Liite 2.

Monte Carlo -simulaatio

Tässä liitteessä esitetään Monte Carlo -simulaatiossa käytetty Python-koodi, joka generoi asetettujen lähtöarvojen perusteella merkkijonosarjoja, joista jokaiselle lasketaan erikseen Continuity of care -indeksi (Coci) sekä Usual provider index (UPC) -arvot.

Merkkijonokohtaisista Coci- ja UPC-arvoista lasketaan keskiarvoina terveyskeskuskohtaiset jatkuvuusindeksiarvot. Jokaisen simulaation terveyskeskuskohtainen jatkuvuusindeksiarvo tallennetaan koodilla luotavaan Excel-tiedostoon.

Koodissa on lihavoituna simulaation ”asetukset”, jotka tutkimuksessa toimivat laskujen lähtöarvoina.

```
import random
import pandas as pd
from tkinter import Tk, messagebox, ttk
def generate_random_scenarios(lengths, counts, additional_chars):
    scenarios = []
    for length, count in zip(lengths, counts):
        for _ in range(count):
            scenario = "".join(random.choice(additional_chars) for _ in range(length))
            scenarios.append(scenario)
    return scenarios
def replace_characters(scenarios, ascii_settings, additional_chars):
    new_scenarios = scenarios.copy()
    total_length = sum(len(s) for s in scenarios)
    for char, percentage in ascii_settings.items():
        total_to_replace = round(total_length * (percentage / 100))
        for _ in range(total_to_replace):
            replaced = False
            while not replaced:
                scenario_index = random.randrange(len(new_scenarios))
                char_index = random.randrange(len(new_scenarios[scenario_index]))
```

```
            if new_scenarios[scenario_index][char_index] in additional_chars:
                new_scenarios[scenario_index] = (new_scenarios[scenario_index][:char_index] +
char +
                                                 new_scenarios[scenario_index][char_index + 1:])
                replaced = True
    return new_scenarios
def calculate_coci(visit_counts, total_visits):
    numerator = sum(n**2 for n in visit_counts) - total_visits
    denominator = total_visits * (total_visits - 1)
    return numerator / denominator if denominator != 0 else 0
def calculate_upc(scenario):
    visit_counts = {char: scenario.count(char) for char in set(scenario)}
    max_visits = max(visit_counts.values())
    return max_visits / len(scenario)
def suorita_simulaatio(lengths, counts, ascii_settings, additional_chars, iteraatio_id,
on_viimeinen_iteraatio=False):
    scenarios = generate_random_scenarios(lengths, counts, additional_chars)
    new_scenarios = replace_characters(scenarios, ascii_settings, additional_chars)
    coci_values = []
    upc_values = []
    total_chars_count = {}
    for scenario in new_scenarios:
        visit_counts = [scenario.count(char) for char in set(scenario)]
        kokonaiskaynnit = len(scenario)
        coci = calculate_coci(visit_counts, kokonaiskaynnit)
        coci_values.append(coci)
        upc = calculate_upc(scenario)
        upc_values.append(upc)
        for char in set(scenario):
            total_chars_count[char] = total_chars_count.get(char, 0) + scenario.count(char)
```

```
    coci_keskiarvo = sum(coci_values) / len(coci_values) if coci_values else 0
    upc_keskiarvo = sum(upc_values) / len(upc_values) if upc_values else 0
    total_chars = sum(total_chars_count.values())
    char_percentages = {char: (count / total_chars * 100) for char, count in
total_chars_count.items()}
    if on_viimeinen_iteraatio:
        print(f"Iteraatio {iteraatio_id} merkkijonot ja niiden COCI-arvot:")
        for scenario, coci in zip(new_scenarios, coci_values):
            print(f"Merkkijono: '{scenario}', COCI: {coci:.4f}")
        print(f"COCI-keskiarvo: {coci_keskiarvo:.4f}")
        print("Merkkien suhteelliset osuudet (%):")
        for char, percentage in char_percentages.items():
            print(f"{char}: {percentage:.2f}%")
    results = {
        'Iteraatio_ID': iteraatio_id,
        'COCI_Keskiarvo': coci_keskiarvo,
        'UPC_Keskiarvo': upc_keskiarvo}
    return pd.DataFrame([results])
def suorita_simulaatiot_ja_tallenna_exceliin(iteraatioiden_maara, settings, tiedostonimi):
    ikkuna = Tk()
    latauspalkki = ttk.Progressbar(ikkuna, length=200, mode='determinate',
maximum=iteraatioiden_maara)
    latauspalkki.pack()
    ikkuna.update()
    kaikki_tulokset = []
    for iteraatio_id in range(1, iteraatioiden_maara + 1):
        on_viimeinen_iteraatio = iteraatio_id == iteraatioiden_maara
        df_yhteenveto = suorita_simulaatio(**settings, iteraatio_id=iteraatio_id,
on_viimeinen_iteraatio=on_viimeinen_iteraatio)
        kaikki_tulokset.append(df_yhteenveto)
        latauspalkki['value'] = iteraatio_id
```

```
        ikkuna.update()
    ikkuna.destroy()
    lopullinen_df = pd.concat(kaikki_tulokset, ignore_index=True)
    lopullinen_df.to_excel(tiedostonimi, index=False)
# Asetukset Monte Carlo simulaatiolle
settings1 = {
    'lengths': [3, 4, 5, 6, 7, 8, 9, 10, 11, 12], # Merkkimääräryhmä
    'counts': [100, 100, 50, 50, 25, 15, 10, 5, 5, 2], # Merkkijonojen määrä
    'ascii_settings': {'A': 20, 'B': 10, 'C': 5.5, 'D': 0, 'E': 0, 'F': 0, 'G': 0, 'H': 0, 'I': 0, 'J': 0,'K': 0, 'L': 0, 'M': 0, 'N': 0, 'O': 0}, #Työsuhteisten lääkärien osuudet käynneistä
    'additional_chars': ['P','Q','R','S'] # Lisälääkärimerkit
}
iteraatioiden_maara = 100
tiedostonimi = 'ArtikkeliEsimerkki.xlsx'
suorita_simulaatiot_ja_tallenna_exceliin(iteraatioiden_maara, settings1, tiedostonimi)
```