
Jan 15
Feb 1
Feb 15
Mar 1
Mar 15
Apr