Wrangling F1 Data With R

A Data Junkie's Guide

Tony Hirst

Wrangling F1 Data With R

A Data Junkie's Guide

Tony Hirst

Tweet This Book!

Please help Tony Hirst by spreading the word about this book on Twitter!

The suggested hashtag for this book is #f1datajunkie.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#f1datajunkie

Thanks… Just because… (sic)

Acknowledgements

FORMULA 1, FORMULA ONE, F1, FIA FORMULA ONE WORLD CHAMPIONSHIP, GRAND PRIX, F1 GRAND PRIX, FORMULA 1 GRAND PRIX and related marks are trademarks of Formula One Licensing BV, a Formula One group company. All rights reserved.

The Ergast Developer API is an experimental web service which provides a historical record of motor racing data for non-commercial purposes. http://ergast.com/mrd/ .

R: A Language and Environment for Statistical Computing, is produced by the R Core Team/R Foundation for Statistical Computing, http://www.R-project.org .

RStudio™ is a trademark of RStudio, Inc. http://www.rstudio.com/ .

ggplot2: elegant graphics for data analysis, Hadley Wickham, Springer New York, 2009. http://ggplot2.org/

knitr: A general-purpose package for dynamic report generation in R, Yihui Xie. http://yihui.name/knitr/

Leanpub: flexible ebook posting, and host of this book at http://leanpub.com/wranglingf1datawithr/

Errata

An update of the RSQLite package to version 1.0.0 (25.10/14) requires the following changes:

```
### COMMENT OUT THE ORIGINAL SET UP
#require(RSQLite)
#ergastdb = dbConnect(drv='SQLite', dbname='./ergastdb13.sqlite')

### REPLACE WITH:
require(DBI)
ergastdb =dbConnect(RSQLite::SQLite(), './ergastdb13.sqlite')
```

Contents

Battlemaps

Battlemaps are a custom chart style designed to illustrate the competition between a particular driver and the cars in race positions immediately ahead or behind them at a particular stage in a race, or the battle for a particular race position.

As well as revealing the gap to the car immediately ahead or behind in race position terms, *battlemap* displays also include cars on a different race lap (either backmarkers, or race leaders on laps ahead when considering lower placed positions) that have a track position in between that of a target vehicle and car in the race position either immediately or behind one position. (The aim here is to illustrate whether there are any off-lap vehicles that may interfere with any positional battles.)

As a graphic that reveals information about track position as well as race position, we need to have access to data that revelas this information. One way of obtaining this information is to derive it from the laptime data published as part of the ergast database.

Identifying Track Position From Accumulated Laptimes

Give a set of laptime data for a particular race, how can we identify track position information from it?

The first observation we might make is that a race track is a closed circuit; the second that the accumulated race time to date is the same for each driver, given that they all start the race at the same time. (The race clock is not started as each driver passes the start finish line - the race clock starts when the lights go green. To this extent, drivers lower placed on the grid server a positional time penalty compared to cars further up grid. This effective time penalty corresponds to the time it takes a lower placed car to physically get as far up the track as the cars in the higher placed grid positions.)

Let’s start by getting hold of all of the lap time data for a particular race:

```
library(DBI)
ergastdb =dbConnect(RSQLite::SQLite(), './ergastdb13.sqlite')

#There should be only a single result from this query,
# so we can index its value directly.
q=paste('SELECT d.driverId,d.driverRef,d.code, l.lap,l.position,l.time,l.milliseconds
        FROM drivers d JOIN lapTimes l JOIN races r
        WHERE year="2012" AND round="1" AND r.raceId=l.raceId AND d.driverId=l.driver\
Id')
lapTimes=dbGetQuery(ergastdb,q)
#Note that we want the driverId as a factor rather than as a value
lapTimes$driverId=factor(lapTimes$driverId)
```

The laptimes are described as a time in milliseconds. At first glance it might appear to be more convenient to work in seconds, calculated by dividing the milliseconds time by 1000.

```
#We want to convert the time in milliseconds to time in seconds
#One way of doing this is to take the time in milliseconds colument
lapTimes$rawtime = lapTimes$milliseconds/1000
```

However, in practice we see that when calculating differences between values represented in this way as floating point numbers, we get floating point errors.

In order to find the track position of a car, we first need to identify which leader's lap each driver is on and then use this as the basis for deciding whether a car is on the same lap - or a different one - compared with any car immediately ahead or behind on track. One way of doing this is on the basis of accumulated race time. If we order the drivers by the accumulated race time, and flag whether or not a particular driver is the leader on particular lap, we can count the accumulated number of "lap leader" flags to give us the current lead lap count irrespective of how many laps a given driver has completed.

```
library(plyr)

#For each driver, calculate their accumulated race time at the end of each lap
lapTimes=ddply(lapTimes, .(driverId), transform,
               acctime=cumsum(milliseconds))

#Order the rows by accumulated lap time
lapTimes=arrange(lapTimes,acctime)
#This ordering need not necessarily respect the ordering by lap.

#Flag the leader of a given lap - this will be the first row in new leader lap block
lapTimes$leadlap= (lapTimes$position==1)
head(lapTimes[lapTimes$position<=3,c('driverRef','leadlap')],n=5)
```

```
##             driverRef leadlap
## 1              button    TRUE
## 2            hamilton   FALSE
## 3  michael_schumacher   FALSE
## 22             button    TRUE
## 23           hamilton   FALSE
```

A Boolean `TRUE` value has numeric value 1, a Boolean `FALSE` numeric value 0.

```
#Calculate a rolling count of leader lap flags.
#Recall that the cars are ordered by accumulated race time.
#The accumulated count of leader flags is the lead lap number each driver is on.
lapTimes$leadlap=cumsum(lapTimes$leadlap)
head(lapTimes[lapTimes$position<=3,c('driverRef','leadlap')],n=6)
```

```
##             driverRef leadlap
## 1              button       1
## 2            hamilton       1
## 3  michael_schumacher       1
## 22             button       2
## 23           hamilton       2
## 24 michael_schumacher       2
```

Let's now calculate the track position for a given lead lap, where the leader in a given lap is in both race position and track position 1, the second car through the start/finish line is in track position 2 (irrespective of their race position), and so on. (In your mind's eye, you might imagine the cars passing the finish line to complete each lap, first the race leader, then either car in second, or a lapped back marker, and so on.) Specifically, we group by leadlap *and then* accumulated race time within that lap, and assign track positions in incremental order.

```
lapTimes=arrange(lapTimes,leadlap,acctime)
lapTimes=ddply(lapTimes,.(leadlap),transform,
               trackpos=1:length(position))
lapTimes[lapTimes$leadlap==33,c('code','lap','position','acctime','leadlap','trackpos\
')]
```

```
##     code lap position acctime leadlap trackpos
## 616  BUT  33        1 3100735      33        1
## 617  HAM  33        2 3111538      33        2
## 618  VET  33        3 3113745      33        3
## 619  SEN  32       16 3115035      33        4
## 620  RIC  32       17 3115829      33        5
## 621  ALO  33        4 3125951      33        6
## 622  WEB  33        5 3131009      33        7
## 623  MAL  33        6 3133006      33        8
## 624  RAI  33        7 3141269      33        9
## 625  KOB  33        8 3147051      33       10
## 626  GLO  32       18 3150703      33       11
## 627  PER  33        9 3153818      33       12
## 628  ROS  33       10 3159053      33       13
## 629  VER  33       11 3162088      33       14
## 630  DIR  33       12 3172712      33       15
## 631  MAS  33       13 3177681      33       16
## 632  PET  33       14 3184974      33       17
## 633  PIC  32       19 3186685      33       18
## 634  KOV  33       15 3188375      33       19
```

In this example, we see Timo Glock (GLO) has only completed 32 laps compared to 33 for the race leader and the majority of the field. *On track*, he is placed between Kobyashi (KOB) and Perez (PER).

Calculating DIFF and GAP times

We can now calculate various gap times, such as the standard GAP to leader and the +/- DIFF times to any car placed directly ahead or behind a particular car.

The GAP (time to leader) is calculated as the difference between the accumulated race time of the race leader at the end of a lap and the accumulated race time of driver when they complete the same race lap.

The accumulated race time $t_{d,N}$ for a driver *d* on lap *N* is given as:

$$t_{d,N} = \sum_{l=1}^{N} t_{d,l}$$

For the leader, declare $d = L$ to give the accumulated race time for the leader at the end of lap *N* as $t_{L,N}$.

The GAP between a driver *d* after *N* laps and the leader at the end of lap *N* is given as:

$$GAP_{d,N} = t_{d,N,GAP} = t_{d,N} - t_{L,N}$$

Alternatively, we can calculate it as the sum of differences between consecutively placed drivers. The DIFF between drivers in positions *m* and *n* at the end of *N* laps and where *m* is ahead of *n* is given as:

$$DIFF_{n,m,N} = t_{n,N} - t_{m,N}$$

The GAP between a driver in position *P* and the leader *L=1* is then:

$$t_{P,N,GAP} = DIFF_{2,1,N} + D_{3,2,N} + .. + DIFF_{P,P-1,N}$$

and where $GAP_{L,N} = t_{L,N,GAP} = 0$.

We can write this more succinctly as:

$$GAP_{P,N} = t_{P,N,GAP} = \sum_{p=2}^{P} DIFF_{p,p-1,N} = \sum_{p=2}^{P} (t_{p,N} - t_{p-1,N})$$

We can implement these calculations directly as follows:

```
#Order the drivers by lap and position
lapTimes=arrange(lapTimes,lap,position)
#Calculate the DIFF between each pair of consecutively placed cars at the end of each\
 race lap
#Then calculate the GAP to the leader as the sum of DIFF times
lapTimes=ddply(lapTimes, .(lap), mutate,
               diff=c(0,diff(acctime)),
               gap=cumsum(diff)  )
```

For completeness, we might also want to capture the DIFF to the car behind, which we shall represent as `chasediff`, further requiring that it is a negative quantity. That is, we have $CHASEDIFF_{q,r} = -DIFF_{r,q}$ for race positions $r > q$ (that is, q is ahead of r).

```
#Order the drivers by lap and reverse position
lapTimes=arrange(lapTimes,lap, -position)
#Calculate the DIFF between each pair of consecutively reverse placed cars at the end\
 of each race lap
lapTimes=ddply(lapTimes, .(lap), mutate,
               chasediff=c(0,diff(acctime)) )

#Print an example
head(lapTimes[lapTimes$lap==35,c('code','lap','diff','chasediff')],n=5)
```

```
##     code lap  diff chasediff
## 658  PIC  35 92327         0
## 659  GLO  35 34462    -92327
## 660  KOV  35 14749    -34462
## 661  RIC  35  1281    -14749
## 662  SEN  35 25011     -1281
```

Typically, timing sheets do not show the GAP to the leader for cars other than those cars on the lead lap. Instead, they provide a count of the number of laps behind the driver is.

```
lapTimes$tradgap=as.character(lapTimes$gap)
lapsbehind=function(lap,leadlap,gap){
  if (lap==leadlap) return(gap)
  paste("LAP+",as.character(leadlap-lap),sep='')
}

lapTimes$tradgap=mapply(lapsbehind,lapTimes$lap,lapTimes$leadlap,lapTimes$gap)

#Print an example
lapTimes[lapTimes$lap==35,c('code','lap','leadlap','tradgap')]
```

```
##     code lap leadlap tradgap
## 658  PIC  35      37   LAP+2
## 659  GLO  35      36   LAP+1
## 660  KOV  35      36   LAP+1
## 661  RIC  35      36   LAP+1
## 662  SEN  35      36   LAP+1
## 663  MAS  35      35   80864
## 664  DIR  35      35   78460
## 665  VER  35      35   60621
## 666  ROS  35      35   57772
## 667  PER  35      35   51594
## 668  ALO  35      35   50737
## 669  KOB  35      35   47972
## 670  RAI  35      35   39980
## 671  MAL  35      35   32088
## 672  WEB  35      35   28514
## 673  VET  35      35   12514
## 674  HAM  35      35   10890
## 675  BUT  35      35       0
```

Here we see that the drivers up to and including Massa (MAS) are on the lead lap, and as such an explicit time gap to the leader is reported. For Bruno Senna (SEN) and the lower placed drivers, they are one lap behind the leader, except for Charles Pic, who is two laps down.

Calculating the time between cars based on track position

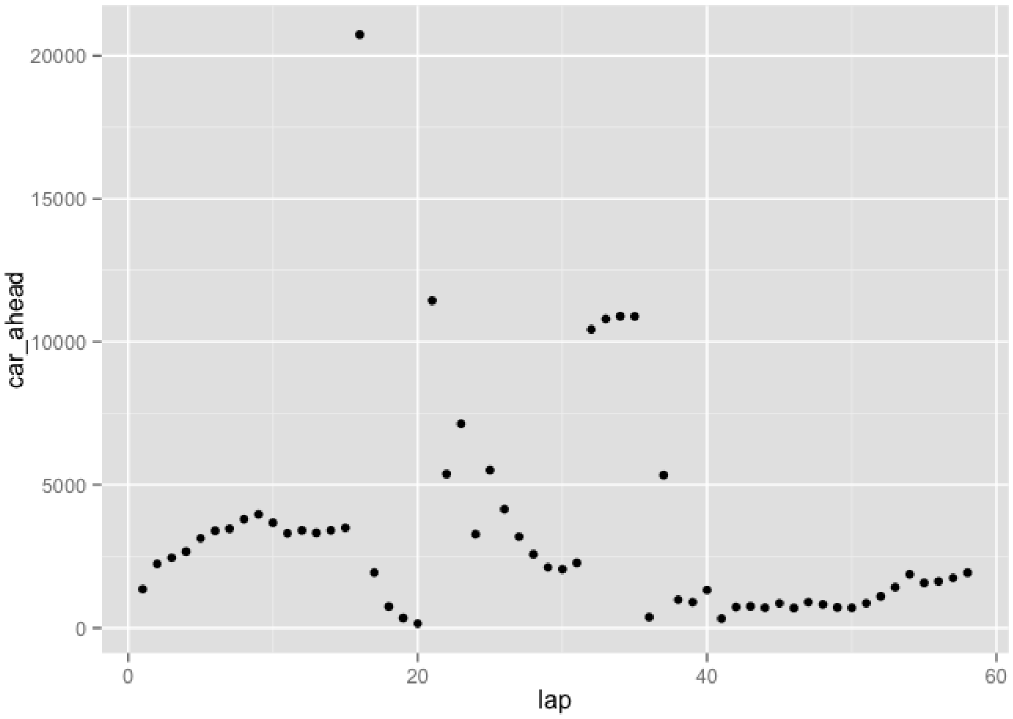
To calculate the time difference to the car ahead on track (`car_ahead`) and the car behind (`car_behind`) on track, we can simply calculate the differences between accumulated laptimes for appropriately ordered rows.

```
#Arrange the drivers in terms of increasing accumulated race time
lapTimes = arrange(lapTimes, acctime)
#For each car, calculate the DIFF time to the car immediately ahead on track
lapTimes$car_ahead=c(0,diff(lapTimes$acctime))
#Identify the code of the driver immediately ahead on track
lapTimes$code_ahead=c(NA,head(lapTimes$code,n=-1))
#Identify the race position of the driver immediately ahead on track
lapTimes$position_ahead=c(NA,head(lapTimes$position,n=-1))

#Now arrange the drivers in terms of decreasing accumulated race time
lapTimes = arrange(lapTimes, -acctime)
#For each car, calculate the DIFF time to the car immediately behind on track
lapTimes$car_behind=c(0,diff(lapTimes$acctime))
#Identify the code of the driver immediately behind on track
lapTimes$code_behind=c(NA,head(lapTimes$code,n=-1))
#Identify the race position of the driver immediately behind on track
lapTimes$position_behind=c(NA,head(lapTimes$position,n=-1))

#put the lapTimes dataframe back to increasing acculamated race time order.
lapTimes = arrange(lapTimes, acctime)
```

Notice how the `diff()` function finds the difference between column values on consecutive rows working down the column. To find the gap to the car ahead, we sort on *increasing* accumulated race time and then apply the `diff()` function. To find the gap to the car behind, we reverse the order, sorting on *decreasing* accumulated lap time, before applying the `diff()` function.

Having calculated the time to car ahead - or the car behind - on track, we can use a simple scatter plot to show the time in milliseconds to the car ahead, for each lap .

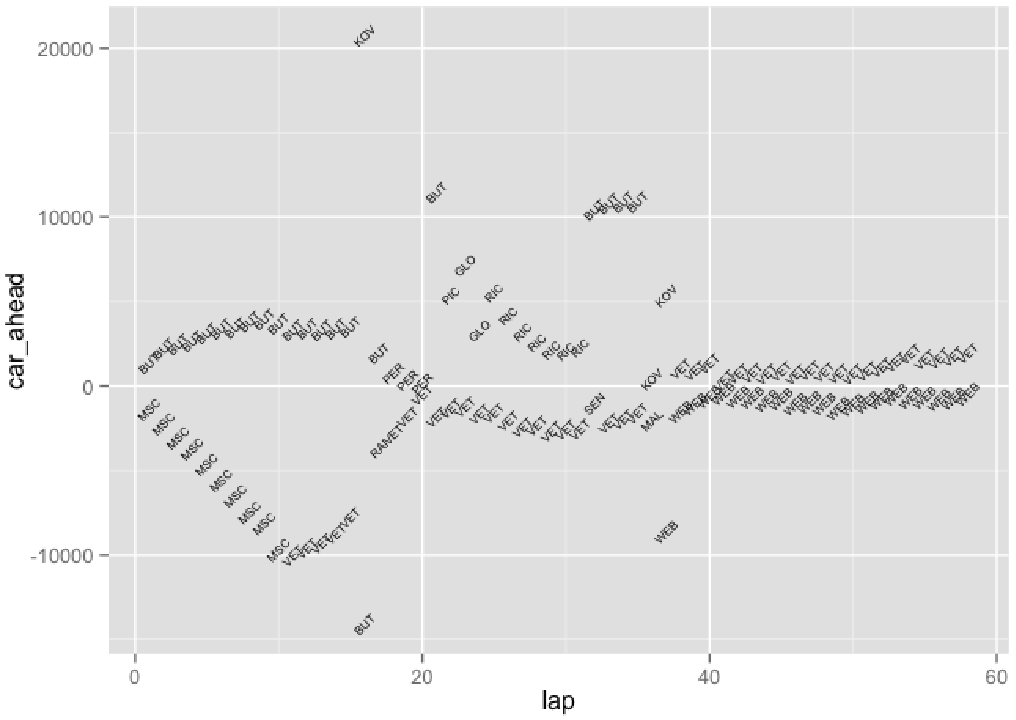
```
library(ggplot2)

g=ggplot(lapTimes[lapTimes['code']=='HAM',])
g+geom_point(aes(x=lap,y=car_ahead))
```



For a selected driver, chart the time to the car ahead

If instead we use a text plot, we can identify which driver in particular was in the car ahead or behind on track.

```
g=g+geom_text(aes(x=lap,y=car_ahead,label=code_ahead),angle=45,size=2)
g+geom_text(aes(x=lap,y=car_behind,label=code_behind),angle=45,size=2)
```



For a selected driver, identify which driver is ahead or behind on track, and by how much time, on each lap

Here we see that Hamilton draws away from Michael Schumacher at a constant rate over the first 10 laps of the race, dropping behind Jenson Button over that same period, then keeps pace with him between laps 10 and 15.

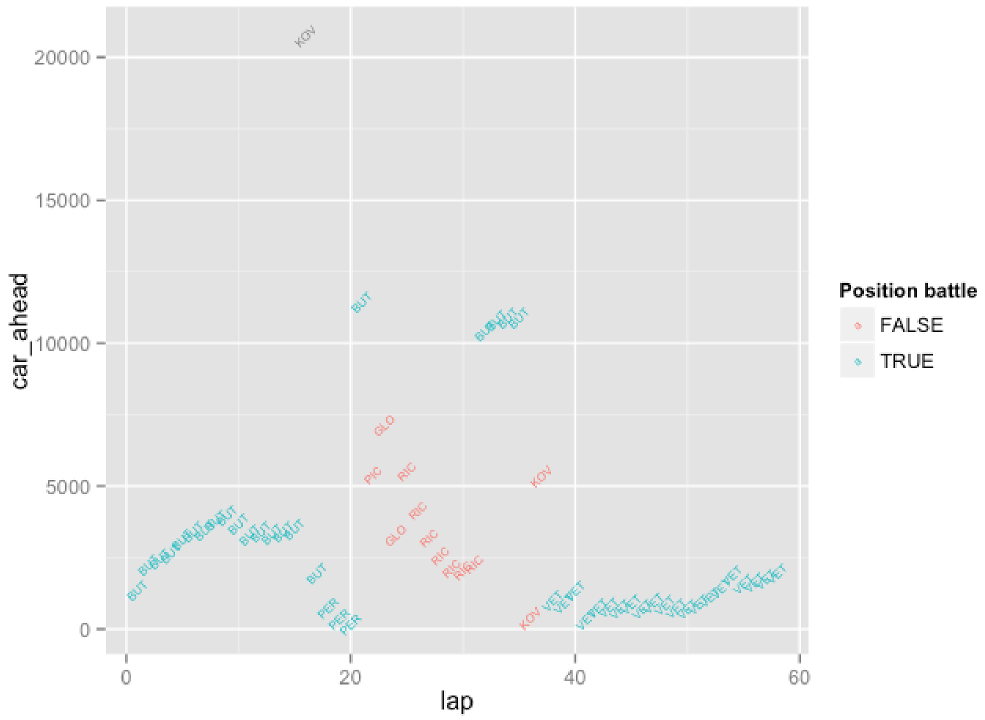
Note that the driver codes specified refer to the driver immediately ahead or behind on the track, irrespective of whether they are on the same lap. The chart would perhaps me more informative if we could indentify whether the car immediately ahead or behind is actually on the same racing lap as our selected driver.

One way to approach this is to generate new columns that identify the driver immediately ahead or behind each car in terms of race position, rather than track position. This new information will allow us to test whether the car ahead or behind on track is in a battle for position with the selected driver.

```
lapTimes = arrange(lapTimes, lap,position)
lapTimes = ddply(lapTimes,.(lap),transform,
                         code_raceahead=c(NA,head(code,n=-1)))

lapTimes = arrange(lapTimes, -lap,-position)
lapTimes = ddply(lapTimes,.(lap),transform,
                         code_racebehind=c(NA,head(code,n=-1)))

lapTimes = arrange(lapTimes, acctime)
```

The test is one simply of equality: *is the driver one place ahead on track the driver in one place ahead in terms of race position?* This answer to this test allows us to visually distinguish between whether there is a battle for position going on with the car directly ahead on track:

```
battlesketch1=function(driverCode){
  g=ggplot(lapTimes[lapTimes['code']==driverCode,])
  g=g+geom_text(aes(x=lap,
                  y=car_ahead,
                  label=code_ahead,
                  #Test whether we are in a direct battle with the car ahead
                  col=factor(code_ahead==code_raceahead)),
              angle=45,size=2)
  g+guides(col=guide_legend(title="Position battle"))
}
battlesketch1('HAM')
```

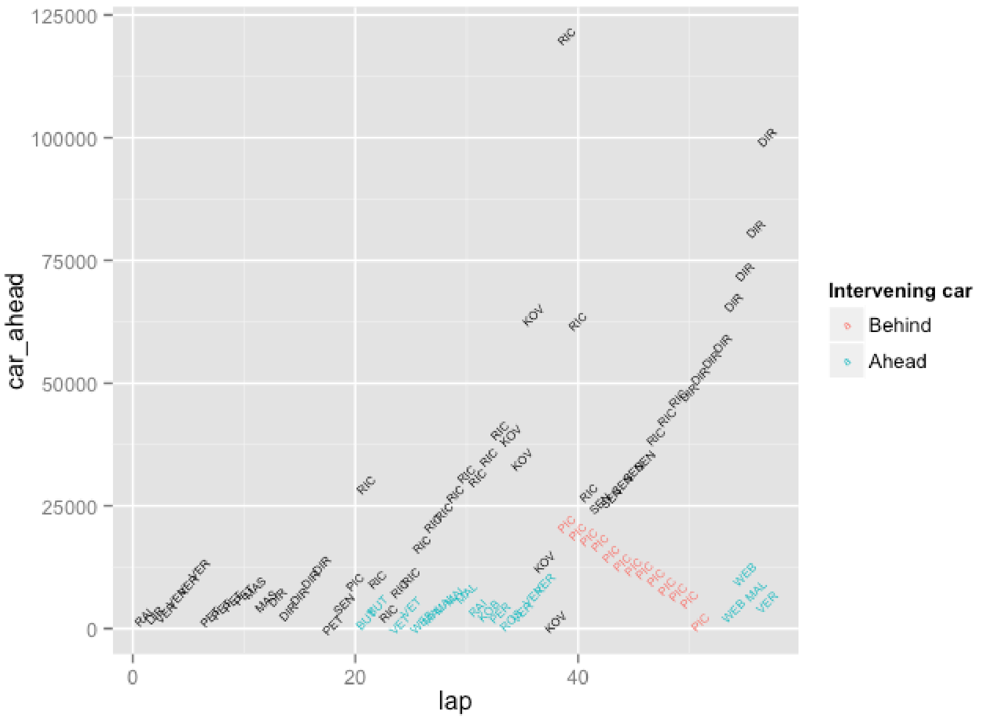


A useful side effect here is that if `code_raceahead` is undefined (because the selected driver is in the lead at the start of a particular lap), the `code_ahead==code_raceahead` test is also undefined, no colour is set, and the text label color defaults to grey.

It would be useful to further refine the chart so that it additionally shows the driver immediately ahead (or behind) in terms of *race* position if the car ahead on track is not the car immediately ahead in terms of position.

We can achieve this by adding another layer:

```
battlemap_ahead=function(driverCode){
  g=ggplot(lapTimes[lapTimes['code']==driverCode,])
  #Plot the offlap cars that aren't directly being raced
  g=g+geom_text(data=lapTimes[(lapTimes['code']==driverCode)
                              & (lapTimes['code_ahead']!=lapTimes['code_raceahead']),\
],
                aes(x=lap,
                  y=car_ahead,
                  label=code_ahead,
                  col=factor(position_ahead<position)),
              angle=45,size=2)
  #Plot the cars being raced directly
  g=g+geom_text(aes(x=lap,
                  y=diff,
                  label=code_raceahead),
              angle=45,size=2)
  g=g+scale_color_discrete(labels=c("Behind","Ahead"))
  g+guides(col=guide_legend(title="Intervening car"))
}
battlemap_ahead('GLO')
```



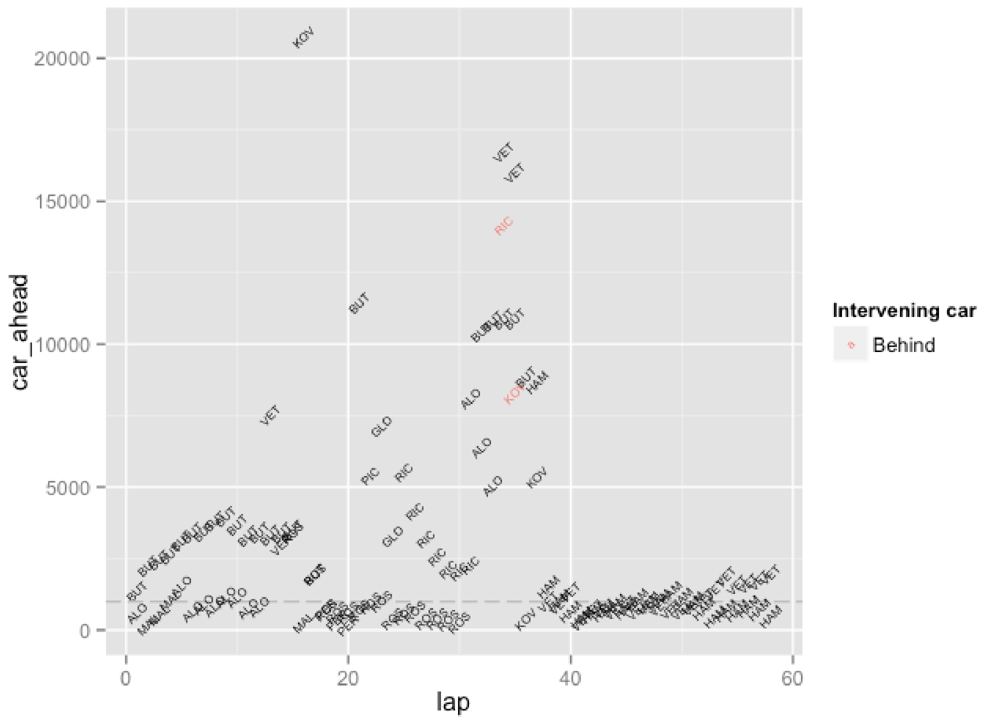
A first attempt at a battle map, showing cars in positions immediately ahead/behing on track, as well as in race position terms

In this case, cars on the same lap are coloured black, and cars in a track position that is out of race position is coloured either aqua (if it is in a race position ahead of the selected car) or orange if it is on at least one lap behind. From the chart, we notice how Timo Glock falls behind the car he is racing time and time again (the waves of activity that go up and to the right). The only car he gains on over is series of laps is that of Charles Pic, from about lap 39 onwards. Every so often cars at least one lap ahead (coloured aqua), are in the space ahead between Timo Glock and the car in the race position ahead of him.

Let's now try to put the pieces together in a full battle map showing the state of the race immediately ahead of a particular driver, as well as immediately behind them, throughout the course of a race. We'll also add in a guide that identifies the DRS (drag reduction system) range of one second (1000ms).

We can generalise the battle map charter so that it can plot the battles with the cars ahead or behind. In the following example, we plot just the battle ahead.

```
dirattr=function(attr,dir='ahead') paste(attr,dir,sep='')

#We shall find it convenenient later on to split out the initial data selection
battlemap_df_driverCode=function(driverCode){
  lapTimes[lapTimes['code']==driverCode,]
}

battlemap_core_chart=function(df,g,dir='ahead'){
  car_X=dirattr('car_',dir)
  code_X=dirattr('code_',dir)
  factor_X=paste('factor(position_',dir,'<position)',sep='')
  code_race_X=dirattr('code_race',dir)
  if (dir=='ahead') diff_X='diff' else diff_X='chasediff'

  if (dir=="ahead") drs=1000 else drs=-1000
  g=g+geom_hline(aes_string(yintercept=drs),linetype=5,col='grey')

  #Plot the offlap cars that aren't directly being raced
  g=g+geom_text(data=df[df[dirattr('code_',dir)]!=df[dirattr('code_race',dir)],],
                aes_string(x='lap',
                  y=car_X,
                  label=code_X,
                  col=factor_X),
              angle=45,size=2)
  #Plot the cars being raced directly
  g=g+geom_text(data=df,
                aes_string(x='lap',
                  y=diff_X,
                  label=code_race_X),
              angle=45,size=2)
  g=g+scale_color_discrete(labels=c("Behind","Ahead"))
  g+guides(col=guide_legend(title="Intervening car"))

}

battle_WEB=battlemap_df_driverCode("WEB")
battlemap_core_chart(battle_WEB,g,'ahead')
```



A first attempt at a complete battlemap.

And then we can add in any threats that are coming up from behind:

```
g=battlemap_core_chart(battle_WEB,ggplot(),dir='behind')
```

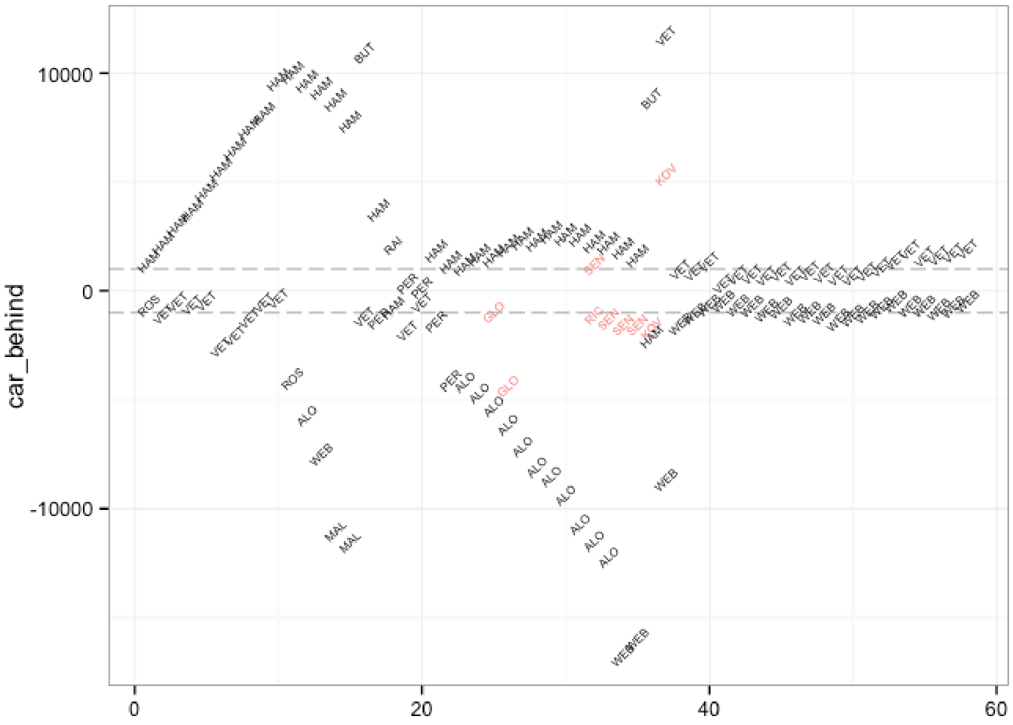
Here we see how Mark Webber kept pace with Alonso, albeit around about a second behind, at the start of the race, at the same time as he drew away from Massa. In the last thrid of the race, he was closely battling with Hamilton whilst drawing away from Alonso.

Battles for a particular position

As well as charting the battles in the vicinity of a particular driver, we can also chart the battle in the context of a particular race position. We can reuse the chart elements and simply need to redefine the filtered dataset we are charting.

For example, if we filter the dataset to just get the data for the car in third position at the end of each lap, we can then generate a battle map of this data.

```
battlemap_df_position=function(position){
  lapTimes[lapTimes['position']==position,]
}

battleForThird=battlemap_df_position(3)

g=battlemap_core_chart(battleForThird,ggplot(),dir='behind')+xlab(NULL)+theme_bw()
g=battlemap_core_chart(battleForThird,g,'ahead')+guides(col=FALSE)
g
```



A position battle chart showing the fight for third in the course of a single race

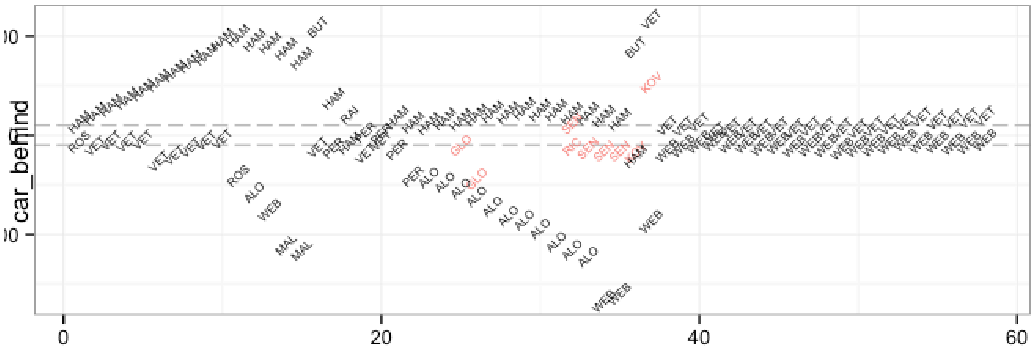
In this case we see how in the opening laps of the race, the battle for third was coming from behind, with Vettel chellenged for position from fourth, as the second placed driver (Lewis Hamilton) pulled away. In the middle third of the race, the car in third kept pace with 2nd placed Hamilton but pulled away from fourth placed Alonso. And in the last third of the race, the car in third is battling hard with Vettel ahead and defending hard against Webber behind.

One thing this chart does not show is which driver was in third position on each lap. We might naturally think to add a layer on to the chart that displays the driver in the position we are charting around along the x-axis (that is, at y=0) but this typically leads to a very cluttered chart.

Instead, we can seek to separate out this information, as the following chart shows.

The following is a work in progress - the vertical sizing/alignment is still broken

```
allblank=theme( panel.border = element_blank(),panel.grid.major = element_blank(), pa\
nel.grid.minor = element_blank(), axis.line = element_blank(),axis.ticks = element_bl\
ank(), axis.text = element_blank())

g2=ggplot(battleForThird)+geom_text(aes(x=lap,y=0,label=code),angle=45,size=2)+ylim(-\
1,1)+theme_bw()+xlab(NULL)+ylab(NULL)+allblank

library(grid)

grid.newpage()
grid.draw(rbind(ggplotGrob(g), ggplotGrob(g2), size = "last"))
```



```
#library(gridExtra)
#grid.arrange(g2,g,heights=c(0.1, 0.9), nrow=2 )
```

Battles Between Particular Drivers

Another sort of battle we might wish to depict is a battle between two particular drivers. *I need to think how to best represent the data to do this...*

Summary

In this chapter, we processed the laptime data in order to identify track position as well as race position. We also generated various time deltas, such as the standard GAP and DIFF values, but also the DIFF to the car in the race position behind, as well as the time to the cars ahead and behind *on track*. The laptime information for each driver on a particular lap was also annotated with the driver code of the drivers immediately ahead and behind, again in terms of both track and racing position. To try to capture some sense of the battles a particular driver was engaged in on track, we started to develop the idea of a *battle map* that shows how close a selected driver is to the cars immediately ahead and behind on each lap in the sense of both race position and track position.