Exercício 2

In [ ]:
```
import plotnine as p
from plotnine import *
import pandas as pd
import numpy as np
```

In [ ]:
```
data = pd.read_csv("murders.csv")
data
```

Out[ ]:

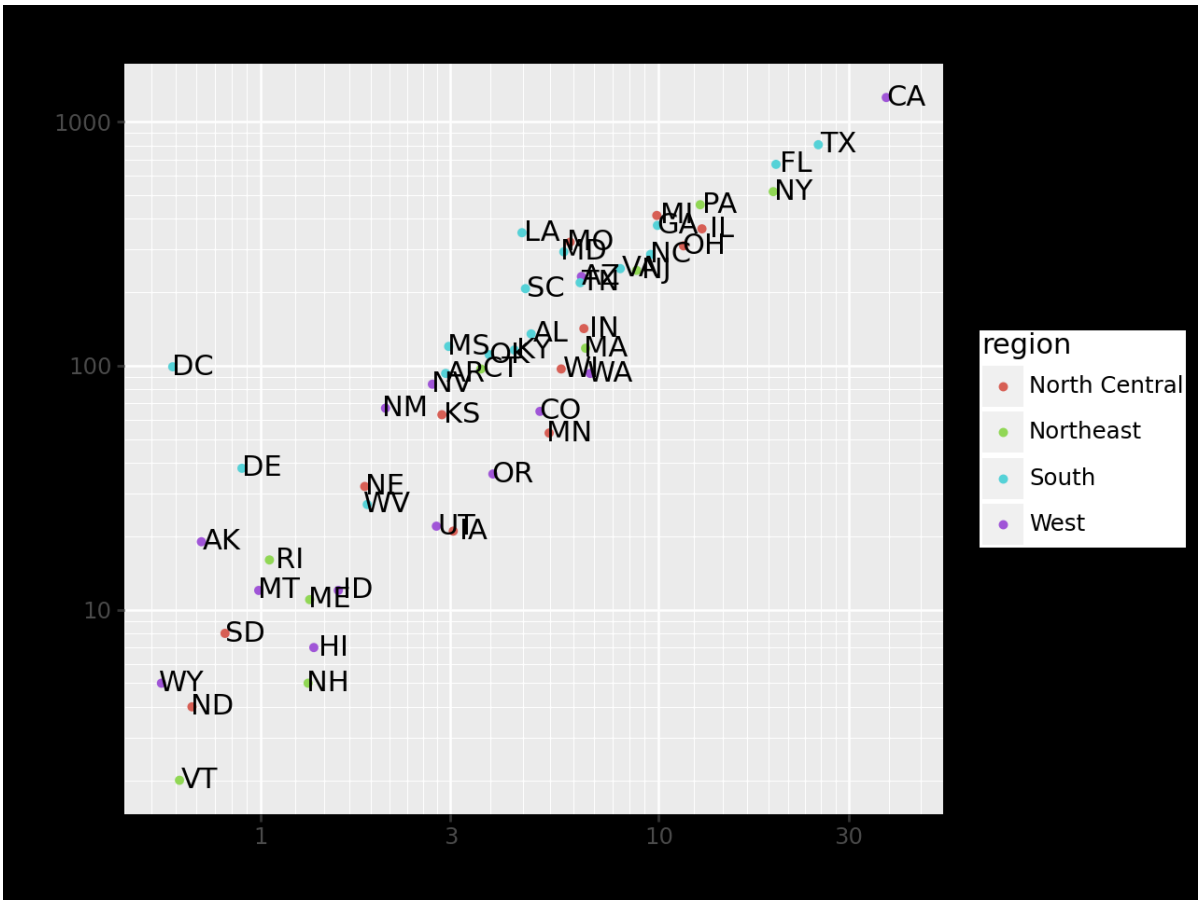
	state	abb	region	population	total
0	Alabama	AL	South	4779736	135
1	Alaska	AK	West	710231	19
2	Arizona	AZ	West	6392017	232
3	Arkansas	AR	South	2915918	93
4	California	CA	West	37253956	1257
5	Colorado	CO	West	5029196	65
6	Connecticut	CT	Northeast	3574097	97
7	Delaware	DE	South	897934	38
8	District of Columbia	DC	South	601723	99
9	Florida	FL	South	19687653	669
10	Georgia	GA	South	9920000	376
11	Hawaii	HI	West	1360301	7
12	Idaho	ID	West	1567582	12
13	Illinois	IL	North Central	12830632	364
14	Indiana	IN	North Central	6483802	142
15	Iowa	IA	North Central	3046355	21
16	Kansas	KS	North Central	2853118	63
17	Kentucky	KY	South	4339367	116
18	Louisiana	LA	South	4533372	351
19	Maine	ME	Northeast	1328361	11
20	Maryland	MD	South	5773552	293
21	Massachusetts	MA	Northeast	6547629	118
22	Michigan	MI	North Central	9883640	413
23	Minnesota	MN	North Central	5303925	53
24	Mississippi	MS	South	2967297	120
25	Missouri	MO	North Central	5988927	321
26	Montana	MT	West	989415	12
27	Nebraska	NE	North Central	1826341	32
28	Nevada	NV	West	2700551	84
29	New Hampshire	NH	Northeast	1316470	5
30	New Jersey	NJ	Northeast	8791894	246
31	New Mexico	NM	West	2059179	67
32	New York	NY	Northeast	19378102	517
33	North Carolina	NC	South	9535483	286
34	North Dakota	ND	North Central	672591	4
35	Ohio	OH	North Central	11536504	310
36	Oklahoma	OK	South	3751351	111
37	Oregon	OR	West	3831074	36
38	Pennsylvania	PA	Northeast	12702379	457

	state	abb	region	population	total
39	Rhode Island	RI	Northeast	1052567	16
40	South Carolina	SC	South	4625364	207
41	South Dakota	SD	North Central	814180	8
42	Tennessee	TN	South	6346105	219
43	Texas	TX	South	25145561	805
44	Utah	UT	West	2763885	22
45	Vermont	VT	Northeast	625741	2
46	Virginia	VA	South	8001024	250
47	Washington	WA	West	6724540	93
48	West Virginia	WV	South	1852994	27
49	Wisconsin	WI	North Central	5686986	97
50	Wyoming	WY	West	563626	5

In [ ]:
```
p = ggplot(data)
type(p)
```

Out[ ]:
```
plotnine.ggplot.ggplot
```

In [ ]:
```
from cmath import log10
p = ggplot(data)
p = p+ geom_point(aes(x=data["population"] / 1e6, y="total", color="region"
p = p+ geom_text(aes(data["population"]/1e6, "total", label = "abb"), nudge_
p = p+ scale_x_log10() + scale_y_log10()
p = p+ xlab("Populations in millions (log scale)") + ylab("Total number of r
p
```

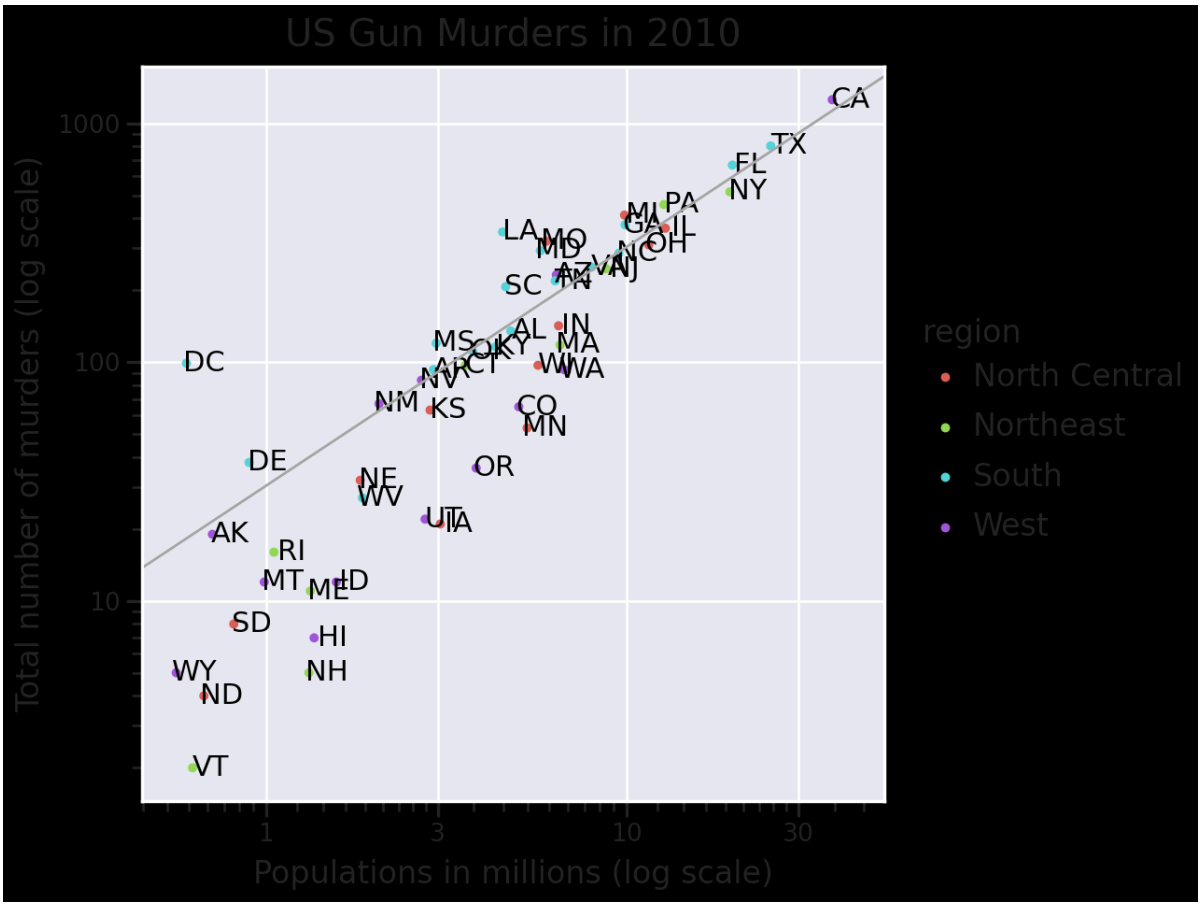


In [ ]:
```
data[['rate']] = (data["total"].sum() / data["population"].sum()) * 10 ** 6
r = data[['rate']]
```

In [ ]:
```
linear_r = np.log10(r)
linear_r
type(linear_r)
```

Out[ ]:
```
pandas.core.frame.DataFrame
```

In [ ]:
```
p = p+ geom_abline(intercept = linear_r, color = "darkgray")
p = p+ theme_seaborn()
plot1 = p
plot1
```



2.2

In [ ]:
```
diamonds = pd.read_csv("diamonds.csv")
diamonds
```

Out[ ]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...	...
53935	53936	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

53940 rows × 11 columns

In [ ]:
```
ideal = diamonds[diamonds["cut"] == "Ideal"]
premium = diamonds[diamonds["cut"] == "Premium"]
premium
```

Out[ ]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
12	13	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33
14	15	0.20	Premium	E	SI2	60.2	62.0	345	3.79	3.75	2.27
15	16	0.32	Premium	E	I1	60.9	58.0	345	4.38	4.42	2.68
...	...	...	...	...	...	...	...	...	...	...	...
53928	53929	0.79	Premium	E	SI2	61.4	58.0	2756	6.03	5.96	3.68
53930	53931	0.71	Premium	E	SI1	60.5	55.0	2756	5.79	5.74	3.49
53931	53932	0.71	Premium	F	SI1	59.8	62.0	2756	5.74	5.73	3.43
53934	53935	0.72	Premium	D	SI1	62.7	59.0	2757	5.69	5.73	3.58
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74

13791 rows × 11 columns

In [ ]:
```
algebra = diamonds[["carat","cut","price"]]
algebra
```

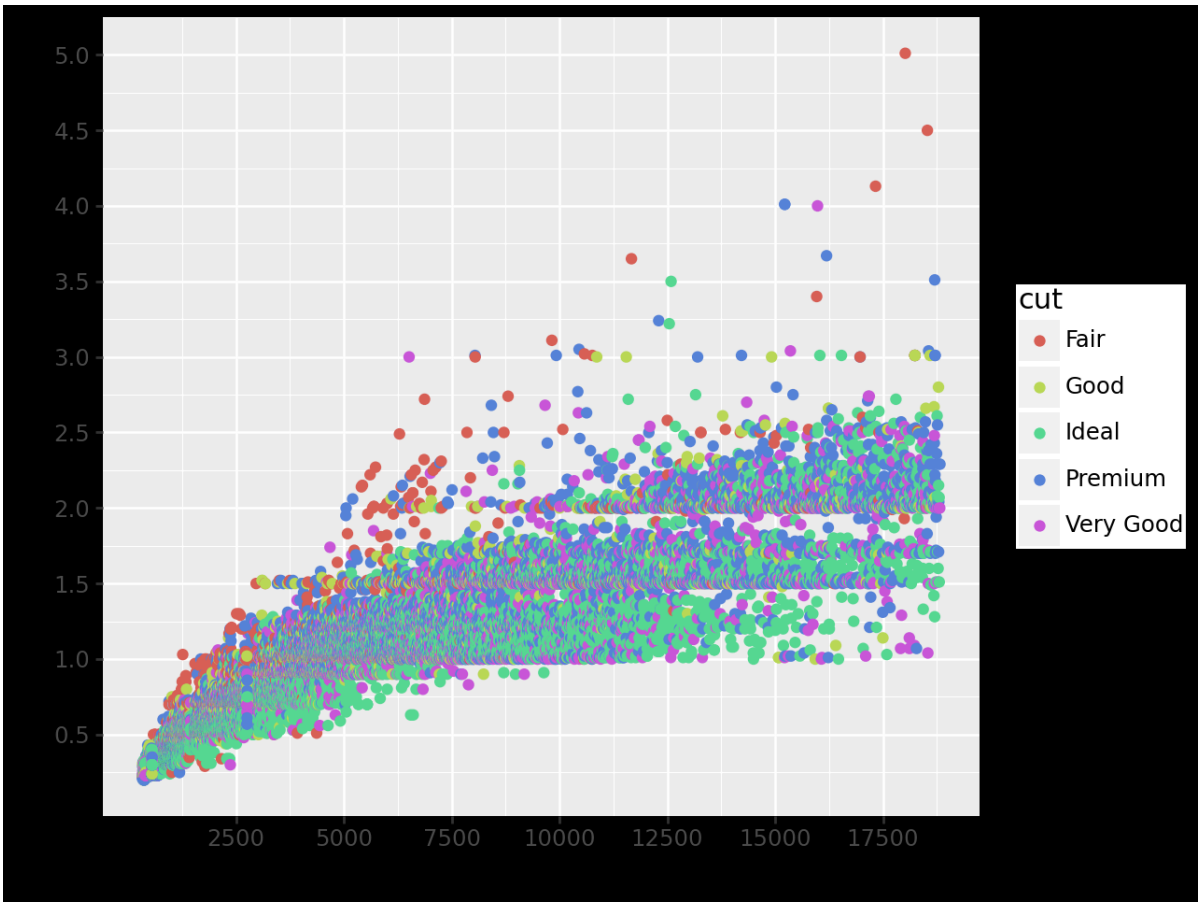
Out[ ]:

	carat	cut	price
0	0.23	Ideal	326
1	0.21	Premium	326
2	0.23	Good	327
3	0.29	Premium	334
4	0.31	Good	335
...	...	...	...
53935	0.72	Ideal	2757
53936	0.72	Good	2757
53937	0.70	Very Good	2757
53938	0.86	Premium	2757
53939	0.75	Ideal	2757

53940 rows × 3 columns

In [ ]:
```
breaks1 = np.arange(2500, 17501, 2500)
breaks2 = np.arange(0.5, 5.1, 0.5)

diamonds_plot = ggplot(algebra)
diamonds_plot += geom_point (aes(x="price",y="carat",color="cut"))
diamonds_plot += scale_x_continuous (breaks = breaks1)
diamonds_plot += scale_y_continuous (breaks = breaks2)
diamonds_plot
```

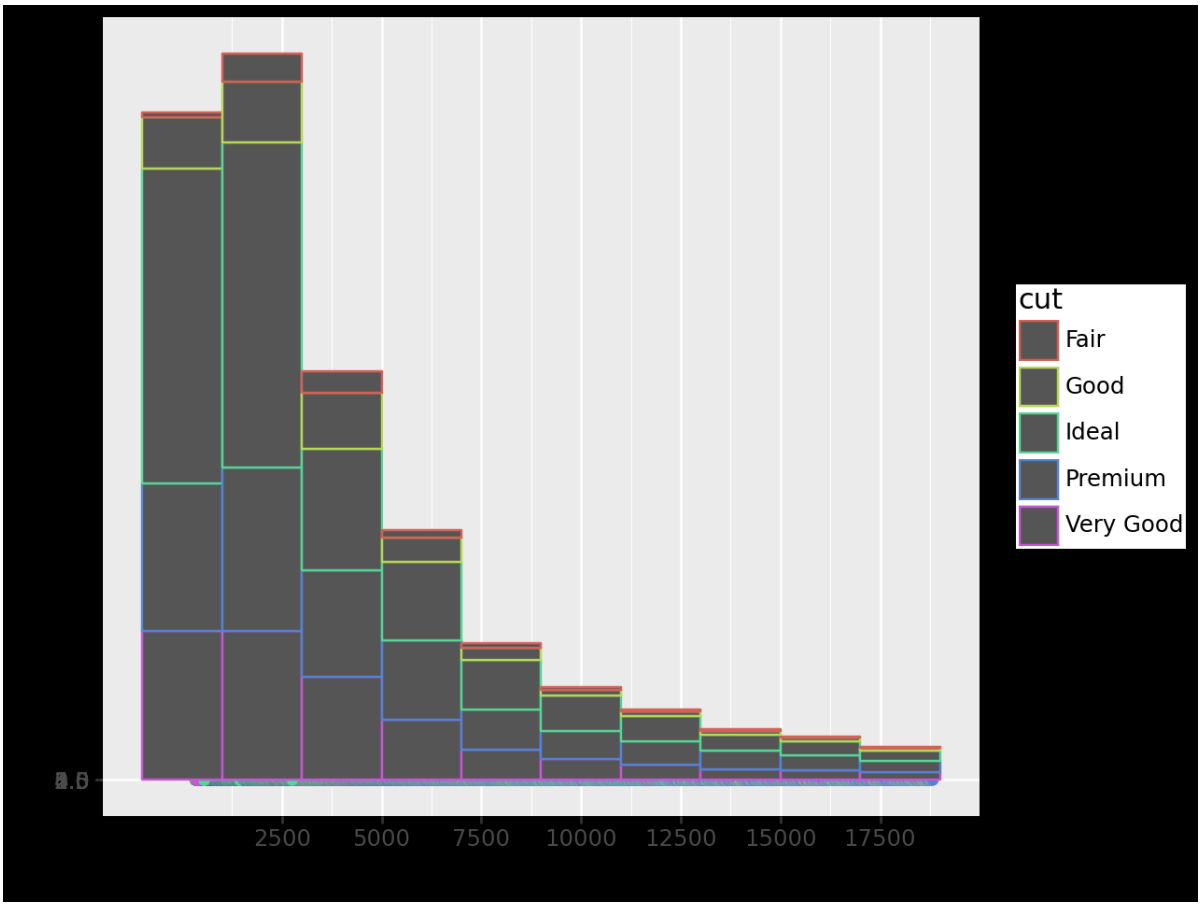


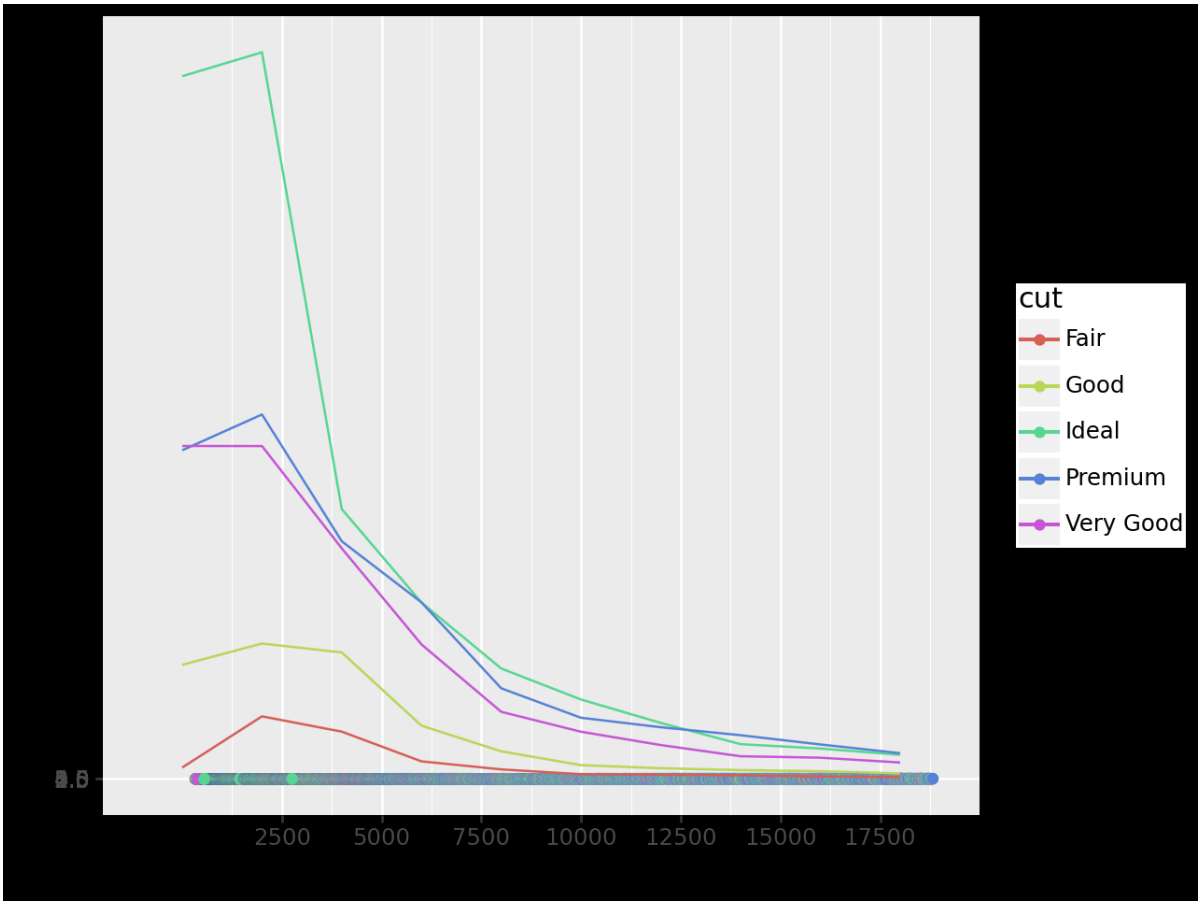
In [ ]:
```
print("u_carat =", algebra["carat"].mean())
print("sd_carat =", algebra["carat"].std())
print("Md_carat =", algebra["carat"].median())
```

```
print("u_price = U$", algebra["price"].mean())
print("sd_price = U$", algebra["price"].std())
print("Md_price = u$", algebra["price"].median())
```

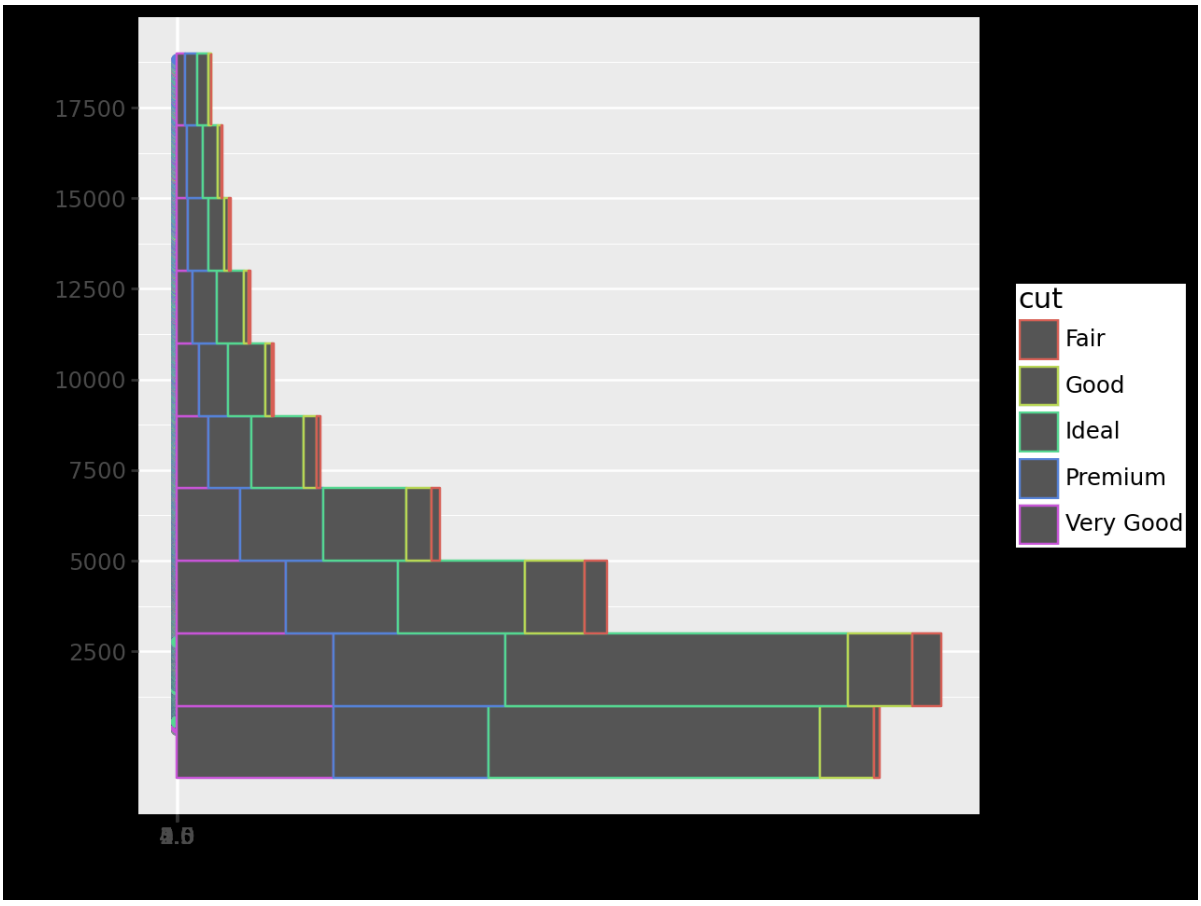
```
u_carat = 0.7979397478680014
sd_carat = 0.47401124440538067
Md_carat = 0.7
u_price = U$ 3932.799721913237
sd_price = U$ 3989.4397381463023
Md_price = u$ 2401.0
```

In [ ]:
```
diamonds_plot_histogram = diamonds_plot+ geom_histogram(aes(x="price", colo
diamonds_plot_freqpoly = diamonds_plot + geom_freqpoly(aes(x="price", color=
diamonds_plot_histogram.show()
diamonds_plot_freqpoly.show()
```

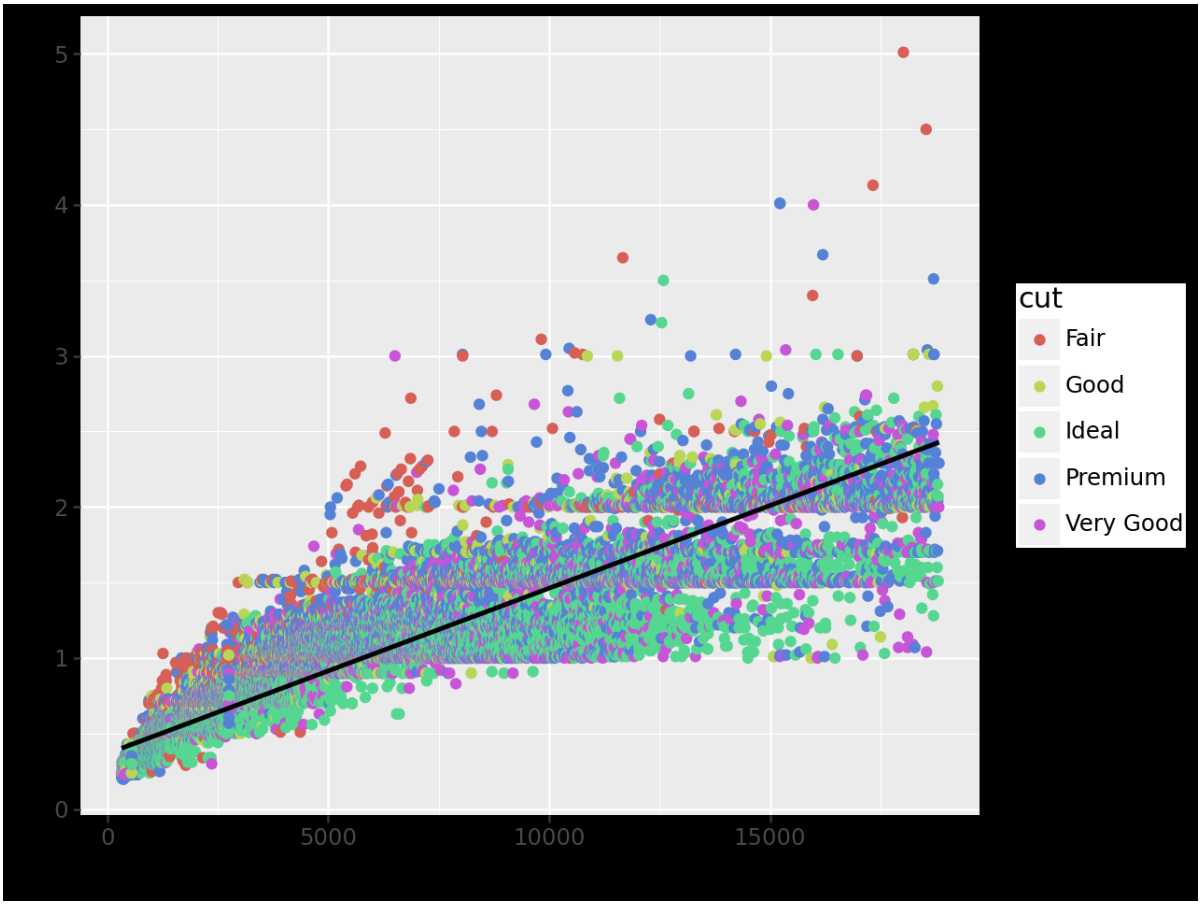




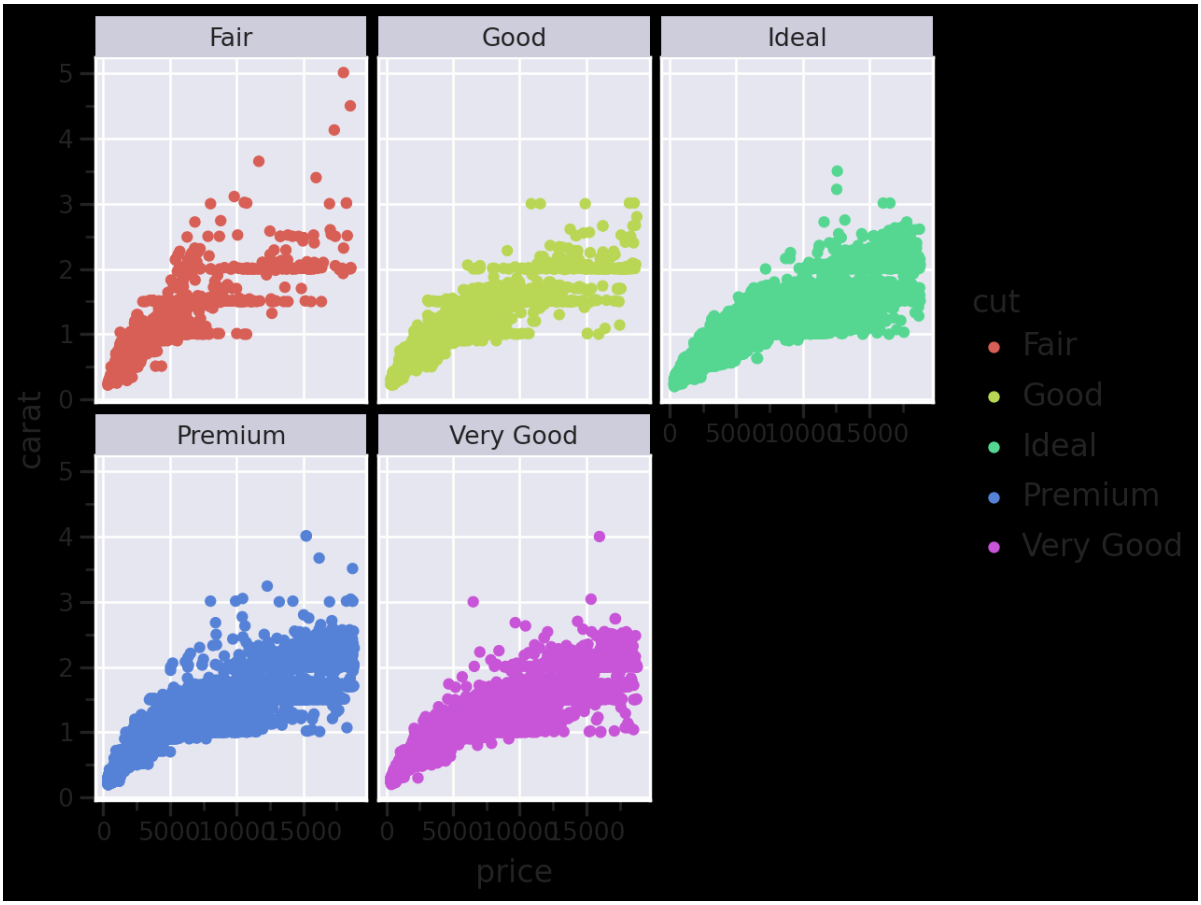
In [ ]:
```
(diamonds_plot_histogram+coord_flip()).show()
# (diamonds_plot_histogram+coord_polar()).show()
```



In [ ]:
```
ggplot(algebra, aes(x="price",y="carat")) + \
geom_point(mapping = aes(color="cut")) + \
geom_smooth(method = "lm", se = False)
```



In [ ]:
```
plot2 =ggplot(algebra, aes(x="price",y="carat", color="cut")) + \
geom_point () + \
facet_wrap ("~cut", nrow=2) + theme_seaborn()
plot2
```



2.3

caso 1 da Seção 10.1 na referência [34]

Para poder utilizar o dataset gapminder é necessário instalção pelo pip como um pacote python.

In [ ]:
```
# !pip install gapminder
from gapminder import gapminder
```

- Verificar as características carregadas:

In [ ]:
```
gapminder.head()
```

Out[ ]:

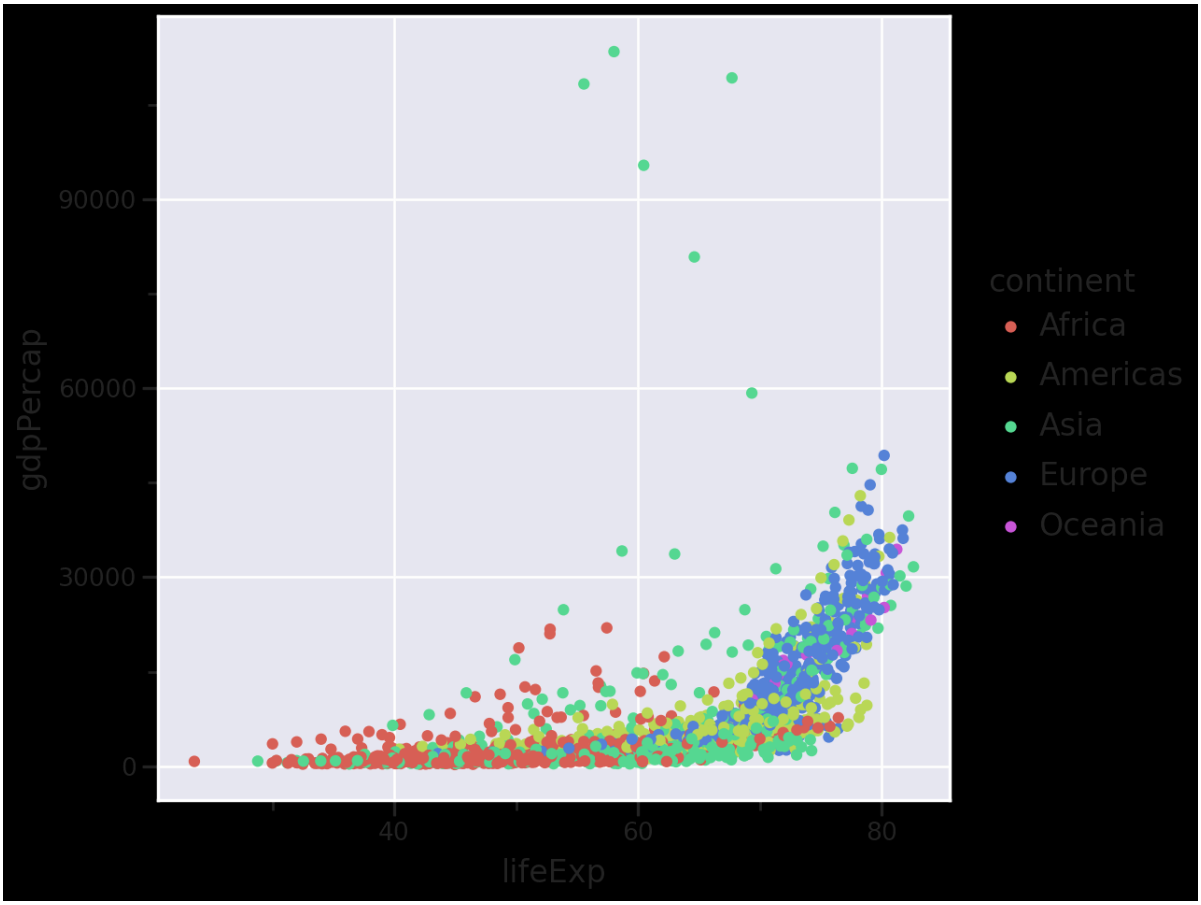
	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
4	Afghanistan	Asia	1972	36.088	13079460	739.981106

In [ ]:
```
gapminder_data = gapminder
gapminder_data = gapminder_data[(gapminder_data["year"] == 2007) & ((gapminc
gapminder_data[["country","lifeExp"]]
```

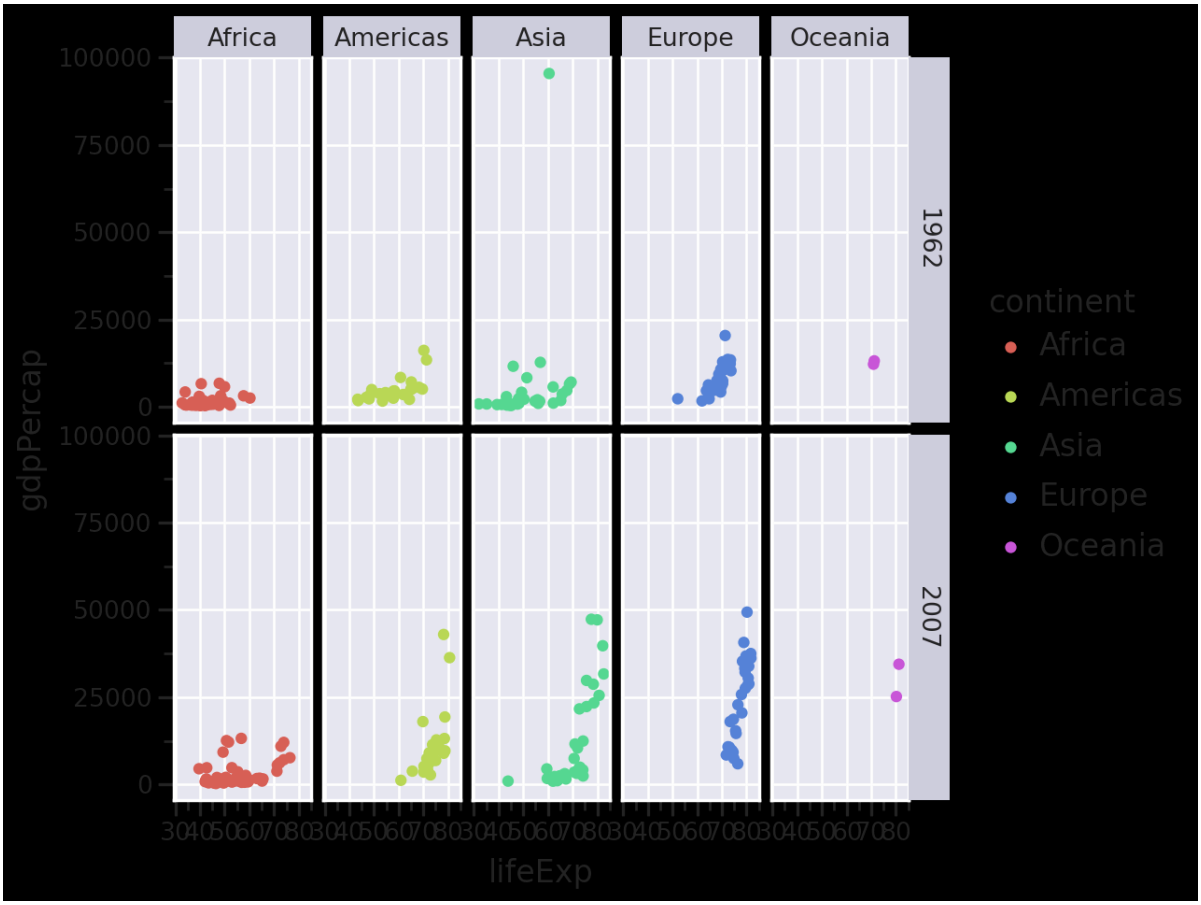
Out[ ]:

	country	lifeExp
1439	Sri Lanka	72.396
1583	Turkey	71.777

In [ ]:
```
gapminder_data = gapminder
# ggplot(algebra, aes(x="price",y="carat"))
gapminder_plot = ggplot(gapminder_data, aes(x="lifeExp", y="gdpPercap", colo
gapminder_plot += geom_point()
gapminder_plot
```

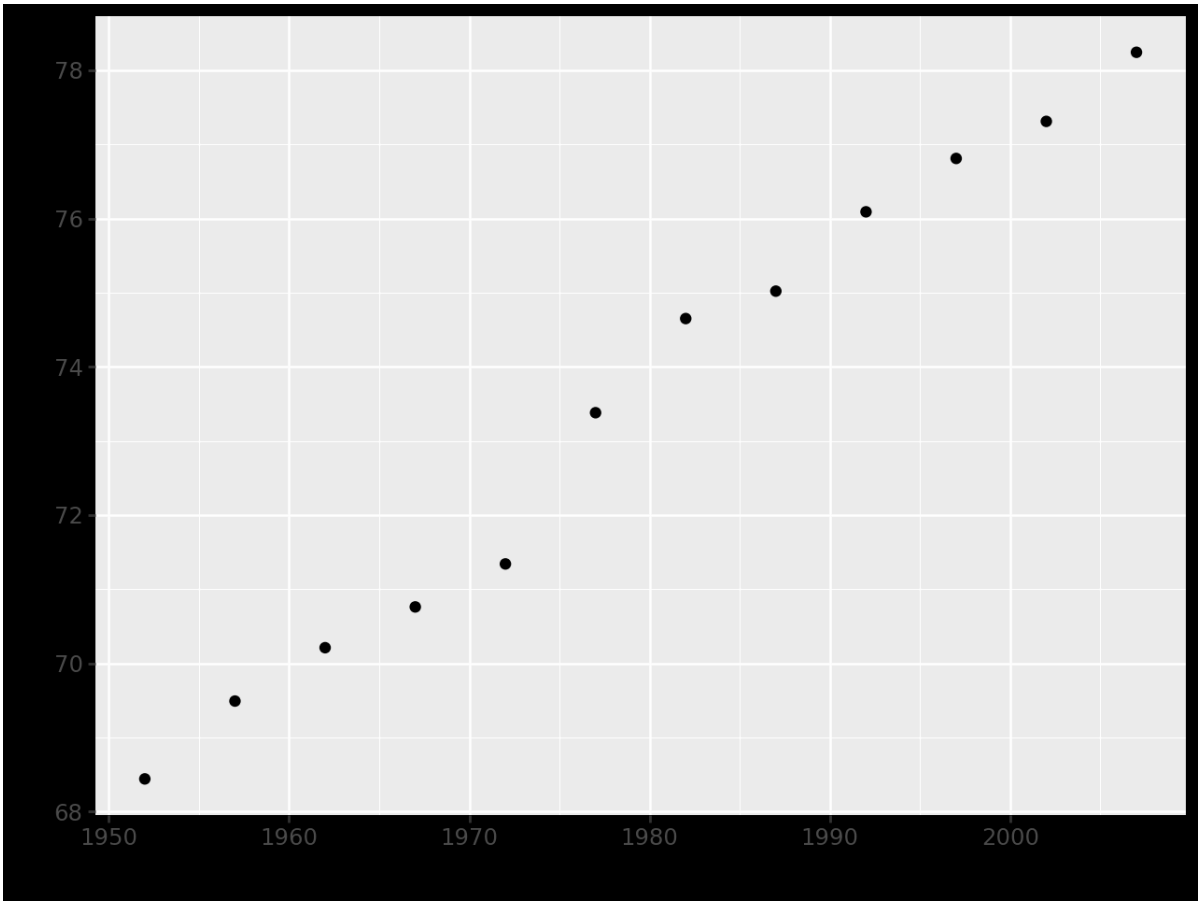


In [ ]:
```
gapminder_data = gapminder_data[(gapminder_data["year"] == 1962) | (gapminde
gapminder_plot = ggplot(gapminder_data, aes(x="lifeExp", y="gdpPercap", colo
gapminder_plot += geom_point()
gapminder_plot += facet_grid("year~continent")
gapminder_plot
```

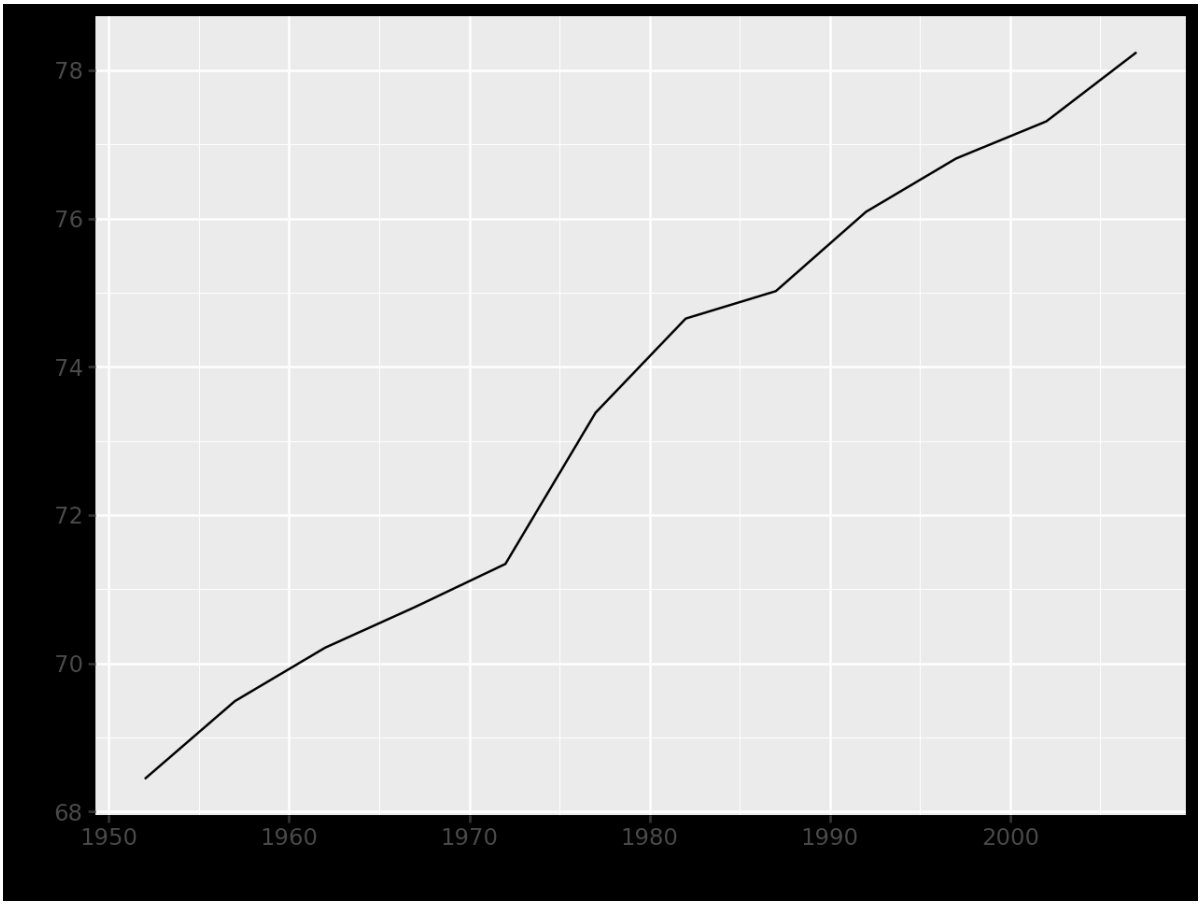


In [ ]:
```
gapminder_plot += facet_wrap("~year", scales="free")
gapminder_plot
```

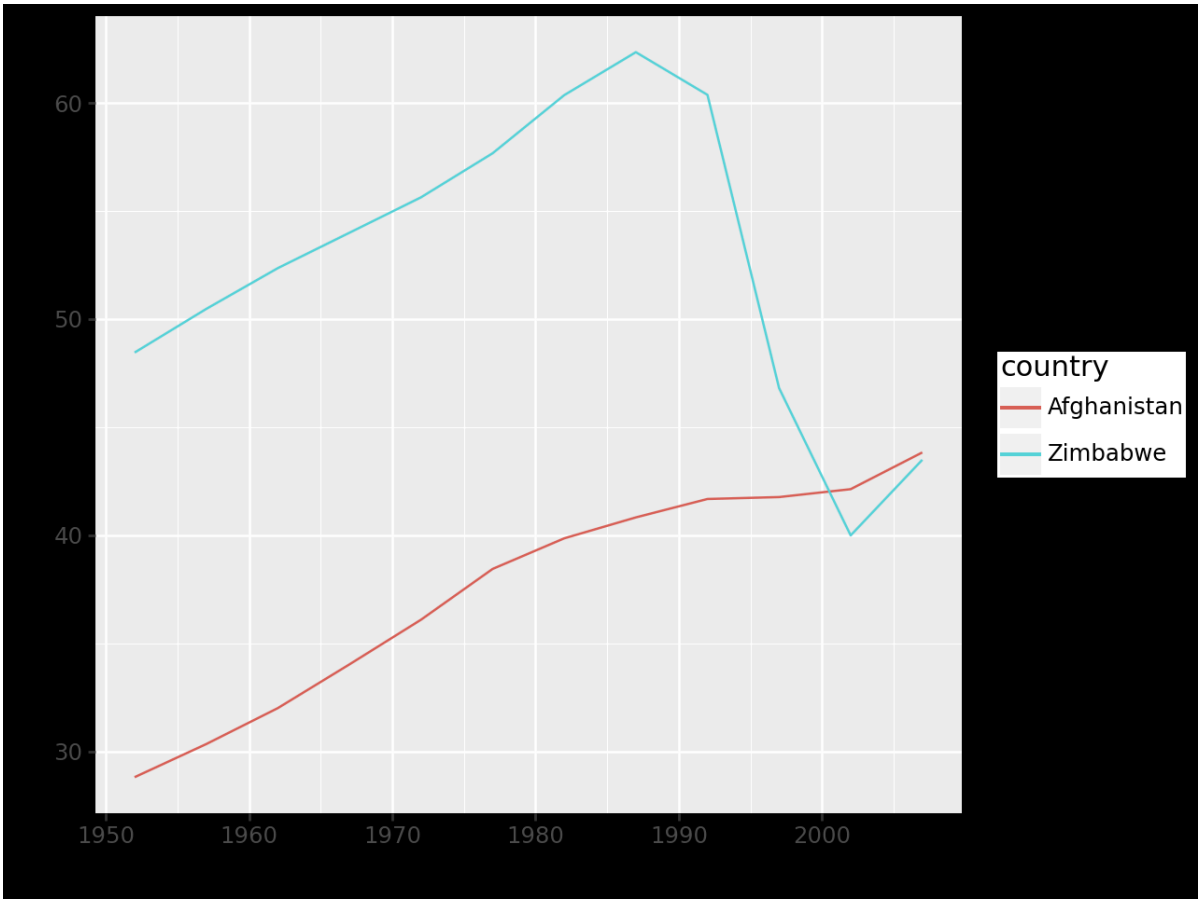
In [ ]:
```
gapminder_data = gapminder
gapminder_data = gapminder_data[gapminder_data["country"] == "United States"
gapminder_plot = ggplot(gapminder_data,aes(x="year",y="lifeExp"))
gapminder_plot += geom_point()
gapminder_plot
```



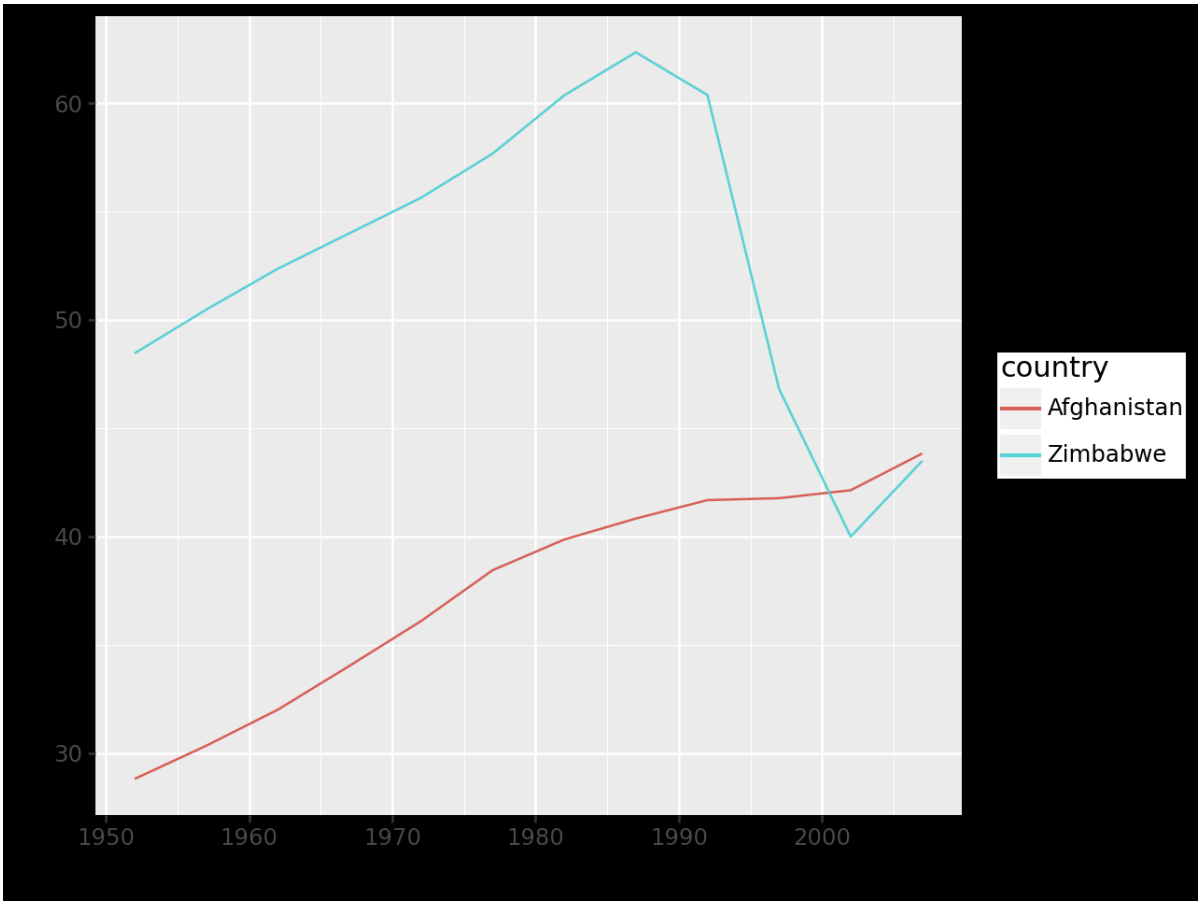
In [ ]:
```
gapminder_data = gapminder
gapminder_data = gapminder_data[gapminder_data["country"] == "United States"
gapminder_plot = ggplot(gapminder_data,aes(x="year",y="lifeExp"))
gapminder_plot += geom_line()
gapminder_plot
```



In [ ]:
```
gapminder_data = gapminder
gapminder_data = gapminder_data[(gapminder_data["country"] == "Zimbabwe") |
gapminder_plot = ggplot(gapminder_data,aes(x="year",y="lifeExp", color="cour
gapminder_plot += geom_line()
gapminder_plot
```



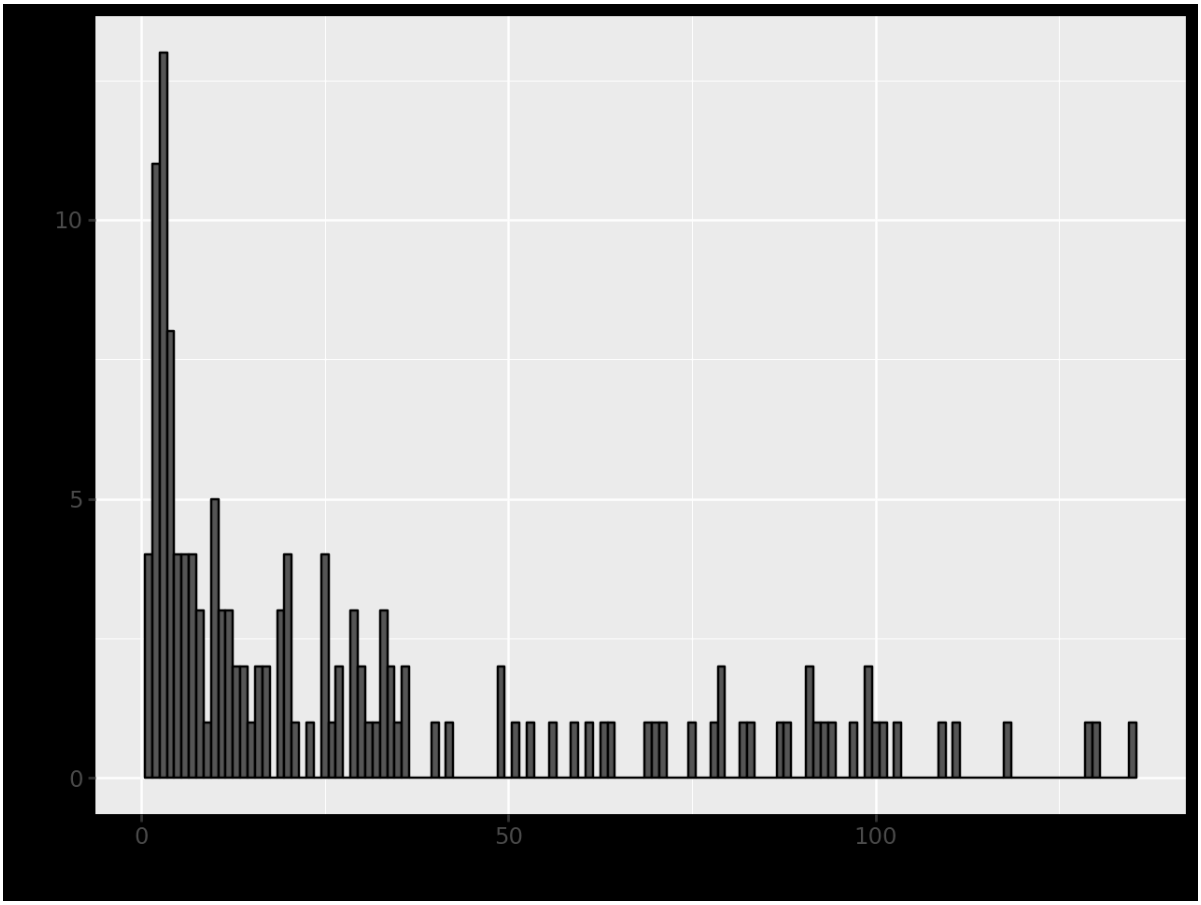
In [ ]:
```
gapminder_data = gapminder
gapminder_data = gapminder_data[(gapminder_data["country"] == "Zimbabwe") |
gapminder_plot = ggplot(gapminder_data,aes(x="year",y="lifeExp", color="cour
gapminder_plot += geom_line()
# gapminder_plot += geom_textpath()
# gapminder_plot += theme(legend.position = "none")
plot3= gapminder_plot
gapminder_plot
```



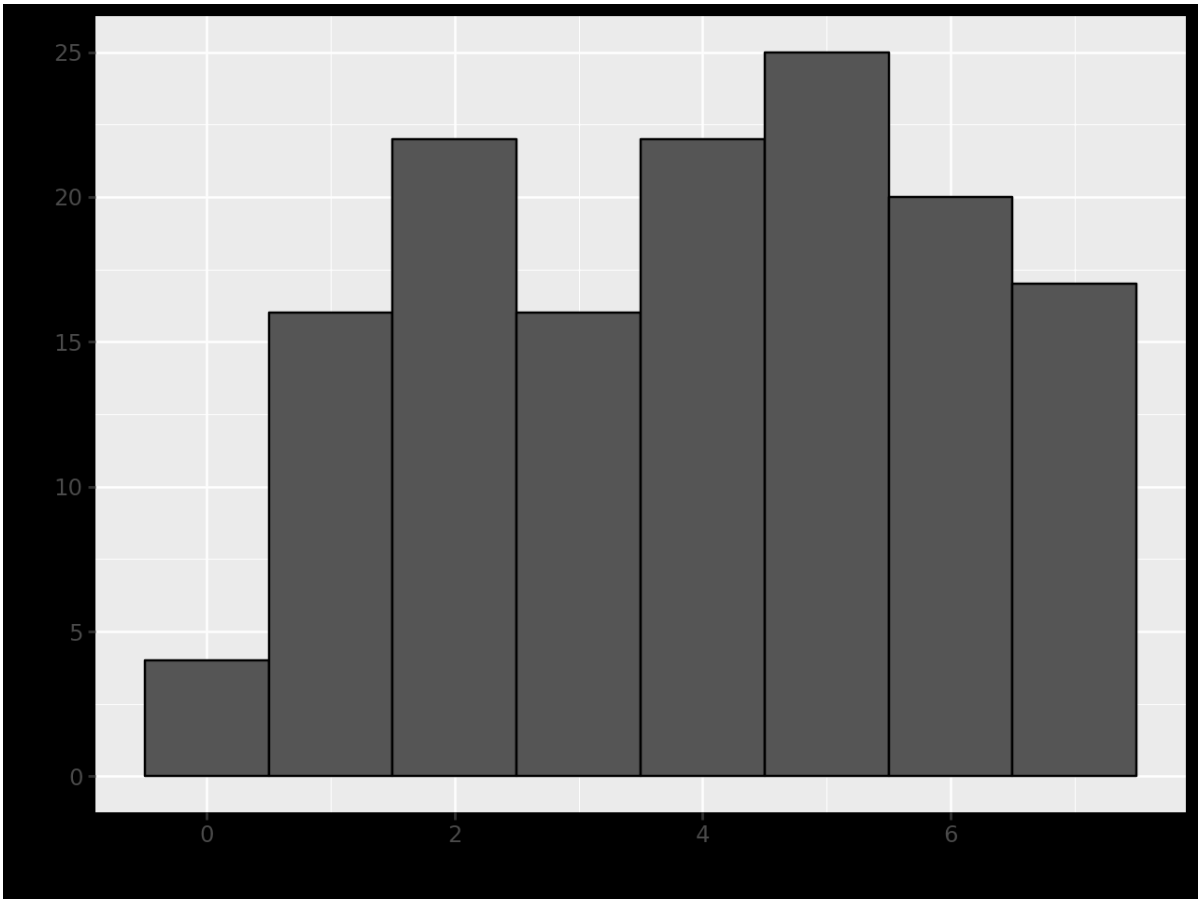
2.4

caso 2 descrito na seção 10.5 na referência [34].

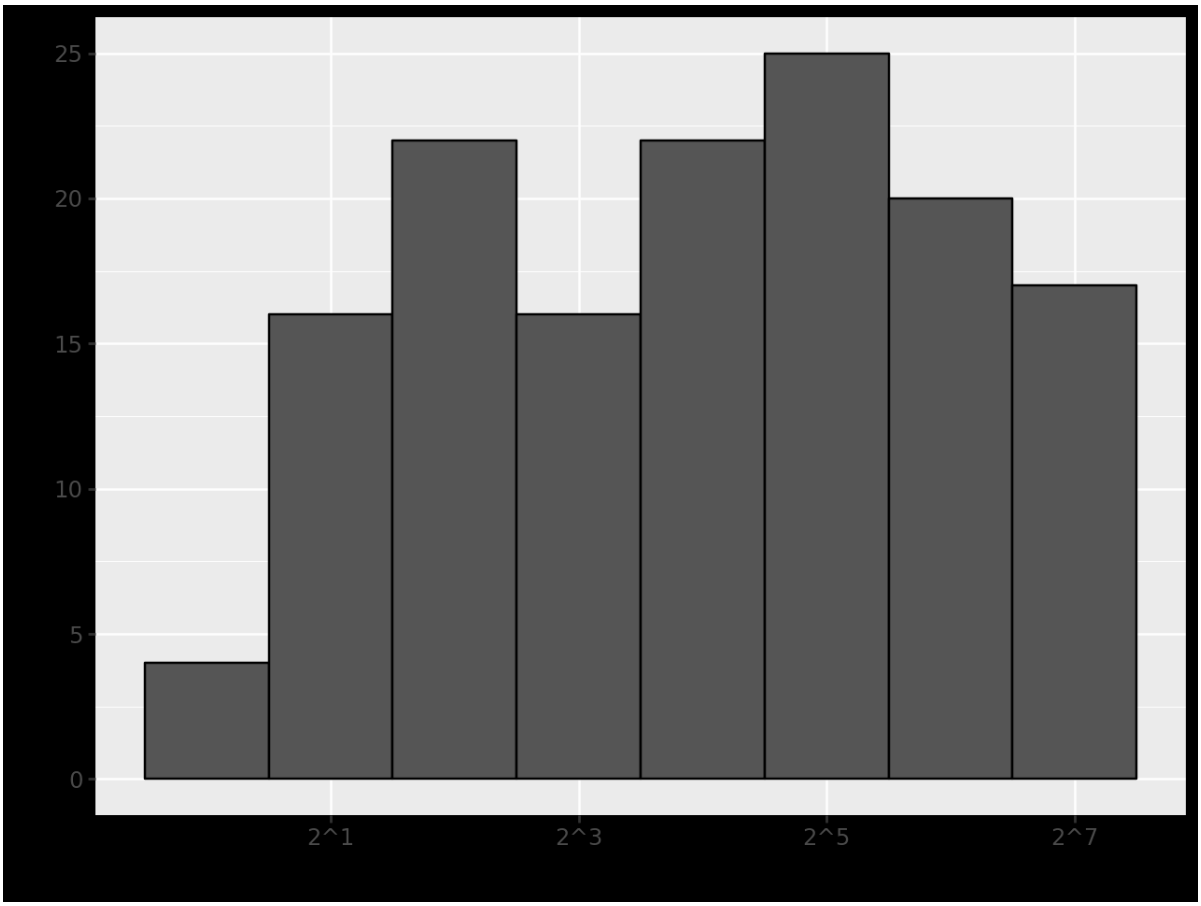
In [ ]:
```
gapminder_data = gapminder
gapminder_data["dollars_per_day"] = gapminder_data["gdpPercap"]/365
gapminder_data = gapminder_data[gapminder_data["year"] == 2007]
gapminder_plot = ggplot(gapminder_data, aes(gapminder_data["dollars_per_day"
gapminder_plot += geom_histogram(binwidth=1,color = "black")
gapminder_plot
```



In [ ]:
```
gapminder_data = gapminder
gapminder_data["dollars_per_day"] = gapminder_data["gdpPercap"]/365
gapminder_data = gapminder_data[gapminder_data["year"] == 2007]
gapminder_plot = ggplot(gapminder_data, aes("np.log2(dollars_per_day)"))
gapminder_plot += geom_histogram(binwidth=1,color = "black")
gapminder_plot
```



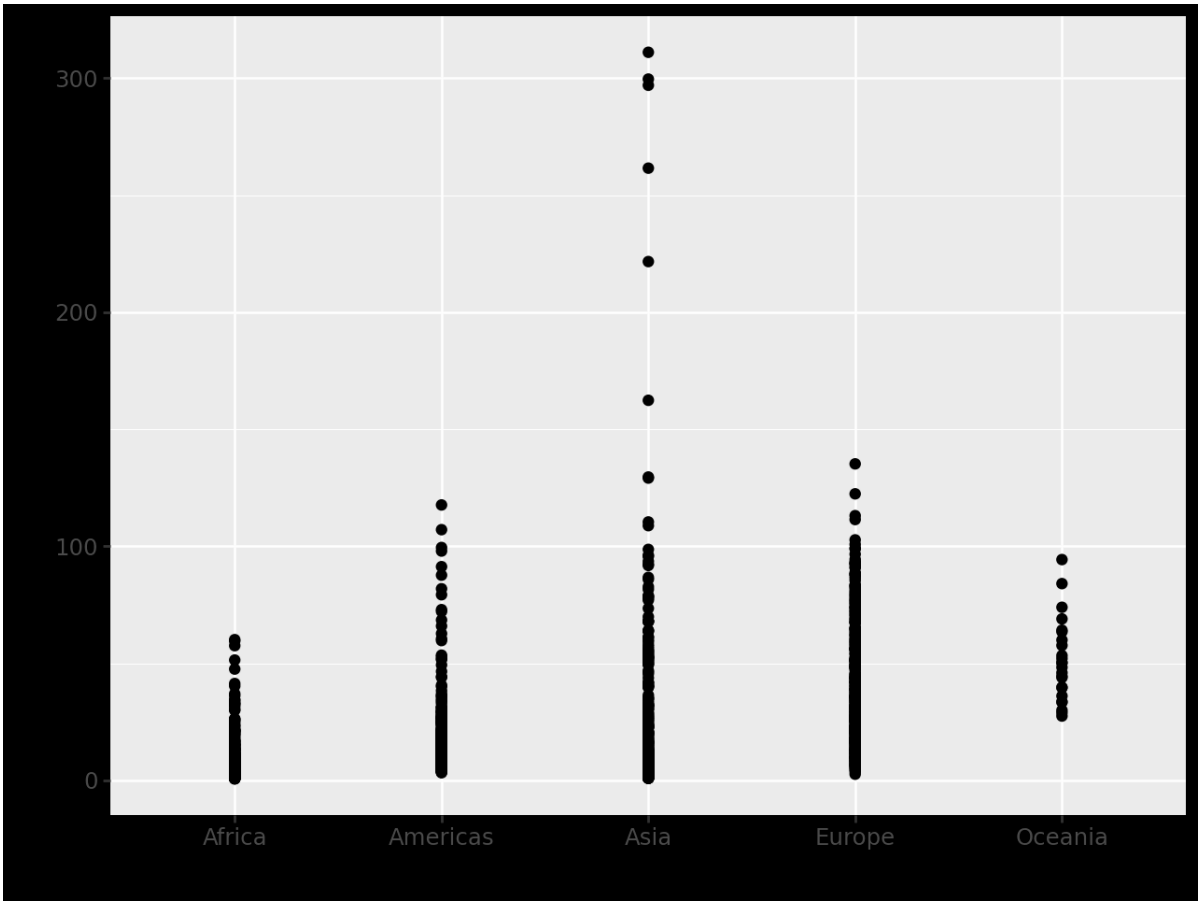
In [ ]:
```
gapminder_data = gapminder
gapminder_data["dollars_per_day"] = gapminder_data["gdpPercap"]/365
gapminder_data = gapminder_data[gapminder_data["year"] == 2007]
gapminder_plot = ggplot(gapminder_data, aes(gapminder_data["dollars_per_day"
gapminder_plot += geom_histogram(binwidth=1,color = "black")
gapminder_plot += scale_x_continuous(trans = "log2")
gapminder_plot
```



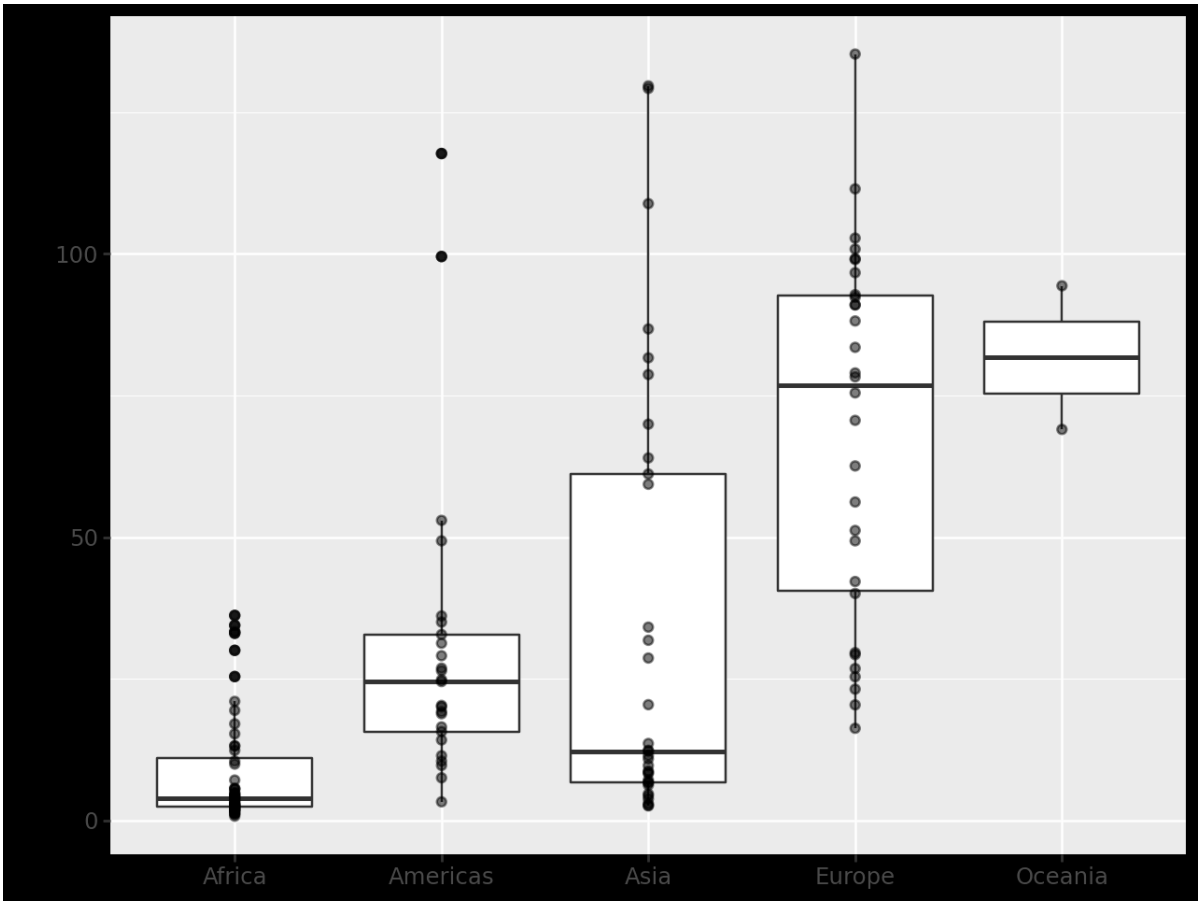
In [ ]:
```
gapminder_data = gapminder
grouped = gapminder_data.groupby("continent")
medians = grouped["dollars_per_day"].transform(np.median)
sorted_order = medians.sort_values(ascending=False).index
gapminder_data = gapminder_data.loc[sorted_order]

gapminder_plot = ggplot(gapminder_data, aes(y = "dollars_per_day", x="contin
gapminder_plot += geom_point()
# gapminder_plot += scale_x_continuous(trans = "log2")
gapminder_plot
```

```
/tmp/ipykernel_58064/899437614.py:3: FutureWarning: The provided callable <
function median at 0x721c80ef1630> is currently using SeriesGroupBy.median.
In a future version of pandas, the provided callable will be used directly.
To keep current behavior pass the string "median" instead.
```

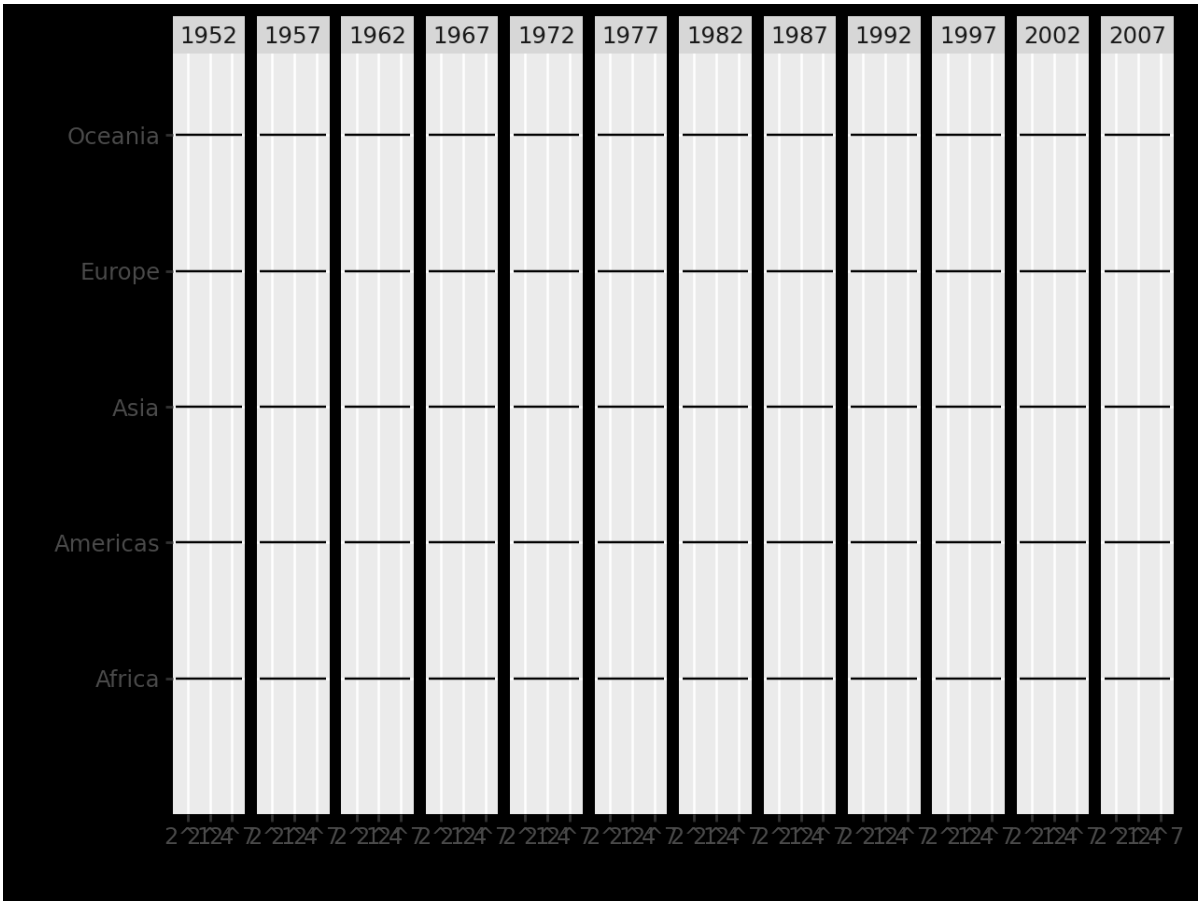


In [ ]:
```
gapminder_data = gapminder
gapminder_data = gapminder_data[gapminder_data["year"] == 2007]
gapminder_plot = ggplot(gapminder_data, aes("continent","dollars_per_day"))
gapminder_plot += geom_boxplot()
# gapminder_plot += scale_x_continuous(trans = "log2")
gapminder_plot += xlab("")
# gapminder_plot += theme(axis.text.x = element_text(angle = 90, hjust = 1),
gapminder_plot += geom_point(alpha = 0.5)
plot4= gapminder_plot
gapminder_plot
```



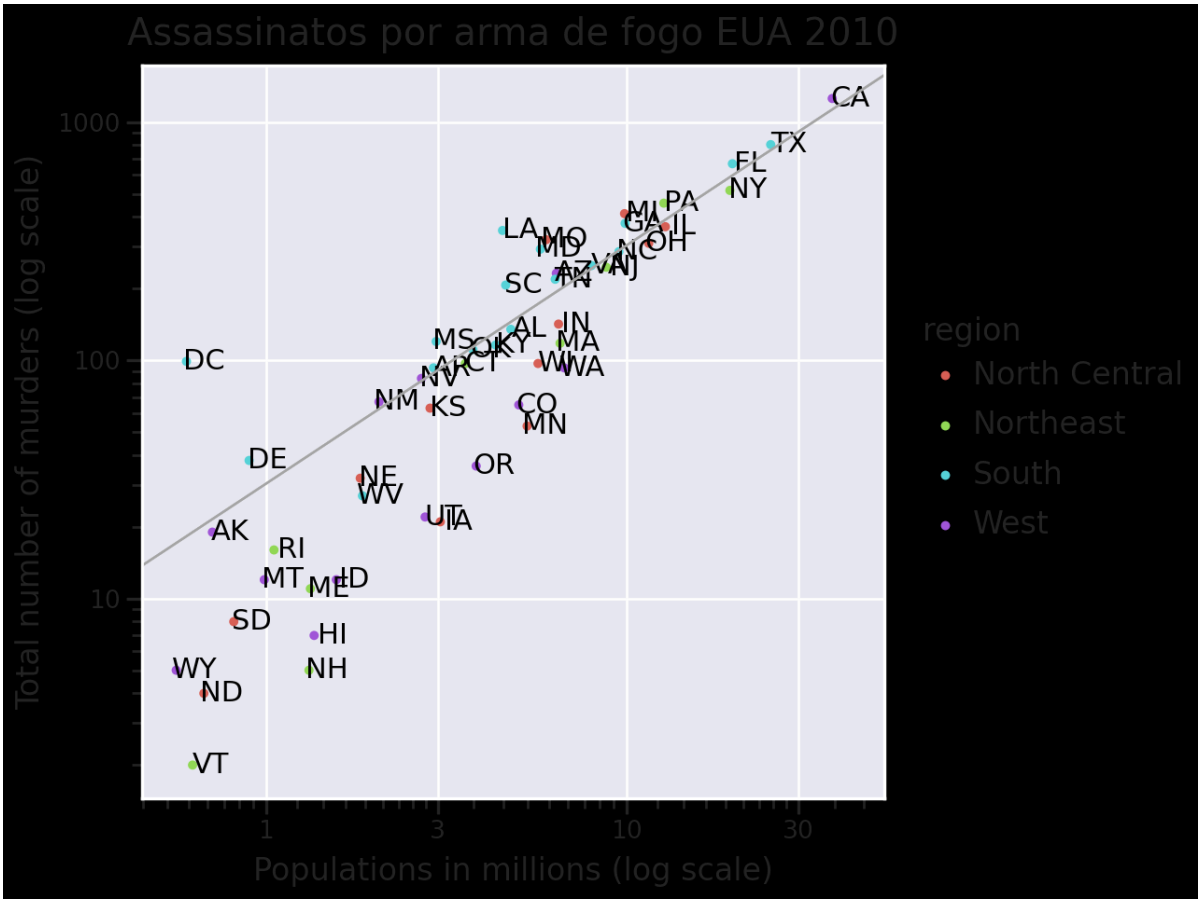
In [ ]:
```
gapminder_data = gapminder
# gapminder_data = gapminder_data[gapminder_data["year"] == 2007]
gapminder_plot = ggplot(gapminder_data, aes("dollars_per_day", "continent"))
gapminder_plot += scale_x_continuous(trans = "log2")
gapminder_plot += geom_density()
gapminder_plot += facet_grid("~year")

gapminder_plot
```

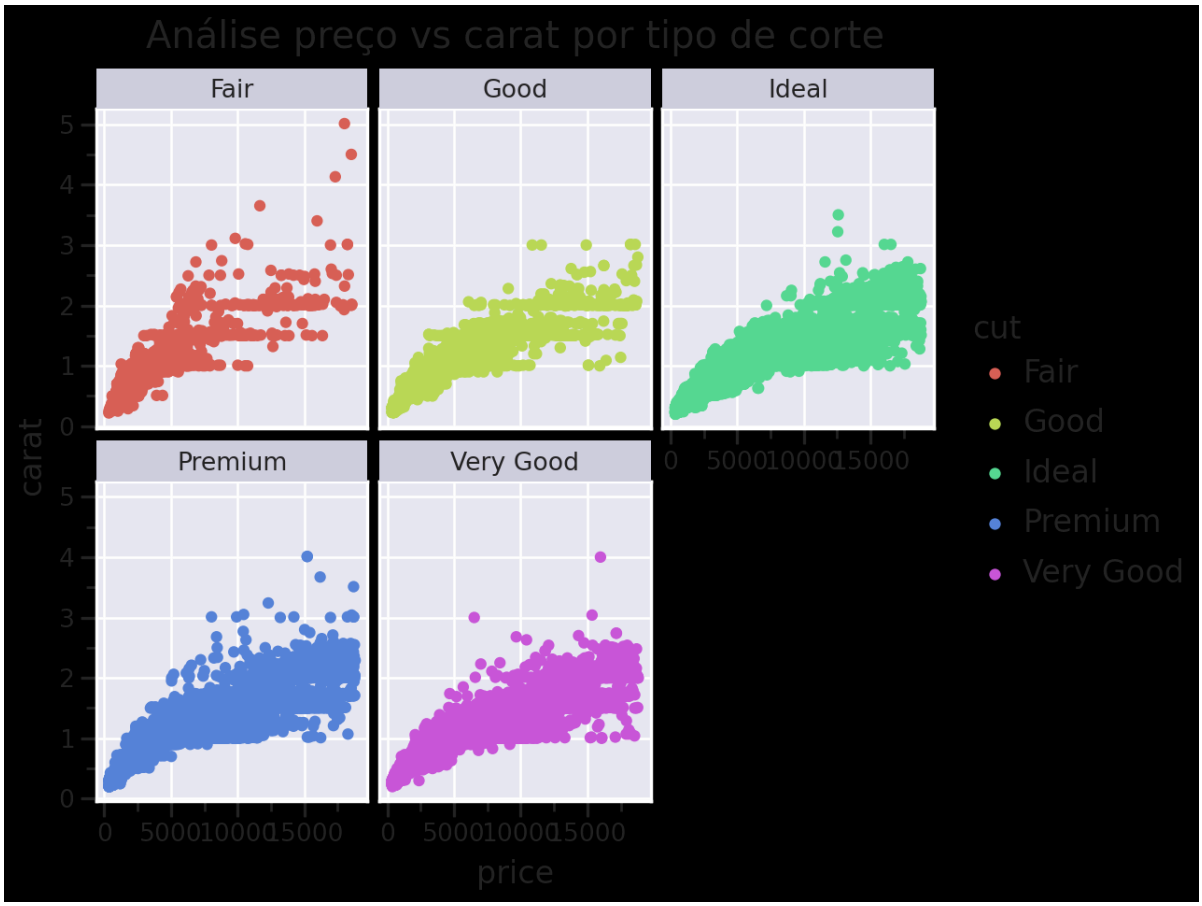


Exercício 3

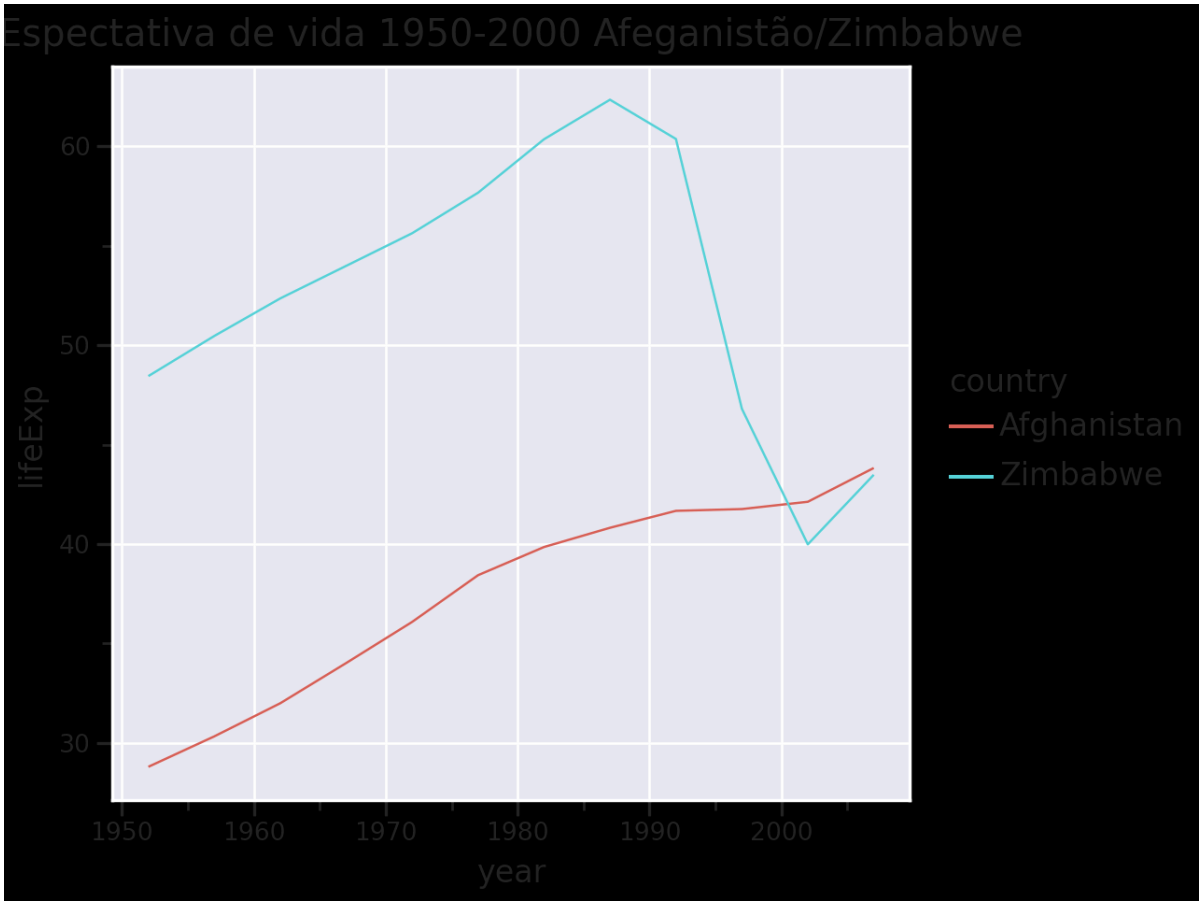
In [ ]:
```
plot1+= ggtitle("Assassinatos por arma de fogo EUA 2010")
plot1
```



In [ ]:
```
plot2+= ggtitle("Análise preço vs carat por tipo de corte")
plot2
```



In [ ]:
```
plot3+= ggtitle("Espectativa de vida 1950-2000 Afeganistão/Zimbabwe")
plot3 += theme_seaborn()
plot3
```



In [ ]:
```
plot4+= ggtitle("Dolar ganahdo por dia nos continentes")
plot4 += theme_seaborn()
plot4
```