WORK IN PROGRESS

Maggifix OBWAHL-Darstellung

**Herzllichen Glückwunsch!** Du hast dich entschieden, den Auszählungsstand einer Kommune bzw. eines Kreises mit der leistungsfähigen, in der High-Performance-Datenanalyse-Sprache R geschriebenen obwahl-Software zu…

…nein, alles gut, keine Panik. Damit die Echtzeit-Darstellung der Wahlauszählung gelingt, musst du die Technik einrichten und die Daten vorbereiten – wenn alle Informationen von der Kommune da sind, ist das an einem Vormittag bequem zu erledigen. Ein wenig zusätzliche Zeit vielleicht noch für einen Testlauf einplanen – und am Wahlabend in Rufweite sein.

Geschrieben habe ich die Skripte für die Auswertung von OB-Wahlen in den kreisfreien Städten – inwieweit sie auch beispielsweise für Landratswahlen im Kreis taugt, habe ich noch nicht untersucht, aber da die Struktur ähnlich ist – eine große Fläche, die sich in etliche kleine aufteilt – funktioniert das wahrscheinlich genauso.

## Bedingung: Du brauchst:

### Im hr: Technik und Systeme

* Linux-Server oder lokale Maschine; SSH-Terminal- und SFTP-Zugang
* Den R-Code von <https://github.com/JanEggers-hr/obwahlen/> (Wäre klüger gewesen, das Repository „obwahl“ zu nennen, ich weiß)
* Datawrapper-Zugang und API-Token
* Jemanden, der dem Code über die Schulter schauen kann

### Von der Kommune: Daten

* Einen Link auf die CSV-Datei auf der Votemanager-Seite
* Einen Stimmzettel (bzw. eine Liste der Kandidaten)
* Einen Shapefile mit den Stadtteilen (ersatzweise: einen Shapefile aller Stimmbezirke)
* Eine Datei mit der Zuordnung der Stimmbezirke – reale und Briefwahl – zu den Stadtteilen

(Das sind in der Regel Dinge, die man vom Wahlleiter bekommt)

### Von diesem Dokument: Ein wenig Basiswissen

Wahlen werden in Hessen mit einer Software namens „Votemanager“ ausgezählt, die von einem Unternehmen namens Vote-IT stammt und von der landeseigenen IT-Firma ekom21 betreut und eingesetzt wird. Der Votemanager hilft den Wahlhelfern vor Ort, die Stimmen einzutragen, übermittelt das Ergebnis im Wahllokal dann an die Wahlleitung und generiert daraus eine so genannte Schnellmeldung mit den gezählten Stimmen – die dann über eine Webseite und eine CSV-Datei auch veröffentlicht wird.

Die Wahl ist aufgeteilt in so genannte Stimmbezirke – das meist einfach die Wahllokale, ab und zu hat ein Wahllokal aber mehrere Stimmbezirke, und dann gibt es dann noch die Briefwahl-Stimmbezirke. Wann immer ein Stimmbezirk ausgezählt ist, veröffentlicht der Votemanager die Schnellmeldung dazu

Die Briefwahl-Stimmbezirke haben übrigens rein rechnerisch null Wahlberechtigte – während die Wahllokale im Wählerverzeichnis nachschauen können, wieviele Wahlberechtigte es gibt, ist ja nicht vorher klar, wie viele Wählerinnen und Wähler die Briefwahl nutzen.

## Vorbereitung: Code auf den Server

Der R-Code läuft am Wahlabend auf einer (virtuellen) Linux-Maschine. Diese Maschine hat Zugang zu einem Google-Bucket – einem Netz-Laufwerk, das von der virtuellen Maschine beschrieben und aus dem Internet gelesen werden kann. Die Daten dort werden über einen Mechanismus namens CORS von Datawrapper gezogen und aktuell angezeigt.

Der Zugang zur virtuellen Maschine ist über einen digitalen Key. Diesen SSH-Key muss man selbst erzeugen und dann bei Ulrich Gayer hinterlegen, damit er ihn auf der virtuellen Maschine hinterlegt.

* Key importieren
* SSH-Connection nach 35.207.97.31
* cd /home/jan\_eggers\_hr\_de/obwahl
  + Unterverzeichnis index
  + md <wahlname> (z.B. md obwahl\_of\_2023)
* Konfigurationsdateien:
  + Zentrale Konfigurationsdatei
  + Datei mit den Kandidaten
  + Datei mit den Stadtteilen und Zentrierpunkten
  + Datei mit der Zuordnung der Stimmbezirke zu den Stadtteilen

## Daten besorgen

Kontakt mit dem Wahlamt der betroffenen Kommune aufnehmen. In Frankfurt geht das direkt – in anderen Gemeinden schiebt sich in der Regel die Pressestelle dazwischen. Klarstellen, dass es um technischen Support geht, nicht um inhaltliche Auskünfte.

Das Wahlamt muss folgende Informationen liefern:

* Wird die Wahl mit dem „Votemanager“ der ekom21 ausgezählt? (Nur dann ist die automatisierte Live-Auswertung der Schnellmeldungen möglich.)
* Unter welcher URL ist die CSV-Datei mit den Schnellmeldungen nach Stimmbezirk abrufbar?
* Einen Shapefile der Kommune – mit den Stadtteilen
* Eine Zuordnung der Stimmbezirke zu den Stadtteilen
* Einen Stimmzettel
* Hilfreich: Eine leere CSV-Beispieldatei

Wenn es keine leere CSV-Beispieldatei gibt: Eine Datei einer vergangenen Wahl nehmen – und umbauen. (Bei den Kommunalwahlen z.B.: Ausländerbeiräte-Stimmbezirke rausnehmen, Anzahl der Stimmen-Spalten D1, D2, D3… an die Kandidatenzahl anpassen)

## Shapefiles vorbereiten

„Shapefiles“ sind Dateien mit den Umrissen der Stadtteile/Stimmbezirke mit zugehörigen Geokoordinaten – im Idealfall enthalten sie alle Informationen, die du brauchst: Orte auf der Karte, Namen und Nummern der Stadtteile, Zuordnungen.

Der Datawrapper kann diese Daten verarbeiten, wenn man sie ins richtige Format gebracht hat – es gibt nämlich für Kartendateien nicht nur unterschiedliche Dateiformate, sondern auch unterschiedliche Koordinaten-Bezugssysteme. (Dran denken: die Erde ist rund, eine Karte ist flach – irgendwie muss man das miteinander verrechnen.) Unser Ziel:

* Eine GeoJSON-Datei
* Das Koordinatensystem EPSG:4326 (WGS 84)

Das Hin- und Herformatieren kann man mit einem Open-Source-Karteneditor namens QGIS erledigen (Download unter [www.qgis.org](http://www.qgis.org) ). Es ist etwas unübersichtlich, aber foge einfach diesen Schritten, um zum gewünschten Ergebnis zu kommen.

### Shapefile in QGIS importieren

Die Shapefile-Datei kommt in der Regel als eine ZIP-Datei, die eine Reihe von Dateien enthält, darunter eine mit der Dateiendung .shp

* Datei in einen Ordner entpacken
* Doppelklickl auf die .shp-Datei im Ordner (startet QGIS)

### GEOJSON im richtigen Koordinatensystem erstellen

* Rechtsklick auf den Layer; Export/Objekte speichern als…
* Format GeoJSON, KBS (Koordinatensystem): EPSG:4326 - WGS 84
* Exportieren

QGIS legt alles, was es exportiert, als Kopie an – also nicht wundern, dass plötzlich ein neuer Layer auftaucht.

* Den alten Layer am besten löschen (Rechtsklick; „Löschen“ wählen

### Daten bearbeiten

* Den neuen Layer einmal anklicken
* F6 drücken – ein Fenster mit den „Attributen“ geht auf, also: eine Liste aller Gebiete mit den zugehörigen Daten. Wer mag, kann sich das Fenster durch einen Klick auf das „Docken“-Symbol in der oberen Symbolleiste ganz rechts ins Hauptfenster holen.
* Das Bleistift-Symbol ganz links in der Attribute-Symbolleiste anklicken und so die Bearbeitung aktivieren
* Wenn es schon Felder gibt, die für jeden Stimmbezirk angeben, wo er dazugehört: prima. Wenn nicht: ein neues Feld generieren

Stadtteile generieren

* Menü “Vektor”, “Geometrieverarbeitungswerkzeuge”, “Auflösen” - und dann in der Dialogbox auswählen “Felder auflösen [optional]”, und dann die Attribute hinzufügen, nach denen zusammengeführt werden soll.
* Nach den Attributen schauen - die Ortsteilnr. ist ein String, kein Integer!
* Rechtsklick auf den neuen Layer; Eigenschaften…, dann den “Felder…”-Editor, da oben auf das kleine Abakus-Symbol klicken,
* Namen für das neue Feld in Ausgabefeldname (z.B. “nr”), dann in Feld Ausdruck eintragen”to\_int(Ortsbezirk)” - und OK klicken.
* Neues Feld wird angelegt.
* Dann das alte Feld löschen (auswählen, oben Klick auf Löschen-Feld).

### Export des neuen Stadtteil-Layers

* Rechtsklick auf den Layer; Exportieren als GEOJSON - nicht vergessen, das Bezugssystem auf WGS84 umzustellen!
* Rechtsklick auf den Layer; Export als XLSX - ggf. Geo-Attribute abschalten

### Mittelpunkte der Stadtteile

* Menü “Vektor”, “Geometrie-Werkzeuge”, “Zentroide”
* Dann noch Geokoordinaten der Zentroidpunkte: Rechte Seite die Toolbox, dort “Vektortabelle” aufklappen, “X/Y-Felder zu Attributen hinzufügen”
* Rechtsklick auf den neu erzeugten Layer, exportieren als XLSX bzw CSV

CSV-/XLSX-Dateien putzen

Brauchen eine Stadtteil-Datei mit nr,name,lon,lat (erzeugt aus den Zentroiden)

Brauchen einen Wahlbezirks-Zuordnung

Reparatur der Darmstadt-Karte

Laden (falsche Geometrie - das erst zum Schluss fixen!)

Mapshaper.org

Vereinfachen: Fläche

Auflösen

Löcher löschen

## Datawrapper vorbereiten

Zunächst müssen die Datawrapper-Grafiken angelegt werden – vier für die Ergebnisdarstellung insgesamt und nach Stadtteil, Dieser Schritt passiert im Augenblick noch von Hand – die Konfiguration der Grafiken, also die Anpassung der Metadaten für die jeweilige Wahl, erfolgt automatisch.

* + Ergebnis Top x <https://www.datawrapper.de/_/QP1bJ/>
  + Alle Stimmen für alle Kandidaten https://www.datawrapper.de/\_/rTsdq/
  + Symbol-Karte Sieger Stadtteile <https://www.datawrapper.de/_/l5K3r/> - Shapefile hochladen!
  + Choropleth-Karte Ergebnis Kandidat <https://www.datawrapper.de/_/Ka71G/> - Shapefile hochladen, Kopien für jede/n Kandidat/in!
  + Tabelle Stärkste/Schwächste Stadtteile je Kandidat <https://www.datawrapper.de/_/floHv/>
  + Tabelle alle Ergebnisse nach Stadtteil <https://www.datawrapper.de/_/vSNkO/>

## Indexdateien anpassen

Zunächst müssen die Datawrapper-Grafiken angelegt wer

|  |  |
| --- | --- |
| **name** | **value (Erklärung)** |
| wahl\_name | Name der Wahl, z.B. ob\_ffm\_2023 |
| stimmbezirke\_url | Die Votemanager-URL, unter der eine CSV-Datei mit den Ergebnissen und dem Meldungsstand nach Stimmbezirk abrufbar ist. (Beim Testen auf eine eigene URL oder einen Dateipfad umbiegen) |
| wahlberechtigt | *derzeit nicht benötigt* |
| briefwahl | *derzeit nicht benötigt* |
| kandidaten\_fname | Dateiname der Kandidaten-Datei (s.u.) |
| zuordnung\_fname | Dateiname der Stimmbezirks-Zuordnungs-Datei (s.u.) |
| stadtteile\_fname | Dateiname der Stadtteil-Datei (s.u.) |
| startdatum | Zeitpunkt, ab dem die Wahlauszählung laufen soll |
| top | Anzahl der führenden Kandidatinnen, die die erste Säulengrafik und die Anzahl der Ergebnis Balken in den Tooltipps der Karten steuert |
| top\_id | Datawrapper-ID für die Säulengrafik für das derzeitige Ergebnis der führenden (top) Kandidatinnen und Kandidaten |
| karte\_sieger\_id | Datawrapper-ID der Symbolkarte mit den Siegern nach Stadtteil |
| karte\_kand1\_id | Die Datawrapper-IDs für die Choropleth-Karten der Kandidatinnen; für jede eine Karte - nummeriert nach der Reihenfolge auf dem Wahlzettel (wie in der Indexdatei kandidaten.xlsx hinterlegt) |
| …karte\_kandn\_id |  |
| tabelle\_alle\_id | Die Datawrapper-ID für die Balkengrafik mit allen Kandidierenden - die technisch eigentlich eine Tabelle ist und deswegen tabelle\_alle\_id heißt |
| hochburgen\_id | Die Datawrapper-ID für die Tabelle mit den besten und schlechtesten Stadtteilergebnissen nach Kandidat |
| tabelle\_stadtteile\_id | Die Datawrapper-ID für die Tabelle mit den Gesamtergebnissen |
| social1\_id | Für Social Media: ID der Top-Säulengrafik |
| social2\_id | Für Social Media: ID einer Kopie der Gesamt-Tabelle/Balkengrafik |
| stadtteil\_str |  |
| obwahl\_q\_name | Quelle, die in Datawrapper-Grafiken genannt sein soll |
| obwahl\_q\_url | Quellen-Link für Datawrapper |

Tabelle ```index/wahlname/kandidaten.csv``` (also z.B. index/obwahl\_of\_2023/kandidaten.csv) enthält folgende Spalten:

* Nummer: laufende Nr. des Kandidierenden nach Wahlzettel als ID
* Vorname
* Name
* Parteikürzel | Kurzform der Partei des Kandidaten (z.B. "PARTEI" statt "Partei für Arbeit, Rechtsstaat, Tierschutz, Elitenförderung und basisdemokratische Initiative"")
* Partei | Vollständiger Parteiname (derzeit nicht verwendet)
* Farbwert | Die Kampagnenfarbe bzw. die Farbe für die Darstellungen des Kandidierenden als Hex-RGB-String, also z.B. "#B92837"
* URL | Verlinkung auf den Hintergrundartikel zum Kandidaten (derzeit nicht verwendet)

Also zum Beispiel:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nummer | Vorname | Name | Parteikürzel | Partei | Farbwert | URL |
| 1 | Karl | Napp | USPD | Unabh. | #ff0000 |  |

`index/wahlname/stadtteile.xlsx```- Die Stadtteil-Datei kann man aus QGIS exportieren, wenn man das Shapefile erstellt (CSV oder XLSX):

* nr | Laufende Nummer, ID des Stadtteils
* name | Name des Stadtteils (dient auch als ID, also auf Tippfehler achten!)
* lon | Längengrad des Zentrierpunkts für den Stadtteil
* lat | Breitengrad des Zentrierpunkts für den Stadtteil

Zum Beispiel:

|  |  |  |  |
| --- | --- | --- | --- |
| nr | name | lon | lat |
| |  |  |  |  | | --- | --- | --- | --- | | 1 |  |  |  | | Briefwahl | 8,809636278 | 50,05723721 |

```index/wahlname/zuordnung\_wahllokale.csv```- Die Stimmbezirks-Datei enthält die Zuordnungen für die Wahlbezirke zu Stadtteilen und wird aus der Open-Data-Beispieldatei des votemanagers erstellt:

* nr | ID des Stimmbezirks
* ortsteilnr | ID des Stadtteils
* ortsteil | Name des Stadtteils

Nicht benötigte Spalten können in der Tabelle bleiben, sollten aber möglichst nicht "name" oder so heißen.

Zum Beispiel:

|  |  |  |  |
| --- | --- | --- | --- |
| nr | Wahllokal | ortsteil | ortsteilnr |
| 1 | Gewerblich-technische Schule | Wahllokal Gewerblich-technische Schule | 17 |
| 2 | Gewerblich-technische Schule | Wahllokal Gewerblich-technische Schule | 17 |
| 3 | Gewerblich-technische Schule | Wahllokal Gewerblich-technische Schule | 17 |
| 4 | Eichendorffschule | Wahllokal Eichendorffschule | 14 |

## Grafik-Anpassung generieren

* Config.csv und convig
* Alles auf den Server laden
* Auf dem Server ausführen:   
  Rscript main\_oneshot.R --TEST --DO\_PREPARE\_MAPS wahl\_name=<wahlname>
  + TEST sorgt für Logging onscreen statt in die Log-Datei; config\_test.csv statt config.csv (so kann man andere Quelle testen)
  + DO\_PREPARE\_MAPS generiert die Metadaten in den Karten
  + Falls Verzeichnis index/<wahlname> existiert, sucht das Programm darin nach config.csv bzw. config\_test.csv. Falls das Verzeichnis nicht existiert, sucht es einfach im Verzeichnis index.
* Metadaten: Quelle, Link

## TODO

* Testdaten generieren – dafür gibt es ein Skript im Ordner R/Vorlagen, das nur eine leere Datei (also eine ohne Stimmen) benötigt, wie es sie vor Beginn der Auszählung auf den Votemanager-Seiten gibt – allerdings muss in dem Skript an einer Stelle die Anzahl der Spalten von Hand angepasst werden. Sorry. („D1+D2+D3+D4…“ um die Anzahl von gewünschten Spalten ergänzen)
* Dann kann mit einer config\_test.csv getestet werden, die sich die Daten unter der url „testdaten/test.csv“ zieht – und die Testdateien kopiert man nacheinander von Hand über test.csv drüber und beobachtet, ob das Programm alles richtig aktualisiert.