Andmed. Tõenäosus. Statistika

Gümnaasiumi reaalharu

Jan Willemson

https://varamu.eu

Saatesõna

Sa hoiad käes Eesti gümnaasiumi reaalharule mõeldud eksperimentaalsesse matemaatikaõpikute sarja kuuluvat õpikut. Sari on leidnud inspiratsiooni 2023. aasta gümnaasiumi laia matemaatika õppekavast, kuid ei vasta sellele üksüheselt. Autori hinnangul vajabki Eesti gümnaasiumi matemaatika ainekava põhjalikku uuendamist. Käesolev sari kujutab endast autori nägemust sellest, milline matemaatika ainekava 21. sajandi 1. veerandi lõpul välja peaks nägema, et üldhariduskooli lõpetajad oleksid valmis jätkama õpinguid ühiskonnale olulistel erialadel.

Sarja sõsarõpikuks on sama autori „Võistlusmatemaatika põhivara", millest huvitatud lugeja leiab palju täiendavaid ülesandeid ning süvendatud materjali. Mõned jaotised ja ülesanded on kahel õppematerjalil ka ühised – aga eks matemaatika, mida neis käsitletakse, ongi ju üks ja seesama.

Sarja õpikud on hetkel mustandi staatuses – neis võib esineda lünki, vigu ning muid vajakajäämisi. Autor on tänulik kommentaaride ja tagasiside eest, mida saab saata aadressile `matemaatika@varamu.eu`.

Sisukord

PEATÜKK 1

Andmed ja nende töötlemine

Nii matemaatika kui elektronarvutid on suures plaanis välja mõeldud andmete töötlemiseks. Juba antiikajal pidid arhitektid hindama, kui suuri kiviplokke läheb vaja templi ehituseks, meresõitjatel oli tarvis teada, kui kaugel nad on oma sihtmärgist, ja sõdade ajal tuli murda vastaste salakirju.

Kõigi nende ülesannete lahendamiseks läheb tarvis teatud tüüpi sisendandmeid (templi soovitav suurus, tähtede asend taevavõlvil või vastaste salakiri), mille põhjal siis arvutusi teha. Kui lihtsamate arvutuste jaoks piisab pliiatsist ja paberist (või isegi puupulgast ja liivast), siis keerukamatel puhkudel hakkab vajalike arvutuste maht kiiresti kasvama.

Üks võimalik keerukuse allikas on ülesande olemuslik raskus. Näiteks salakirjad konstrueeritaksegi nii, et isegi lühikese sõnumi murdmine nõuaks astronoomiliselt palju rehkendamist.

Teine põhjus, miks arvutused käest ära lähevad, on töödeldavate andmete hulk. Tänapäeval räägime üha rohkem *suurandmetest* – tohututest andmekogumitest, mida me oma igapäevase tegevusega toodame. Meie nutiseadmed suhtlevad kogu aeg mobiilivõrguga, et edastada meie asukohta, saata sõnumeid ning võtta vastu teisi, mis meile saadetakse. Isegi kui serverid ei salvesta sõnumite sisu, pannakse nende kohta kirja palju *metainfot* – kes, kellele, millal ja millisest seadmest sõnumi saatis. Arvestades maailmas iga päev saadetavate sõnumite hulka on see tohutu andmekogus ja selle toodab ainult üks konkreetne rakendus.

Samas on kogutud andmed ka väga väärtuslikud. Briti matemaatik ja ettevõtja Clive Humby ütles oma 2006 aastal peetud kõnes: „Andmed on uus nafta!“, pidades sellega silmas, et toorandmeid leidub palju, aga selleks, et neist väärtus kätte saada, tuleb neid enne töödelda. Käesolevas kursuses tutvumegi peamiste andmetöötlusmeetodite ning nende matemaatiliste alustega.

1.1 Andmete esitamine

Kõige levinum samaliigiliste andmete esitamise vorm on nende kogumine *tabelisse*, st andmestruktuuri, mille read (ehk *kirjed*) vastavad andmeobjektidele, veerud aga eri tüüpi *tunnustele*, mis meid nende objektide kohta huvitavad.

Tabelis 1.1 on näiteks toodud põhilised andmed Eesti linnade kohta 2025. aasta seisuga.

Vasta selle tabeli põhjal järgmistele küsimustele.

Ülesanne 1.1 Mitu linna on Eestis 2025. aasta seisuga?

Ülesanne 1.2 Milline on rahvaarvult suurim Eesti linn, milline aga väikseim?

Ülesanne 1.3 Tartu kohta on teada, et ta sai linnaõigused millalgi 1225. ja 1262. aasta vahel, teiste linnade kohta on nende linnaõigused täpsemalt dokumenteeritud. Millised on kolm vanimat Eesti linna? Milline linn on kõige noorem?

Ülesanne 1.4 Mitu Eesti asulat on linnaks saanud pärast aastat 1900?

Ülesanne 1.5 Milline on väikseim 13. sajandil linnaõigused saanud asula?

Nendele küsimustele on võimalik vastata näpuga tabelis järge ajades, kuigi see on paiguti üsna tüütu. Juba tabeli ridade ühekaupa kokkulugemine ülesande 1.1 jaoks võib kergesti sassi minna ja vale tulemuse anda, rääkimata kõige väiksema või kõige noorema linna ülesotsimisest.

Seepärast ongi andmeid puudutavatele küsimustele vastamisel mõistlik kasutada arvutite abi. Et arvutid meid aidata saaks, tuleb andmed kõigepealt esitada neile arusaadavalt, masinoetaval kujul.

Üks lihtsamaid ja levinumaid tabelandmete esitusviise on CSV-failivorming, kus CSV on lühend inglisekeelsest fraasist *Comma-Separated Values*. Igale kirjele vastab failis üks rida ning kirje väljad on eraldatud mingi kindla sümboliga. Inglisekeelses kultuuriruumis valitakse eraldajaks sageli koma, kuid kuna Eestis eraldab koma reaalarvu kümnendkohti, tuleb meil kasutada mõnda teist sümbolit, näiteks semikoolonit. Faili esimesse ritta pannakse tavaliselt tabeli veergude (ehk uuritavate tunnuste) nimed. Nõnda võib tabelile 1.1 vastava CSV-faili algus näha välja näiteks järgmine:

```
Nimi;Linnaõigus;Pindala;Rahvaarv
Abja-Paluoja;1993;4,5;1049
Antsla;1938;2,9;1204
Elva;1938;9,9;5622
Haapsalu;1279;11,1;9499
Jõgeva;1938;3,9;5049
```

Kogu tabel on esitatud õpikuga kaasas käivas failivaramus failis `eestilinnad.csv` (kus Tartu linnaõiguse saamise aastaks oleme konkreetsuse huvides määranud 1262).

Andmefailide lugemiseks ning tabelitega opereerimiseks kasutame Pythoni teeki `pandas`:

```
import pandas as pd
```

Tabel 1.1. Eesti linnad

Nimi	Linnaõigus	Pindala (km^2)	Rahvaarv (2025)
Abja-Paluoja	1993	4,5	1049
Antsla	1938	2,9	1204
Elva	1938	9,9	5622
Haapsalu	1279	11,1	9499
Jõgeva	1938	3,9	5049
Jõhvi	1938	7,6	10720
Kallaste	1938	2,3	651
Kärdla	1938	4,5	2892
Karksi-Nuia	1993	2,9	1429
Kehra	1993	3,9	2815
Keila	1938	11,2	11024
Kilingi-Nõmme	1938	4,3	1566
Kiviõli	1946	11,8	4701
Kohtla-Järve	1946	39,3	32839
Kunda	1938	10,1	2990
Kuressaare	1563	15,5	12989
Lihula	1993	4,2	1226
Loksa	1993	3,8	2492
Maardu	1980	23,4	16875
Mõisaküla	1938	2,2	732
Mustvee	1938	5,7	1127
Narva	1345	68,7	52495
Narva-Jõesuu	1993	10,2	2595
Otepää	1936	6,2	2112
Paide	1291	10,1	7936
Paldiski	1783	59,9	4081
Pärnu	1251	33,2	41529
Põltsamaa	1926	6	4090
Põlva	1993	5,5	5392
Püssi	1993	2,1	860
Rakvere	1302	10,64	15668
Räpina	1993	3,8	2037
Rapla	1993	4,7	5320
Saue	1993	4,4	6217
Sillamäe	1957	12,1	12153
Sindi	1938	5	3766
Suure-Jaani	1938	2,9	1125
Tallinn	1248	159	456518
Tamsalu	1996	6,3	2320
Tapa	1926	17,4	5481
Tartu	1225-1262	39	97304
Tõrva	1926	4,8	2600
Türi	1926	6,9	5133
Valga	1584	16,7	11999
Viljandi	1283	14,7	17157
Võhma	1993	1,9	1233
Võru	1784	14	12024

```
df = pd.read_csv('eestilinnad.csv', sep=';', decimal=',')
print(df)
```

Esimene rida impordib `pandas`-teegi ja annab sellele lühinime `pd`. Teine rida loeb andmed failist (pane tähele, et eraldajaks on määratud semikoolon ja kümnendkoha eraldajaks koma!) ning annab tulemusele nime `df`. See nimi on lühend inglisekeelsest fraasist *data frame* ehk andmefreim. Põhimõtteliselt võib talle anda suvalise nime, aga `df` on sagedane kokkulepe, mis aitab kohe mõista, millega on tegu. Kolmas rida trükib andmefreimi `df` välja ja kui kõik läks hästi, tuleb väljatrükk selline:

```
              Nimi  Linnaõigus  Pindala  Rahvaarv
0    Abja-Paluoja         1993     4.50      1049
1          Antsla         1938     2.90      1204
2             Elva         1938     9.90      5622
3         Haapsalu         1279    11.10      9499
...
43           Valga         1584    16.70     11999
44        Viljandi         1283    14.70     17157
45           Võhma         1993     1.90      1233
46            Võru         1784    14.00     12024
```

Paneme tähele, et kuna kirjete loendamine algab indeksist 0, on 47 kirje puhul viimase rea numbriks 46. Samuti paneme tähele, et pindala veerus on kümnendkohtade eraldajaks punkt – see tähendab, et Python on interpreteerinud pindalad edukalt reaalarvudena (mitte näiteks tekstistringidena).

1.2 Andmefreimi töötlemine

Andmefreim töötab sarnaselt teiste Pythoni andmestruktuuridega, näiteks loendi ja sõnastikuga. Nii saame küsida freimi kirjete arvu:

```
>>> len(df)
47
```

Veidi ootamatult ei saa niisama lihtsalt pärida freimi ühte rida. Kontrolli järele, et käsk

```
>>> df[2]
```

annab vea. Põhjus on selles, et andmefreim pole ette nähtud üksikute väärtuste või kirjete, vaid andmestiku kui terviku uurimiseks. Küll saame freimist pärida *alamfreimi*, näiteks kolme esimese rea leidmiseks saame anda käsu

```
>>> df[0:3]
           Nimi  Linnaõigus  Pindala  Rahvaarv
0  Abja-Paluoja         1993      4.5      1049
1        Antsla         1938      2.9      1204
2          Elva         1938      9.9      5622
```

Väikese trikitamisega saame muidugi kätte ka üherealise alamfreimi:

```
>>> df[2:3]
   Nimi  Linnaõigus  Pindala  Rahvaarv
2  Elva         1938      9.9      5622
```

Samas on tähtis mõista, et tegemist pole lihtsalt ühe reaga tabelist, vaid ühe-realise alamfreimiga, millele saab rakendada kõiki freimitöötlusmeetodeid.

Alamfreimi saame moodustada ka mõnest tabeli veerust. Sel juhul pöördume freimi kui sõnastiku poole, kasutades võtmena vastava veeru nime:

```
>>> df['Rahvaarv']
0      Abja-Paluoja
1            Antsla
2              Elva
...
44         Viljandi
45            Võhma
46             Võru
```

Kui on tarvis alamfreimi algse tabeli mitmest veerust, anname ette vastavate veerunimede loendi:

```
>>> df[['Nimi','Rahvaarv']]
             Nimi  Rahvaarv
0    Abja-Paluoja      1049
1          Antsla      1204
2            Elva      5622
...
44       Viljandi     17157
45          Võhma      1233
46           Võru     12024
```

Pandas-teegi abil Pythonisse laaditud andmeid saame nüüd juba mugavamalt uurida. Näiteks selleks, et selgitada välja suurimad ja vähimad linnad, on mõistlik andmefreim sorteerida veeru `Rahvaarv` järgi:

```
>>> df.sort_values('Rahvaarv')
             Nimi  Linnaõigus  Pindala  Rahvaarv
6        Kallaste        1938     2.30       651
19      Mõisaküla        1938     2.20       732
29          Püssi        1993     2.10       860
...
21          Narva        1345    68.70     52495
40          Tartu        1262    39.00     97304
37         Tallinn        1248   159.00    456518
```

Ülesanne 1.6 Leia andmefreimi `df` sorteerimise abil Eesti vanimad ja noorimad linnad.

Sageli on tarvis andmefreimist eraldada mingile tingimusele vastav osa. Näiteks ülesandes 1.4 küsiti, mitu asulat on saanud linnaõigused pärast 1900. aastat. Vastuse väljaselgitamiseks moodustame alamfreimi, mis vastabtingimusele `df['Linnaõigus']>1900`. Selleks kasutab Pandas süntaksit `df[...]`, kus kantsulgude vahele kirjutatakse vastav tingimus:

```
>>> df[df['Linnaõigus']>1900]
             Nimi  Linnaõigus  Pindala  Rahvaarv
0    Abja-Paluoja        1993      4.5      1049
1          Antsla        1938      2.9      1204
2            Elva        1938      9.9      5622
```

```
4          Jõgeva         1938       3.9       5049
...
```

Näeme, et tulemuseks on uus andmefreim. Kuna ülesandes 1.4 küsiti ainult tingimusele vastavate linnade arvu, võime kohe rakendada `len`-funktsiooni:

```
>>> len(df[df['Linnaõigus']>1900])
35
```

Ülesandes 1.5 küsiti väikseimat 13. sajandil linnaõigused saanud asulat. Selleks peame moodustama alamfreimi, kus linnaõiguste aasta vastab tingimustele `df['Linnaõigus']>1200` ja `df['Linnaõigus']<=1300`. Kahte tingimust saab kombineerida sümboliga `&`, samuti tuleb tingimuste ümber panna sulud:

```
>>> df[(df['Linnaõigus']>1200) & (df['Linnaõigus']<=1300)]
        Nimi  Linnaõigus  Pindala  Rahvaarv
3   Haapsalu         1279     11.1      9499
24     Paide         1291     10.1      7936
26     Pärnu         1251     33.2     41529
37   Tallinn         1248    159.0    456518
40     Tartu         1262     39.0     97304
44  Viljandi         1283     14.7     17157
```

See andmefreim on piisavalt väike vahetuks uurimiseks, aga soovi korral saame teda muidugi sorteerida funktsiooni `sort_values` abil:

```
>>> df[(df['Linnaõigus']>1200) &
... (df['Linnaõigus']<=1300)].sort_values('Rahvaarv')
        Nimi  Linnaõigus  Pindala  Rahvaarv
24     Paide         1291     10.1      7936
3   Haapsalu         1279     11.1      9499
44  Viljandi         1283     14.7     17157
26     Pärnu         1251     33.2     41529
40     Tartu         1262     39.0     97304
37   Tallinn         1248    159.0    456518
```

Andmefreimile saab olemasolevate tunnuste põhjal lisada ka uusi. Näiteks kui me tahame selget ülevaadet, kui vanad Eesti linnad on, võime amdmefreimi täiendada tunnusega `Vanus`:

```
>>> df['Vanus'] = 2025 - df['Linnaõigus']
>>> df
             Nimi  Linnaõigus  Pindala  Rahvaarv  Vanus
0    Abja-Paluoja         1993     4.50      1049     32
1          Antsla         1938     2.90      1204     87
2             Elva         1938     9.90      5622     87
3        Haapsalu         1279    11.10      9499    746
4           Jõgeva         1938     3.90      5049     87
5            Jõhvi         1938     7.60     10720     87
...
```

Näeme, et andmefreimi on lisatud uus veerg ja selle väärtuseks on pandud 2025 miinus linnaõiguse saamise aasta (mis oli linna vanuseks selle raamatu kirjutamise aastal 2025).

Ülesanne 1.7 Kuidas kirjutada kood ümber nii, et 2025 asemel kasutataks jooksva aasta numbrit?

Ülesanne 1.8 Millised on kolm kõige suurema rahvastikutihedusega linna Eestis? Aga kolm kõige väiksema rahvastikutihedusega linna?

1.3 Andmete visualiseerimine

Failis `hinded.csv` on ühe klassi õpilaste nimed ja nende matemaatika kontrolltöö hinded. Loeme selle tabeli andmefreimi ja püüame temast sotti saada.

```
>>> import pandas as pd
>>> df = pd.read_csv('hinded.csv')
>>> print(df)

       Nimi  Hinne
0      Karl      4
1     Laura      5
2    Martin      4
...
31   Birgit      3
32     Aron      2
33   Stella      5
```

See tabel ei ole eriti ülevaatlik ja temast pole esimese hooga selge, kuidas klassil kontrolltöö õigupoolest läks. Kuna tunnusel `Hinne` pole palju võimalikke väärtusi, on mõistlik hinnete andmed esitada *sagedustabelina*, kus tunnuse iga väärtuse kohta on näidatud, mitu korda ta esineb. Sagedustabeli saame koostada funktsiooni `value_counts()` abil, mille rakendame andmefreimi vastavale veerule:

```
>>> df['Hinne'].value_counts()

Hinne
4    13
5    11
3     7
2     3
Name: count, dtype: int64
```

See on juba tunduvalt ülevaatlikum esitus. Tõsi, hetkel on sagedustabel järjestatud mitte hinnete, vaid sageduste järgi. Seda viga saab parandada funktsiooni `sort_index()` abil, mis sorteerib tabeli indeksi järgi eeldusel, et indeksil (antud juhul hinnetel) on järjestus olemas:

```
>>> sagedused = df['Hinne'].value_counts()
>>> sagedused = sagedused.sort_index()
>>> sagedused

Hinne
2     3
3     7
4    13
```

```
5    11
Name: count, dtype: int64
```

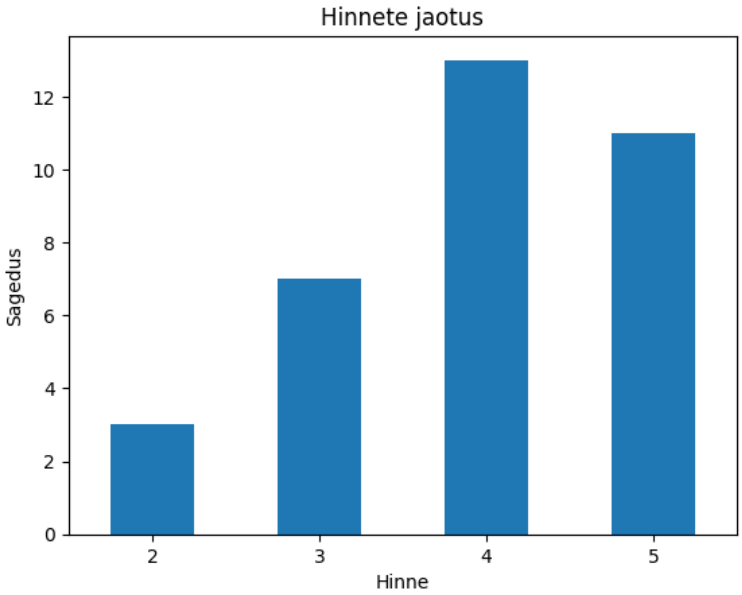
Tasub tähele panna, et andemstruktuur `sagedused` pole tüübilt mitte freim, vaid rida (*series*):

```
>>> type(sagedused)
<class 'pandas.core.series.Series'>
```

Sagedustabelit on tavaks visualiseerida ka graafiliselt. Üks levinumaid graafilisi sagedustabeli esitusi on *tulpdiagramm*, kus iga kategooria sagedust esitatakse vastava kõrgusega tulba abil. Pythonis on tulpdiagrammide joonistamiseks palju võimalusi. Näiteks saab sellega hakkama ka meile juba tuttav teek `matplotlib`:

```
import matplotlib.pyplot as plt
sagedused.plot(kind="bar", rot=0)
plt.xlabel("Hinne")
plt.ylabel("Sagedus")
plt.title("Hinnete jaotus")
plt.show()
```

Tulemusena saame niisuguse diagrammi:



Teine ülevaade, mida me võime tahta andmerea põhjal saada, on hinnete protsentuaalne jaotus. Leiame kõigepealt protsentide tabeli. Selleks saame kasutada Pandase võimalust teha tehteid kõigi rea elementidega korraga:

```
>>> sagedused / sagedused.sum() * 100
Hinne
4    38.235294
5    32.352941
3    20.588235
```

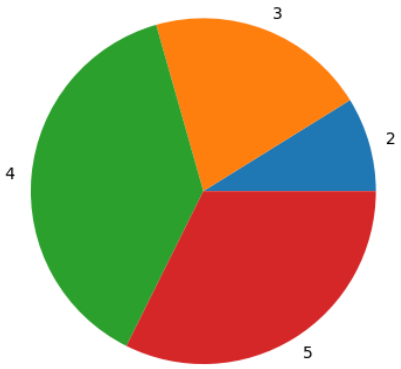
```
2      8.823529
Name: count, dtype: float64
```

Mis siin toimub? Rea `sagedused` pealt arvutame kõigepealt tema elementide summa `sagedused.sum()` (kontrolli, et see tuleb 34!). Seejärel jagame kõik rea elemendid selle summaga, saades osamäärad, ja korrutame 100-ga, mis annab protsendid. Tulemuseks on uus rida protsentidest.

Protsentide tabelit visaliseeritakse sageli *sektordiagrammiga*, kus ring jagatakse väärtustega võrdelisteks sektoriteks:

```
plt.pie(sagedused, labels=sagedused.index)
plt.show()
```

Tulemus näeb välja selline:



Ülesanne 1.9 Failis `eksamitulemused.csv` on ühe maakonna gümnaasiumilõpetajate matemaatika riigieksami tulemused. Koosta nende põhjal tulp- ja sektordiagrammid. Kui ülevaatlikud need tulevad?

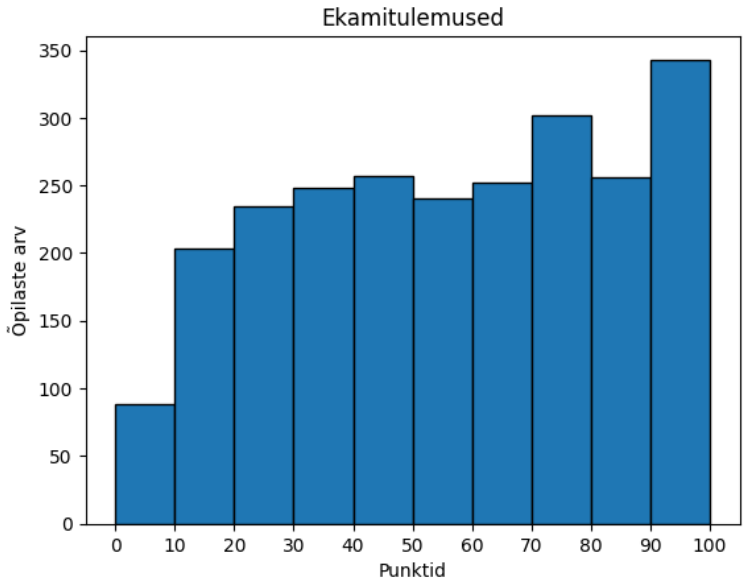
Kui andmepunktidel on palju võimalikke väärtusi, pole me enamasti enam huvitatud igast üksikust neist; pigem tahame aru saada üldisest trendist. Praegusel juhul võime näiteks küsida, kui palju õpilasi sai alla 10 punkti, kui paljud $10 \ldots 19$ jne. Selleks anname Pythonile ette soovitavad vahemikupiirid ning laseme tal joonistada *histogrammi* `matplotlib`-teegi käsuga `hist`:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('eksamitulemused.csv')

vahemikud = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

plt.hist(df['Eksamitulemused'], vahemikud, edgecolor="black")
plt.xlabel('Punktid')
plt.ylabel('Õpilaste arv')
plt.title('Ekamitulemused')
plt.xticks(vahemikud)
plt.show()
```



Lahendused

1.1 Eestis oli 2025. aasta seisuga 47 linna.

1.2 Rahvaarvult suurim on Tallinn (456518 elanikku), väikseim aga Kallaste (651 elanikku).

1.3 Eesti vanimad linnad on Tallinn, Tartu ja Pärnu. Kõige noorem on 1996. aastal linnaõigused saanud Tamsalu.

1.4 Pärast 1900. aastat on Eestis linnaks saanud koguni 35 asulat.

1.5 Nii rahvaarvult kui pindalalt kõige väiksem 13. sajandil linnaõigused saanud linn on Paide (7936 elanikku, $10{,}1\mathrm{km}^2$).

1.6 Andmefreim tuleb sorteerida veeru `Linnaõigus` järgi:

```
>>> df.sort_values('Linnaõigus')
```

1.7 Üks võimalus on kasutada teeki `datetime`:

```
>>> from datetime import datetime
>>> df['Vanus'] = datetime.now().year - df['Linnaõigus']
```

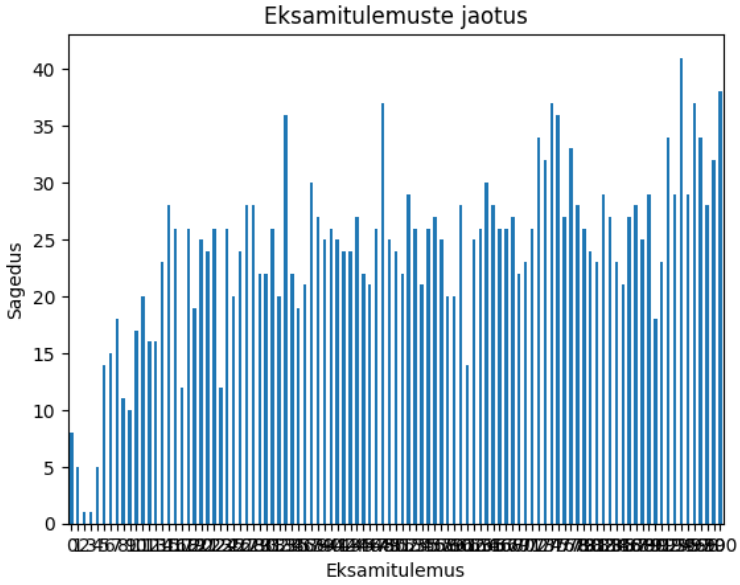
1.8 Lisame andmefreimile tunnuse `Tihedus`, mis on linna rahvaarvu ja pindala jagatis:

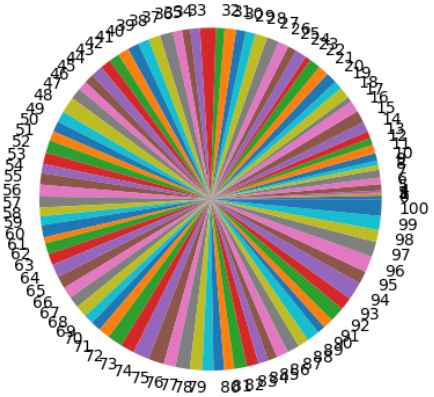
```
>>> df['Tihedus'] = df['Rahvaarv'] / df['Pindala']
>>> df.sort_values('Tihedus')[['Nimi','Tihedus']]
             Nimi      Tihedus
25       Paldiski    68.130217
20        Mustvee   197.719298
```

```
0     Abja-Paluoja    233.111111
...
30         Rakvere   1472.556391
40           Tartu   2494.974359
37         Tallinn   2871.182390
```

Tuletame meelde, et konstruktsioon `[['Nimi','Tihedus']]` võimaldab suure hulga tunnustega tabelist kuvada ainult need, mis meile huvi pakuvad; antud juhul siis `Nimi` ja `Tihedus`.

1.9 Antud eksamitulemused on täisarvud vahemikus $0 \ldots 100$. See tähendab, et tulpasid ja sektoreid tuleb vastavatele diagrammidele väga palju ning saadud punktide väärtused jooksevad üksteisega kokku.





PEATÜKK 2

Statistika

2.1 Andmestiku statistilised näitajad

Uurime 1. peatükist tuttavat Eesti linnade andmestikku lähemalt.

Ülesanne 2.1 Kui palju elas 2025. aasta seisuga Eesti linnades inimesi kokku? Statistikaameti hinnangul elas 2025. aastal Eestis umbes $1{,}37$ miljonit inimest. Kui suur oli linnainimeste protsent Eesti elanike seas?

Saame kasutada funktsiooni `sum` üle tabeli veeru `Rahvaarv`:

```
>>> sum(df['Rahvaarv'])
908636
>>> sum(df['Rahvaarv'])/1370000
0.6632379562043795
```

Seega elas 2025. aastal Eestis linnades 908636 inimest ehk umbes $66{,}3\%$ rahvastikust.

Ülesanne 2.2 Kui palju elab inimesi keskmises Eesti linnas?

Sellele küsimusele vastamiseks tuleb kõigepealt läbi mõelda, mida õigupoolest tähendab fraas „keskmine Eesti linn“. Üks võimalus on kasutada aritmeetilist keskmist. Tuletame meelde, et väärtuste $x_1, x_2, \ldots, x_n$ *aritmeetiliseks keskmiseks* nimetatakse suurust

$$\frac{x_1 + x_2 + \ldots + x_n}{n}.$$

Seega võime arvutada

```
>>> sum(df['Rahvaarv'])/len(df)
19332.68085106383
```

või funktsiooniga `mean()`

```
>>> df['Rahvaarv'].mean()
19332.68085106383
```

Niisiis elab Eesti linnas keskmiselt umbes 19332 inimest. Milline see keskmine Eesti linn on? Sorteerime tabeli rahvaarvu järgi kahanevalt:

```
>>> df.sort_values(by='Rahvaarv',ascending=False)
             Nimi  Linnaõigus  Pindala  Rahvaarv
37         Tallinn        1248   159.00    456518
40           Tartu        1262    39.00     97304
21           Narva        1345    68.70     52495
26           Pärnu        1251    33.20     41529
13   Kohtla-Järve        1946    39.30     32839
44        Viljandi        1283    14.70     17157
18          Maardu        1980    23.40     16875
30         Rakvere        1302    10.64     15668
15     Kuressaare        1563    15.50     12989
34        Sillamäe        1957    12.10     12153
46            Võru        1784    14.00     12024
...
```

Nõnda arutledes osutub keskmiseks Eesti linnaks Viljandi, mis on rahvaarvult 6. kohal. Linnu oli aga koguni 47, miks siis keskmiseks osutub kuues?

Probleem seisneb selles, et Eestis on mõned väga suured linnad, mis kallutavad aritmeetilise keskmise enda kasuks. Olukord, kus andmestikus esinevad väga erinevad väärtused, on praktikas üsna tavaline. Seetõttu osutub aritmeetilise keskmise asemel sageli mõistlikumaks vaadelda *mediaani*, st väärtust, millest mõlemale poole jääb andmestikus sama palju väärtusi.[1] Pandas-teegi abil saame tabeliveeru mediaani leida funktsiooniga `median()`:

```
>>> df['Rahvaarv'].median()
4701.0
```

Niisiis elab mediaanlinnas 4701 inimest. Selleks, et leida, mis linn see on, saame kasutada funktsiooni `loc`, mis väljastab väärtusele vastava rea:

```
>>> mediaan = df['Rahvaarv'].median()
>>> df.loc[df['Rahvaarv'] == mediaan]
       Nimi  Linnaõigus  Pindala  Rahvaarv
12  Kiviõli        1946     11.8      4701
```

Ülesanne 2.3 Leia Statistikaameti kodulehelt Eesti viimase kvartali keskmine ja mediaanpalk. Miks nad erinevad? Kumb neist peegeldab paremini rahvastiku keskmist toimetulekut?

2.2 Lahendused

2.3 Eesti 2025. aasta III kvartali keskmine brutopalk (st palk enne üksikisiku makse) oli 2075 eurot, mediaanpalk aga 1722 eurot. Sinu leitud andmed

[1] Kui andmestikus on paarisarv väärtusi, loetakse mediaaniks kahe keskmise väärtuse aritmeetiline keskmine.

on ilmselt veidi teistsugused, sest nii keskmine kui mediaanpalk tõusevad ajas, aga nende erinevus on arvatavasti endiselt märkimisväärne. Nii on keskmine palk meidaanpalgast 2025. aasta III kvartalis umbes 20% kõrgem. Põhjus on sama, mis keskmise suurusega linna otsimise puhul – andmestikus on mõned väga suured väärtused, antud juhul siis vähesed inimesed, kelle palk on võrdlemisi kõrge. Rikka vähemuse järgi ei saa aga hinnata rahvastiku keskmist toimetulekut; sellest annab mediaanpalk paremini aimu.

Indeks