Programmeertechnieken 2018
Opdracht 1: Data Analysis Pipelines

Deadline: Vrijdag 2 maart 23:59.

1 Introductie

Het analyseren van datasets is een veelvoorkomende taak in de informatica. Er zijn vele verschillende typen datasets, zoals bijvoorbeeld een dataset met resultaten van een set experimenten, "access logs" van een webserver of voorraadbeheer van een bedrijf opgeslagen in een relationeel databasesysteem. Op een dataset kunnen analyses worden uitgevoerd met variërende complexiteit: eenvoudige analyses zoals het tellen hoe vaak een bepaalde karakteristiek voorkomt (hoe vaak is een bepaalde pagina van een website bezocht in de laatste maand?) tot ingewikkelde analyses waarbij er bijvoorbeeld wordt gezocht naar patronen in de data (als iemand product A koopt, wat voor ander type producten zitten er dan in dezelfde aankoop?). In deze opdracht beperken we ons tot de eenvoudigere analyses. Het zoeken naar patronen in data komt later in de studie uitgebreid aan de orde bij het vak Data Mining.

In deze opdracht gaan we voertuiggegevens analyseren. De RDW houdt gegevens bij over alle voertuigen met een kenteken. De opdracht bestaat uit het genereren van verschillende statistieken gegeven de open data van de RDW. Bij het uitvoeren van de analyses is het de bedoeling optimaal gebruik te maken van de kracht van UNIX-systemen: in plaats van zelf één groot programma te schrijven dat de analyse uitvoert, maken we een aaneenschakeling van al bestaande tools op een UNIX systeem. In een dergelijke aaneenschakeling wordt de uitvoer van het ene programma steeds doorgestuurd naar een ander programma. Het doorsturen van data wordt gedaan met behulp van een "pipe", vandaar dat voor dergelijke aaneenschakelingen vaak de term "pipeline" wordt gebruikt. Informatie over standaard tools kan worden opgezocht met het commando *man*: met `man -k <zoekterm>` kan er worden gezocht naar de naam van een programma, een gedetailleerde handleiding kan worden opgevraagd met `man <programmanaam>`.

Het komt wel eens voor dat er nog geen geschikt programma bestaat om in de pipeline op te nemen. In dat geval zit er niets anders op dan het programma zelf te schrijven. Vaak wordt er gebruik gemaakt van een scripttaal, zoals Perl of Python, om in weinig tijd een geschikt programma te schrijven. We zullen dat in deze opdracht ook doen, in ieder geval bij het opzoeken van het land waarin een IP-adres zich bevindt en voor het maken van grafieken op basis van de data. Houd bij het ontwikkelen van deze extra programma's de UNIX filosofie in het achterhoofd! Maak de programma's zo generiek mogelijk, zodat je ze in de toekomst zou kunnen hergebruiken.

2 Dataset

Als dataset zullen we gebruik maken van de dataset "Gekentekende voertuigen" van de RDW[1]. Het betreft hier een publieke dataset. Je kan deze dataset downloaden door te kiezen voor "Exporteer" en vervolgens "CSV". Ongecomprimeerd omvat de database ongeveer 7 GB aan data (14 miljoen regels). Je kunt je programma's testen met een deel van de data (met behulp van `head`). Voor het verslag is het de bedoeling dat *alle* data wordt verwerkt.

De bestanden zijn tekstbestanden geformatteerd als een kommagescheiden bestand (CSV, comma-separated values). Elke regel in het bestand is één voertuig. De gegevens worden gescheiden door komma's. Een overzicht van de kolommen staat op de gelinkte website van de RDW en de eerste regel van het bestand geeft ook de veldnamen aan. Het `cut`-commando is handig om specifieke kolommen uit het bestand te selecteren.

Op Blackboard, onder "Assignment 1" kan je het bestand "Gekentekende_voertuigen.csv.bz2" downloaden. Dit is een gecomprimeerde versie (met bzip2) van de eerste 27 kolommen van het databestand die een stuk minder ruimte inneemt. Het is de bedoeling dat je voor de opdracht deze versie gebruikt. Je kan `bzcat` gebruiken als deel van je pipeline om dit bestand uit te pakken.

[1] `https://opendata.rdw.nl/Voertuigen/Open-Data-RDW-Gekentekende_voertuigen/m9d7-ebf2`

3 Te verrichten analyses

De opdracht is nu als volgt: bouw geschikte pipelines om de onderstaande statistieken te genereren. Plaats de pipeline steeds in een shell script (`pipeline1.sh`, enz.). De shell scripts hebben één argument, namelijk de locatie van het gecomprimeerde databestand. Dit verwijst naar een bestand met de versie van de dataset die je op Blackboard kan downloaden.

1. Een lijst van alle voertuigsoorten anders dan "Personenauto" (kijk bijvoorbeeld naar `grep -v`). Zorg ervoor dat geen voertuigsoort meer dan eens voorkomt (zie `sort` en `uniq`).

2. Overzicht top tien merken van personenauto's, aflopend gesorteerd op aantal personenauto's met dat merk. (zie `uniq -c`, `sort -n`, en `head -n`).

3. Bereken voor elk merk de gemiddelde cataloguswaarde van personenauto's van dat merk. Er is geen tool die het gemiddelde berekent dus daarvoor zal je zelf een Python script moeten schrijven.

4. Een grafiek die voor elke dag van de week laat zien hoeveel auto's er op die dag tenaamgesteld zijn (op basis van de kolom "Datum tenaamstelling"). De grafiek heeft dus de 7 dagen van de wwek op de x-as en het gemiddelde aantal personenauto's op de y-as. Gebruik een "bar plot", zie ook de appendix. De waarde afgebeeld voor dinsdag is dus bijvoorbeeld het totaal aantal personenauto's dat op een dindag tenaamgesteld is in de dataset.

   (Schrijf bijvoorbeeld een script dat een datum om kan zetten naar een dag van de week (kijk eens naar de `datetime` *module). Dan staat alle data klaar om de totalen te berekenen. Tenslotte zou je een Python script kunnen schrijven dat gegeven een lijstje van dagen van de week en totaal aantal tenaamgestelde auto's op die dag een PDF-bestand met een grafiek genereert (bijvoorbeeld* NumPy *en* matplotlib *of* pandas*, zie ook de appendix)).*

5. Een boom die voor personenauto's per merk, handelsbenaming, en bouwjaar aangeeft hoe zuinig ze gemiddeld zijn. Het merk is het eerste niveau in de boom, de handelsbenaming het tweede niveau, en het bouwjaar het derde niveau. Het bouwjaar bepaal je op basis van het veld "Datum eerste toelating'. Je berekent de gemiddelde zuinigheid door voor elke auto het zuinigheidslabel (een letter van A tot en met G) om te zetten in een getal (A=1, B=2, ..., G=7) en daarvan het gemiddelde te berekenen. Bij sommige voertuigen mist het zuinigheidslabel, deze kan je negeren. Geef het gemiddelde met één decimaal weer naast het bouwjaar. Zo staat bijvoorbeeld op het hoogste niveau "VOLKSWAGEN" van alle Volkswagens, op het tweede niveau "GOLF", en op het derde niveau "2017 - 3.0" als de gemiddelde zuinigheid van alle Volkswagen Golf uit 2017 3.0 is (dat wil zeggen, gemiddeld zuinigheidslabel C).

   Denk hierbij aan het volgende:

   - *Selecteer de relevante kolommen.*
   - *Je kunt weer gebruik maken van* `uniq -c` *om te tellen.*
   - *Schrijf een script dat* `uniq -c` *uitvoer accepteert en op basis hiervan het gemiddelde zuinigheidslabel berekent.*
   - *Schrijf een script dat hier een ASCII boom voor genereert. Er mag gebruik worden gemaakt van bestaande Python modules om de boom te genereren, bijvoorbeeld* `asciitree`.

4 Vereisten

Er mag worden gewerkt in teams van twee personen. Het volgende moet worden ingeleverd:

- De source code van alle geschreven scripts.
- Shell scripts welke de geschreven pipelines uitvoeren.

- Een kort verslag in PDF formaat, gemaakt met behulp van LaTeX, waarin per analyse wordt uitgelegd hoe de pipeline is opgebouwd, de uitvoer van de pipeline wordt getoond (grafieken, tabellen of tekst) en de genereerde statistieken kort worden bediscussieerd. Voeg de gemaakte tabellen/grafieken dus in je verslag! De code hoef je *niet* in het verslag te zetten.
- Let op: het hele databestand moeten worden meegenomen in de analyse.

Zorg ervoor dat alle bestanden die worden ingeleverd (source code, verslag, enz.) zijn voorzien van naam en studentnummer! Plaats alle bestanden om in te leveren in een aparte directory (bijv., `opdracht1`) en maak een "gzipped tar" bestand:

```
tar -czvf opdracht1-sXXXXXXX-sYYYYYYY.tar.gz opdracht1/
```

Vul op de plek van `XXXXXXX` en `YYYYYYY` de bijbehorende studentnummers in. De inzendingen kunnen worden verzonden per e-mail naar `pt2018@lists.liacs.nl` met als onderwerp "PT Opdracht 1 AAA en BBB", waarbij je AAA en BBB vervangt door de namen van de studenten in je groep. Het verslag hoeft *niet* op papier te worden ingeleverd, zoals bij Programmeermethoden het geval was.

Normering: Verslag (3/10), werking (4/10), kwaliteit code/commentaar/modulariteit (3/10).

Appendix: Tips & Tricks

Eenvoudig plot voorbeeld

Het volgende voorbeeld leest een reeks regels van *stdin* welke bestaan uit (bijvoorbeeld) het uur van de dag en het aantal verstuurde bytes, gescheiden door een spatie. Gebaseerd op deze data wordt een plot gemaakt. Je kunt dit als startpunt gebruiken. Bekijk de documentatie van `matplotlib` om de plot verder op te maken.

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

D = pandas.read_csv(sys.stdin, sep=",", header=None, index_col=0)
D.plot(kind="bar", rot=30, legend=False)
plt.title("Mijn plot")
plt.show()
exit(0)
```