Ps2

November 3, 2020

[53]:
```
import pandas as pd
import matplotlib.pyplot as plt
import math
import numpy as np
from tabulate import tabulate
import seaborn as sns
```

1 Założenie: próbki oznaczają wagę mężczyzn

1.1 Wygenerowano po 1000 próbek dla każdego z rozkładu

1.2 Przy rozkładzie normalnym określono następujące parametry:

1.2.1 μ: 90 (średnia)

1.2.2 : 10 (odchylenie standardowe)

1.3 Dla rozkładu jednostajnego określono następujące parametry:

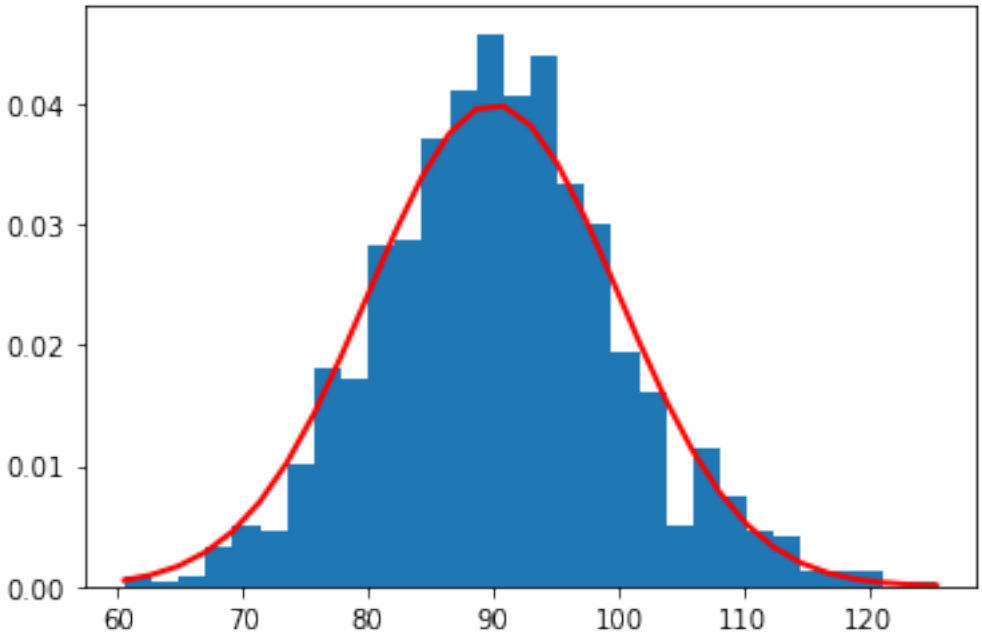
1.3.1 minimum: 60

1.3.2 maksimum: 130

[54]:
```
mu, sigma = 90, 10
s = np.random.normal(mu, sigma, 1000)
```

1.4 Wygenerowana próbka z rozkładu normalnego:

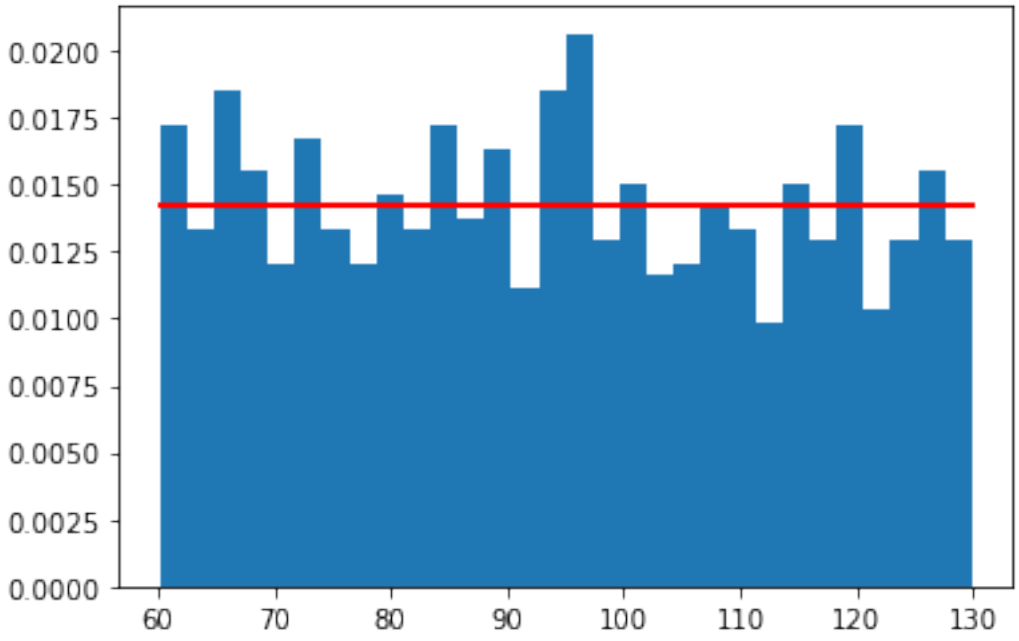
[55]:
```
count, bins, ignored = plt.hist(s, 30, density=True)
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
               np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
         linewidth=2, color='r')
plt.show()
```



[56]:
```
s2 = np.random.uniform(60,130,1000)
```

1.5 Wygenerowana próbka z rozkładu jednostajnego:

[57]:
```
x = np.arange(31, dtype=float)
count, bins, ignored = plt.hist(s2, 30, density=True)
plt.plot(bins, np.full_like(x, 1/(130-60)), linewidth=2, color='r')
plt.show()
```



[58]:
```
df = pd.DataFrame(s, columns = ["Rozkład normalny"])
df2 = pd.DataFrame(s2, columns= ["Rozkład jednostajny"])
```

1.6 Statystyki opisowe wygenerowanych próbek:

[59]:
```
df_both = pd.concat([df,df2], axis=1)
df_working_copy = df_both.copy()
df_both.describe()
```

[59]:

	Rozkład normalny	Rozkład jednostajny
count	1000.000000	1000.000000
mean	90.517546	94.007779
std	9.640677	20.132000
min	60.519264	60.151003
25%	84.275147	76.444585
50%	90.183994	93.852006
75%	96.241508	110.731056
max	125.412710	129.897715

[60]:
```
sample_to_drop = df_working_copy.sample(frac=0.1)
df90 = df_working_copy.drop(sample_to_drop.index)
```

[61]:
```
def print_statistics(data_, label):
    i = 100
    for d in data_:
        i = i-10
```

```
        print()
        print('Tabela ' + str(i) + label)
        print(tabulate(d.describe(), headers = ['Nazwa', 'Rozkład normalny',␣
↪'Rozkład jednostajny'], tablefmt = 'fancy_grid'))
```

1.7 Statystyki opisowe próbek po usunięciu 10 - 80% rekordów:

[62]:
```
df80 = df90.drop(df90.sample(100).index)
df70 = df80.drop(df80.sample(100).index)
df60 = df70.drop(df70.sample(100).index)
df50 = df60.drop(df60.sample(100).index)
df40 = df50.drop(df50.sample(100).index)
df30 = df40.drop(df40.sample(100).index)
df20 = df30.drop(df30.sample(100).index)
df_dropped = [df90, df80, df70, df60, df50, df40, df30, df20]
print_statistics(df_dropped, "%")
```

```
Tabela 90%

╒═══════╤═══════════════════╤═══════════════════════╕
│ Nazwa │ Rozkład normalny  │ Rozkład jednostajny   │
╞═══════╪═══════════════════╪═══════════════════════╡
│ count │ 900               │ 900                   │
├───────┼───────────────────┼───────────────────────┤
│ mean  │ 90.6096           │ 94.3302               │
├───────┼───────────────────┼───────────────────────┤
│ std   │ 9.73085           │ 20.061                │
├───────┼───────────────────┼───────────────────────┤
│ min   │ 60.5193           │ 60.151                │
├───────┼───────────────────┼───────────────────────┤
│ 25%   │ 84.2894           │ 76.7611               │
├───────┼───────────────────┼───────────────────────┤
│ 50%   │ 90.4489           │ 94.3554               │
├───────┼───────────────────┼───────────────────────┤
│ 75%   │ 96.4081           │ 111.523               │
├───────┼───────────────────┼───────────────────────┤
│ max   │ 125.413           │ 129.898               │
╘═══════╧═══════════════════╧═══════════════════════╛

Tabela 80%

╒═══════╤═══════════════════╤═══════════════════════╕
│ Nazwa │ Rozkład normalny  │ Rozkład jednostajny   │
╞═══════╪═══════════════════╪═══════════════════════╡
│ count │ 800               │ 800                   │
├───────┼───────────────────┼───────────────────────┤
│ mean  │ 90.7955           │ 94.4233               │
├───────┼───────────────────┼───────────────────────┤
│ std   │ 9.74853           │ 20.1902               │
```

min	60.5193	60.151
25%	84.4716	76.6586
50%	90.5511	94.4796
75%	96.5397	112.074
max	125.413	129.898

Tabela 70%

Nazwa	Rozkład normalny	Rozkład jednostajny
count	700	700
mean	90.7584	94.3424
std	9.82311	20.3718
min	60.5193	60.151
25%	84.2817	76.4446
50%	90.6049	94.1678
75%	96.5953	113.567
max	125.413	129.898

Tabela 60%

Nazwa	Rozkład normalny	Rozkład jednostajny
count	600	600
mean	90.7072	94.0353
std	9.84407	20.2966
min	60.5193	60.151
25%	84.2817	76.5701
50%	90.4631	93.7959

75%	96.4656	112.408
max	125.413	129.898

Tabela 50%

Nazwa	Rozkład normalny	Rozkład jednostajny
count	500	500
mean	90.5267	94.2719
std	9.69796	20.3429
min	60.5193	60.151
25%	84.2764	77.0106
50%	90.1324	93.3846
75%	96.2903	113.031
max	125.413	129.898

Tabela 40%

Nazwa	Rozkład normalny	Rozkład jednostajny
count	400	400
mean	90.4849	94.4709
std	9.61411	20.6616
min	62.3961	60.151
25%	84.2817	76.7662
50%	90.1324	93.9196
75%	96.1228	113.703
max	125.413	129.208

Tabela 30%

Nazwa	Rozkład normalny	Rozkład jednostajny
count	300	300
mean	90.2622	94.1199
std	9.43267	20.2322
min	62.3961	60.151
25%	84.3921	77.7568
50%	90.4631	93.2399
75%	96.0113	112.164
max	125.413	129.208

Tabela 20%

Nazwa	Rozkład normalny	Rozkład jednostajny
count	200	200
mean	90.1493	92.736
std	9.13639	20.4798
min	62.3961	60.151
25%	84.2764	75.2329
50%	90.4752	89.9639
75%	95.8012	111.614
max	125.413	128.802

[63]:
```
def fill_df_with_mean(data_):
    return data_.append(pd.DataFrame({
                                        'Rozkład normalny': np.
↪full(1000-len(data_),data_['Rozkład normalny'].mean()),
```

```
                                        'Rozkład jednostajny': np.
↪full(1000-len(data_),data_['Rozkład jednostajny'].mean())
                                    })
                        )
```

1.8 Statystyki opisowe próbek po uzupełnieniu brakujących rekordów średnią wyliczoną z pozostałych danych:

[64]:
```
df90_mean = fill_df_with_mean(df90)
df80_mean = fill_df_with_mean(df80)
df70_mean = fill_df_with_mean(df70)
df60_mean = fill_df_with_mean(df60)
df50_mean = fill_df_with_mean(df50)
df40_mean = fill_df_with_mean(df40)
df30_mean = fill_df_with_mean(df30)
df20_mean = fill_df_with_mean(df20)
df_mean = [df90_mean, df80_mean, df70_mean, df60_mean, df50_mean, df40_mean,␣
 ↪df30_mean, df20_mean]
print_statistics(df_mean, "% + uzupełnienie średnią")
```

```
Tabela 90% + uzupełnienie średnią

 Nazwa       Rozkład normalny     Rozkład jednostajny

 count              1000                      1000

 mean               90.6096                   94.3302

 std                 9.23099                  19.0305

 min                60.5193                   60.151

 25%                85.2751                   79.496

 50%                90.6096                   94.3302

 75%                95.6735                  109.515

 max               125.413                    129.898


Tabela 80% + uzupełnienie średnią

 Nazwa       Rozkład normalny     Rozkład jednostajny

 count              1000                      1000
```

mean	90.7955	94.4233
std	8.71826	18.0564
min	60.5193	60.151
25%	86.2099	81.8611
50%	90.7955	94.4233
75%	94.907	107.247
max	125.413	129.898

Tabela 70% + uzupełnienie średnią

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.7584	94.3424
std	8.21684	17.0406
min	60.5193	60.151
25%	87.0143	84.217
50%	90.7584	94.3424
75%	94.0713	103.273
max	125.413	129.898

Tabela 60% + uzupełnienie średnią

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.7072	94.0353
std	7.62264	15.7165
min	60.5193	60.151

25%	88.7388	87.8699
50%	90.7072	94.0353
75%	92.6479	97.7185
max	125.413	129.898

Tabela 50% + uzupełnienie średnią

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.5267	94.2719
std	6.85406	14.3774
min	60.5193	60.151
25%	90.1378	93.4242
50%	90.5267	94.2719
75%	90.5267	94.2719
max	125.413	129.898

Tabela 40% + uzupełnienie średnią

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.4849	94.4709
std	6.07593	13.0577
min	62.3961	60.151
25%	90.4849	94.4709
50%	90.4849	94.4709
75%	90.4849	94.4709

max	125.413	129.208

Tabela 30% + uzupełnienie średnią

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.2622	94.1199
std	5.16045	11.0687
min	62.3961	60.151
25%	90.2622	94.1199
50%	90.2622	94.1199
75%	90.2622	94.1199
max	125.413	129.208

Tabela 20% + uzupełnienie średnią

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.1493	92.736
std	4.07773	9.14049
min	62.3961	60.151
25%	90.1493	92.736
50%	90.1493	92.736
75%	90.1493	92.736
max	125.413	128.802

[65]:
```
def fill_df_with_median(data_):
    return data_.append(pd.DataFrame({
                                        'Rozkład normalny': np.
 ↪full(1000-len(data_),data_['Rozkład normalny'].median()),
                                        'Rozkład jednostajny': np.
 ↪full(1000-len(data_),data_['Rozkład jednostajny'].median())
                                      })
                        )
```

1.9 Statystyki opisowe próbek po uzupełnieniu brakujących rekordów medianą wyliczoną z pozostałych danych:

[66]:
```
df90_median = fill_df_with_median(df90)
df80_median = fill_df_with_median(df80)
df70_median = fill_df_with_median(df70)
df60_median = fill_df_with_median(df60)
df50_median = fill_df_with_median(df50)
df40_median = fill_df_with_median(df40)
df30_median = fill_df_with_median(df30)
df20_median = fill_df_with_median(df20)
df_median = [df90_median, df80_median, df70_median, df60_median, df50_median,␣
 ↪df40_median, df30_median, df20_median]
print_statistics(df_median, "% + uzupełnienie medianą")
```

```
Tabela 90% + uzupełnienie medianą

 Nazwa       Rozkład normalny     Rozkład jednostajny

 count              1000                    1000

 mean               90.5935                 94.3327

 std                 9.23111                19.0305

 min                60.5193                 60.151

 25%                85.2751                 79.496

 50%                90.4489                 94.3554

 75%                95.6735                109.515

 max               125.413                  129.898


Tabela 80% + uzupełnienie medianą
```

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.7466	94.4345
std	8.7188	18.0564
min	60.5193	60.151
25%	86.2099	81.8611
50%	90.5511	94.4796
75%	94.907	107.247
max	125.413	129.898

Tabela 70% + uzupełnienie medianą

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.7124	94.29
std	8.21714	17.0408
min	60.5193	60.151
25%	87.0143	84.217
50%	90.6049	94.1678
75%	94.0713	103.273
max	125.413	129.898

Tabela 60% + uzupełnienie medianą

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.6096	93.9396

std	7.62358	15.7169
min	60.5193	60.151
25%	88.7388	87.8699
50%	90.4631	93.7959
75%	92.6479	97.7185
max	125.413	129.898

Tabela 50% + uzupełnienie medianą

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.3296	93.8283
std	6.85689	14.3842
min	60.5193	60.151
25%	90.1298	93.3649
50%	90.1324	93.3846
75%	90.1351	93.4044
max	125.413	129.898

Tabela 40% + uzupełnienie medianą

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.2734	94.1401
std	6.07839	13.0605
min	62.3961	60.151
25%	90.1324	93.9196

50%	90.1324	93.9196
75%	90.1324	93.9196
max	125.413	129.208

Tabela 30% + uzupełnienie medianą

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.4028	93.5039
std	5.16127	11.0761
min	62.3961	60.151
25%	90.4631	93.2399
50%	90.4631	93.2399
75%	90.4631	93.2399
max	125.413	129.208

Tabela 20% + uzupełnienie medianą

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.41	90.5184
std	4.07981	9.20757
min	62.3961	60.151
25%	90.4752	89.9639
50%	90.4752	89.9639
75%	90.4752	89.9639
max	125.413	128.802

[67]:
```
def fill_df_with_random_numbers(data_):
    return data_.append(pd.DataFrame({
                                        'Rozkład normalny': np.random.
↪normal(data_['Rozkład normalny'].mean(), data_['Rozkład normalny'].std(),␣
↪1000-len(data_)),
                                        'Rozkład jednostajny': np.random.
↪uniform(data_['Rozkład jednostajny'].min(),data_['Rozkład jednostajny'].
↪max(),1000-len(data_))
                                       })
                        )
```

1.10 Statystyki opisowe próbek po uzupełnieniu brakujących rekordów losowymi wartościami w zależności od rozkładu:

1.10.1 - dla rozkładu normalnego wykorzystano średnią i odchylenie standardowe z pozostałych danych

1.10.2 - dla rozkładu jednostajnego wykorzystano minimum i maksimum z pozostałych wartości

[68]:
```
df90_random = fill_df_with_random_numbers(df90)
df80_random = fill_df_with_random_numbers(df80)
df70_random = fill_df_with_random_numbers(df70)
df60_random = fill_df_with_random_numbers(df60)
df50_random = fill_df_with_random_numbers(df50)
df40_random = fill_df_with_random_numbers(df40)
df30_random = fill_df_with_random_numbers(df30)
df20_random = fill_df_with_random_numbers(df20)
df_random = [df90_random, df80_random, df70_random, df60_random, df50_random,␣
 ↪df40_random, df30_random, df20_random]
print_statistics(df_random, "% + uzupełnienie losowymi wartościami")
```

Tabela 90% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.5947	94.2611
std	9.63551	19.9211
min	60.5193	60.151
25%	84.4036	76.9682

50%	90.2067	93.9196
75%	96.288	111.486
max	125.413	129.898

Tabela 80% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.845	94.4663
std	9.85393	20.0891
min	51.3768	60.151
25%	84.4036	76.7611
50%	90.6702	94.3242
75%	96.9038	111.819
max	127.847	129.898

Tabela 70% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.737	94.4413
std	9.64445	20.1627
min	60.4427	60.151
25%	84.2817	77.1376
50%	90.7872	94.2547
75%	96.9313	112.324
max	125.413	129.898

Tabela 60% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.8999	93.7277
std	9.87884	19.9196
min	60.5193	60.151
25%	84.2764	76.95
50%	90.8531	93.9196
75%	96.9126	110.499
max	125.413	129.898

Tabela 50% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.9137	94.869
std	9.7827	20.399
min	60.5193	60.151
25%	84.2764	78.3305
50%	90.6018	93.9196
75%	97.2032	113.16
max	128.979	129.898

Tabela 40% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000

mean	90.0425	94.3912
std	9.29198	19.7403
min	57.5652	60.151
25%	83.6639	78.3742
50%	90.0547	93.4746
75%	96.127	111.663
max	125.413	129.208

Tabela 30% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.1438	93.8035
std	9.47528	19.6188
min	62.3961	60.151
25%	83.3591	78.0086
50%	90.424	93.5938
75%	96.3192	109.63
max	127.624	129.208

Tabela 20% + uzupełnienie losowymi wartościami

Nazwa	Rozkład normalny	Rozkład jednostajny
count	1000	1000
mean	90.1717	93.7929
std	9.12936	19.9163
min	60.7273	60.151

25%	83.8166	76.5908
50%	89.8313	93.266
75%	95.9705	110.536
max	125.413	128.802

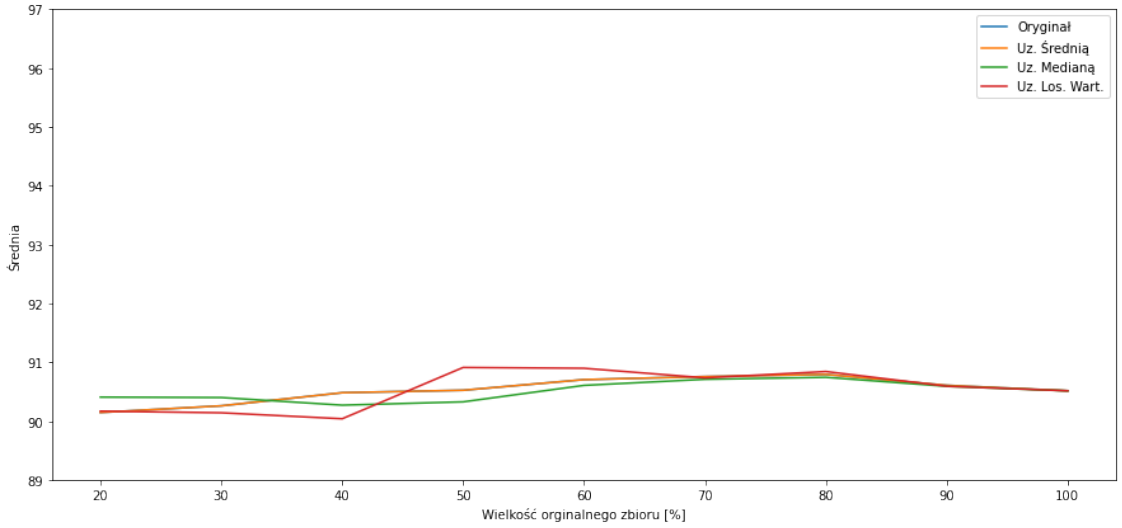
[69]:
```
def get_mean_std_normal(data_):
    data_mean = [d['Rozkład normalny'].mean() for d in data_]
    data_mean.reverse()
    data_mean.append(df_both['Rozkład normalny'].mean())
    data_std = [d['Rozkład normalny'].std() for d in data_]
    data_std.reverse()
    data_std.append(df_both['Rozkład normalny'].std())
    return data_mean, data_std
```

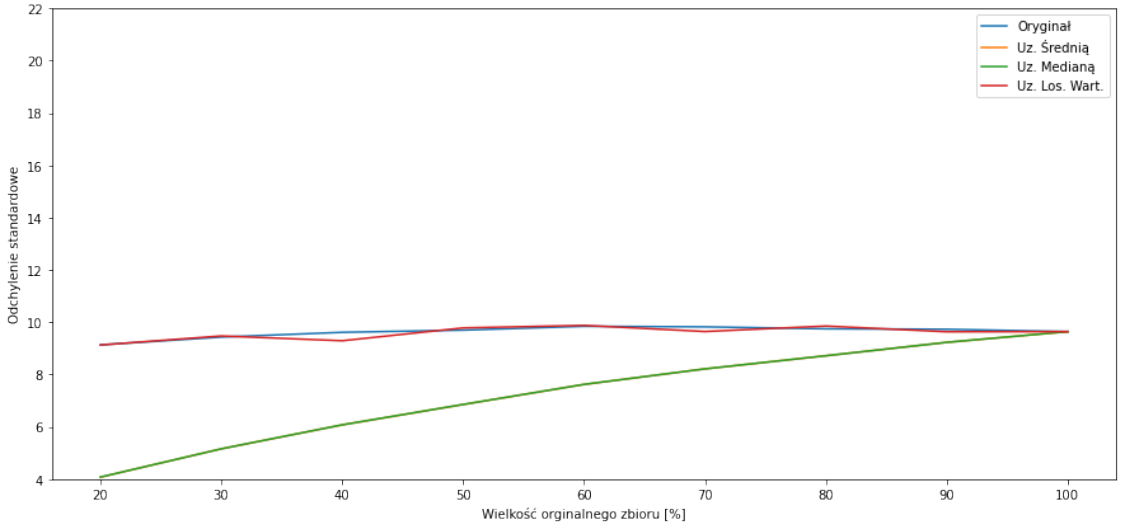
[70]:
```
def get_mean_std_uniform(data_):
    data_mean = [d['Rozkład jednostajny'].mean() for d in data_]
    data_mean.reverse()
    data_mean.append(df_both['Rozkład jednostajny'].mean())
    data_std = [d['Rozkład jednostajny'].std() for d in data_]
    data_std.reverse()
    data_std.append(df_both['Rozkład jednostajny'].std())
    return data_mean, data_std
```

[71]:
```
x = ['20', '30', '40','50','60','70','80','90','100']
df_dropped_mean, df_dropped_std = get_mean_std_normal(df_dropped)
df_mean_mean, df_mean_std = get_mean_std_normal(df_mean)
df_median_mean, df_median_std = get_mean_std_normal(df_median)
df_random_mean, df_random_std = get_mean_std_normal(df_random)

f = plt.figure(figsize=(15,15))
ax = f.add_subplot(211)
ax2 = f.add_subplot(212)

ax.plot(x, df_dropped_mean, label = "Oryginał")
ax.plot(x, df_mean_mean, label = "Uz. Średnią")
ax.plot(x, df_median_mean, label = "Uz. Medianą")
ax.plot(x, df_random_mean, label = "Uz. Los. Wart.")
ax.set_xlabel("Wielkość orginalnego zbioru [%]")
ax.set_ylabel("Średnia")
ax.set_ylim([89,97])
ax.legend()
```

```
ax2.plot(x, df_dropped_std, label = "Oryginał")
ax2.plot(x, df_mean_std, label = "Uz. Średnią")
ax2.plot(x, df_median_std, label = "Uz. Medianą")
ax2.plot(x, df_random_std, label = "Uz. Los. Wart.")
ax2.set_xlabel("Wielkość orginalnego zbioru [%]")
ax2.set_ylabel("Odchylenie standardowe")
ax2.set_ylim([4,22])
ax2.legend()
```

[71]: <matplotlib.legend.Legend at 0x2208ed30f10>





1.11 **Im większe braki w zbiorze danych, tym bardziej statystyki różnią się w porównaniu do danych oryginalnych, gdy danych brakuje więcej niż około 30% różnice na wykresie średniej stają się dużo bardziej zauważalne.**

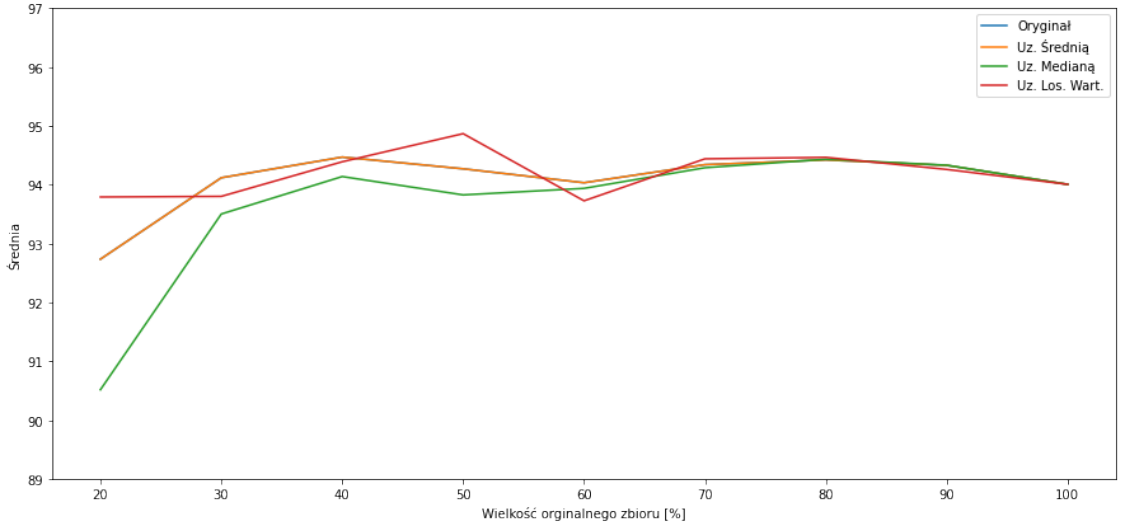
1.12 **Na wykresie opisującym zachowanie oddchylenia standardowego w zależności od procentowej ilości braku danych najsłabiej zachowuje się uzupełnienie danych medianą oraz średnią.**

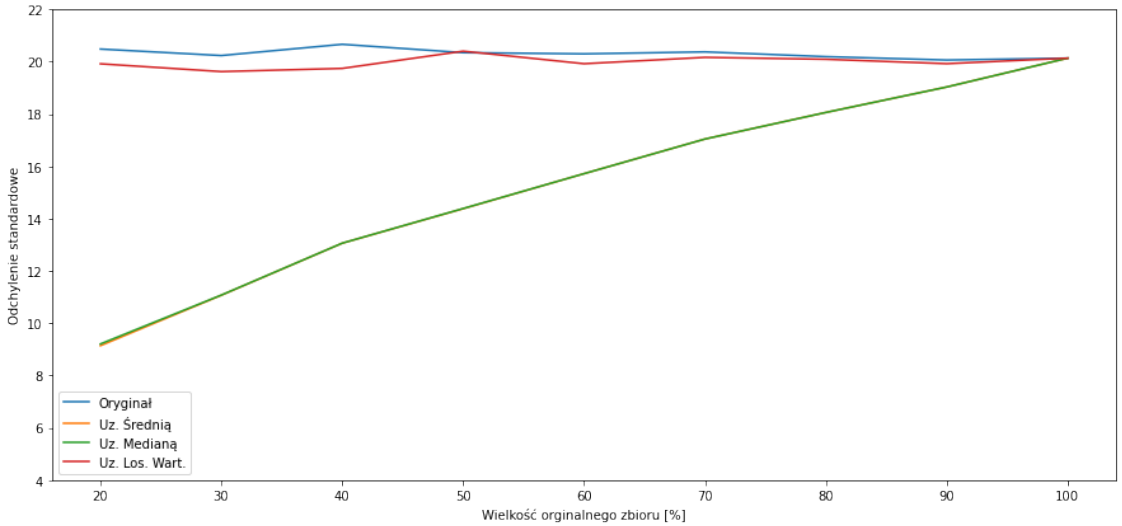
1.13 **Im większe braki, tym odchylenie standardowe odbiega od oryginalnej wartości (jest mniejsze - dzieje się tak, ponieważ uzupełniona część rekordów leży w środkowej części, co powoduje zmiejszanie się wartości odchylenia). Na tym polu wypełnianie braków wartościami losowymi wypada bardzo dobrze.**

1.14 **Patrząc na wykres średniej - wszystkie 3 sposoby prezetują się stosunkowo podobnie.**

[72]:
```
x = ['20', '30', '40','50','60','70','80','90','100']
df_dropped_mean, df_dropped_std = get_mean_std_uniform(df_dropped)
df_mean_mean, df_mean_std = get_mean_std_uniform(df_mean)
df_median_mean, df_median_std = get_mean_std_uniform(df_median)
df_random_mean, df_random_std = get_mean_std_uniform(df_random)

f = plt.figure(figsize=(15,15))
ax = f.add_subplot(211)
ax2 = f.add_subplot(212)

ax.plot(x, df_dropped_mean, label = "Oryginał")
ax.plot(x, df_mean_mean, label = "Uz. Średnią")
ax.plot(x, df_median_mean, label = "Uz. Medianą")
ax.plot(x, df_random_mean, label = "Uz. Los. Wart.")
ax.set_xlabel("Wielkość orginalnego zbioru [%]")
ax.set_ylabel("Średnia")
ax.set_ylim([89,97])
ax.legend()

ax2.plot(x, df_dropped_std, label = "Oryginał")
ax2.plot(x, df_mean_std, label = "Uz. Średnią")
ax2.plot(x, df_median_std, label = "Uz. Medianą")
ax2.plot(x, df_random_std, label = "Uz. Los. Wart.")
ax2.set_xlabel("Wielkość orginalnego zbioru [%]")
ax2.set_ylabel("Odchylenie standardowe")
ax2.set_ylim([4,22])
ax2.legend()
```

[72]:
```
<matplotlib.legend.Legend at 0x2208ede1ca0>
```


Średnia
Wielkość orginalnego zbioru [%]
Oryginał
Uz. Średnią
Uz. Medianą
Uz. Los. Wart.


Odchylenie standardowe
Wielkość orginalnego zbioru [%]
Oryginał
Uz. Średnią
Uz. Medianą
Uz. Los. Wart.

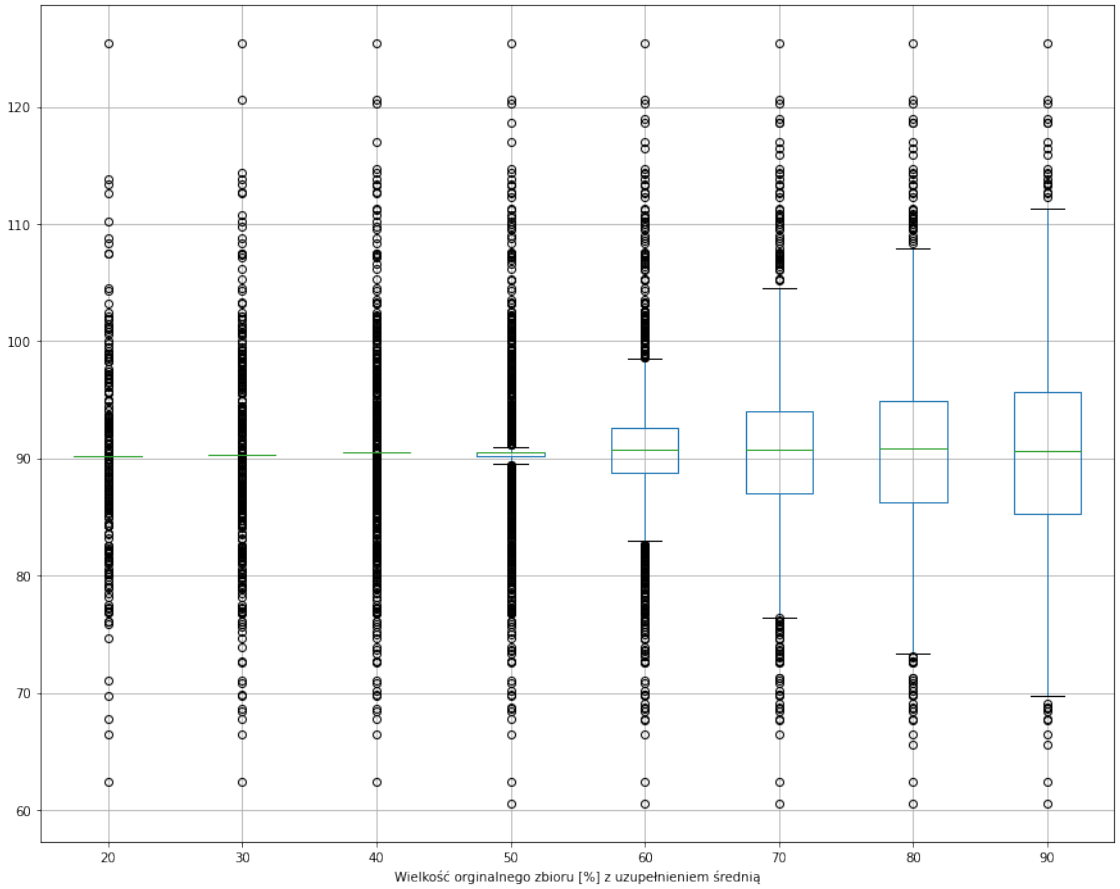
1.15 **Im większe braki w zbiorze danych, tym bardziej statystyki różnią się w porównaniu do danych oryginalnych, gdy danych brakuje więcej niż około 30-35% różnice na wykresie średniej stają się dużo bardziej zauważalne.**

1.16 **Na wykresie opisującym zachowanie oddchylenia standardowego sytuacja jest identyczna jak w przypadku rozkładu normalnego.**

1.17 **Patrząc na wykres średniej można zauważyć, że w sytuacji, gdy mamy jedynie 20% danych, a resztę uzupełniamy, losowanie wartości dało efekt najbliższy oryginałowi.**

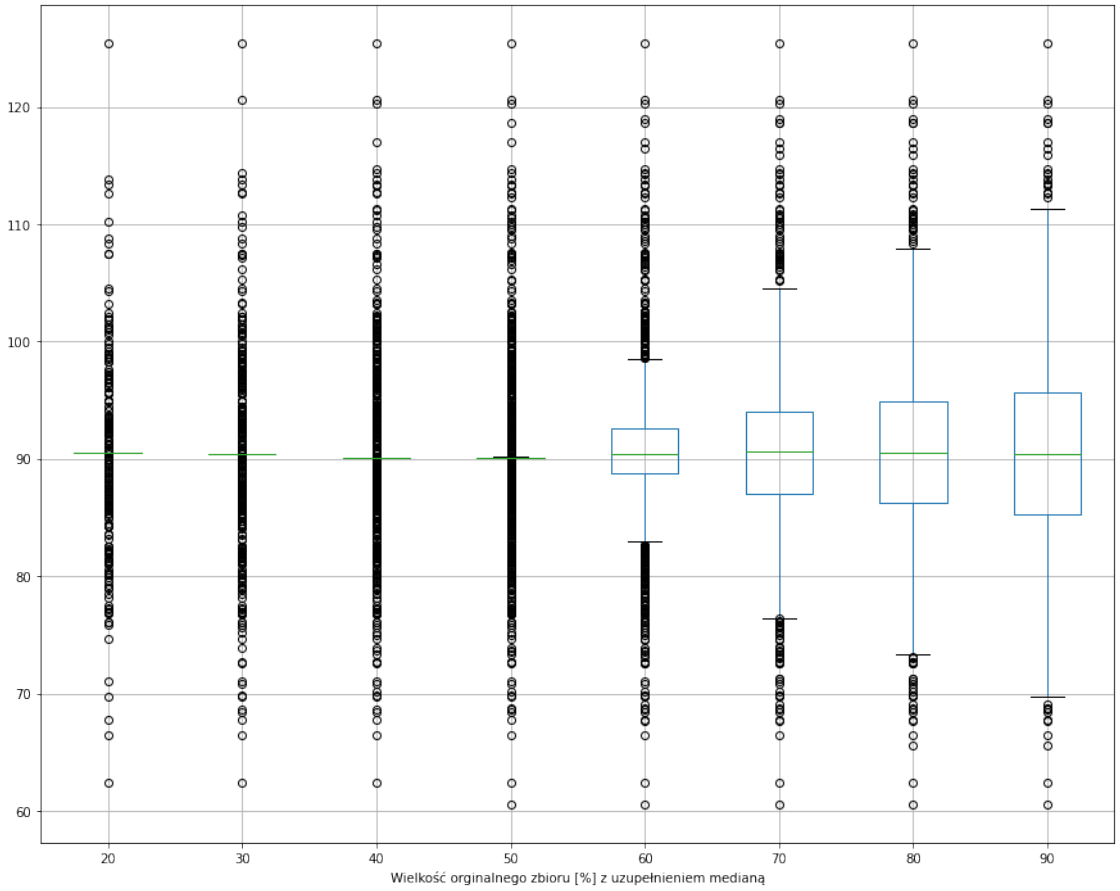
[73]:
```
df_mean_normal = [df_['Rozkład normalny'].copy() for df_ in df_mean]
for df_ in df_mean_normal:
    df_.index = range(1, len(df_) + 1)

df_mean_normal.reverse()
df_mean_normal_merged = pd.concat(df_mean_normal, axis = 1)
df_mean_normal_merged.columns = ['20', '30', '40','50','60','70','80','90']
ax = df_mean_normal_merged.boxplot(column = ['20', '30',␣
 ↪'40','50','60','70','80','90'], figsize = [15,12])
ax.set_xlabel("Wielkość orginalnego zbioru [%] z uzupełnieniem średnią")
```

[73]:
```
Text(0.5, 0, 'Wielkość orginalnego zbioru [%] z uzupełnieniem średnią')
```



[74]:
```
df_median_normal = [df_['Rozkład normalny'].copy() for df_ in df_median]
for df_ in df_median_normal:
    df_.index = range(1, len(df_) + 1)

df_median_normal.reverse()
df_median_normal_merged = pd.concat(df_median_normal, axis = 1)
df_median_normal_merged.columns = ['20', '30', '40','50','60','70','80','90']
ax = df_median_normal_merged.boxplot(column = ['20', '30',␣
 ↪'40','50','60','70','80','90'], figsize = [15,12])
ax.set_xlabel("Wielkość orginalnego zbioru [%] z uzupełnieniem medianą")
```

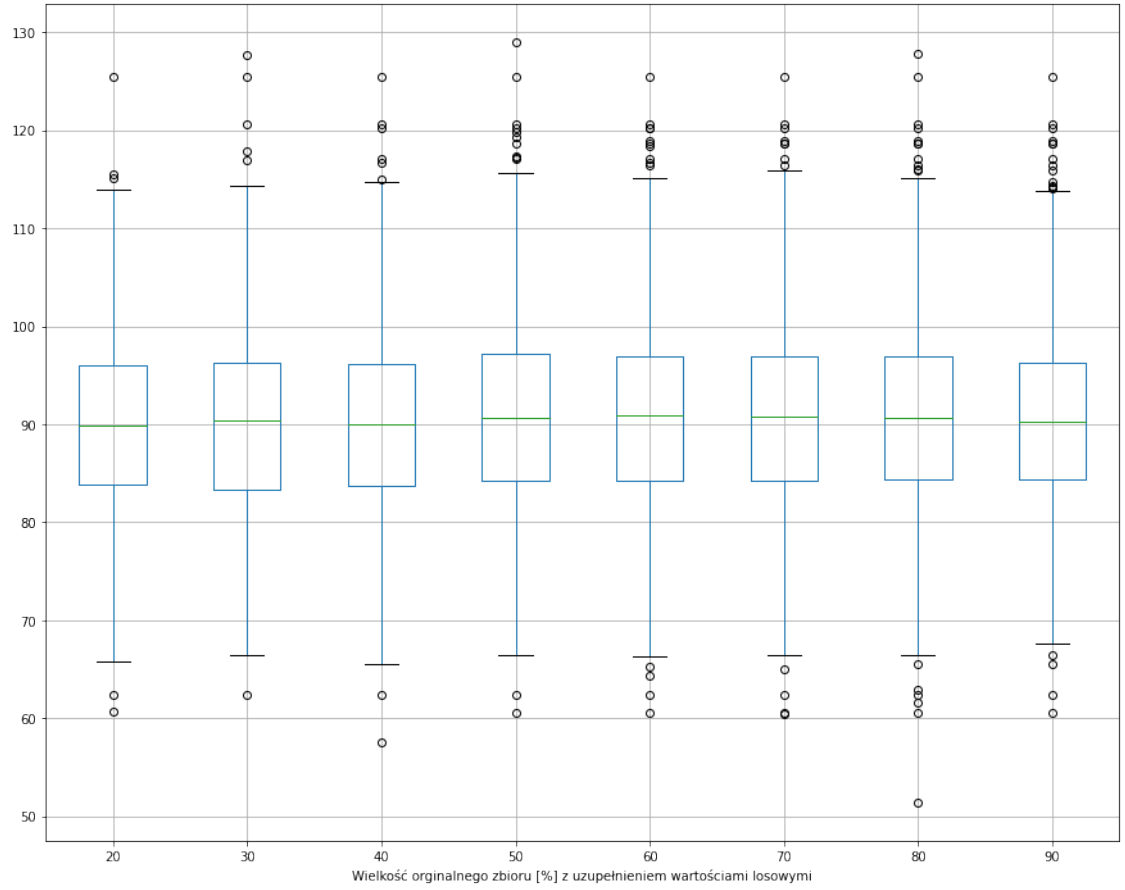
[74]:
```
Text(0.5, 0, 'Wielkość orginalnego zbioru [%] z uzupełnieniem medianą')
```



[75]:
```
df_random_normal = [df_['Rozkład normalny'].copy() for df_ in df_random]
for df_ in df_random_normal:
    df_.index = range(1, len(df_) + 1)
```

```
df_random_normal.reverse()
df_random_normal_merged = pd.concat(df_random_normal, axis = 1)
df_random_normal_merged.columns = ['20', '30', '40','50','60','70','80','90']
ax = df_random_normal_merged.boxplot(column = ['20', '30',␣
↪'40','50','60','70','80','90'], figsize = [15,12])
ax.set_xlabel("Wielkość orginalnego zbioru [%] z uzupełnieniem wartościami␣
↪losowymi")
```

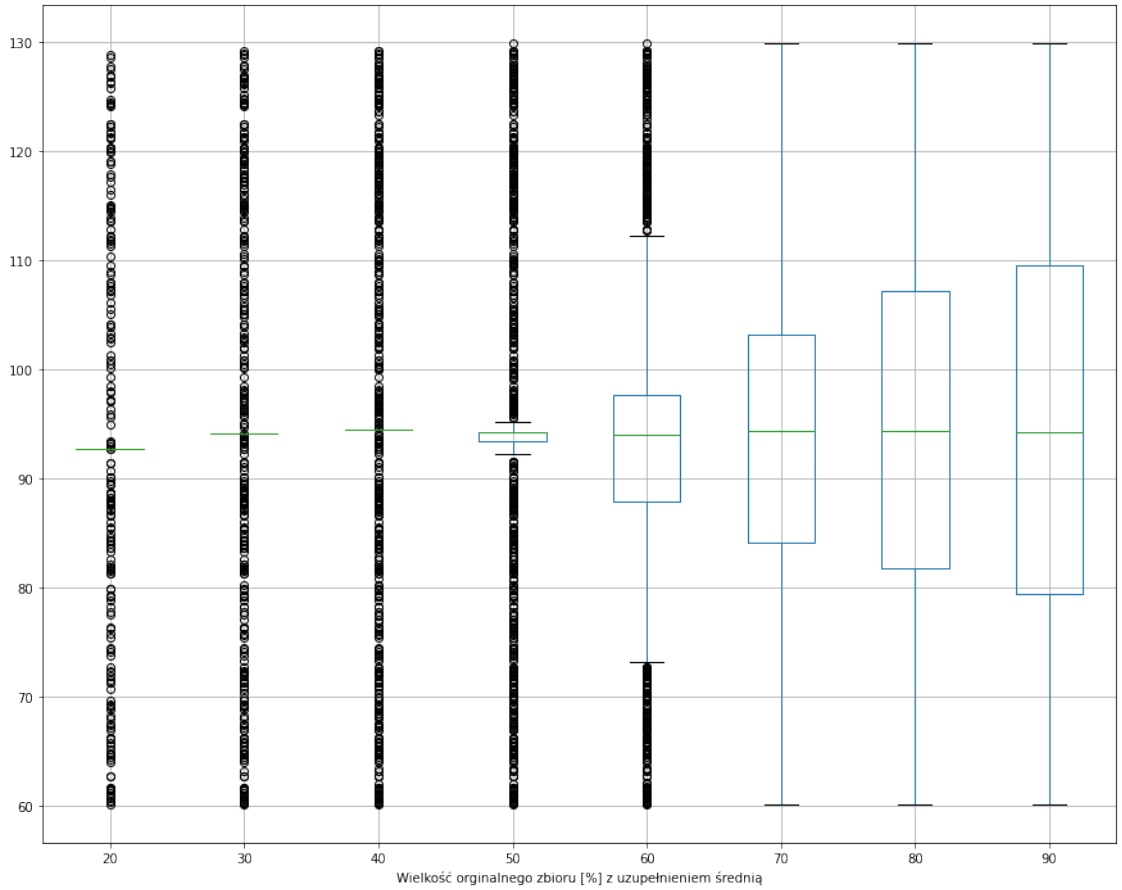
[75]:
```
Text(0.5, 0, 'Wielkość orginalnego zbioru [%] z uzupełnieniem wartościami
losowymi')
```



[76]:
```
df_mean_uniform = [df_['Rozkład jednostajny'].copy() for df_ in df_mean]
for df_ in df_mean_uniform:
    df_.index = range(1, len(df_) + 1)

df_mean_uniform.reverse()
df_mean_uniform_merged = pd.concat(df_mean_uniform, axis = 1)
df_mean_uniform_merged.columns = ['20', '30', '40','50','60','70','80','90']
```

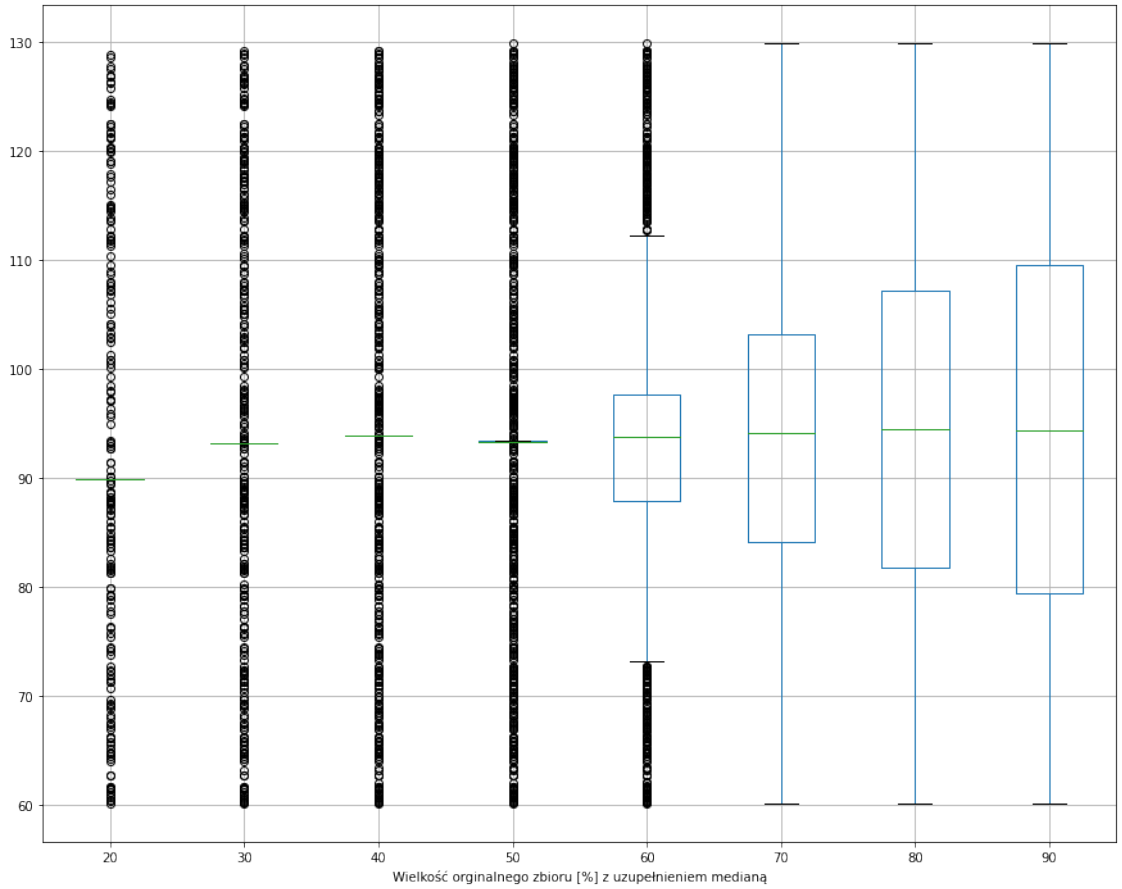
```
ax = df_mean_uniform_merged.boxplot(column = ['20', '30',␣
 ↪'40','50','60','70','80','90'], figsize = [15,12])
ax.set_xlabel("Wielkość orginalnego zbioru [%] z uzupełnieniem średnią")
```

[76]: Text(0.5, 0, 'Wielkość orginalnego zbioru [%] z uzupełnieniem średnią')



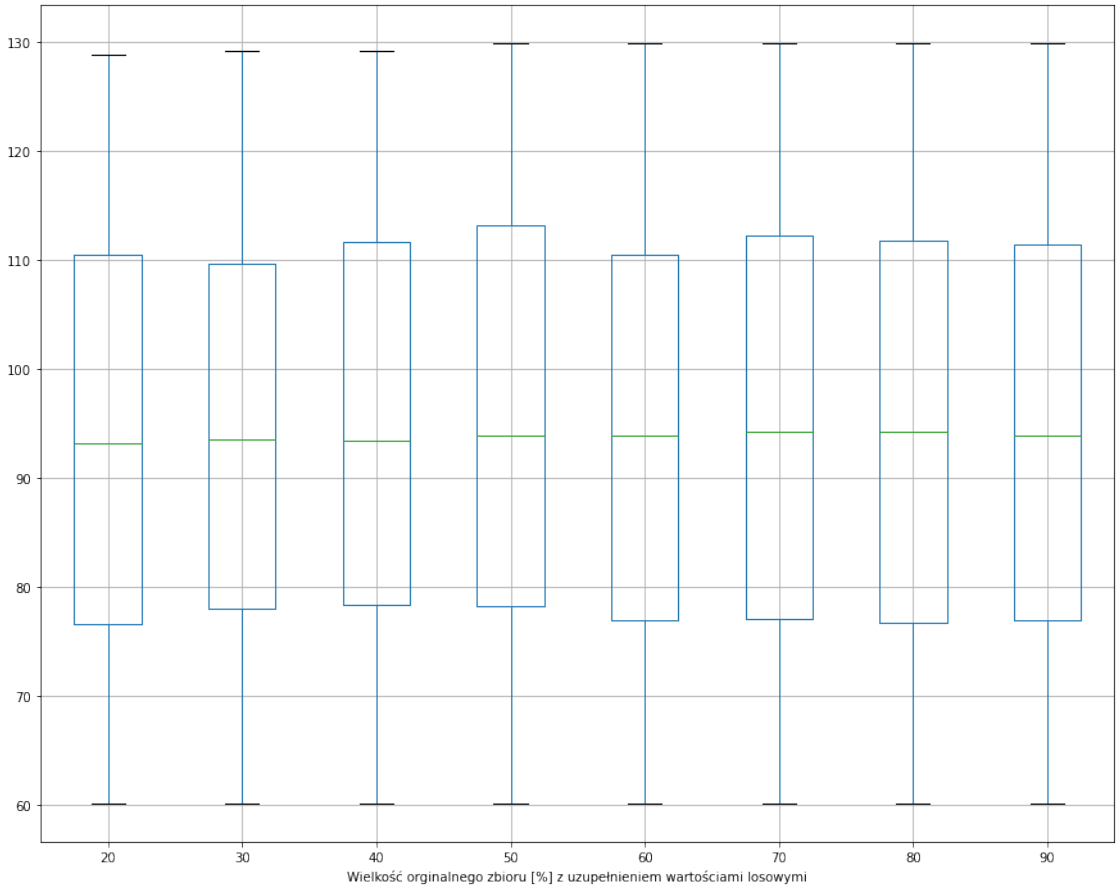
[77]:
```
df_median_uniform = [df_['Rozkład jednostajny'].copy() for df_ in df_median]
for df_ in df_median_uniform:
    df_.index = range(1, len(df_) + 1)

df_median_uniform.reverse()
df_median_uniform_merged = pd.concat(df_median_uniform, axis = 1)
df_median_uniform_merged.columns = ['20', '30', '40','50','60','70','80','90']
ax = df_median_uniform_merged.boxplot(column = ['20', '30',␣
 ↪'40','50','60','70','80','90'], figsize = [15,12])
ax.set_xlabel("Wielkość orginalnego zbioru [%] z uzupełnieniem medianą")
```

[77]: Text(0.5, 0, 'Wielkość orginalnego zbioru [%] z uzupełnieniem medianą')



[78]:
```
df_random_uniform = [df_['Rozkład jednostajny'].copy() for df_ in df_random]
for df_ in df_random_uniform:
    df_.index = range(1, len(df_) + 1)

df_random_uniform.reverse()
df_random_uniform_merged = pd.concat(df_random_uniform, axis = 1)
df_random_uniform_merged.columns = ['20', '30', '40','50','60','70','80','90']
ax = df_random_uniform_merged.boxplot(column = ['20', '30',␣
 ↪'40','50','60','70','80','90'], figsize = [15,12])
ax.set_xlabel("Wielkość orginalnego zbioru [%] z uzupełnieniem wartościami␣
 ↪losowymi")
```

[78]:
```
Text(0.5, 0, 'Wielkość orginalnego zbioru [%] z uzupełnieniem wartościami
losowymi')
```



1.18 **Zarówno dla rozkładu normalnego jak i jednostajnego sytuacja ma się w zasadzie identycznie.**

1.19 **Po uzupełnieniu braków zarówno średnią jak i medianą wykresy skrzynka-wąsy widać wyraźnie, że gdy ilość danych, które uzupełniamy rośnie następuje spłaszczenie skrzynek - większa liczba danych mieści się w samym środku (lub średniej), a sam wykres wydaje się zdecydowanie odbiegający od rzeczywistoci. Jedynie uzupełniając dane wartościami losowymi otrzymujemy wyniki sprawiające wrażenie, że mogłyby być rzeczywiste - mamy niewielką ilość danych odstających, skrzynki mają podobne wielkości niezależnie od tego ile danych pozostało nam uzupełnić.**