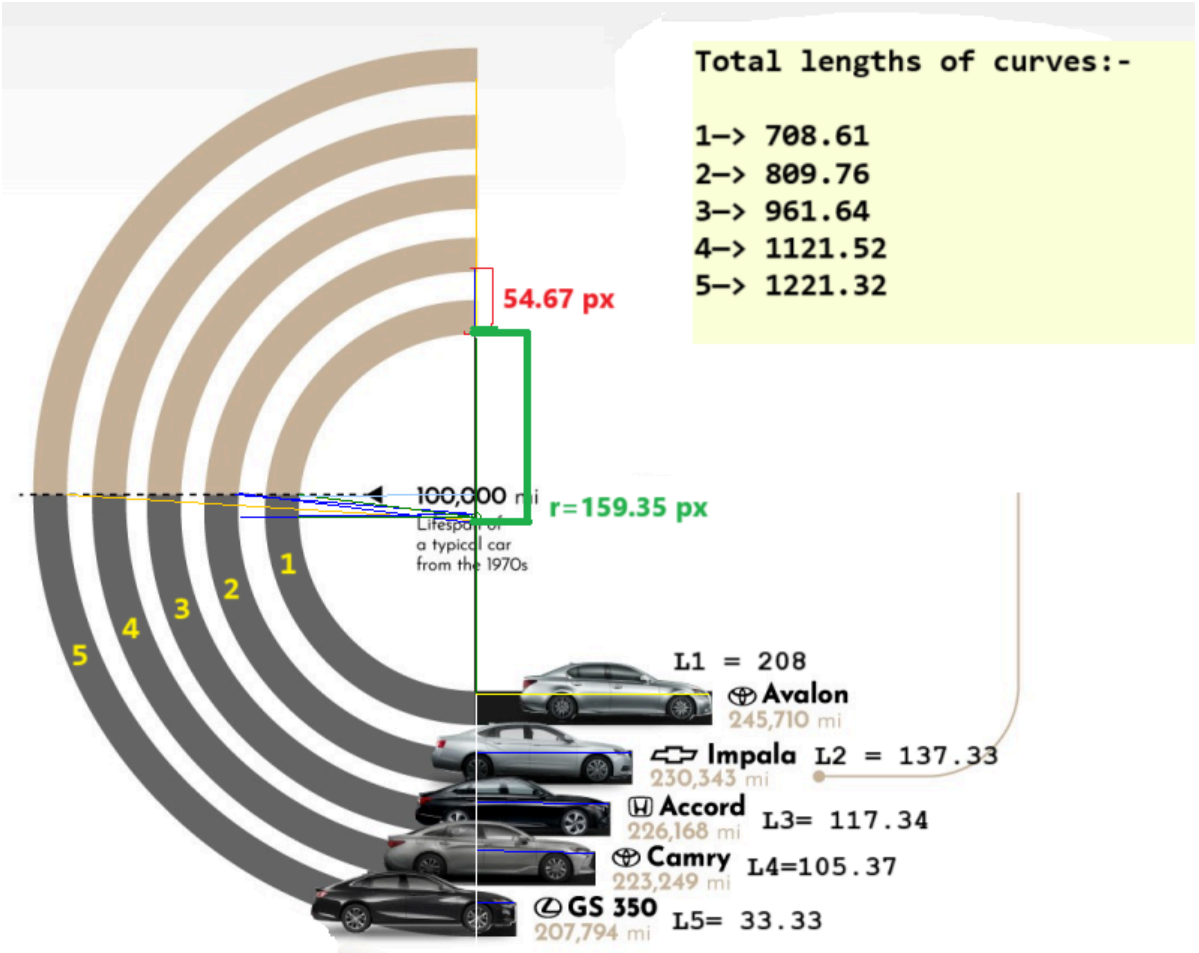
Original Plot—everything that is wrong with it (with precise calculations)



The 100,000 km mark line

At first glance, it looks normal and somewhat accurate but it's when you dig a little deeper that it comes to light how dubious it actually is.

For starters, you can now see that the 100,000 km mark is actually not at the center of the supposed semi-circle, but shifted backwards by some angle. Moreover, as the bars are curved but the dashed-100k-km-mark-line is straight, it lies to our face that the lengths of curves are supposed to be equal.

After calculating, it's off by 7.3° for the first curve, which—if we calculate how many kilometers each unit of length covers—is responsible for a shift in 20.3 units of length and alone adds approximately 7,000 kms to the curve, exaggerating the increase in mileage from the 1970s.

Furthermore, the angle uniformly decreases for each curve until it gets down to just 3.4° for the final curve. But that's not it, due to the curvature the final arc is bigger and thus each unit of its length carries less number of kilometers and adds only a little less than 4000 additional kilometers. Hence, exaggerating the difference between car-1 and car-5.

The curvature—biggest liar of all

Would you believe that the bar of the car with the least mileage is actually the longest? Yeah, right! And by far (check out the top-right table in plot).

But why isn't it obvious at first? Because we ignore the curvature as it's common to all bars (I've drawn a straight line in the plot to emphasize the fact).

As if that wasn't enough, the plot actually cheats. If you were to ignore the length—the semicircle— which is common to all, you should be able to get a plot which fairly compares the differences between the mileages. But even those extra lengths—colored in dark black and hideously masked by 2D car models—are not to scale and have a lie-factor > 1, confirming my suspicions that the mileage of car-1 looks significantly more than it actually is.

Lie Factor data of original plot

```
LieFactor of consecutive pairs (considering complete curves)
1.87
8.56
10.9
1.1
```

```
LieFactor of consecutive pairs (considering only black lengths)
7.72
9.21
8.69
29.06
```

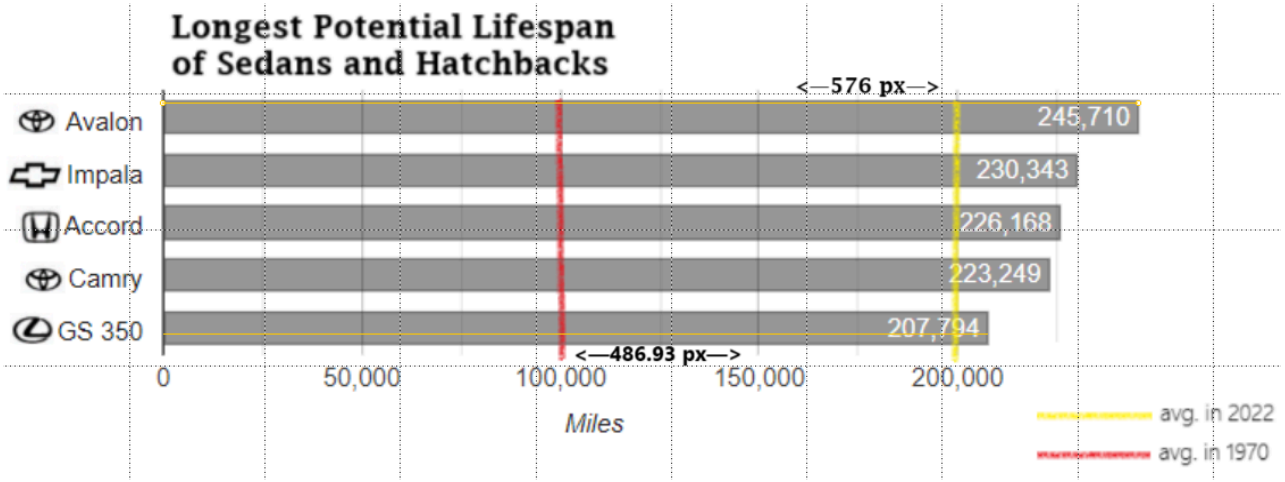
```
LieFactor in extremes
Complete Curve:  2.3
black length: 28.72
```

Program Code to calculate lie factor (python)

```python
import math

pi = math.pi
r1 = 159.35
x = 54.7
vals = (245710, 230343, 226168, 223249, 207794)
exL = (208, 137.3, 117.34, 105.37, 33.33)
```

```
print("\nlengths of black segments", *exL, sep='\n')

curv = list()
for n in range(5):
   curv.append(round((pi*(r1 + n*x)+exL[n]), 2))
print("\ntotal lengths of curves", *curv, sep='\n')

RCinplot = list()
for a in range(4):
   RCinplot.append((curv[a]-curv[a+1])/curv[a+1])

RCindata = list()
for a in range(4):
   RCindata.append((vals[a]-vals[a+1])/vals[a+1])

print("\nLieFactor of consecutive pairs (considering complete curves)")
for a in range(4):
   lf = RCinplot[a]/RCindata[a]
   print(round(abs(lf), 2))

RCinlen = list()
for a in range(4):
   RCinlen.append((exL[a]-exL[a+1])/exL[a+1])

print("\nLieFactor of consecutive pairs (considering only black
lengths)")
for a in range(4):
   lf = RCinlen[a]/RCindata[a]
   print(round(abs(lf), 2))

print("\nLieFactor in extremes")

LFcurv =
round(abs(((curv[0]-curv[4])/curv[4])/((vals[0]-vals[4])/vals[4])), 2)
print("Complete Curve: ", LFcurv)

LFlen = round(abs(((exL[0]-exL[4])/exL[4])/((vals[0]-vals[4])/vals[4])),
2)
print("black length:", LFlen)
```

Redesigned plot



Lie Factor calculation

$$\frac{\frac{576-486.93}{486.93}}{\frac{245710-207794}{207794}} = \frac{0.182}{0.182} = 1$$

As the redesigned plot has a lie factor = 1, it accurately represents the data without overstating or understating its effects.