BIGDATA by 2P
TORONTO 2017

June 20 & 21, 2017 • Metro Toronto Convention Centre

2 co-located show :

AITORONTO by 2P 2017

CONNECTED+ by 2P TORONTO 2017

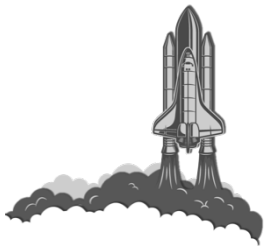
BIG DATA TORONTO

MASTER THE DATA GALAXY

Dive into the era of algorithmic intelligence!

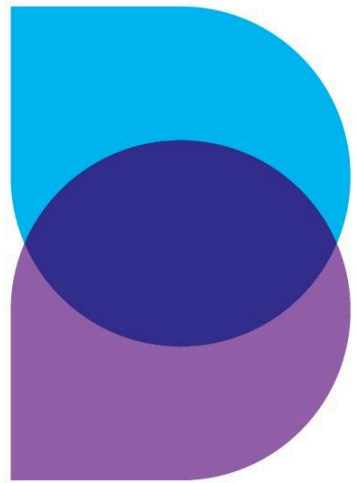
www.bigdata-toronto.com

data
driven
deviations



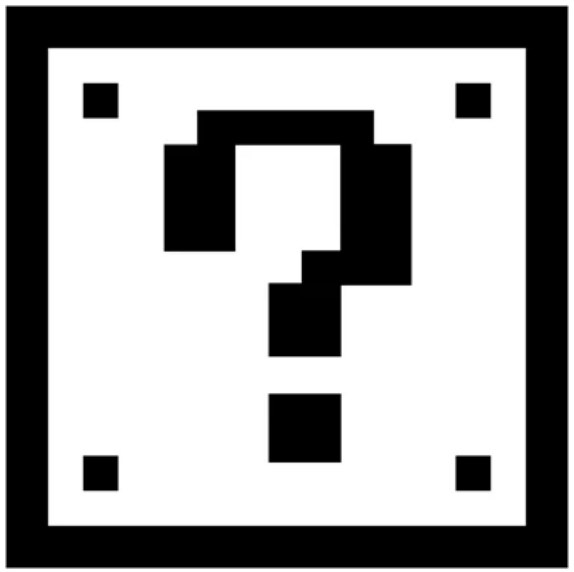
whoami


cisco
P




802
Excellent

250000+



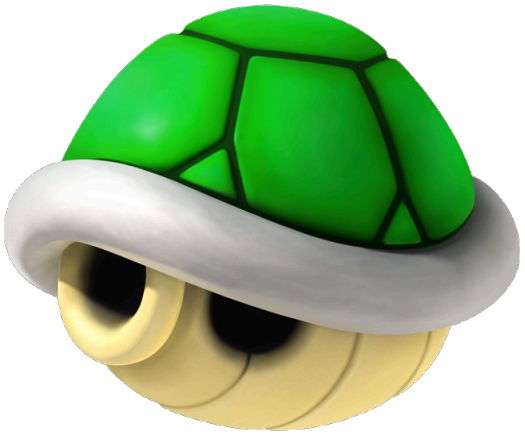
whoareu


FEATURE
PRESENTATION

3rd party data

infrastructure

investors



Green Shell Insurance

cough *cough*


MARIOKART

1kg

5kg

10kg

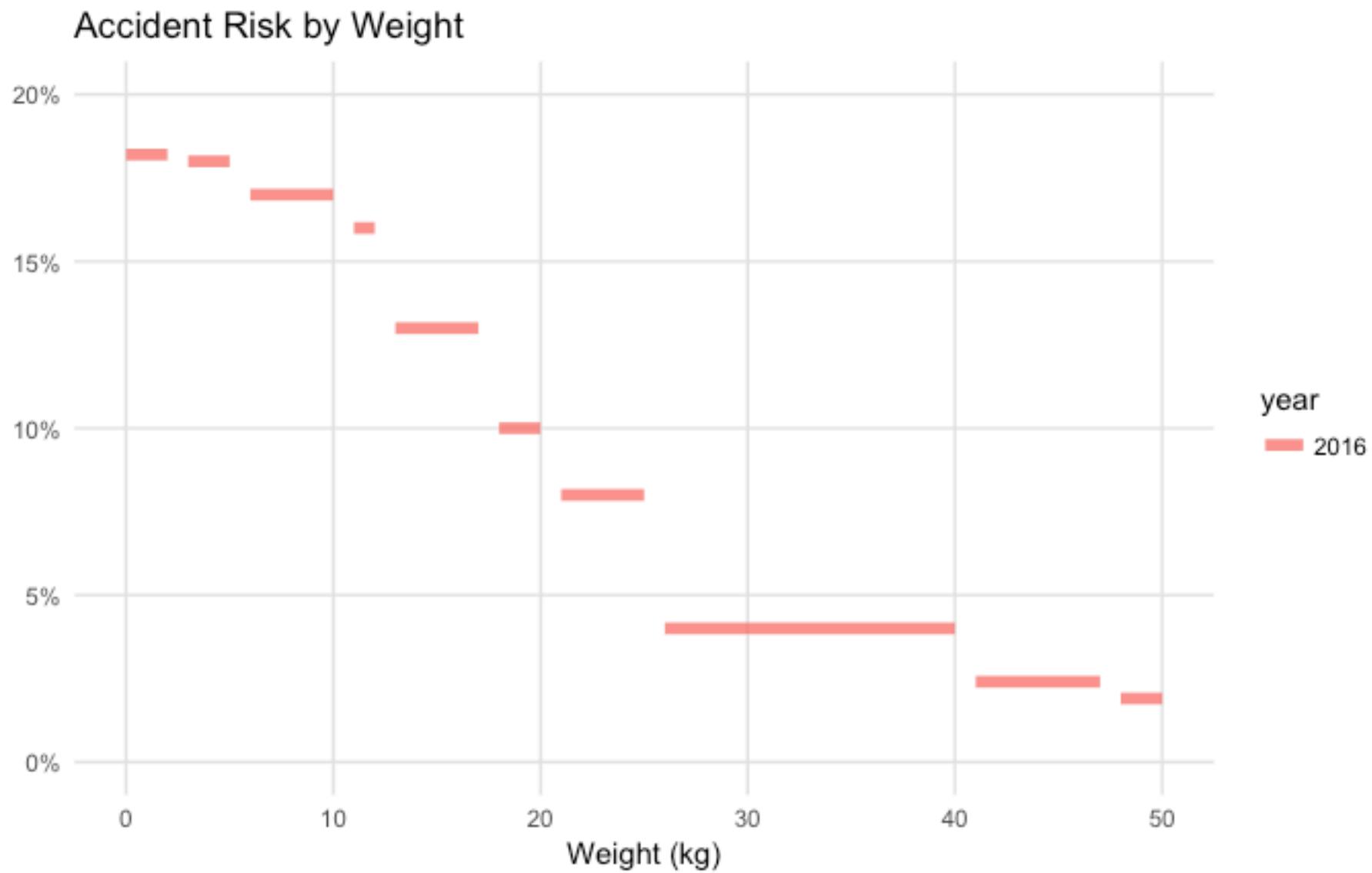
40kg

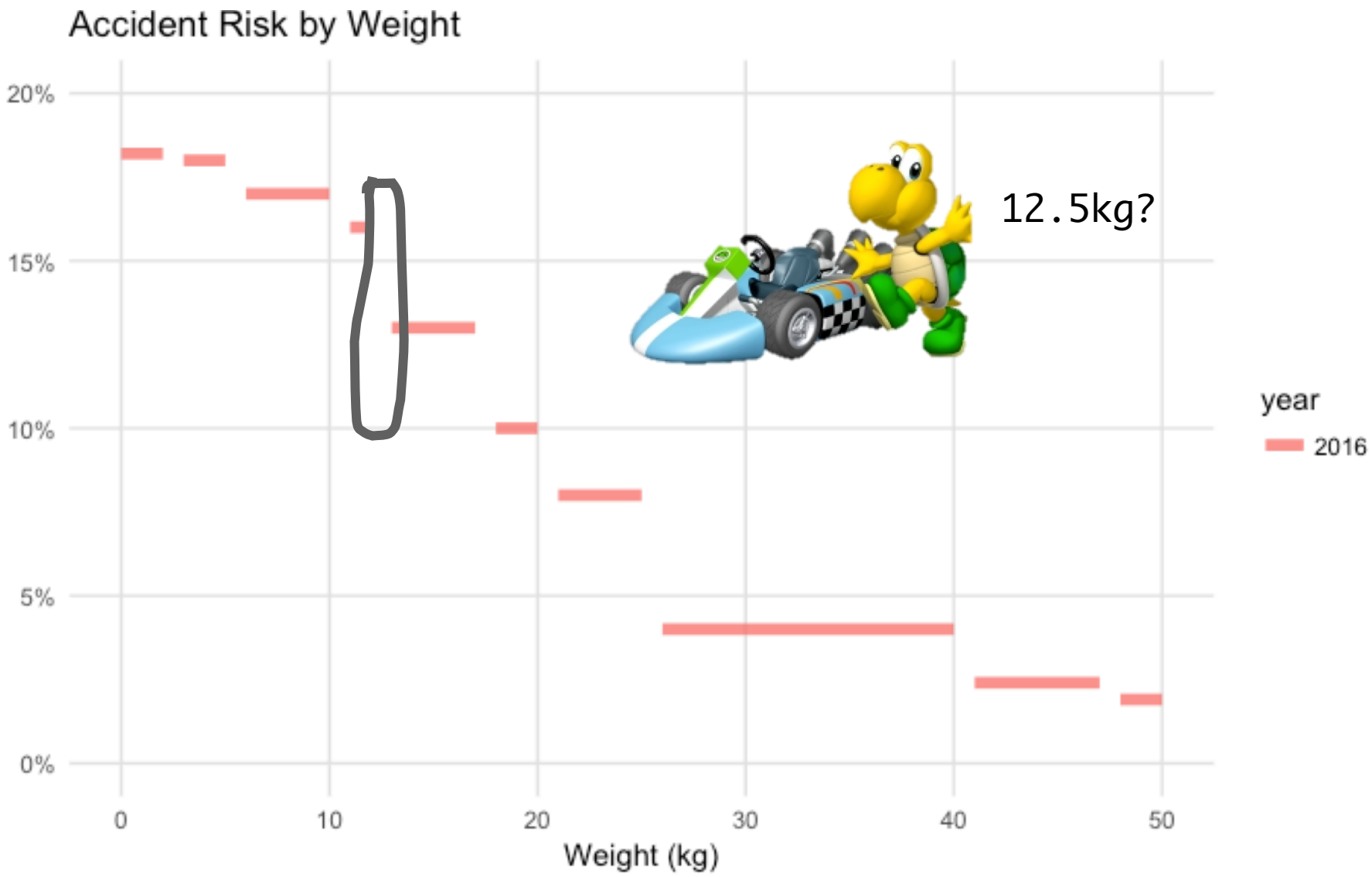

MUSHROOM CUP
MUSHROOM CUP
MUSHROOM CUP
Lemmy's
TIRE SERVICE
GOLDEN WHEEL
BOWSER

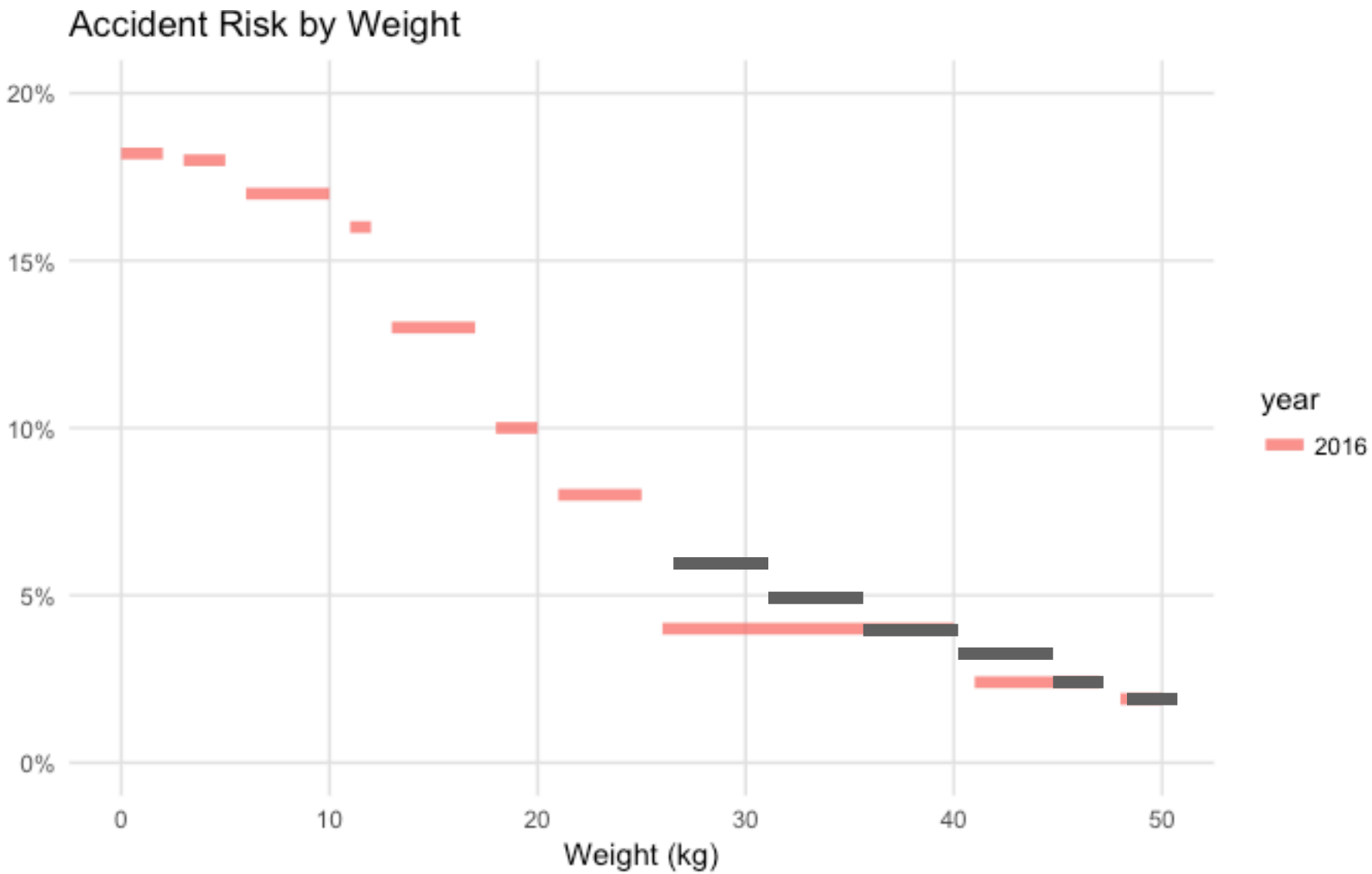
3rd party data

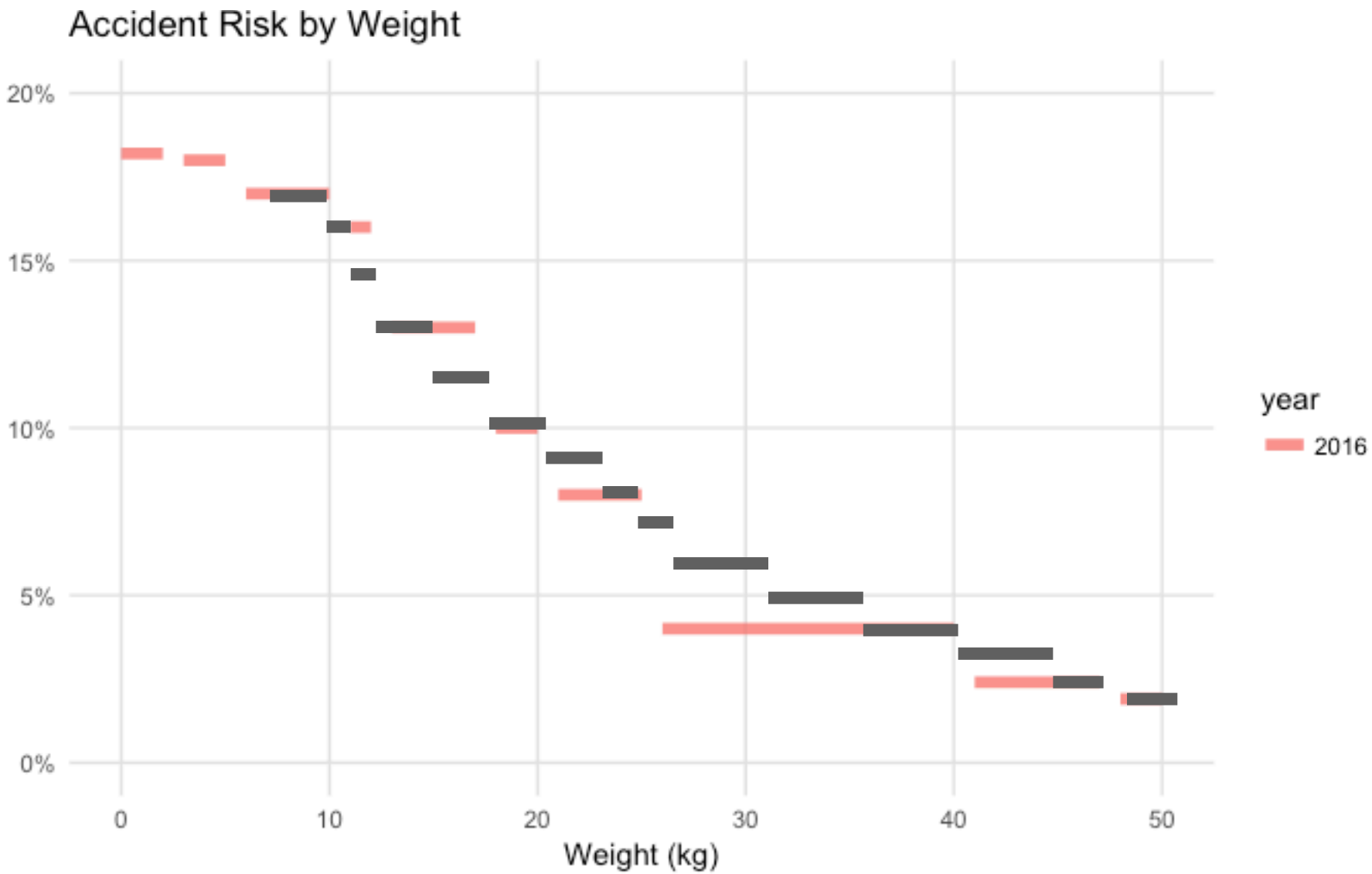
Mushroom Kingdom

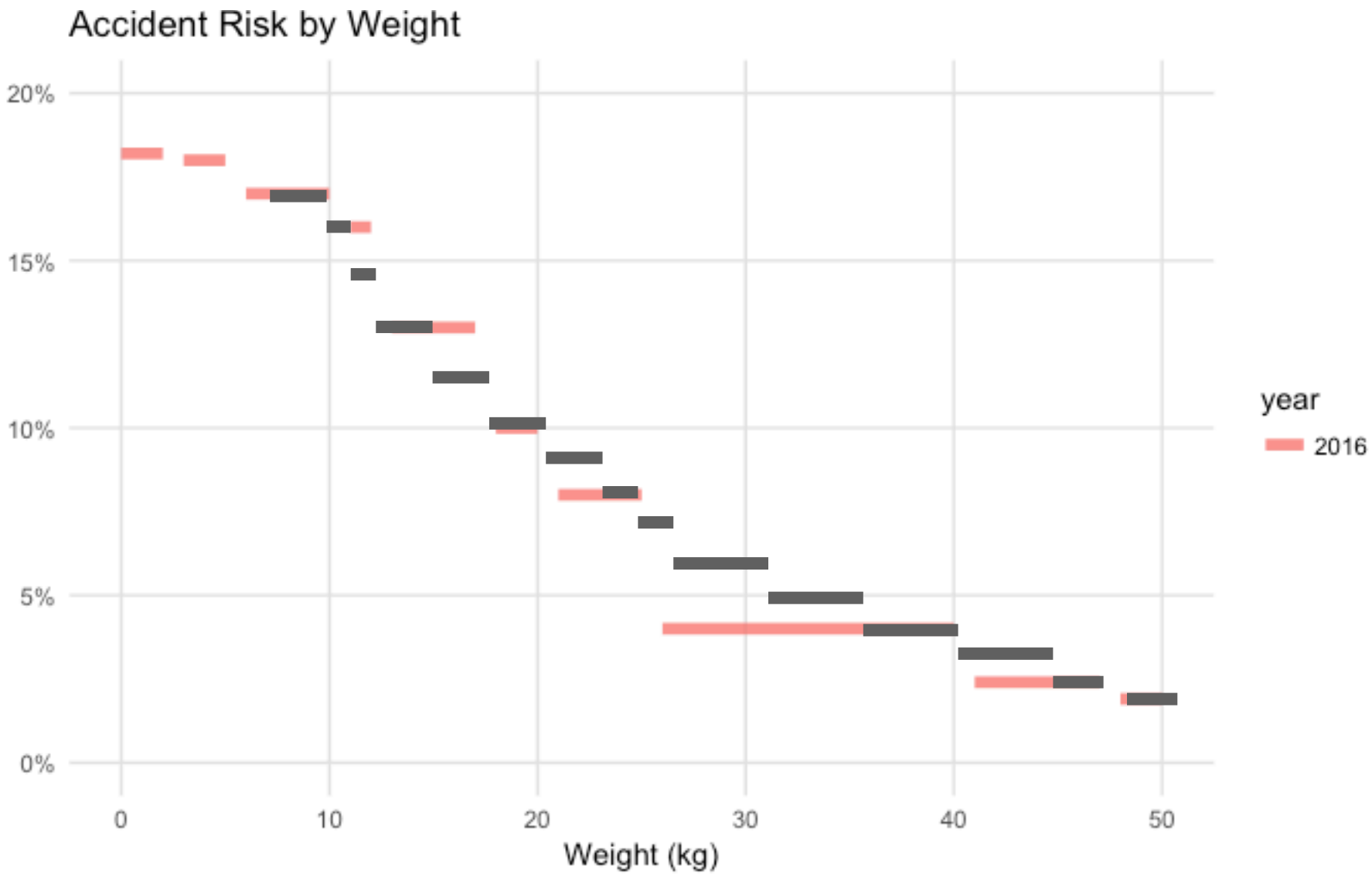
Weight	Risk
0-2	18.2%
3-5	18.0%
6-10	17.0%
11-12	16.0%
13-17	13.0%
18-20	10.0%
21-25	8.00%
26-40	4.00%
41-47	2.40%
48-50	1.90%


Accident Risk by Weight
20%
15%
10%
5%
0%
0
10
20
30
40
50
Weight (kg)
year
2016


Accident Risk by Weight
20%
15%
10%
5%
0%
0
10
20
30
40
50
Weight (kg)
12.5kg?
year
2016

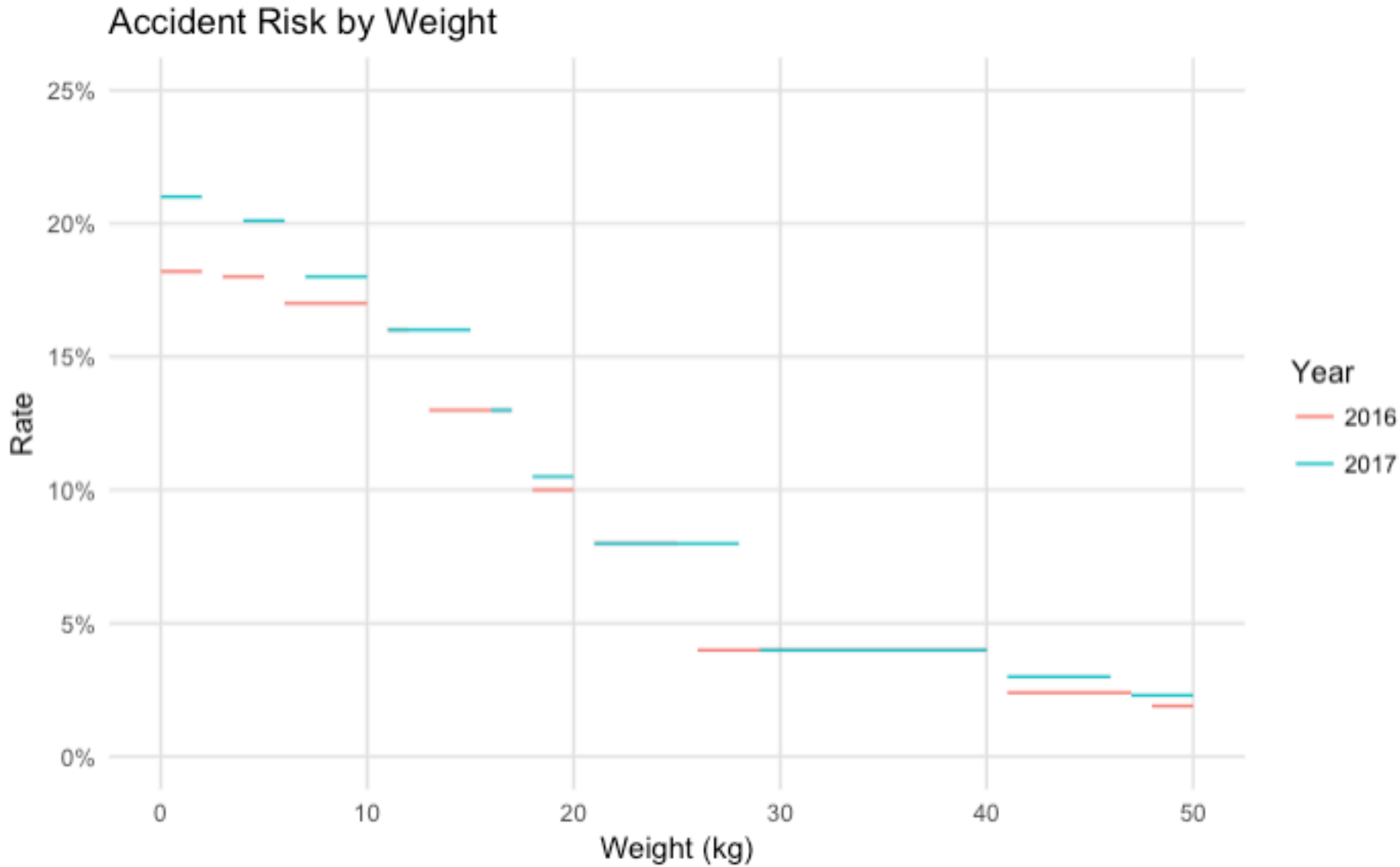

Accident Risk by Weight
20%
15%
10%
5%
0%
0
10
20
30
40
50
Weight (kg)
year
2016

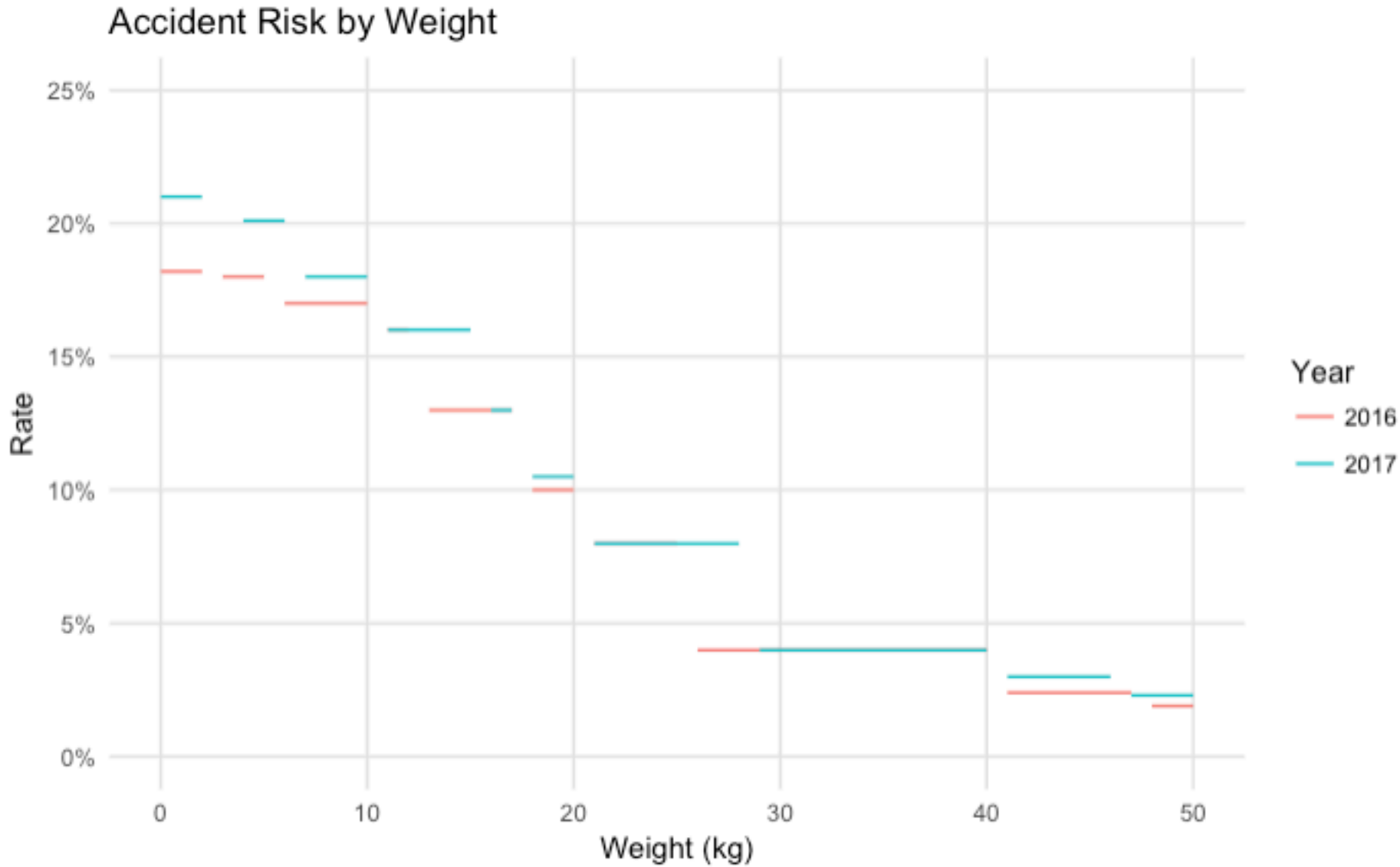

Accident Risk by Weight
20%
15%
10%
5%
0%
0
10
20
30
40
50
Weight (kg)
year
2016

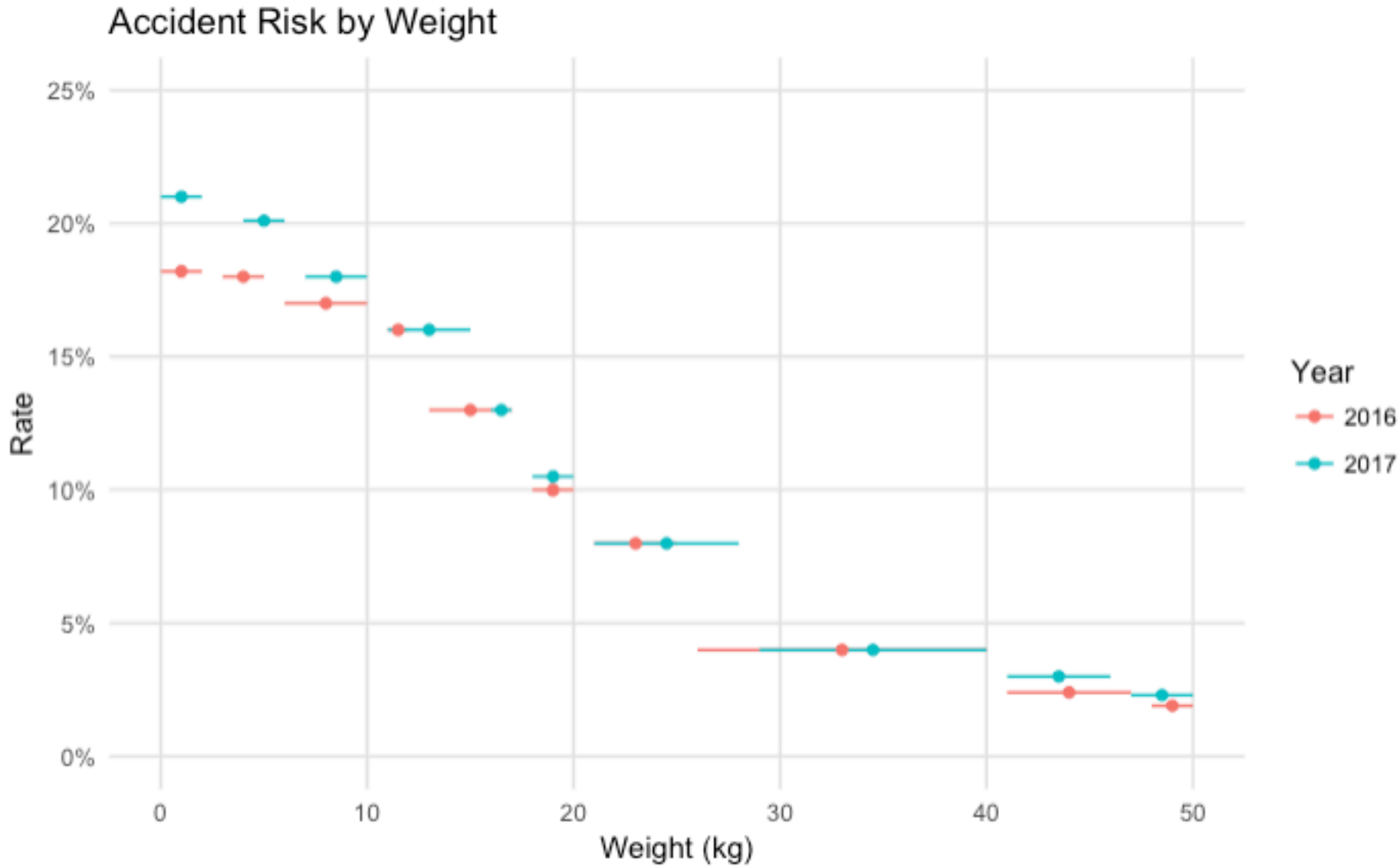

Accident Risk by Weight
20%
15%
10%
5%
0%
0
10
20
30
40
50
Weight (kg)
year
2016

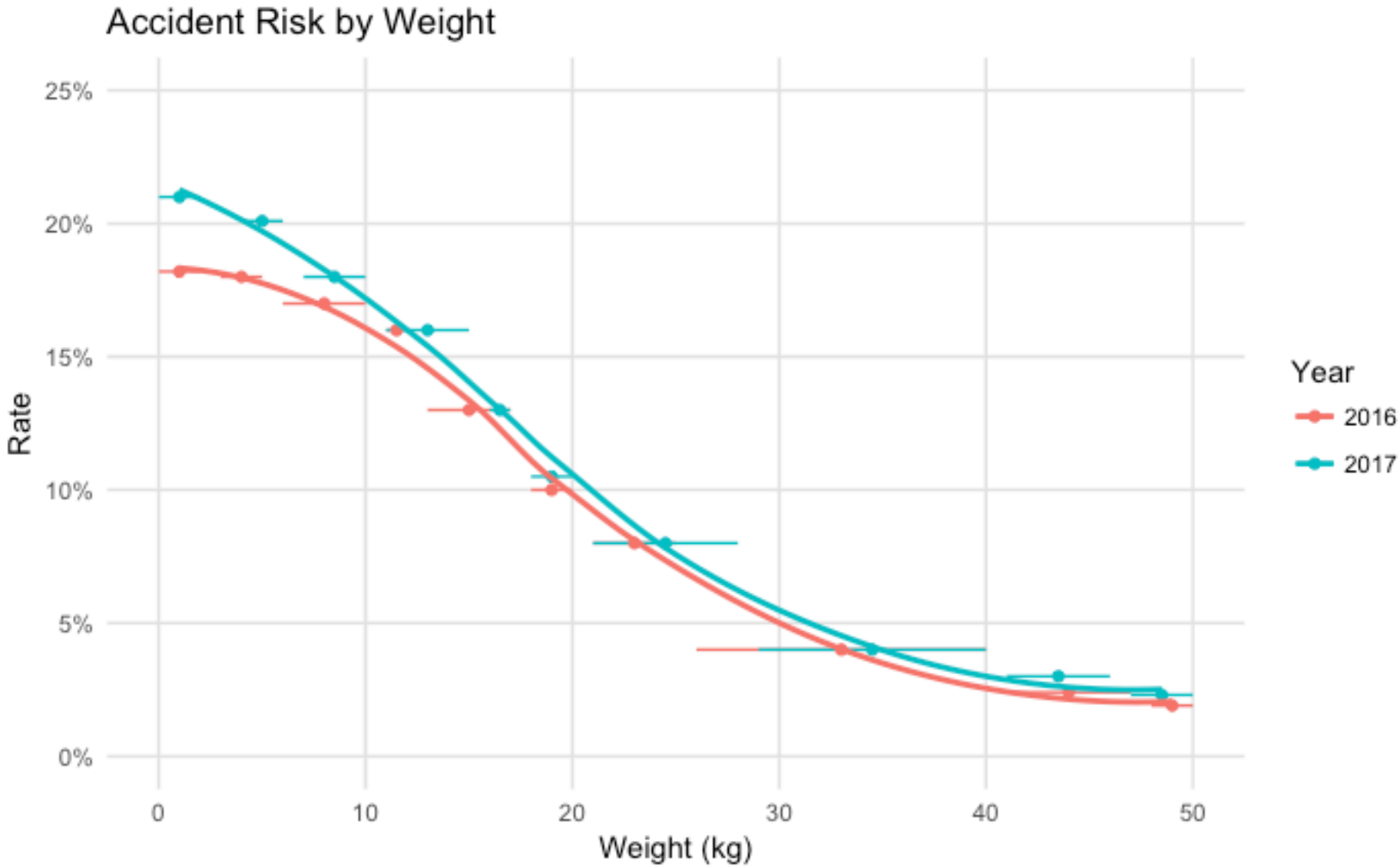
Weight	Risk
0-2	18.2%
3-5	18.0%
6-10	17.0%
11-12	16.0%
13-17	13.0%
18-20	10.0%
21-25	8.00%
26-40	4.00%
41-47	2.40%
48-50	1.90%

Weight	Risk
0-2	21.0%
4-6	20.1%
7-10	18.0%
11-15	16.0%
16-17	13.0%
18-20	10.5%
21-28	8.00%
29-40	4.00%
41-46	3.00%
47-50	2.30%


Accident Risk by Weight
25%
20%
15%
10%
5%
0%
Rate
0
10
20
30
40
50
Weight (kg)
Year
2016
2017

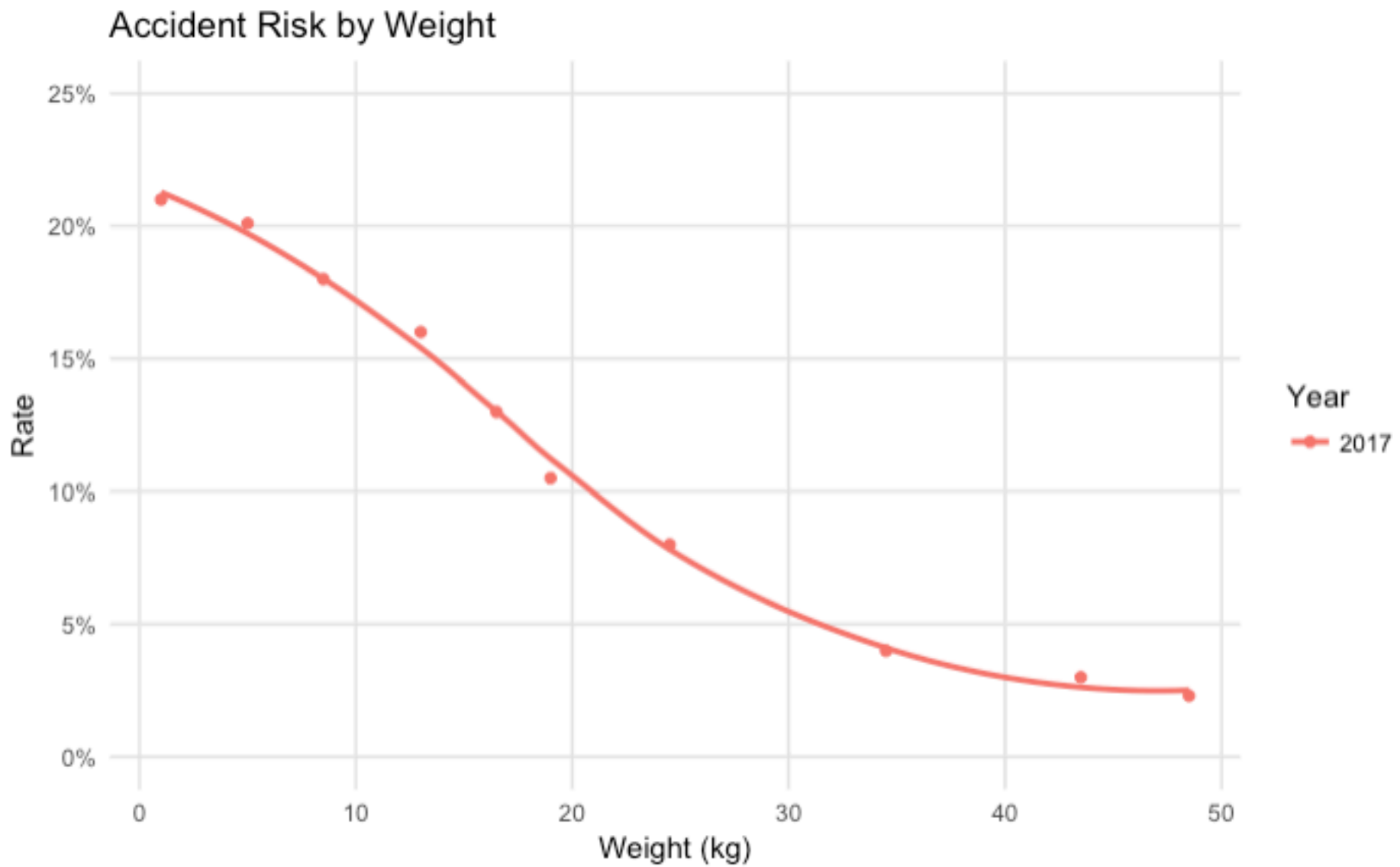

Accident Risk by Weight
25%
20%
15%
10%
5%
0%
Rate
0
10
20
30
40
50
Weight (kg)
Year
2016
2017

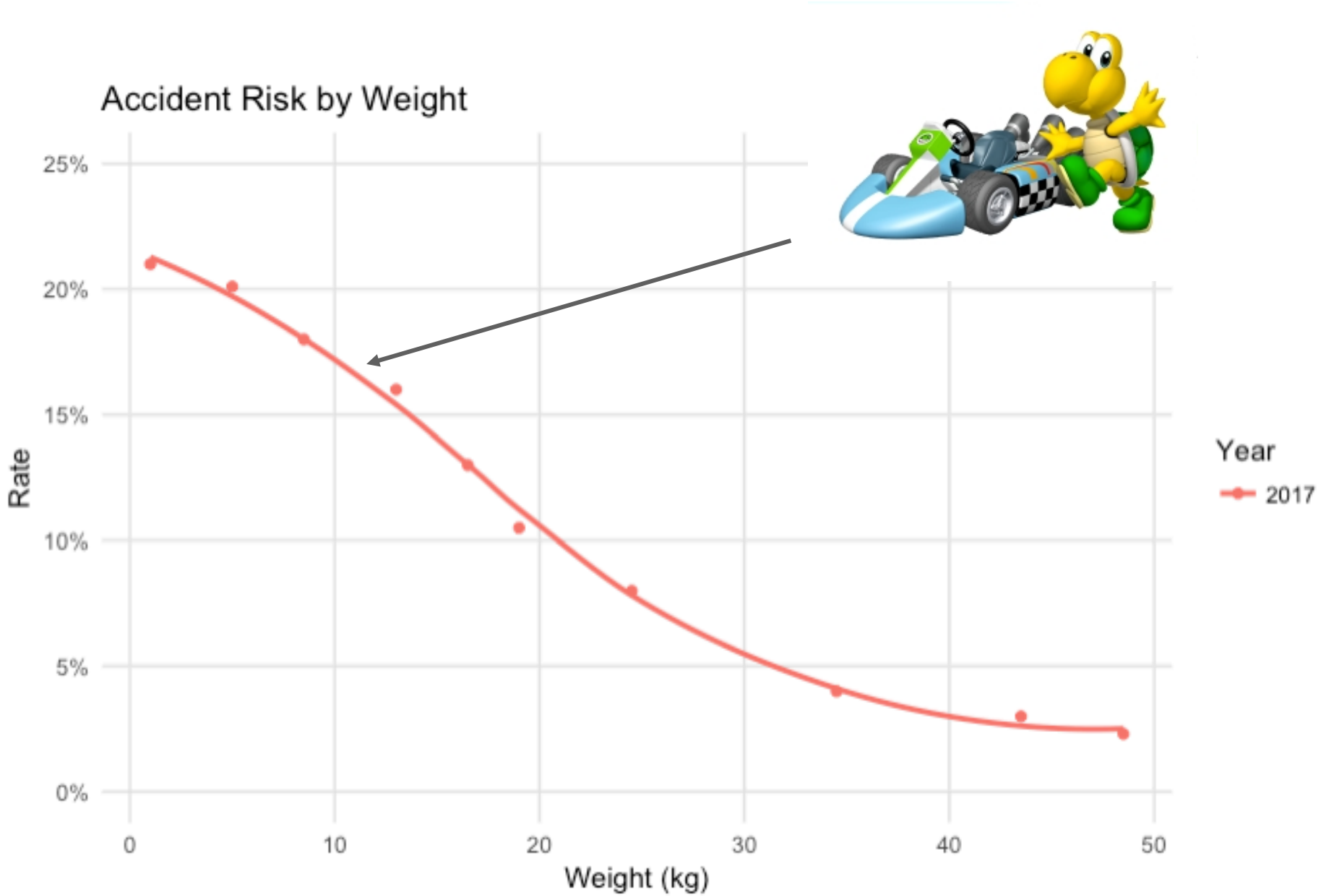

Accident Risk by Weight
25%
20%
15%
10%
5%
0%
Rate
0
10
20
30
40
50
Weight (kg)
Year
2016
2017


Accident Risk by Weight
25%
20%
15%
10%
5%
0%
Rate
0
10
20
30
40
50
Weight (kg)
Year
2016
2017

Weight	Risk
1	21.0%
5	20.1%
8.5	18.0%
13	16.0%
16.5	13.0%
19	10.5%
24.5	8.00%
34.5	4.00%
43.5	3.00%
48.5	2.30%

```
library(tidyverse); library(modelr)

mod <- loess(Risk ~ Weight, data=data, span=0.8)

predict(mod, tibble(Weight=12.5))

grid <- tibble(Weight = seq(0, 50, 0.5)) %>%
    add_predictions(mod, var = "Risk")
```


Accident Risk by Weight
25%
20%
15%
10%
5%
0%
Rate
0
10
20
30
40
50
Weight (kg)
Year
2017


Accident Risk by Weight
25%
20%
15%
10%
5%
0%
Rate
0
10
20
30
40
50
Weight (kg)
Year
2017

data
driven

deviate

infrastructure

Weight	Experience	Speed	Accident
-0.5	-0.3	1.3	1
2.1	-0.8	-1.3	1
-0.1	1.0	-0.3	0
-0.6	-1.2	-2.0	0
0.5	-1.2	-0.6	1
0.7	-1.6	-0.5	1
0.4	0.5	0.3	0
1.6	0.6	0.8	0
-0.6	-0.8	1.1	1
0.9	-1.4	-0.3	1
-0.1	1.5	-1.0	0
-1.2	-1.0	-0.9	0
2.1	-0.7	-1.3	1
1.3	-0.8	-1.1	1
0.3	-1.1	-0.5	1

learning

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(ncols,)))
model.add(Dense(2, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=["accuracy"])
model.fit(X_train, y_train, epochs=10, batch_size=1, verbose=1);

loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy = {:.2f}".format(accuracy))

Accuracy = 0.94
```

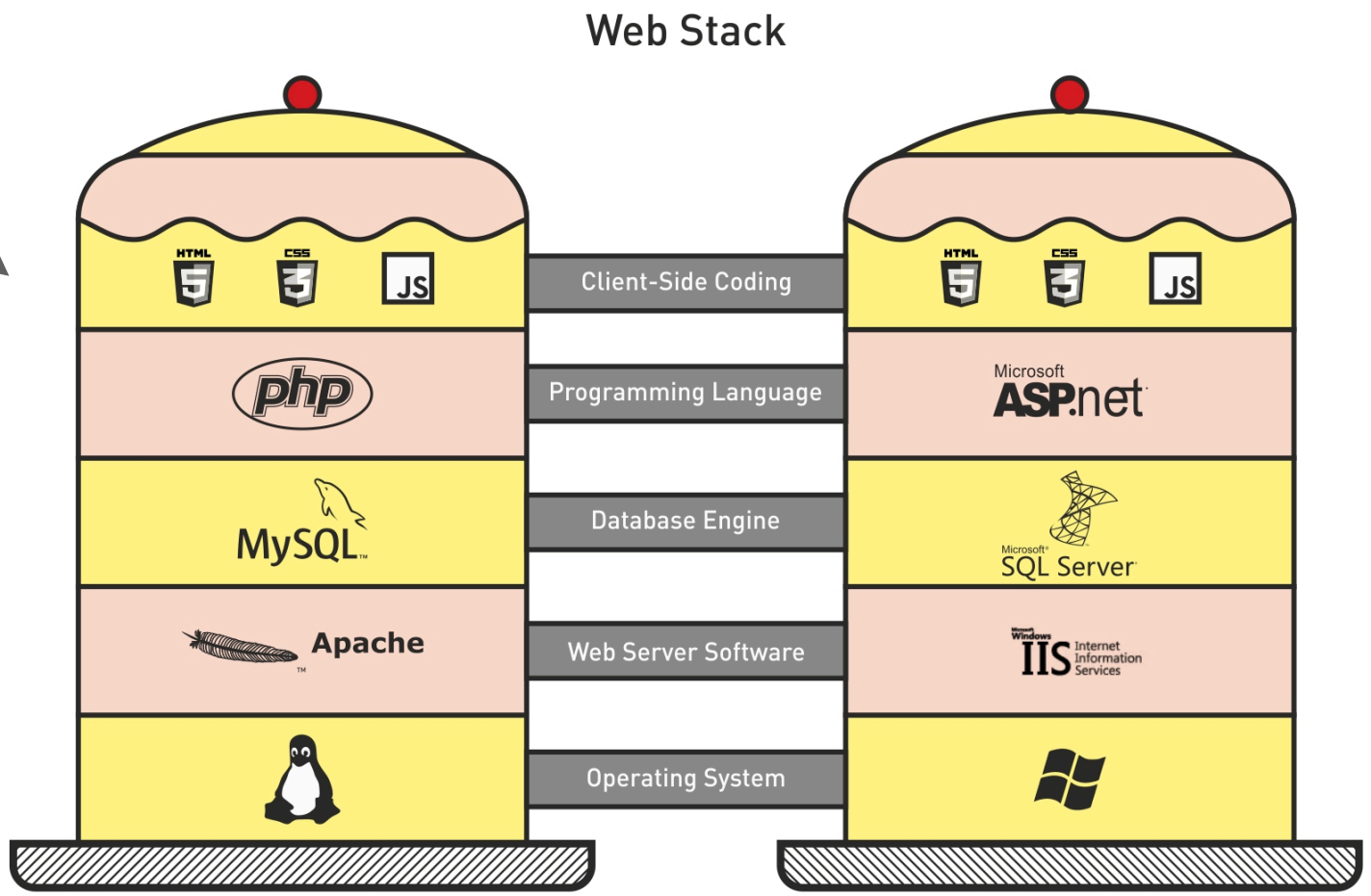
[-2, -2, 0.7]

[-2, -2, 0.7]

```
new_data = np.array([[-2, -2, 0.7]])
model.predict(new_data)
```

0.1964

```
model.predict()
```

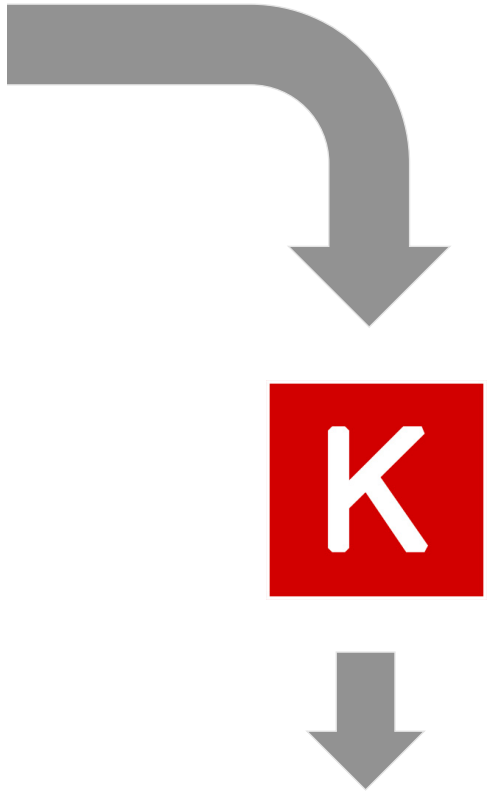

Web Stack
HTML
CSS
JS
Client-Side Coding
php
Programming Language
Microsoft
ASP.net
MySQL
Database Engine
Microsoft
SQL Server
Apache
Web Server Software
Windows
IIS
Internet
Information
Services
Operating System
LAMP
WISA


SUPER MARIO
MARIO
Edition cristal
Edición cristal
6+
CRYSTAL
amiibo
EDITION
Nintendo
amiibo

```
 combos = {
     'Weight': np.arange(-2, 2, 0.1),
     'Experience': np.arange(-2, 2, 0.1),
     'Speed': np.arange(-2, 2, 0.1)
 }


def expand_grid(data_dict):
    """Create a dataframe from every combination of given values."""
    rows = product(*data_dict.values())
    return pd.DataFrame.from_records(rows, columns=data_dict.keys())

crystal = expand_grid(combos)
```

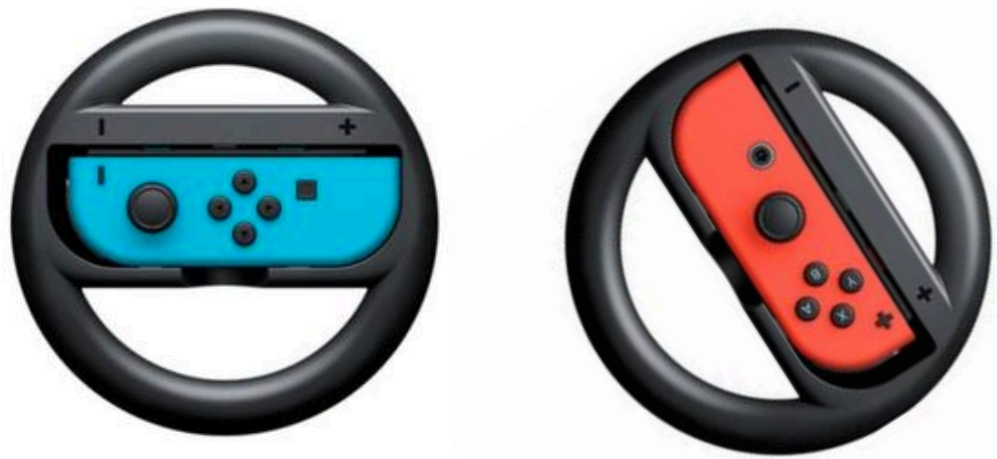
Weight	Experience	Top Speed
-2	-2	-2
-2	-2	-1.9
-2	-2	-1.8
-2	-2	-1.7
-2	-2	-1.6
-2	-2	-1.5
-2	-2	-1.4
-2	-2	-1.3
-2	-2	-1.2
-2	-2	-1.1

Weight	Experience	Top Speed
-2	-2	-2
-2	-2	-1.9
-2	-2	-1.8
-2	-2	-1.7
-2	-2	-1.6
-2	-2	-1.5
-2	-2	-1.4
-2	-2	-1.3
-2	-2	-1.2
-2	-2	-1.1




```
crystal_in = np.array(crystal.values.tolist())
crystal_pred = pd.DataFrame(model.predict(crystal_in))

df_c = pd.concat([crystal.reset_index(drop=True), crystal_pred], axis=1)
```

	Weight	Experience	Top Speed	0	1
4000	-1.8	0	-2	0.997615	0.002385
4001	-1.8	0	-1.9	0.997345	0.002655
4002	-1.8	0	-1.8	0.997044	0.002956
4003	-1.8	0	-1.7	0.996669	0.003331
4004	-1.8	0	-1.6	0.996207	0.003793
39000	0.4	-0.5	-2	0.252056	0.747944
39001	0.4	-0.5	-1.9	0.239986	0.760014
39002	0.4	-0.5	-1.8	0.228317	0.771683
39003	0.4	-0.5	-1.7	0.217054	0.782946
39004	0.4	-0.5	-1.6	0.207301	0.792699
50000	1.1	-1	-2	0.044396	0.955604
50001	1.1	-1	-1.9	0.041424	0.958576
50002	1.1	-1	-1.8	0.038643	0.961357
50003	1.1	-1	-1.7	0.036042	0.963958
50004	1.1	-1	-1.6	0.03361	0.96639



investors

AI™

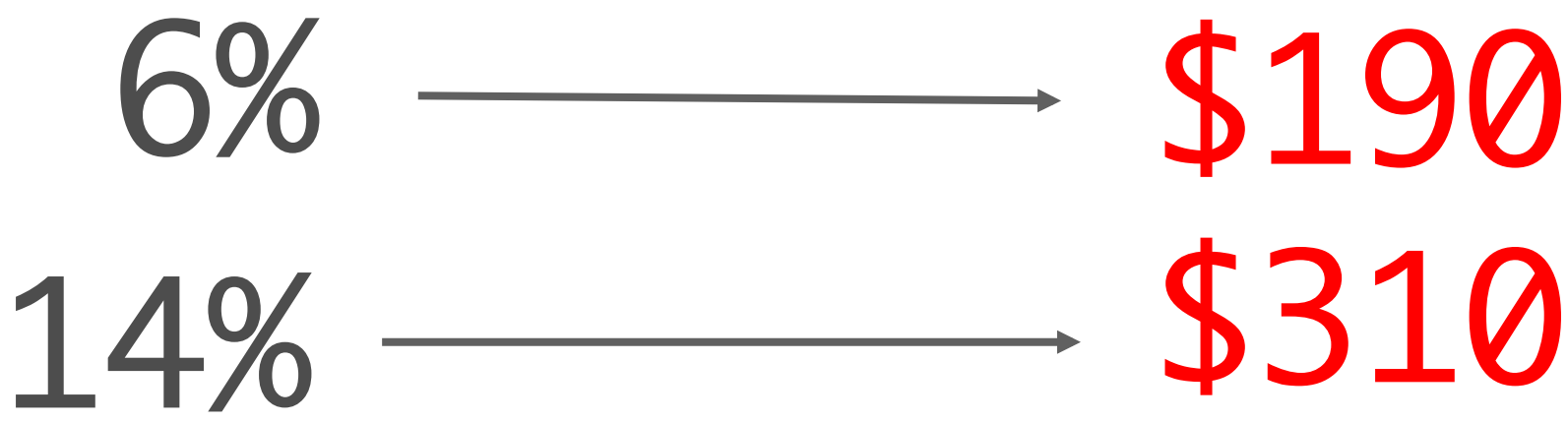
AI™

6%

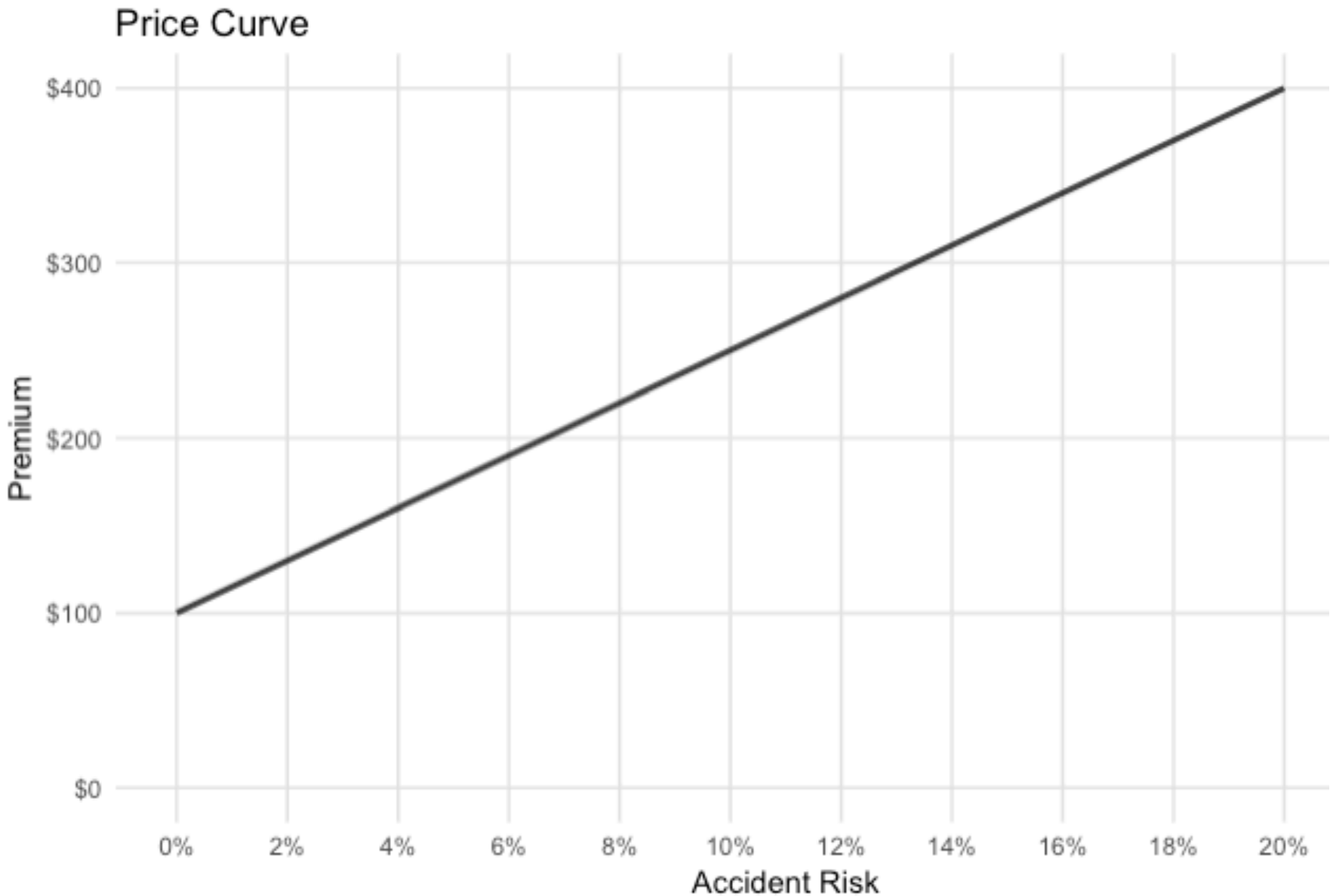
AI™

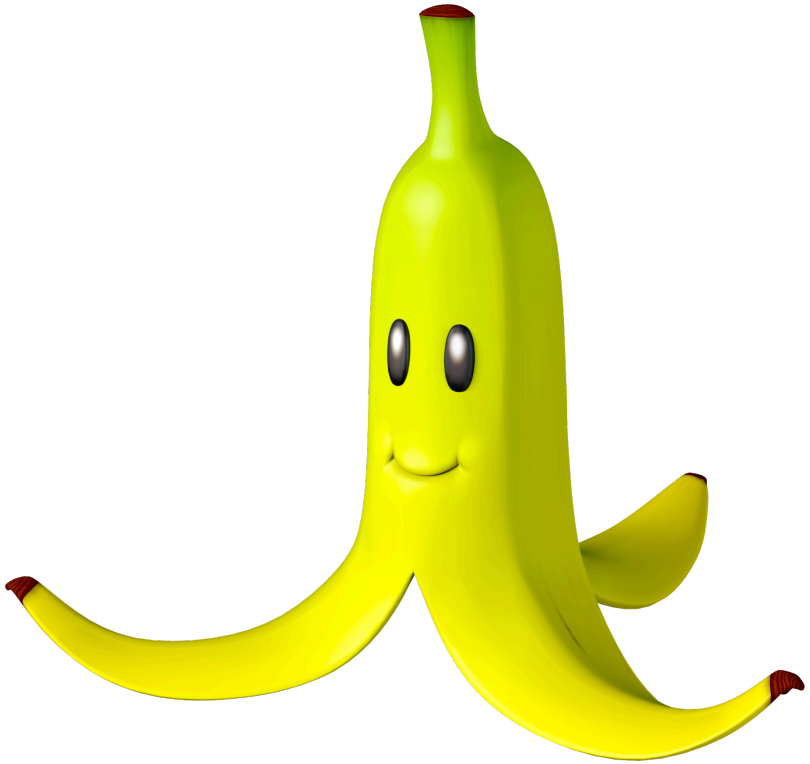
14%

$$y = 1500x + 100$$

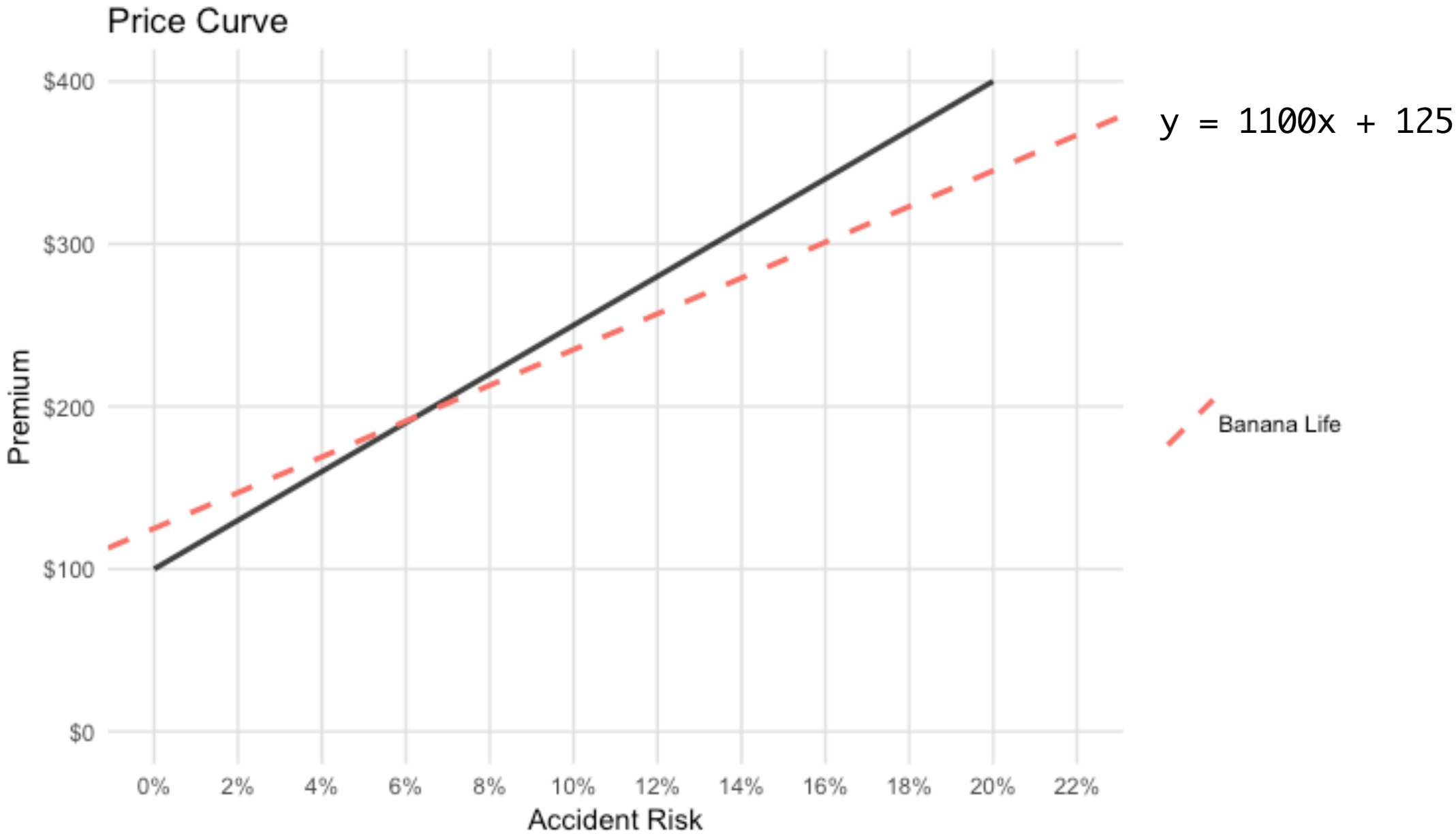

6%
$190
14%
$310

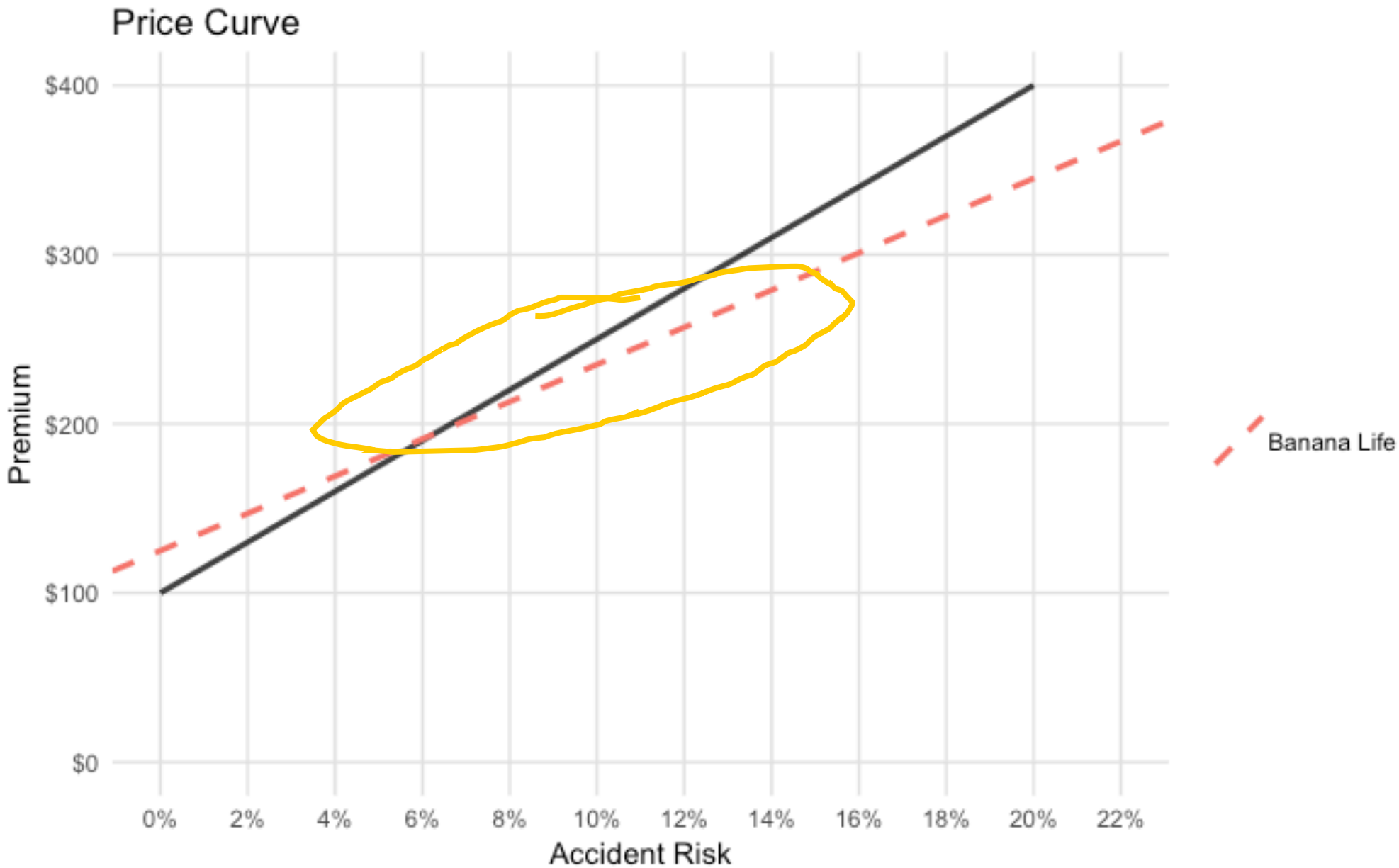
Risk	Premium
2%	$130
4%	$160
6%	$190
8%	$220
10%	$250
12%	$280
14%	$310
16%	$340
18%	$370
20%	$400

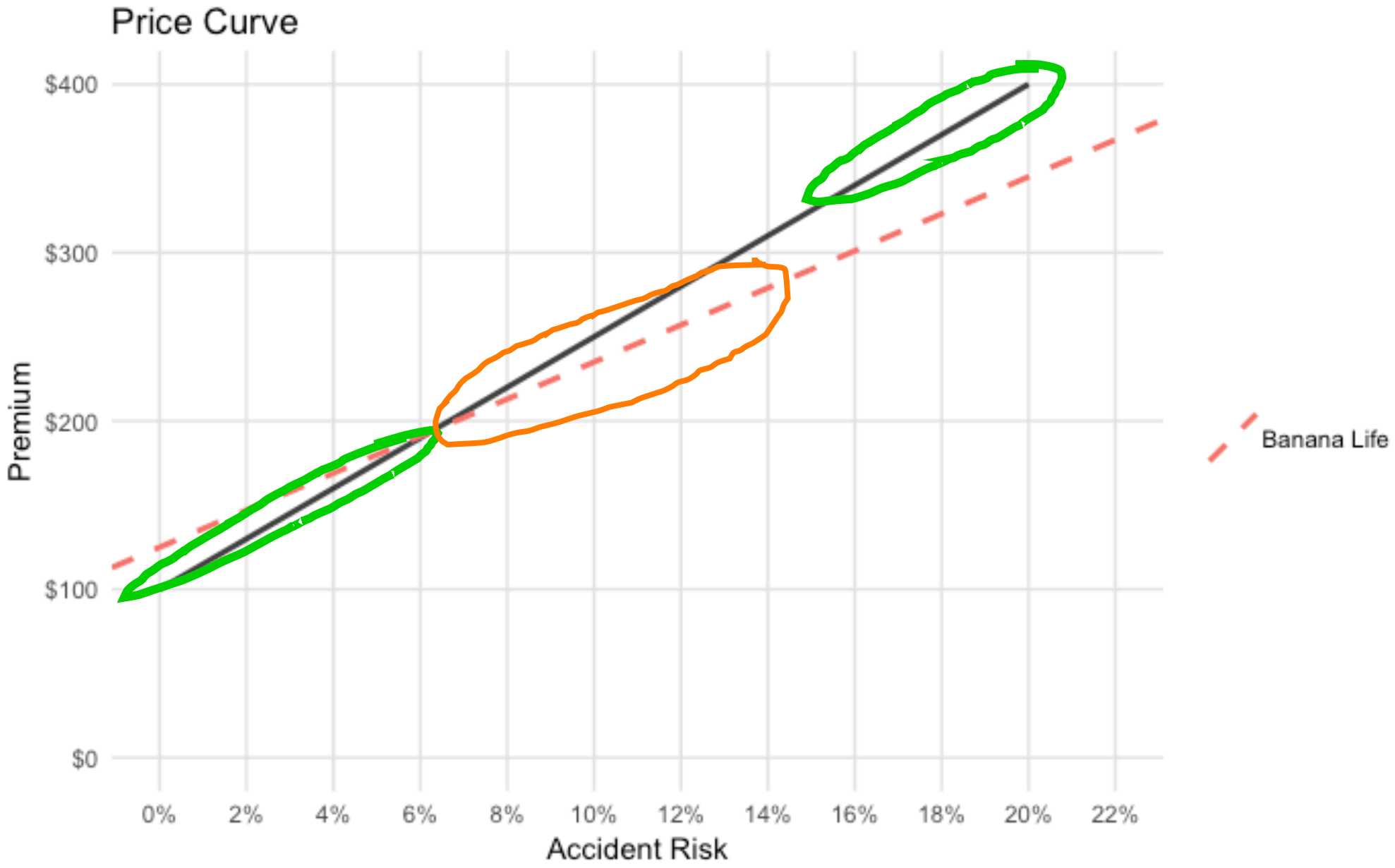

Price Curve
$400
$300
$200
$100
$0
Premium
0%
2%
4%
6%
8%
10%
12%
14%
16%
18%
20%
Accident Risk

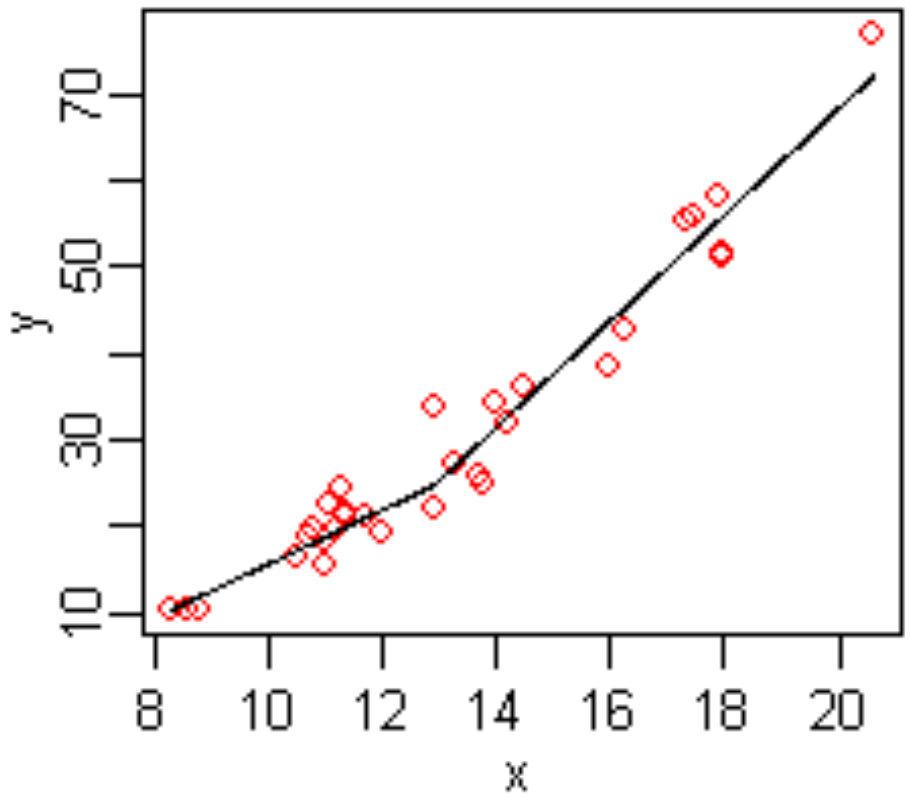


Banana Life Financial

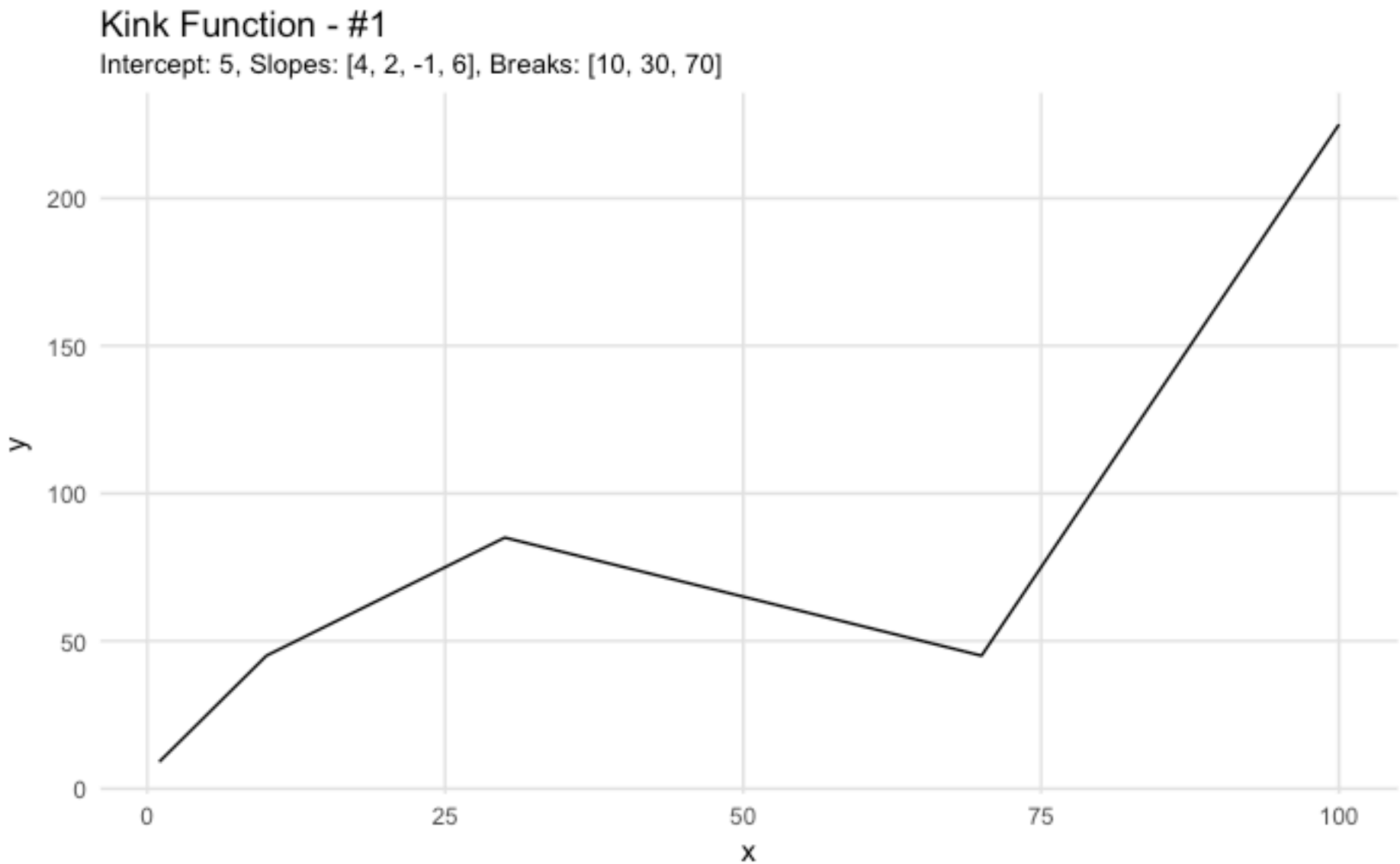

Price Curve
$400
$300
$200
$100
$0
Premium
0%
2%
4%
6%
8%
10%
12%
14%
16%
18%
20%
22%
Accident Risk
y = 1100x + 125
Banana Life

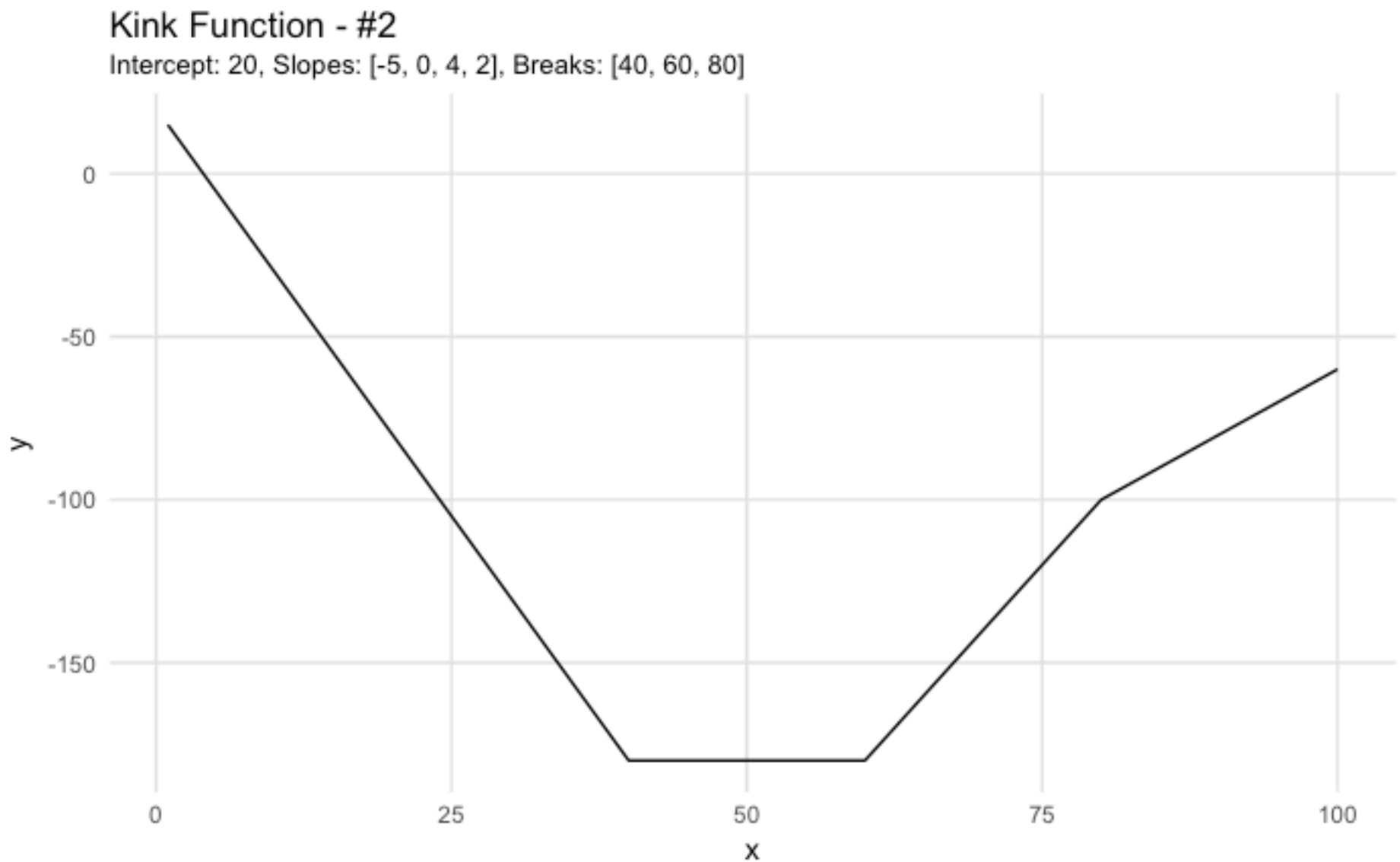

Price Curve
$400
$300
$200
$100
$0
Premium
0%
2%
4%
6%
8%
10%
12%
14%
16%
18%
20%
22%
Accident Risk
Banana Life

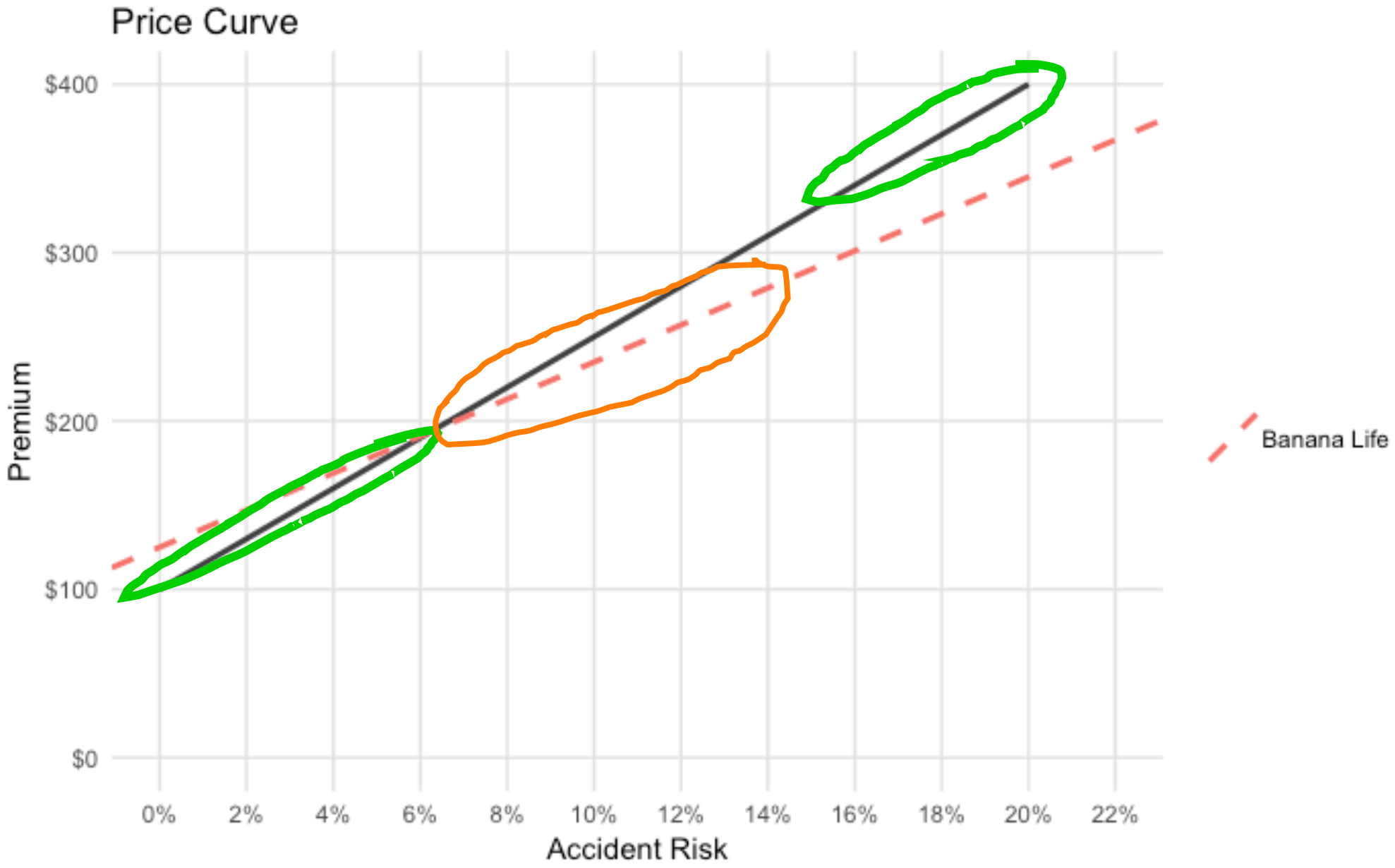

Price Curve
$400
$300
$200
$100
$0
Premium
0%
2%
4%
6%
8%
10%
12%
14%
16%
18%
20%
22%
Accident Risk
Banana Life

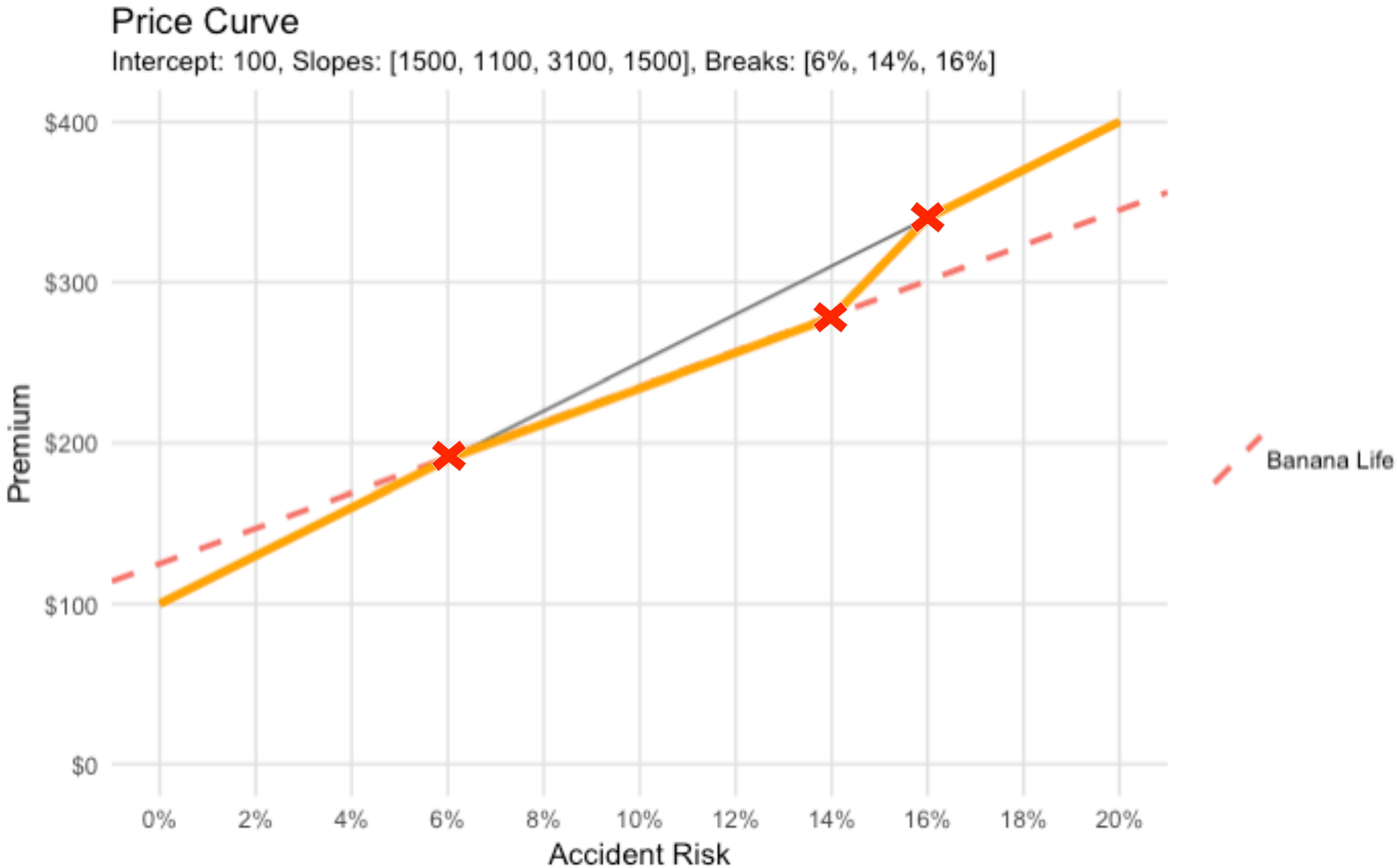

y
10
30
50
70
8
10
12
14
16
18
20
x

```
kink <- function(x, intercept, slopes, breaks) {

    assertive::assert_is_of_length(intercept, n = 1)
    assertive::assert_is_of_length(breaks, n = length(slopes) - 1)

    intercepts <- c(intercept)

    for(i in 1:length(slopes)-1) {
        intercept <- intercepts[i] + slopes[i] * breaks[i] - slopes[i+1] * breaks[i]
        intercepts <- c(intercepts, intercept)
    }

    i = 1 + findInterval(x, breaks)
    y = slopes[i] * x + intercepts[i]

    return(y)

}
```


Kink Function - #1
Intercept: 5, Slopes: [4, 2, -1, 6], Breaks: [10, 30, 70]
200
150
100
50
0
y
0
25
50
75
100
x

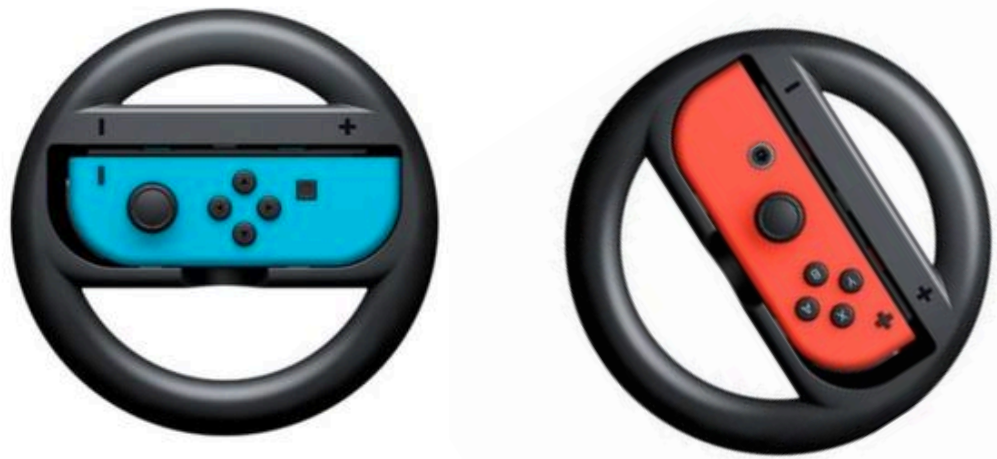

Kink Function - #2
Intercept: 20, Slopes: [-5, 0, 4, 2], Breaks: [40, 60, 80]
0
-50
-100
-150
y
0
25
50
75
100
x


Price Curve
$400
$300
$200
$100
$0
Premium
0%
2%
4%
6%
8%
10%
12%
14%
16%
18%
20%
22%
Accident Risk
Banana Life


Price Curve
Intercept: 100, Slopes: [1500, 1100, 3100, 1500], Breaks: [6%, 14%, 16%]
$400
$300
$200
$100
$0
Premium
0%
2%
4%
6%
8%
10%
12%
14%
16%
18%
20%
Accident Risk
Banana Life

```
kink(
    x = 0.132,
    intercept = 100,
    slopes = c(1500, 1100, 3100, 1500),
    breaks = c(0.06, 0.14, 0.16)
)
```

```
[1] 269.2
```



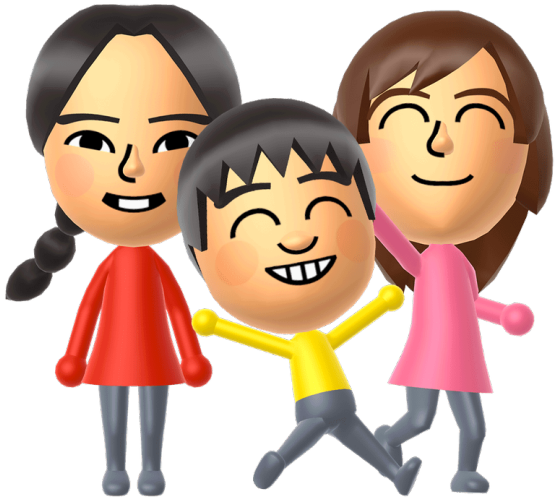

MARIO MOTO
MARIOKART
Nintendo
MARIO CIRCUIT

0 to 

3rd party data

infrastructure

investors



0 to 

Ahmed Badruddin
CEO

WATRHUB

Shea Balish
CEO & Co-Founder

REP.AI

Eva Wong
Co-founder & COO

BORROWELL

Cian O'Sullivan
Founder

BEAGLE.AI

9:15 - 10:00 AM - PANEL DISCUSSION

x

NEXT GEN: THE START-UPS PANEL

Big Data, Machine Learning and AI affect numerous verticals and have resulted in a plethora of innovative start ups and solutions. This panel discussion brings together Toronto's brightest and freshest minds who will share with you best practices and tips for Big Data implementation (and success).

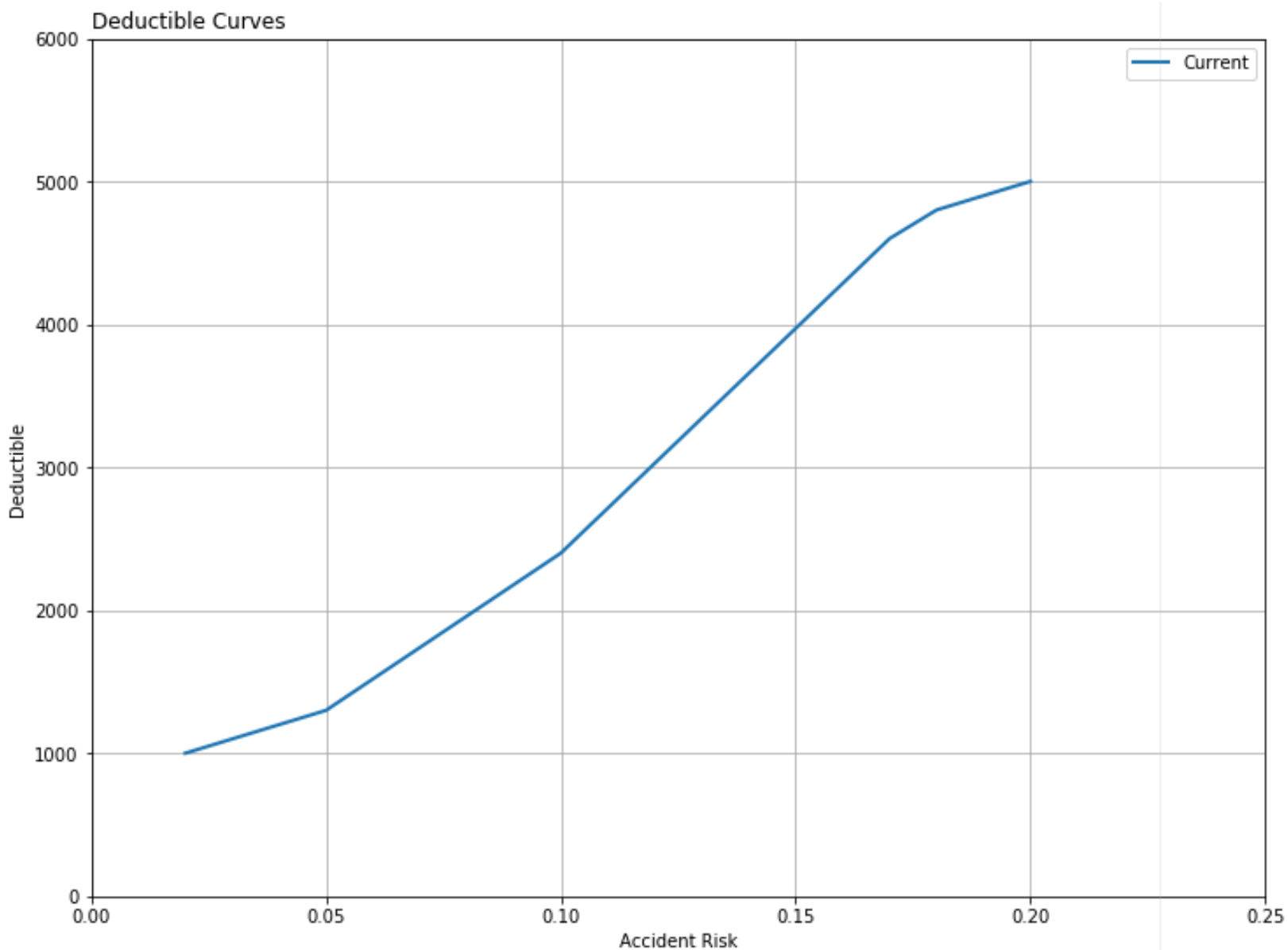
maxhumber

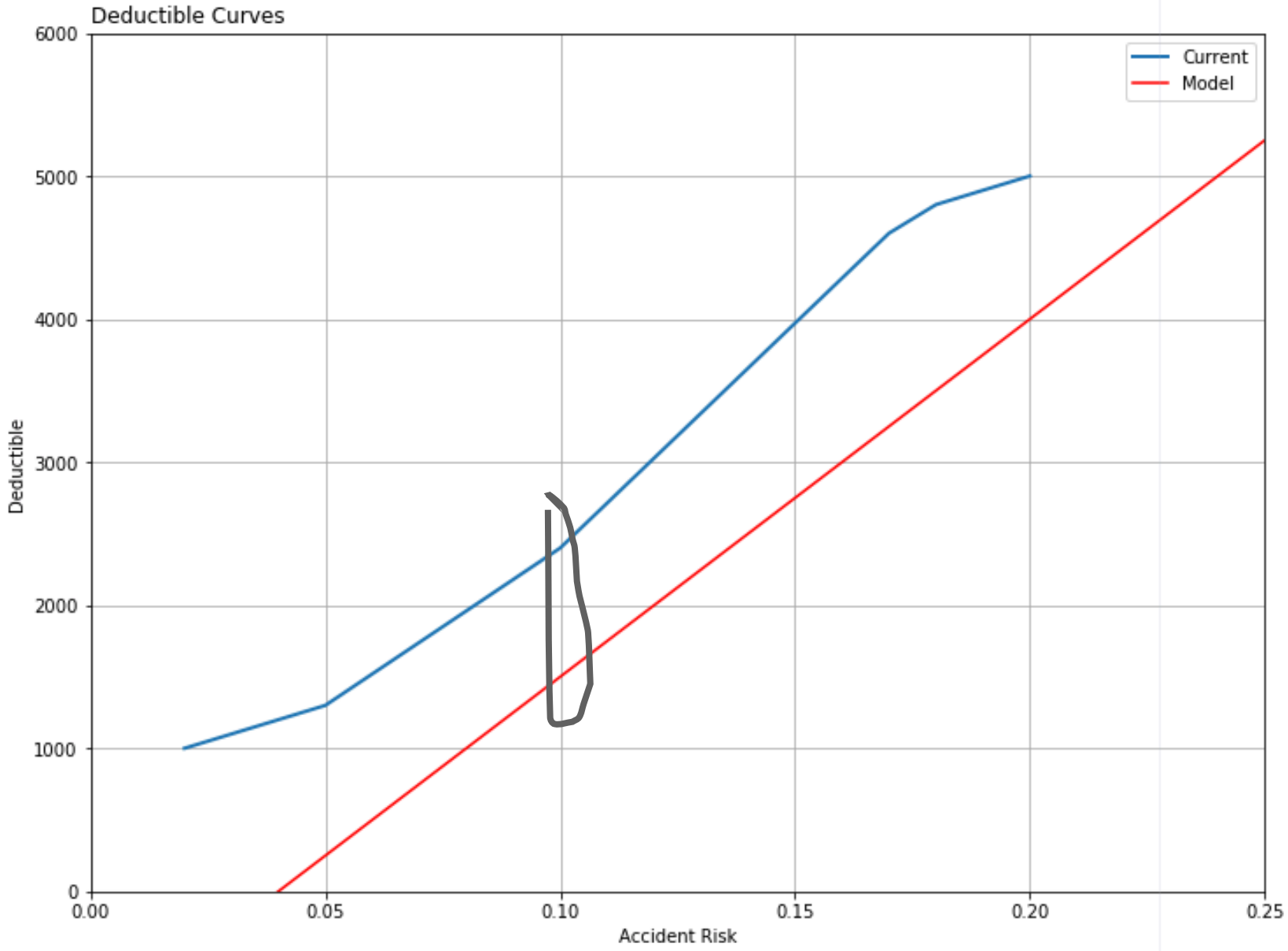


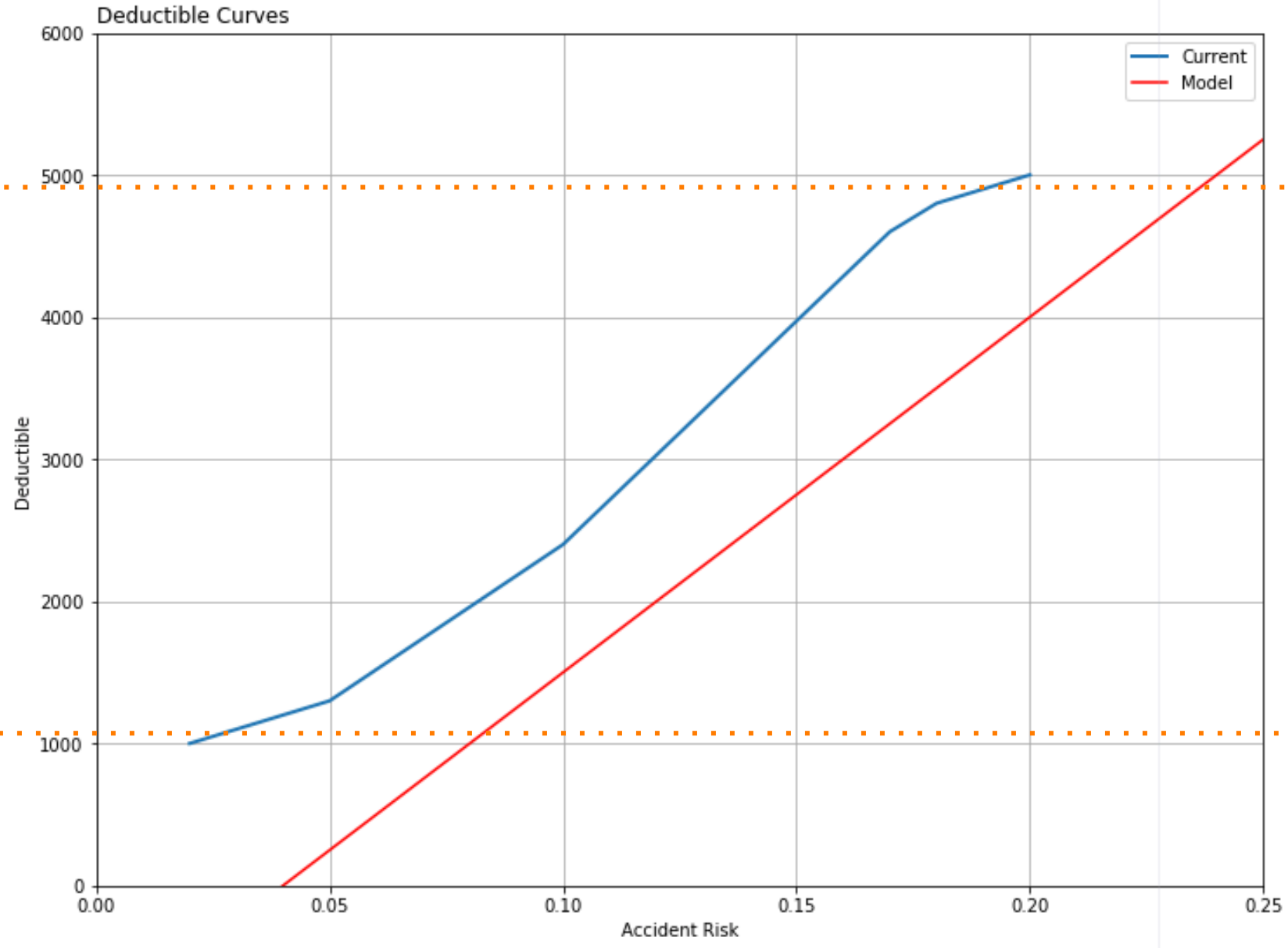
bonus

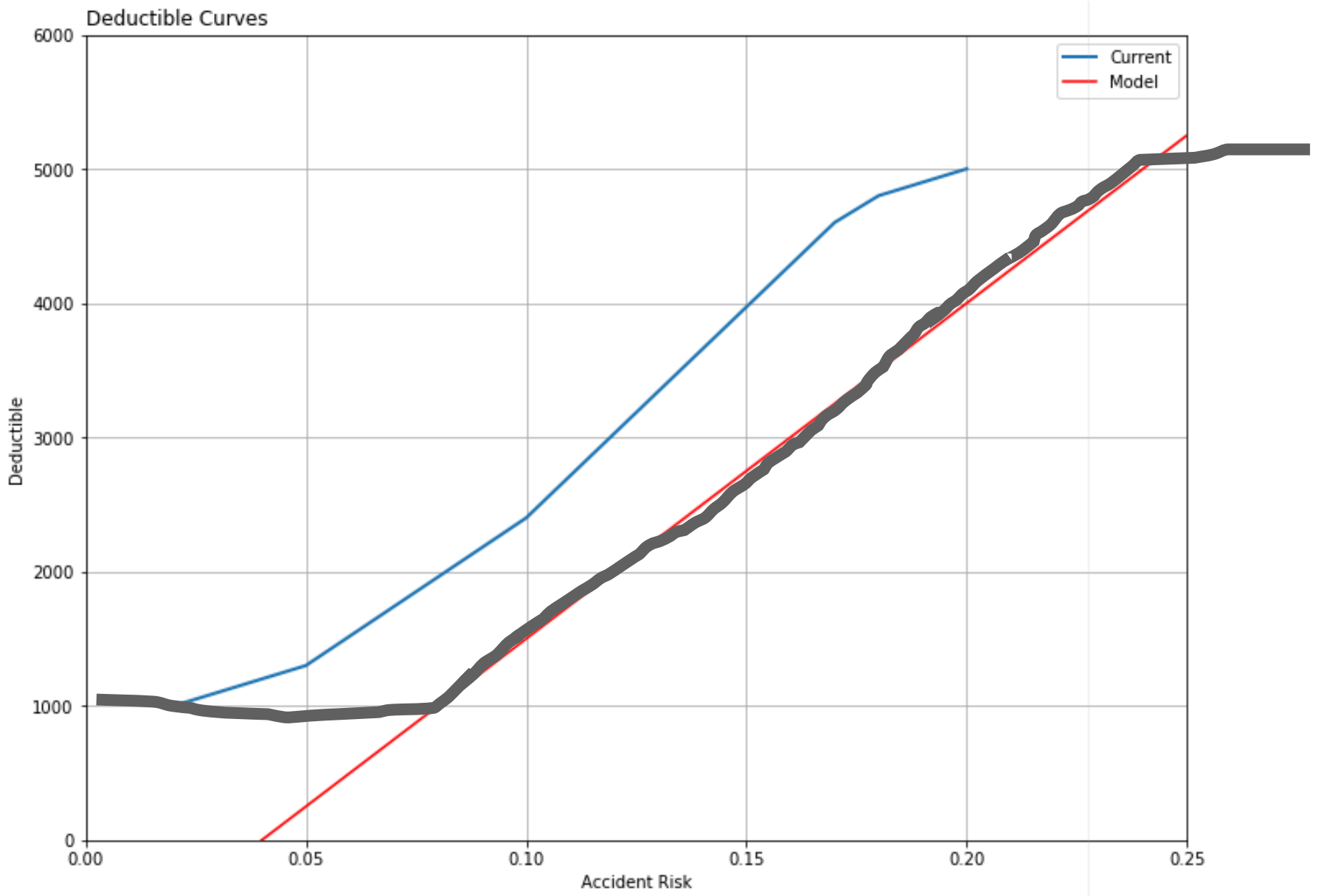
regulators

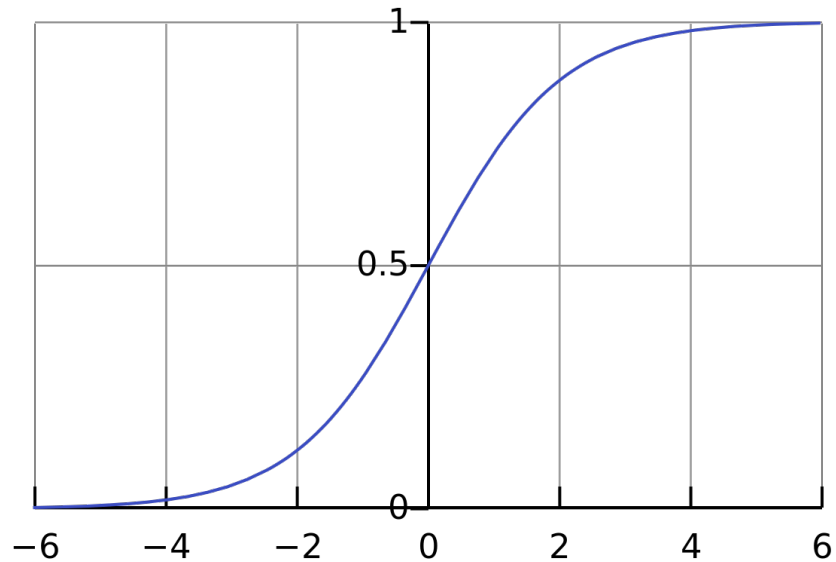
Risk	Deductible
20%	$5000
18%	$4800
17%	$4600
10%	$2400
5%	$1300
4%	$1200
2%	$1000


Deductible Curves
Current
Deductible
Accident Risk
6000
5000
4000
3000
2000
1000
0
0.00
0.05
0.10
0.15
0.20
0.25

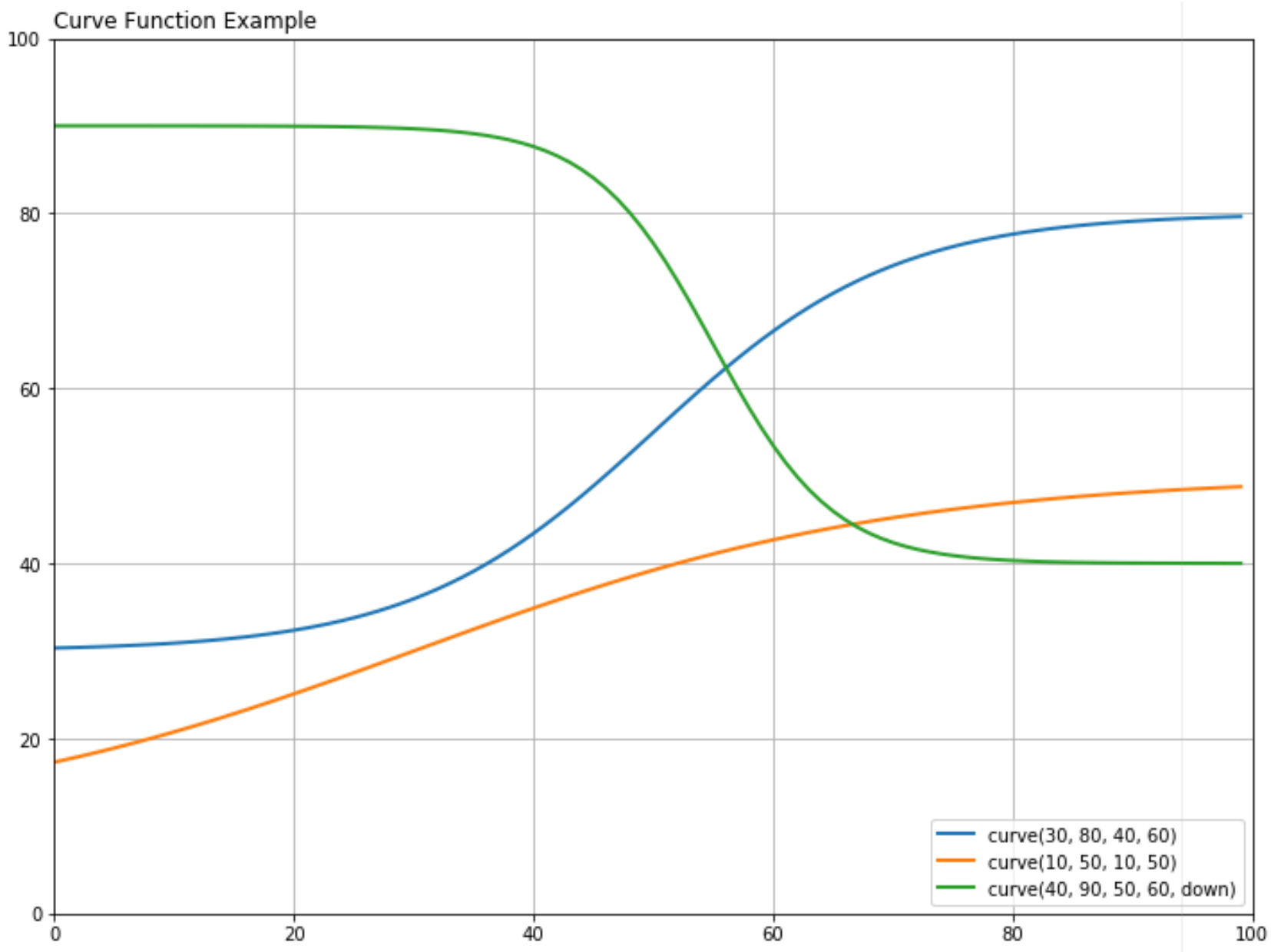

Deductible Curves
Current
Model
Deductible
Accident Risk
6000
5000
4000
3000
2000
1000
0
0.00
0.05
0.10
0.15
0.20
0.25

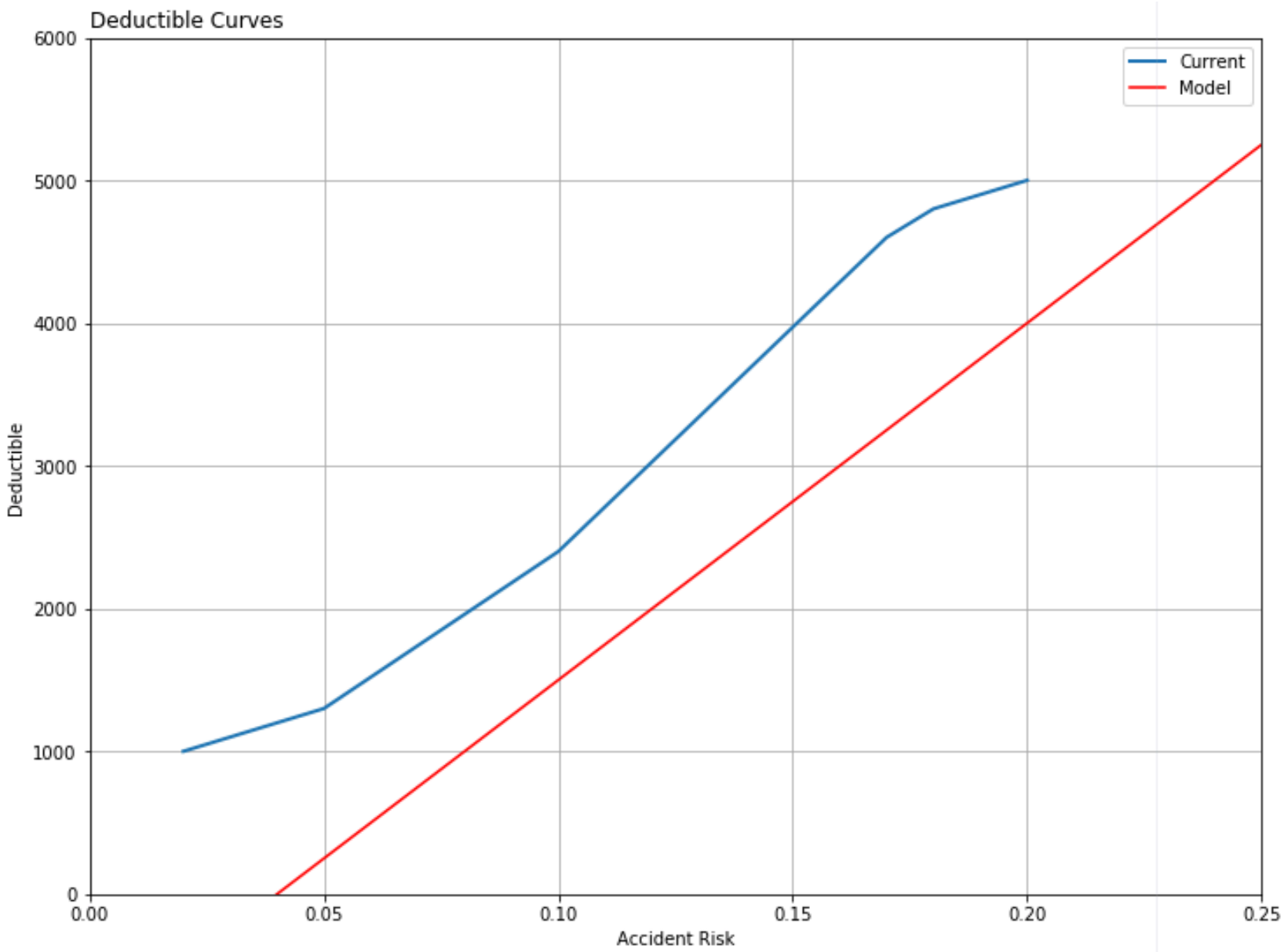

Deductible Curves
Current
Model
Deductible
Accident Risk
6000
5000
4000
3000
2000
1000
0
0.00
0.05
0.10
0.15
0.20
0.25

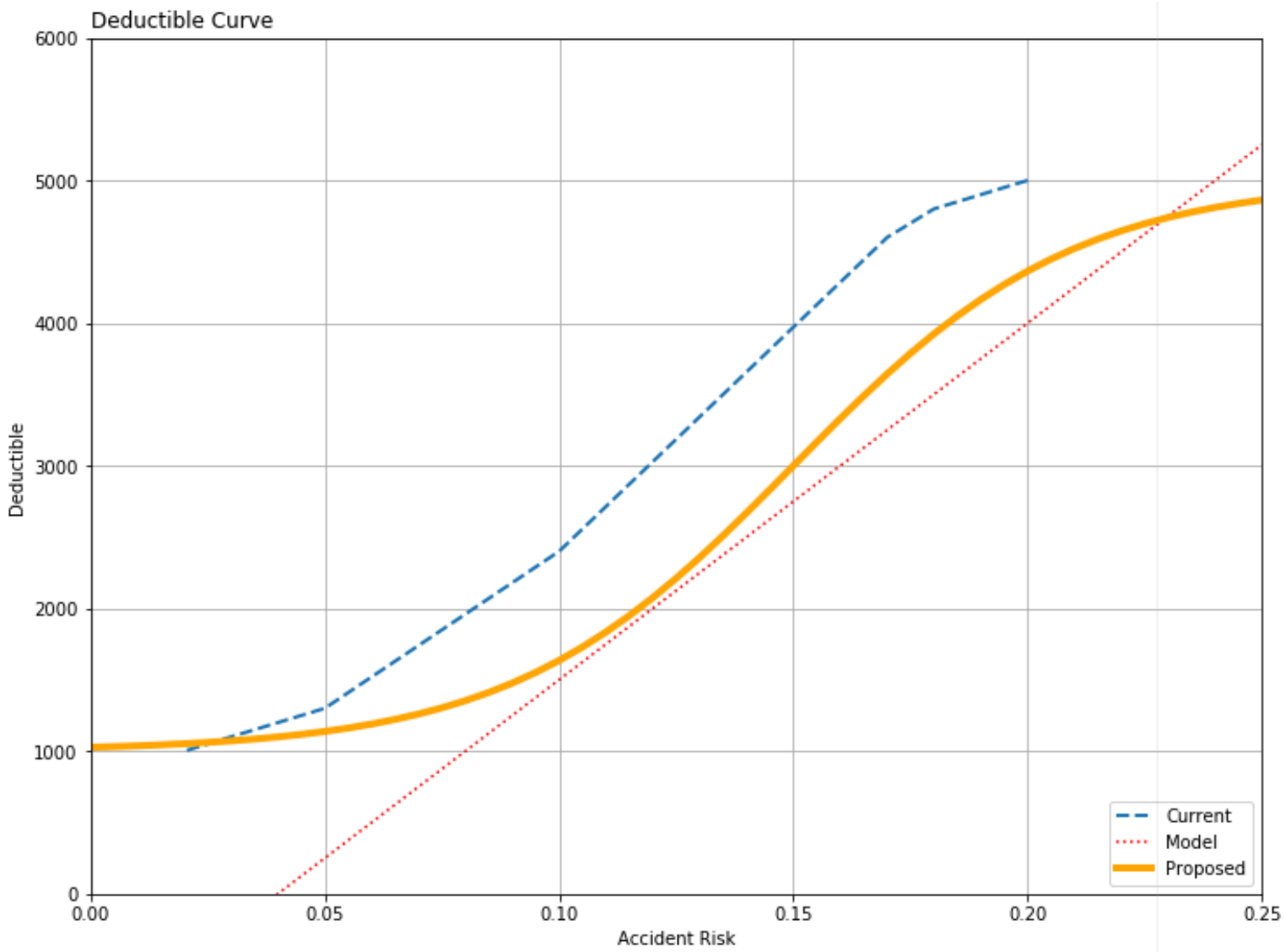

Deductible Curves
Current
Model
Deductible
Accident Risk
6000
5000
4000
3000
2000
1000
0
0.00
0.05
0.10
0.15
0.20
0.25


1
0.5
0
−6
−4
−2
0
2
4
6

```
def curve(x, ymin, ymax, xhl, xhu, up=True):
    a = (xhl + xhu) / 2
    b = 2 / abs(xhl - xhu)
    c = ymin
    d = ymax - c
    if up == True:
        y = c + ( d / ( 1 + np.exp(1)**( -b * (x - a) ) ) )
    elif up == False:
        y = c + ( d / ( 1 + np.exp( b * (x - a) ) ) )
    else:
        None
    return y
```


Curve Function Example
100
80
60
40
20
0
0
20
40
60
80
100
curve(30, 80, 40, 60)
curve(10, 50, 10, 50)
curve(40, 90, 50, 60, down)


Deductible Curves
Current
Model
Deductible
Accident Risk
6000
5000
4000
3000
2000
1000
0
0.00
0.05
0.10
0.15
0.20
0.25




```
df_new = pd.DataFrame({'Risk': np.arange(0, 0.30, 0.005)})
df_new = df_new.assign(Deductible=curve(df_new.prob, ymin=1000, ymax=5000, xhl=0.12, xhu=0.18))
```