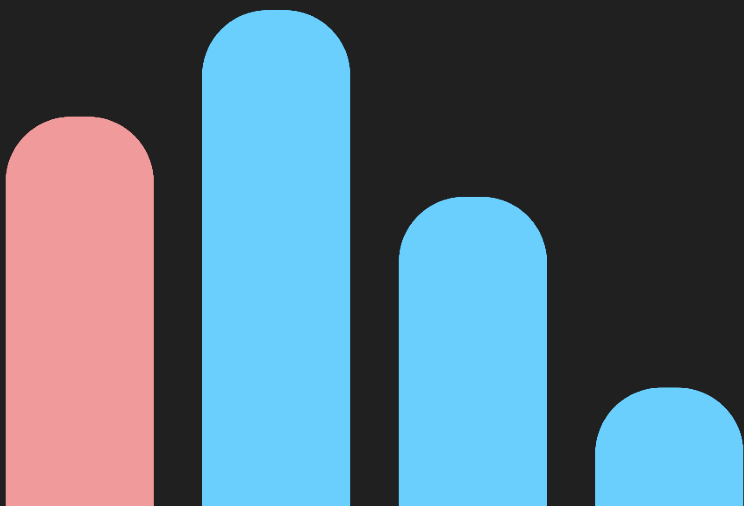


ElectionPredict

Dokumentation und Benutzungsanleitung



1. Dokumentation

1.1 Einstieg

Die App wurde im Xamarin Framework geschrieben, wodurch sich einige Eigenschaften geben. Alle Seiten in der App bestehen aus einem XAML-Dokument, in welchem die Grundstruktur angegeben wird, sowie einem .cs-Dokument im Hintergrund, welches die Funktionalität im Hintergrund kontrolliert. Weitere Klassen befinden sich in separaten .cs-Dokumenten. Das ganze Projekt ist in vier Unter-Projekte eingeteilt. Im zentralen «ElectionPredictFinal»-Subprojekt befindet sich der meiste relevante Code. Desweiteren ist auch «ElectionPredictFinal.iOS» von Interesse; hier befinden sich die für iOS relevanten Anpassungen, welche allerdings jeweils nur für Design eingesetzt wurden. Standardmässig ist das Dokument «App.xaml.cs» unter «ElectionPredictFinal» von Interesse. Dieses bildet den Einstieg in die App.

1.1.1 App.xaml und App.xaml.cs

Diese Dokument besteht aus einer einzigen relevanten Funktion, der Funktion, welche das App-Objekt initialisiert. Hier wird als MainPage, welche standardmässig angezeigt wird, eine neue AppShellPage initialisiert. Diese bildet das Grundgerüst der App.

1.1.2 AppShellPage.xaml und AppShellPage.xaml.cs

AppShellPage ist ein Standardelement und sorgt für die Navigationsfunktionalität. In der Initialisierungsfunktion werden alle drei funktionsspezifischen initialisiert und den drei jeweiligen Tabs zugewiesen. Diese drei Tabs, sowie ihre Icons, werden im .xaml-Dokument definiert. Die drei Seiten sind: PredictionModel, PartyInfluence und DataViewer.

1.2 PredictionModel.xaml.cs

PredictionModel besitzt einige auf der Seite globale Variablen:

Referendums: Eine Liste aller Abstimmungen. Die Klasse Referendum.cs wird separat erklärt.

Cantons: Eine Liste aller Kantone. Die Klasse Canton.cs wird separat erklärt.

LabelStackLink: Ein Dictionary, welches verwendet wird, um die einklappbaren Menus auf der rechten Seite zu kontrollieren, indem die Titel den jeweiligen Sub-Elementen zugeordnet werden.

PartySwitchStateLink: Ein Dictionary, welches die Party-Klasse, welche die Partei angibt und unter Party.cs erklärt wird, und den zugehörigen Parolen-Auswahl-Knopf verbindet. ThreeWaySwitch ist unter Controls.cs abgelegt.

areas: Auswahl-Element für den Politikbereich, der Typ ThreeLevelSelection befindet sich in Controls.cs, areas wird unter LoadAreaSelector() initialisiert.

type: Auswahl-Element für die Rechtsform, der Typ Selection befindet sich in Controls.cs, type wird unter LoadTypeSelector() initialisiert.

year: Auswahl-Element für das Jahr, der Typ Selection befindet sich in Controls.cs, year wird unter LoadYearSelector() initialisiert.

Simulation: Die `SimResults` berechnet die Resultate und macht sie abrufbar. Ausführliche Erklärung unter `SimResults.cs`.

Die erste Funktion, welche ausgelöst wird, ist `LoadAll()`, welche sich um das initialisieren der wichtigsten Elemente kümmert.

1.2.1 LoadAll()

Startet drei weitere Funktionen, `LoadSideBar()`, `LoadReferendums()` und `LoadCantons()`.

1.2.2 LoadSideBar()

Kümmert sich um die Initialisierung und ruft die vier Funktionen auf, welche die einzelnen Selektionselemente und deren Titel auf der rechten Seitenleiste generieren. Diese sind: `LoadYearSelector()`, `LoadTypeSelector()`, `LoadAreaSelector()` und `LoadParties()`. Zuletzt fügt diese Funktion den Hinweis auf `Swissvotes` auf der Seitenleiste ein.

1.2.3 LoadYearSelector()

Generiert eine Liste aller relevanten Jahre, erstellt den Titel und erstellt dessen Klick-Funktion `SubTitle_Tapped()`. Das `Selection-Element` (`year`), welches mit diesen Jahren initialisiert wird, wird schliesslich, zusammen mit dem Titel dem `SelectionStackLayout`, dem Behälter für alle Seitenleiste-Elemente, hinzugefügt.

1.2.4 LoadTypeSelector()

Liest aus `Rechtsform.txt` alle Rechtsformen aus und fügt sie in ein `Selection-Element` (`type`) ein. Erstellt den Titel und erstellt dessen Klick-Funktion `SubTitle_Tapped()`. Daraufhin werden die Elemente zu `SelectionStackLayout` hinzugefügt.

1.2.5 LoadAreaSelector()

Lädt das dreistufige Auswahl-Element (`areas`) für die Politikbereiche anhand von `Politikbereiche.txt`, erstellt den zu dem Element gehörenden Titel und erstellt dessen Klick-Funktion `SubTitle_Tapped()`. Daraufhin werden die Elemente zu `SelectionStackLayout` hinzugefügt.

1.2.6 LoadParties()

Lädt alle Parteien und deren Eigenschaften anhand von `modelparties.txt` und der Funktionalität von `Party.cs`, und erstellt für jede ein Steuerelement auf rechten Seite. Für jede Partei werden `Parteiename`, `Kürzel` und `Logo` geladen und ein `Dreifachauswahlelement` erstellt. Dieser `ThreeWaySwitch` erlaubt die Auswahl der Parole. Das `Party-Element` und der `ThreeWaySwitch` werden mit `PartySwitchStateLink` verknüpft. Zusammen mit dem Titel und seiner Klick-Funktion `SubTitle_Tapped()` werden die Elemente zu `SelectionStackLayout` hinzugefügt.

1.2.7 LoadReferendums()

Liest alle Daten aus `modelbasedata.tsv`, erstellt daraus eine Instanz der `Referendum-Klasse` und speichert sie in der Liste `Referendums`.

1.2.8 LoadCantons()

Liest alle Daten aus `Cantons.tsv`, formatiert jede Zeile in die einzelnen Datenpunkte und erstellt mit diesen Daten die jeweiligen Instanzen der `Canton-Klasse` und speichert diese unter `Cantons`. Ausserdem werden jedem Kanton alle Referenden hinzugefügt, wobei die jeweiligen früheren Abstimmungsergebnisse des Kantons gespeichert werden.

1.2.9 SubTitle_Tapped()

Wird durch Antippen eines Titels auf der rechten Seitenleiste ausgelöst. Durch LabelStackLink, in welchem durch LoadYearSelector(), LoadTypeSelector(), LoadAreaSelector() und LoadParties() die jeweiligen Titel und Steuerelemente abgelegt wurden, kann das entsprechende Steuerelement gefunden werden. Entsprechend des aktuellen Status des Titels (ein- oder ausgeklappt) wird der Titel verändert und die Sichtbarkeit des Steuerelements angepasst.

1.2.10 Button_Clicked()

Diese Funktion wird durch Anklicken des «Berechnen»-Knopfes auf der rechten Seitenleiste ausgelöst. Überprüft die Zustände aller Steuerelemente und gibt Fehlermeldungen zurück, falls eines unzureichend angewählt wurde. Sind alle Auswahlen gemacht worden, wird für jedes Element von Referendums dessen individuelle Ähnlichkeit (entsprechend Referendum.cs). Daraufhin wird die Liste Referendums an jeden Kanton gegeben, sodass jeder Kanton anhand seiner individuellen Resultate eine Normalverteilung der zu erwartenden Resultate erstellen kann. Diese Funktionalität wird unter Canton.cs erklärt. Zuletzt löst die Funktion MonteCarloSimulation() aus.

1.2.11 MonteCarloSimulation()

Grösstenteils visuell. Erstellt die SimResults-Instanz anhand der fertiggestellten Cantons-Liste und führt die entsprechend Funktionen aus, um die Resultate zu generieren. Diese Funktionen werden unter SimResults.cs erklärt. Ausserdem löscht die Funktion jegliche vorherige Elemente von vorherigen Simulationen auf der linken Display-Hälfte, dem ResultsStack. Es werden dem Fortschritt der SimResults-Funktionen entsprechende Statusmeldungen im LoadingLabel ausgegeben und zeitweise wird auch ein Fortschrittsbalken eingeblendet. Zuletzt wird CreateResultsStack() ausgelöst, welches die Resultate visualisiert.

1.2.12 CreateResultsStack()

Erstellt alle Element im ResultsStack, welche zur Anzeige des Resultates benutzt werden. Dies sind der Titel, die Verteilungs-Graphik anhand der Graph-Instanz (unter Controls.cs), welche anhand der Verteilung aus SimResults und der Angabe der Unterteilungen sowie des Titels erstellt wird. Die Informationen in Text-Form werden direkt aus SimResults ausgelesen, mit TopRefs() wird eine Liste der ähnlichsten vorherigen Abstimmungen erstellt und zurückgegeben. Zuletzt folgt die Liste der Kantone. Für jedes Element aus Cantons werden dessen Eckdaten aus der jeweiligen Canton-Instanz ausgelesen. Desweiteren wird eine Karte des Kantons mit CantonRender (unter CantonRender.cs) erstellt.

1.2.13 TopRefs()

Geht die Liste Referendums durch und erstellt eine neue Liste mit den fünf Referendum-Instanzen mit der höchsten Ähnlichkeit (.similarity-Wert, welche unter Button_Clicked berechnet wurden). Diese wird zurückgegeben.

1.3 Canton.cs

Die Canton-Klasse dient als Behälter der kantonalen Daten, deren Abstimmungsergebnisse usw. Die Klasse besitzt einige globale Variablen:

myshorthand: Das Kantons-Kürzel, bspw. TG im Thurgau.

myname: Der Name des Kantons.

mymean: Der Ja-Stimmen-Prozentsatz-Mittelwert, berechnet anhand der Ähnlichkeiten der einzelnen Abstimmungen (Referendum-Instanzen). Immer zwischen 0 und 1.

mystdev: Die Standardabweichung des Ja-Stimmen-Prozentsatzes, in Dezimalschreibweise, berechnet anhand der einzelnen Abstimmungen (Referendum-Instanzen).

mymeanvotes: Die durchschnittliche Anzahl der Stimmen, die erwartete Anzahl Wählerstimmen, anhand der Referendum-Instanzen.

myyesvotes: Liste aller früheren Abstimmungen. Index ist die Nummer der Abstimmung, gleich derer, welche für diese Abstimmung in der entsprechenden Referendum-Klasse abgelegt ist. Die zweite Zahl ist die Anzahl zu einer Abstimmung gehörenden Ja-Stimmen.

mynovotes: Liste aller früheren Abstimmungen. Index ist die Nummer der Abstimmung, gleich derer, welche für diese Abstimmung in der entsprechenden Referendum-Klasse abgelegt ist. Die zweite Zahl ist die Anzahl zu einer Abstimmung gehörenden Nein-Stimmen.

myweight: Anzahl der Ständestimmen des Kantons.

Die zugehörigen Eigenschaften mit `get-` und `set-`Zugriffsoptionen sollten selbsterklärend sein. Erstellt wird `Canton.cs` durch die `Canton()`-Funktion, wobei `myshorthand`, `myname` und `myweight` zugewiesen werden.

1.3.1 AddReferendum()

Fügt dem Kanton eine Abstimmung hinzu; die Resultate der Abstimmung werden in `myyesvotes` und `mynovotes` abgelegt.

1.3.2 ReferendumDistribution()

Berechnet die Verteilung der Vorhersage des Kantons anhand der Ähnlichkeiten der Kantone. Die individuellen Ähnlichkeiten werden ausserdem in `myweights` abgelegt. Berechnet werden `myn`, welches die Summe aller Referendum-Ähnlichkeiten ist. Während jede Referendum-Instanz durchgegangen wird, wird jeweils dessen Ja-Stimmen-Anteil berechnet. Dieser wird mit der Ähnlichkeit der Abstimmung multipliziert. Aus der Summe dessen kann dann der Erwartungswert berechnet werden und unter `mymean` abgelegt werden. In einem zweiten Durchgang kann auch noch die Standardabweichung berechnet und unter `mystdev` abgelegt werden. `mymeanvotes` wird berechnet, indem der gewichtete Durchschnitt aller Ja-Stimmen-Anteile berechnet wird.

1.3.3 CantonCovariance()

Berechnet anhand der Formel für die nicht-gewichtete Kovarianz die Kovarianz zweier Kantone. Geht durch die Formel und gibt das Resultat zurück.

1.4 Referendum.cs

Die Referendum-Klasse ist der Behälter für eine Volksabstimmung und kann deren Ähnlichkeit zu einer anderen Abstimmung berechnen. Sie verfügt über folgende Variablen, welche ausser den vier letzten beim Aufruf von `Referendum()` aus dem Input-String generiert werden:

myindex: Zahl, welche die Nummer der Abstimmung gemäss `Swissvotes-Datensatz` mal zehn wiedergibt. Der Faktor wird genutzt, damit jede Nummer aus dem Datensatz ganzzahlig ist.

myyear: Zahl, welche das Jahr der Abstimmung angibt (1866 - heute).

mytitle: Der deutschsprachige Titel der Abstimmungsvorlage.

mytype: Gibt die Rechtsform der Abstimmung an, gemäss Code von Swissvotes (1. Obligatorisches Referendum, 2. Fakultatives Referendum, 3. Volksinitiative, 4. Direkter Gegenentwurf, 5. Stichfrage).

myarea: Gibt den Politikbereich gemäss Nummerierung von Swissvotes an. Da eine Abstimmungsfrage mehrere Politikbereiche betreffen kann, sind diese in einem Array abgelegt.

mycantonvotes: Gibt die Resultate in den jeweiligen Kantonen an, Key ist das Kantonskürzel, das Array beinhaltet unter [0] die Anzahl Ja-Stimmen und unter [1] die Anzahl Nein-Stimmen. Wird nur von der Canton-Klasse unter AddReferendum() benutzt, um die kantonseigenen Daten auszulesen und im Kanton zu speichern.

mypartydecisions: Gibt die Parolen der Parteien an. Key ist das Parteikürzel; die Zahl steht für die Parole gemäss Swissvotes (1. Ja-Parole, 2. Nein-Parole, sowie weitere, welche vom Modell aber allesamt ignoriert werden).

mypartyvotes: In jeder Instanz der Klasse identisch. Aus modelpartyvotes.tsv werden die Abstimmungsergebnisse aller Parteien ausgelesen. Diese werden als Gewichtungen verwendet. Key ist das Parteikürzel, die zurückgegebene Zahl ist der Stimmenanteil (in Dezimal, also bei bspw. 20% Anteil 0.2).

mysimilarity: Speichert die Ähnlichkeiten auf politischer Ebene (Rechtsform und Politikbereich).

mydirection: Speichert die Ähnlichkeit der Partei-Parolen.

myyearcloseness: Speichert die Differenz der Jahre der Vorhersage und myyear.

myweight: Summe der Gewichtungen nach der Aktivierungsfunktion. Wird durch .similarity zurückgegeben.

Die zugehörigen Eigenschaften mit get- und set-Zugriffsoptionen sollten selbsterklärend sein (Achtung: .similarity gibt die Eigenschaft myweight zurück).

1.4.1 CalculateSimilarites()

Berechnet myweight anhand der anzugebenden Parameter. Diese werden mit den Parametern der Referendum-Klasse verglichen. Anfangs wird die politische Ähnlichkeit verglichen. Ist die Rechtsform identisch, wird mysimilarity um 0.270 erhöht. Sind die Politikbereiche (der angegebene und einer der im Array myarea abgelegten) genau identisch, wird mysimilarity um weitere 0.589 erhöht. Zur Ermittlung von mydirection wird jede Partei durchgegangen. Sind die Parolen der beiden Abstimmungen identisch, wird mydirection um den Stimmenanteil der Partei (gemäss mypartyvotes) erhöht, andernfalls wird mydirection um ebendiesen Stimmenanteil verkleinert. Die Summe von mydirection wird mit der Gewichtung 2.593 multipliziert. Zuletzt wird myyearcloseness berechnet. Die Differenz des angegebenen Jahres year und myyear wird berechnet und durch die Differenz des aktuellen Jahres und 1866 geteilt. Diese

Zahl wird mit der Gewichtung 2.180 multipliziert. Die Summe von `mysimilarity`, `mydirection` und `myearcloseness` wird als `x` in die Aktivierungsfunktion

$$y = \frac{1}{1 + e^{-k \cdot (x - x_0)}}$$

eingesetzt, wobei $k = 6.113$ und $x_0 = 8.225$ beträgt. Diese Berechnung wird in `myweight` gespeichert und kann mit dem Zugriff auf `.similarity` abgerufen werden.

1.5 CantonRender.cs

Die Klasse `CantonRender` wird genutzt, um die Karte eines Kantones mit entsprechender Färbung anzuzeigen. Benutzt wird die Klasse unter `PredicitionModel.xaml.cs`, um bei den Resultaten unter dem Abschnitt Kantone eine Karte des jeweiligen Kantons anzuzeigen. Die Klasse besitzt einige Variablen:

`sksvg`: Element der `SkiaSharp`-Erweiterung und Behälter für die im SVG-Format abgelegte Karte.

`mysvgdata`: Text, welcher den Text des SVG-Dokuments beinhaltet, damit dieser bearbeitet werden kann.

`mypanel`: Element, welches das fertiggestellte Bild anzeigt und auf Anfrage zurückgegeben wird.

Mit Aufruf der `CantonRender()`-Funktion muss ein Kanton (Instanz einer `Canton`-Klasse) angegeben werden. In einem Dictionary sind die Farben und deren zugehörigen Werte angegeben; der Kanton wird entsprechend seiner Wahrscheinlichkeit, ein Stände-Ja abzugeben bewertet. Im Ordner `CantonSVG` sind die Quelldokumente für alle Kantone abgelegt. Diese Dokumente werden aufgerufen und in `mysvgdata` abgelegt. Daraufhin wird das Dictionary durchgegangen und anhand der in der `Canton`-Klasse abgelegten Verteilung (`.distribution`) wird bestimmt, welche Farbe der Kanton erhält. Diese Farbe wird in `mysvgdata` eingesetzt, welches in ein `ByteArray` und daraufhin in einen Stream umgewandelt wird, aus welchem dann `sksvg` erstellt werden kann. Zu `mypanel` wird die Funktion `mappanel_PaintSurface()` hinzugefügt, welche `sksvg` auf `mypanel` zeichnet. Durch `.map` kann auf `mypanel` zugegriffen werden.

1.6 Graph in Controls.cs

Die Klasse `Graph` erstellt das Diagramm der Normalverteilung, welche als Resultat der Modellberechnungen generiert wird. Genutzt werden einige Variablen:

`mydist`: Normalverteilung, aufgrund welcher die Grafik generiert wird.

`mymaindict`: Für jeden Balken wird dessen Höhe hinterlegt.

`mystep`: Die Schrittgrösse in Prozent, 0.05 würde Balken generieren, welche jeweils 5% Prozent der Verteilung beinhalten, also gesamt 20 Balken.

`nocolor`: Farbe für Balken, deren Resultate unter 50% liegen, also abgelehnt werden würden.

`yescolor`: Farbe für Balken, deren Resultate über 50% liegen, also angenommen werden würden.

`mytitle`: Titel des Diagramms

`mydist` wird bereits bei Aufruf von `Graph()` gesetzt, wie auch `mystep` und `mytitle`. Ausserdem wird `mymaindict` generiert. Bei Zugriff auf `Stack` wird die Grafik generiert. Die maximale Höhe wird durch die Ermittlung des grössten Werts im `mymaindict` gefunden. Daraufhin wird für jedes Element ein Balken erstellt und entsprechend seiner Position (über oder unter 50%) koloriert. Auch die Beschriftungen werden schrittweise generiert. Dazu kommen weitere Design-Elemente wie bspw. der Titel. Schliesslich werden alle Elemente in `wrapStack` verpackt und zurückgegeben.

1.7a Selection in Controls.cs

Die `Selection`-Klasse beinhaltet ein Auswahl-Element. Hierfür benötigt sie einige Variablen:

`myactivestring`: Beinhaltet den Text der aktuellen Auswahl.

`myoptions`: Liste aller möglichen Optionen.

`myFrameList`: Liste aller auswählbaren Elemente in der Liste. Jedes Element in der Liste beinhaltet den einen Text aus `myoptions`.

`myFrame`: Äusserer Behälter für das Auswahl-Element.

`myMainStack`: Äusserster Behälter, in welchem sich alle anderen Elemente befinden.

`myScrollView`: Scroll-Behälter, welcher das Scrolling ermöglicht.

`myStackLayout`: Inhalt von `myScrollView`. In `myStackLayout` befinden sich die einzelnen Optionen aus `myFrameList`.

Die Klasse kann über zwei Optionen generiert werden. Bei einer wird als Argument eine Liste angegeben, wobei für jedes Element eine Auswahloption erstellt wird. Bei der anderen wird ein Dokument angegeben, wobei jede Zeile dieses Dokuments zu einer Auswahloption wird. Für jede Option wird ein `StackLayout` erstellt, welches ein Label mit dem Text beinhaltet. Dieses wird `myStackLayout` und `myFrameList` hinzugefügt. Ausserdem erhält es eine Antipp-Erkennung. Wird das Element angewählt, wird die Funktion `LabelTap` ausgeführt, welche die Einfärbung des Elements übernimmt, alle anderen Elemente auf ihre Ausgangsfarbe zurückfärbt und `myactivestring` verändert. Die zugehörigen Eigenschaften mit `get`- und `set`-Zugriffsoptionen sollten selbsterklärend sein, einzig `.selectedIndex` ist nicht offensichtlich. Es wird der Index von `myactivestring` in `myoptions` zurückgegeben (Um eins erhöht, da der `Swissvotes`-Datensatz seine Nummerierungen mit 0 beginnt).

1.7b ThreeLevelSelection in Controls.cs

Diese Klasse ist der `Selection`-Klasse ähnlich, wurde aber speziell darauf ausgelegt, für die Auswahl des Politikbereichs benutzt zu werden. Die Variablen sind ähnlich:

`myselectedarea`: Beinhaltet den Politikbereich.

`LabelAreaLink`: Verbindet das Element mit dem zugehörigen Text.

`AreaLabelLink`: Umkehrung von `LabelAreaLink`, alle Paare sind identisch.

`myMainStack`: Äusserster Behälter für alle Elemente.

`myTopLevelStack`, `mySecondLevelStack`, `myThirdLevelStack`: Für alle Ebenen die Behälter mit den jeweiligen auswählbaren Elementen.

`myTopLevelScrollView`, `mySecondLevelScrollView`, `myThirdLevelScrollView`: Ermöglicht das Scrollen durch die entsprechenden `StackLayout`, welche sich jeweils in diesen `ScrollViews` befinden.

`myTopLevelFrame`, `mySecondLevelFrame`, `myThirdLevelFrame`: Container für die `ScrollViews`. Sind sichtbar als die Rahmen der einzelnen Optionen.

Die `ThreeLevelSelection` wird mit der Angabe eines Textdokuments erstellt. In diesem Dokument befinden sich alle möglichen Politikbereiche. Für alle wird ein auswählbares Element erstellt, welches zusammen mit dem beinhalteten Text `LabelAreaLink` und `AreaLabelLink` hinzugefügt werden. Ausserdem werden die Elemente anwählbar gemacht. Werden sie angetippt, wird die Funktion `LabelTap` ausgeführt. Diese ändert `myselectedarea` und macht gegebenenfalls `mySecondLevelFrame` oder `myThirdLevelFrame` sichtbar. Diese werden dabei mit allen Unterelementen geführt, welche bereits generiert wurden und über die beiden Dictionaries adressierbar sind. Die Funktion `PadSelection` wird zur Ausgabe des von `myselectedarea` genutzt, um gegebenenfalls Nullen anzufügen, wodurch jeder String drei Teilgebiete erhält.

1.8 ThreeWaySwitch in Controls.cs

Fast identisch mit `Selection`, nur das Layout ist ein anderes und die Auswahl ist auf nicht-scrollbare Elemente begrenzt. Folgende Variablen werden benutzt:

`myactivestate`: Die ausgewählte Option.

`myreport`: Genutzt für `PartyInfluence.xaml.cs`. Falls aktiv wird bei Änderung von `myactivestate` ein Signal an `PartyInfluence` gesandt, damit dieses aktualisiert.

`mybgcolor`: Die Hintergrundfarbe.

`myactivecolor`: Die Hintergrundfarbe des ausgewählten Objekts.

`f`: Äusserer Rahmen des Objekts, welcher bei Anfrage zurückgegeben wird.

`s`: Behälter innerhalb von `f`, welcher die drei Auswahloptionen beinhaltet.

Bei Erstellung der Klasse müssen die Optionen angegeben werden. Ausserdem ist die Partei relevant und es muss angegeben, ob `myreport` wahr sein soll. Für alle Optionen wird ein anklickbares Element geschaffen. Diese werden zu `s` hinzugefügt und erhalten die Funktion `OptionClicked`. Diese Funktion überprüft `myactivestate` und verändert es entsprechend. Ist `myreport` wahr wird die Änderung an `PartyInfluence` gemeldet, wo aufgrund eines Listeners weitere Veränderungen vorgenommen werden. Schliesslich wird `CheckActive()` aufgerufen, welches das angewählte Element einfärbt.

1.9 Party.cs

Die `Party`-Klasse ist der Behälter für eine Partei, welche einige Variablen besitzt:

`mymaindict`: Beinhaltet Instanzen der `Vote`-Klasse. Diese bildet jeweils eine Abstimmung ab, ist aber spezifisch für eine Instanz der `Party`-Klasse. Der Key ist jeweils der Index der Abstimmung.

`mypartyname`: Voller Name der Partei.

`mypartyshorthand`: Abkürzung für den Parteinamen.

Bei Erstellung der `Party`-Instanz muss ein String angegeben werden. Dieser ist einerseits der Parteiname, andererseits auch der Dateiname eines zugehörigen Dokuments, in welchem die Daten für die Erstellung aller relevanten `Vote`-Klasse-Instanzen vorhanden sind. Auf der ersten Zeil des Dokuments befindet sich ausserdem das Parteikürzel. Die zugehörigen Eigenschaften mit `get`- und `set`-Zugriffsoptionen sind meist selbsterklärend. Es können auf zwei Arten Instanzen der `Vote`-Klasse abgefragt werden, welche sich innerhalb von `mymaindict` befinden. Einerseits durch deren Index; eine Liste wird in `RequestVotes()` angegeben, und es kommt eine Liste mit den entsprechenden Instanzen zurück. Es kann auch eine Parole angegeben werden, und es kommen alle `Vote`-Instanzen zurück, in welchen die Partei ebendiese Parole vorgeschlagen hat.

1.10 `SimResults.cs`

Diese Klasse ist der Behälter für die simulierten Resultate einer Abstimmung. Diese Simulationen funktionieren über eine multivariate Normalverteilung, eine Simulation wird durch den Aufruf mehrerer Funktionen ausgeführt. Die benutzten Variablen sind:

`mybaselist`: Liste aller Instanzen der `Canton`-Klasse.

`mymeanvector`: Ein Array aus `doubles`, welches den Mittelwertvektor repräsentiert. Die Länge von `mymeanvector` entspricht der von `mybaselist`, da jeder Kanton einen Mittelwert beiträgt.

`mycovariancematrix`: Ein zweidimensionales Array aus `doubles`, welches die Kovarianzenmatrix repräsentiert. In beiden Dimensionen hat diese Matrix die Länge von `mybaselist`, da jeder Kanton kombiniert mit jedem Kanton eine Kovarianz erzeugt.

`myStändemehr`: Wahrheitswert, welcher angibt, ob es sich um eine Abstimmung handelt, in welcher das Ständemehr relevant ist. Falls dies so ist, wird das Ständemehr miteingerechnet, wenn das Modell den Wert für den Prozentsatz der Simulationen mit angenommener Vorlage berechnet.

`mytotnationalvotes`: Erwartete Stimmenmenge, Summe aller erwarteten Stimmenmengen der Kantone.

`mydist`: Multivariate Normalverteilung, erzeugt aus `mymeanvector` und `mycovariancematrix`.

`mysamples`: Liste der generierten Simulationen, wobei jedes `double`-Array in der Liste ein Resultat repräsentiert und die einzelnen Werte die erwartete Ja-Stimmen in den jeweiligen Kantonen angeben.

`mymeanpopvotes`: Mittelwert aller Stimmen-Prozentsätze; das vermutete Resultat. Wird über `.distribution` ausgegeben.

`mymeanstände`: Mittelwert der Ja-Stände, die erwartete Anzahl Stände, welche mit Ja stimmen werden.

`mypercentageyes`: Prozentsatz der Simulationen, in welchen die Vorlage angenommen wurde.

`mytotcantonweights`: Summe der Ständestimmen, Anzahl der Stände, mit der aktuellen Konfiguration der Kantone immer 23.

`myvariance`: Standardabweichung aller Stimmen-Prozentsätze, die Unsicherheit des vermuteten Resultats. Wird über `.distribution` ausgegeben.

`mypercentages`: Liste aller Ja-Prozentsätze, wird benötigt, um die Standardabweichung zu berechnen.

Die zugehörigen Eigenschaften mit `get`- und `set`-Zugriffsoptionen sollten selbsterklärend sein. Bei der Erstellung einer `SimResults`-Instanz müssen die Kantone angegeben werden, welche ausgewertet werden sollen. Diese werden in `mybaselist` gespeichert. Ausserdem muss angegeben werden, ob das Ständemehr berücksichtigt werden soll; `myStändemehr` wird festgelegt. Anhand der Länge von `mybaselist` werden auch `mymeanvector` und `mycovariancematrix` generiert.

1.10.1 GenerateStructure()

Der erste Schritt in der Generierung der Simulation. Diese Funktion befüllt `mymeanvector` und `mycovariancematrix` und erzeugt schliesslich daraus `mydist`. Hierfür wird für jedes Element einerseits der Mittelwert der Ja-Stimmen abgerufen, andererseits wird auch ein weiteres Mal alle Canton-Instanzen durchgegangen und für alle die Kovarianz berechnet, welche dann in die Kovarianzenmatrix eingefügt wird. Desweiteren werden `mytotnationalvotes` und `mytotcantonweights` berechnet.

1.10.2 AddSims()

Beim Aufruf dieser Funktion muss eine Zahl angegeben werden, welche angibt, wie viele Simulationen ausgeführt werden sollen. Diese werden zu `mysamples` hinzugefügt. Die Funktion wird mehrere Male aufgerufen, um `mysamples` schrittweise aufzubauen. Somit kann auf einfache Weise eine Fortschrittsanzeige ausgegeben werden (Siehe hierfür `PredictionModel.xaml.cs`).

1.10.3 LoopAndCalc()

Diese Funktion wertet die generierten Simulationen aus. Die Variablen `mymeanpopvotes`, `mymeanstände`, `mypercentageyes` und `myvariance` werden berechnet. Hierfür wird jede Simulation in `mysamples` durchgegangen und ein Total wird ausgerechnet, welches dann anhand der Anzahl Elemente von `mysamples` zu einem Durchschnitt umgerechnet wird. Für die Berechnung der Varianz ist ein zweiter Durchgang aller Elemente in `mysamples` nötig. Alle berechneten Elemente können über `get`-Eigenschaften abgerufen werden.

1.11 Vote.cs

Wie auch `Referendum.cs` ein Container für eine einzelne Vorlage, hier allerdings für `PartyInfluence.xaml.cs`. Bei der Klasse handelt es sich um einen Behälter, welcher für Vergleiche und Berechnungen benutzt wird, selbst aber keine Funktionalität besitzt. Seine gesamte Funktion sind die Variablen, welche ausgegeben werden können:

`myindex`: Nummer der Vorlage, identisch mit der Nummerierung des Swiss-votes-Datensatzes multipliziert mit zehn (Im Original-Datensatz gibt

es Zahlen mit einer Kommastelle, mit der Multiplikation wird jeder Index ganzzahlig).

mytitle: Titel der Vorlage in deutscher Sprache.

myyear: Zahl, welche das Jahr der Vorlage angibt.

mydomain: String, welcher den Politikbereich angibt. Nur der grobe Bereich in Form einer Zahl, entsprechend der Einordnung von Swissvotes.

myadopted: Enthält die Angabe darüber, ob die Vorlage angenommen wurde. Entweder «Angenommen» oder «Abgelehnt».

mypercentageyes: Prozentsatz der Ja-Stimmen; In Prozent mit zwei Nachkommastellen.

mypartystrength: Wähleranteil der Partei zum Zeitpunkt der Abstimmung, berechnet aus linearem Modell zwischen den beiden nächstgelegenen Nationalratswahlen. In einem Wahljahr identisch mit dem Resultat, in einem Jahr in der Mitte zwischen zwei Nationalratswahlen der Durchschnitt der beiden Ergebnisse.

myendorsement: Parole der Partei, entweder «Ja», «Nein» oder «Neutral»

mynumsections: Anzahl Sektionen mit abweichender Parole

mysections: Array der Sektionen mit abweichender Parole.

Bei der Erstellung einer Instanz der Vote-Klasse muss ein String angegeben werden, welcher alle relevanten Informationen enthält. Diese Strings sind in separaten Dokumenten abgelegt. Die Variablen der Klasse können über get-Eigenschaften abgerufen werden. Deren Namensgebung sollte selbsterklärend sein. Einzig .sections macht mehr, als nur eine Variable zurückzugeben; das Array wird in einen String mit passend gesetzten Kommas umgewandelt.

1.12 PartyInfluence.xaml.cs

PartyInfluence ist der zweite Funktionsbereich der App. Der Bereich ist in der Tab-Leiste unter Partei-Parolen abrufbar. Es können diverse Organisationen ausgewählt werden, für welche eine Parole ausgewählt wird. Alle Vorlagen, bei welchen die Organisation diese Parole ergriffen hat, werden dann ausgewertet und einige Kennwerte werden ausgegeben. Hierfür sind einige Variablen nötig:

buttonpartylink: In der rechten Seitenleiste erscheinen die auswählbaren Parteien als einzelne Knöpfe, welche angewählt werden können, um die Organisation zur Berechnung hinzuzufügen. Dieses Dictionary enthält diese Button-Elemente und die zugehörigen Party-Instanzen.

framepartylink: Ist eine Organisation ausgewählt, erscheint auf der linken Bildschirmseite ein grösseres Element, auf welchem die zu analysierende Parole ausgewählt werden kann und die zugehörigen Vorlagen ausgegeben werden. Diese Elemente werden über framepartylink mit der jeweiligen Instanz der Party-Klasse verlinkt.

listpartylink: Auf den Elementen der aktiven Organisationen, welche in framepartylink abgelegt sind, befindet sich ein Textfeld, in welchem

alle zu der ausgewählten Parole passenden Vorlagen ausgegeben werden. `listpartylink` ist die Verknüpfung der Party-Instanz und dieses Labels.

`subtitleorder`: Die Organisationen auf der rechten Displayseite sind in einzelne Gruppen (Parteien, Verbände, usw.) unterteilt. Diese Gruppen haben Titel, welche durch antippen die ihnen zugehörigen Elemente unsichtbar machen können. Durch `subtitleorder` werden diese Titel mit den sich unter ihnen befindlichen Elementen verknüpft.

`activevotes`: Liste der aktiven Parteien und der Listen der ausgewählten Vorlagen. Wenn eine Organisation aktiv ist (also von der Liste auf der rechten Bildschirmseite ausgewählt wurde), wird sie der Liste hinzugefügt. Wird eine zu analysierende Parole ausgewählt, wird die Liste der damit übereinstimmenden Vote-Instanzen dieser Partei zu `activevotes` hinzugefügt. Dieses hinzufügen funktioniert über `ThreewaySwitch`, welcher, wenn die `report-Boolean` wahr ist, auf `activevotes` zugreift und von der zugehörigen Party-Instanz über `RequestVotes()` die entsprechenden Votes-Instanzen anfordert. Bei jeder Änderung von `activevotes` werden die auf der rechten Bildschirmhälfte angegebenen Kennwerte neu berechnet.

`selectableparties`: Liste der Party-Instanzen, welche nicht für die aktuelle Berechnung berücksichtigt werden. Alle Elemente dieser Liste sind auf der linken Bildschirmseite als Buttons vorhanden.

`activeparties`: Instanzen der Party-Klasse, welche für die aktuelle Berechnung berücksichtigt wird. Für jede dieser Parteien ist ein Element mit Auswahlmöglichkeiten auf der rechten Bildschirmseite sichtbar.

`visibleselectableparties`: Teilmenge von `selectableparties`, derer Instanzen der Party-Klasse, deren zugehörigen Buttons auf der linken Bildschirmseite auch sichtbar sind. Ausgeblendet können sie entweder sein, da sich die Partei in der Liste `activeparties` befindet, oder sich durch ein Antippen des zugehörigen Titels unsichtbar gemacht wurde.

Wenn `PartyInfluence` erstellt wird, werden zwei Hauptkomponenten ausgeführt. Einerseits werden mit `LoadParties()` die wichtigsten Elemente initialisiert. Andererseits wird `Activevotes_CollectionChanged` zu `activevotes` hinzugefügt. Die Funktion wird jedes Mal ausgeführt, wenn `activevotes` verändert wird.

1.12.1 LoadParties()

Erst wird das Dokument `partyindex.txt` ausgelesen. Dieses erhält die Namen aller Organisationen sowie auch die Titel von Gruppen von Organisationen. Diese Titel sind im Text-Dokument mit einem \$ gekennzeichnet. Wird ein Titel erkannt, wird dieser als solcher der Seitenauswahl hinzugefügt. Ausserdem erhält er die Funktion `SubTitle_Tapped()`, welche ausgelöst wird, wenn der Titel angetippt wird. Alle auf einen Titel folgenden Organisationen, für welche eine Instanz der Party-Klasse erzeugt wird, werden mit `subtitleorder` zu dem betreffenden Untertitel hinzugefügt. Die erzeugte Party-Instanz wird ausserdem mit `CreateSelectableParty()` benutzt, um einen Knopf auf der linken Seitenleiste zu erstellen. Zuletzt wird die Datenherkunft in einem Textfeld ganz unten auf der Seitenleiste angegeben.

1.12.2 SubTitle_Tapped()

Wird ausgelöst, wenn ein Titel auf der Seitenleiste angetippt wird. Ändert die Sichtbarkeit der zugehörigen Knöpfe und bearbeitet `visibleselectableparties` entsprechend.

1.12.3 CheckSubtitles()

Wenn eine aktive Partei geschlossen wird, überprüft diese Funktion ob der zugehörige Knopf auf der Seitenleiste aktuell, basierend auf dem Titel, sichtbar sein sollte.

1.12.4 CreateSelectableParty()

Diese Funktion erzeugt den Knopf auf der Seitenleiste und fügt die Partei zu `selectableparties` hinzu. Die Funktion `SelectableButtonClicked` wird dem Knopf hinzugefügt.

1.12.5 SelectableButtonClicked()

Beim Aufruf dieser Funktion wird `CreateActiveParty()` ausgelöst, sowie mit `CheckSubtitles()` die Sichtbarkeit aller relevanten Elemente überprüft.

1.12.6 CreateActiveParty()

Diese Funktion macht den Knopf auf der Seitenleiste unsichtbar und erstellt mit `ActivePartyFrame()` das Auswahlelement für eine Organisation auf der rechten Bildschirmseite. Diese Funktion ändert diverse Listen so, dass sie die korrekten Angaben enthalten. Die Party-Instanz wird aus `selectableparties` entfernt und zu `activeparties` hinzugefügt. Das erzeugte Element wird `framepartylink` und zum vordefinierten `StackLayout` für die aktiven Elemente hinzugefügt.

1.12.7 ActivePartyFrame()

Erstellt das grosse Auswahlelement, welches für eine ausgewählte Organisation auf der rechten Bildschirmseite erscheint. Name und Kürzel werden aus der Party-Instanz ausgelesen. Das Parteilogo wird geladen. Ein Knopf zum schliessen des Elements wird erstellt und mit der Funktion `CloseButtonClicked()` versehen. Zuletzt wird auch noch ein Label hinzugefügt, welches auch in `listlabellink` gespeichert wird. In diesem Label werden alle relevanten Vorlagen angezeigt.

1.12.8 CloseButtonClicked()

Entfernt das Element für eine aktive Organisation und macht den entsprechenden Button auf der linken Seitenleiste wieder sichtbar. Entfernt die Party-Instanz aus `activevotes`.

1.12.9 Activevotes_CollectionChanged()

Wird aufgerufen, wenn sich `activevotes` ändert. Falls keine Vote-Instanzen in `activevotes` vorhanden sind, wird ein entsprechender Text angezeigt. Andernfalls wird die Funktion `CommonVotes()` genutzt, um aus den vielen verschiedenen Listen von Vote-Instanzen eine Liste zu erzeugen, in welcher alle Instanzen gesammelt sind, welche in allen Listen vorhanden waren. So wird eine Liste aller relevanten Instanzen erzeugt. Welche dann analysiert werden. Es wird durch alle Vote-Instanzen durchgegangen und der durchschnittliche Ja-Prozentsatz, der Anteil der angenommenen Vorlagen, der höchste und tiefste Ja-Prozentsatz sowie die durchschnittliche Parteistärke ermittelt. Alle diese Zahlen werden in vordefinierten Textfeldern wiedergegeben. Ausserdem aktualisiert diese Funktion für jedes Element aktiver Organisationen die

Liste der relevanten Vorlagen. Diese Liste enthält das Jahr, den Titel und die Sektionen mit abweichenden Parolen.

1.12.10 CommonVotes()

CommonVotes() analysiert mehrere Listen und erstellt eine neue Listen mit Vote-Instanzen, welche die Summe aller gemeinsamen Instanzen in den gegebenen Listen sind. Hierfür werden neue Instanzen erzeugt, welche in den meisten Angaben den anderen Vote-Instanzen ähnlich sind, aber als Parteistärke wird die Summe der Stärken angegeben.

1.13 DataViewer.xaml.cs

DataViewer ist die älteste Komponente der App und dient der Darstellung einzelner Datensätze. Ausserdem können diese Daten verglichen und minimal modifiziert werden. Wenige Variablen werden genutzt:

GeneratedControls: Liste der durch Code erzeugten Elemente, damit diese nach ihrer Benutzung wieder entfernt werden können.

Optionssource: Gibt die Quelle für die Auswahloptionen an. Ist vom gewählten Land (USA/Schweiz) abhängig.

countrysourcestring: Gibt den Dateinamen der Datei in Optionssource an.

sksvg: Das SVG-Element, welches die Karte anzeigt. Wird immer wieder mit neuer Karte angezeigt.

Viele der Elemente in DataViewer wurden bereits im XAML-Code vordefiniert. Darunter sind vor allem die Auswahlelemente auf der linken Bildschirmseite, aber auch die Anzeigeelemente auf der rechten Seite. Ausserdem wird sehr viel auf abgelegt Dateien zurückgegriffen. Hierbei definieren usoption.tsv und swissoption.tsv die Kategorien und deren Unterpunkte. Die einzelnen Datenpunkte sind dann in einzelnen Dokumenten abgelegt, welche nach dem Schema «Kategorienname» + «Datensatznummer (bspw. Jahr)» + «.tsv» abgelegt sind, ein Beispiel wäre Gallup1980.tsv. Diese Datensätze haben verschiedene Visualisierungstypen:

MapVis: Enthält Daten zu allen Staaten/Kantonen, in der Visualisierung wird eine Karte angezeigt.

MapVisPopVote: Identisch zu MapVis, nur werden die Resultat nicht in Prozentsätzen, sondern in Stimmenzahlen abgelegt. Dies ist für Schweizer Volksabstimmungen notwendig, um zu berechnen, wie viele Ja- und Nein-Stimmen es landesweit gab.

StatVis: Erzeugt mehrere Balken mit einzelnen Teilen, welche Resultate einer Umfrage angeben. Das Dokument enthält die Titel diese Umfragen sowie die Resultate.

TableVis: Stellt Resultate als eine Art Tabelle, bestehend aus einer Frage/einem Titel und der dazugehörigen Antwort. Wird benutzt, um die 13-Keys-Daten anzuzeigen.

Der Einstieg zum Programm stellt die Funktion LoadCountries() dar.

1.13.1 LoadCountries()

Die Funktion fügt dem obersten Auswahlelement, bei welchem das zu untersuchende Land angegeben wird, die beiden Möglichkeiten, USA und Schweiz hinzu. Bereits im .xaml definiert ist, dass bei Auswahl einer dieser Optionen CountryListView_ItemSelected() ausgelöst wird.

1.13.2 CountryListView_ItemSelected()

Diese Funktion wird ausgelöst, wenn der Benutzer ein zu untersuchendes Land auswählt. Sie macht alle nicht-Landesauswahl-Elemente auf der linken Seitenleiste sichtbar. Je nach der getätigten Auswahl wird Optionssource und countrysourcestring festgelegt. Schlussendlich wird LoadOptions() ausgelöst.

1.13.3 LoadOptions()

Diese Funktion befüllt die vordefinierten Auswahlelemente mit den entsprechenden Optionen. Für die USA sind dies die einzelnen Kategorien (Echte Daten, Gallup, usw.), für die Schweiz sind es die Daten, an welchen Abstimmungen stattfanden. Die Optionen werden sowohl dem Hauptauswahlelement als auch, insofern es sich nicht um die «13 Keys»-Kategorie handelt, dem Auswahlelement zum Vergleich von Optionen hinzugefügt. Im .xaml vordefiniert ist, dass sobald eine Option angewählt wurde, HistoricalListView_ItemSelected() ausgelöst wird.

1.13.4 HistoricalListView_ItemSelected()

Diese Funktion wird ausgelöst, sobald der Benutzer eine Kategorie auswählt und befüllt das nächste Auswahlelement. Wählt der Benutzer beispielsweise die Gallup-Daten aus, so fügt diese Funktion dem entsprechenden Auswahlelement die Jahre hinzu, für welche Gallup Daten vorhanden sind. Diese Informationen sind in Optionssource vorhanden und werden mit der Funktion GetOptionYear() ausgelesen. Wird das Auswahlelement befüllt, welches die möglichen Vergleichsoptionen enthält, so wird mit IsMapVis() überprüft, ob es sich um einen Datensatz handelt, welcher mit anderen Datensätzen verglichen werden kann.

1.13.5 GetOptionYear()

Liest die Quell-Dateien aus und erstellt eine Liste mit allen angegebenen Informationen. Im Dokument usoption.tsv sind beispielweise alle Kategorien gelistet, und es wird angegeben, für welche Datenpunkte Daten vorhanden sind (Beispielsweise für die Gallup-Daten die Jahre 1936, 1940, usw.). Alle diese Punkte werden ausgelesen und einer Liste hinzugefügt. Um Fehler zu vermeiden, wird auf die Liste RemoveEmptyInfo() angewendet, welches leere Einträge in der Liste entfernt.

1.13.6 IsMapVis()

Diese Funktion überprüft, ob ein bestimmter Datensatz den Typ MapVis hat. Dieser Anzeigetyp wird mit einer Karte dargestellt und enthält Daten auf Kantons/Staatsebene. Die Überprüfung ist wichtig, da nur dieser Datentyp mit anderen Datensätzen verglichen werden kann. Zum Abrufen des Datensatz-Dokuments wird SourceFormatType() benutzt.

1.13.7 RemoveEmptyInfo()

Akzeptiert als Input ein Array, entfernt alle leeren Einträge aus der Liste, welche aufgrund der Benutzung von .Split() und überflüssigen Trenn-Zeichen entstehen können.

1.13.8 SourceFormatType()

Liest aus einem Dokument mit einem Datensatz die erste Zeile aus, welche den Datensatztypus angibt. Dieser Text wird als String zurückgegeben.

1.13.9 CheckVisibility()

Da es mehrere Möglichkeiten gibt, wie ein Element auf der Seitenleiste sichtbar oder unsichtbar werden kann, überprüft diese Funktion, ob alle Titel und die zugehörigen Ausklapp-Elemente richtig angezeigt werden. Die Vergleichs- und Veränderungs-Elemente können einerseits beim Anwählen ihres Übertitels sichtbar oder unsichtbar gemacht werden oder ihre Sichtbarkeit ändern, wenn der Benutzer einen Datensatz auswählt welcher verglichen oder nicht verglichen werden kann, womit die Optionen, welche genutzt werden um dies zu tun, nötig beziehungsweise überflüssig werden.

1.13.10 TextCell_Tapped(), TextCell_Tapped_1(), TextCellTapped_2(), usw.

Diese Funktionen werden aufgerufen, wenn ein Element in den Auswahllisten angetippt wird. Sie sind dafür verantwortlich, die Hintergrundfarbe der angewählten Zelle zu verändern. Zuletzt wird CheckVisibility() ausgelöst.

1.13.11 TapGestureRecognizer_Tapped() und TapGestureRecognizer_Tapped_1()

Diese Funktionen werden ausgelöst, wenn ein Titel auf der Seitenleiste angewählt wird. Sie machen die zugehörigen Elemente sichtbar oder unsichtbar.

1.13.12 mappanel_PaintSurface()

Funktion, welche für das Funktionieren der Karten-Anzeige nötig ist. Wenn das Bild geändert wird, wird die Funktion ausgelöst.

1.13.13 Button_Clicked()

Diese Funktion wird ausgelöst, wenn der Knopf zum Berechnen auf der linken Seitenleiste angewählt wird. Sie löst die eigentlich wichtigste Funktion MainLoadFunc() aus.

1.13.14 MainLoadFunc()

Diese Funktion liest alle auf der Seitenleiste vorhandenen Daten aus und erzeugt den Output auf der rechten Bildschirmseite. Erst wird ClearDisplay() ausgeführt, um alle Elemente auf der rechten Bildschirmseite zu entfernen. Erst wird überprüft, ob bei allen Auswahlelementen Daten ausgewählt wurden. Wenn irgendwo in dieser Funktion ein Fehler festgestellt wird, wird dies über das im .xaml definierte ErrorLabel ausgegeben. Mit SourceFormatType() wird analysiert, welche Art von Datensatz analysiert werden soll. Wenn ausserdem ein weiterer Datensatz ausgewählt ist, welcher verglichen werden soll, werden beiden Optionen entsprechend ausgelesen und verglichen. Neben der Option, dass es sich um einen Datensatz des Typen MapVis handelt, gibt es noch zwei weitere Optionen. Entweder kann es sich um Umfragewerte (StatVis) handeln oder um eine Tabelle (TableVis), welche für das Anzeigen von 13-Keys-Werten genutzt wird. Einige Funktionen, wie MetaExtract(), GetGradient() oder MainParties() werden für alle Möglichkeiten genutzt, andere sind nur für gewisse Visualisierungstypen notwendig. Wofür die Funktion benötigt wird, ist jeweils bei der Funktion nachzulesen.

1.13.15 ClearDisplay()

Löscht alle Elemente, welche durch Code erzeugt wurden und in GeneratedControls abgelegt wurden. Ausserdem werden für die im .xaml definierten Titel leere Texte gesetzt und die Kartenanzeige wird mit einem leeren Element

gefüllt und `mappanel_PaintSurface` per `.InvalidateSurface()` ausgelöst. Die Funktion lässt die rechte Bildschirmseite leer erscheinen.

1.13.16 MetaExtract()

Liest die zweite Zeile in einem Datensatz-Dokument, wo gewisse Meta-Informationen abgelegt sind. Es wird das zu benutzende Karten-Dokument angegeben, die Dateien, welche die Informationen zu den zu benutzenden Farben enthalten werden angegeben usw. Diese Funktion wird sehr häufig aufgerufen, um diese Informationen als Argumente in anderen Funktionen zu nutzen.

1.13.17 CreateDictionary()

Der Darstellung in MapVis liegt ein Dictionary zugrunde, welches für jeden Kanton/Staat angibt, was die Parteitendenz ist, sowie wie viele Elektorenstimmen der Staat besitzt. Hierbei wird das Datensatz-Dokument ab der fünften Zeile ausgelesen, wobei jede Zeile für einen Kanton oder Staat steht. Abhängig davon, ob es sich beim Datensatz um den Typen MapVis oder MapVisPopVote handelt, werden Stimm-Anzahlen in Prozente umgerechnet. Falls über die entsprechenden Steuer-Elemente auf der linken Seitenleiste ausgewählt wurde, dass das Resultat modifiziert werden soll, wird es entsprechend verschoben. Hierbei sind die Daten aus `ColorCategories()`, welche alle möglichen Zustände (bspw. Demokratisch, Tendenz Demokratisch, Unentschieden, Tendenz Republikanisch, Republikanisch) sowie aus `MainParties()`, welche die beiden grossen Parteien angibt, zwischen welchen verschoben werden soll. Dabei wird so beurteilt, dass nach Verschiebungen Stimmendifferenzen von bis zu 5% als Tendenzen erachtet werden und deswegen in diese Kategorie eingestuft werden.

1.13.18 CreateComparedDictionary()

Sollte ein MapVis-Datensatz zum Vergleich angewählt sein, wird unter `MainLoadfunc()` diese Funktion ausgelöst. Diese fügt zwei Dictionaries von zwei Datensätzen wie folgt zusammen: Sind die beiden Resultate für einen Staat identisch, wird die gleiche Farbe beibehalten (Wenn ein Staat bspw. in beiden Datensätzen für die Demokraten stimmt). Andernfalls wird eine «neue» Kategorie erstellt, welche nach der Veränderung benannt ist (bspw. Republikanisch -> Demokratisch).

1.13.19 DrawMap()

Zeichnet aus den Dictionary-Daten die SVG-Karte. Hierbei muss der Text des SVG-Dokuments mit den entsprechenden Farben verändert werden und dann wieder in ein SVG umgewandelt und schliesslich angezeigt werden. Das Element der Karte ist bereits vordefiniert.

1.13.20 GetSvgFromDirectory()

Lädt das in einem Datensatz angegebene SVG-Dokument aus dem Speicher und gibt einen Stream mit den ausgelesenen Daten zurück.

1.13.21 GetGradient()

In den Datensätzen sind Dokumente angegeben, welche die Farben, welche für die einzelnen Parteien/Kategorien zu benutzen sind, angegeben sind (bspw. welche Farbe für eine Tendenz Demokratisch zu benutzen ist). Diese Funktion liest das entsprechende Dokument und wendet `ColorsExtract()` darauf an. Es kann eine Liste mit Farben zurückgegeben werden.

1.13.22 ColorCategories()

Liest aus einem Datensatz-Dokument alle möglichen Kategorien für Parteien und Partei-Tendenzen aus, welche auf der dritten Zeile abgelegt sind. Diese werden in einem Array zurückgegeben.

1.13.23 ColorComparedCategories()

Wenn durch CreateComparedDictionary() «neue» Kategorien erstellt werden, ist es sinnvoll diese zu ordnen. Diese Funktion erstellt eine Liste aller möglichen Kombinationen und filtert die, die nicht von CreateComparedDictionary() erstellt wurden. Die Liste der benutzten neuen Vergleichs-Kategorien kann benutzt werden, um jeder dieser Kategorien eine Farbe zuzuschreiben.

1.13.24 SplitVote()

Berechnet, falls es sich um den Datensatztypen MapVisPopVote handelt, die korrekte Verschiebung, wie sie durch den Benutzer auf der linken Seitenleiste ausgewählt wurde.

1.13.25 DrawElectoralVotes()

Funktion, welche eine Leiste zeichnet, welche Stimmenanteile anzeigt. Die Breite einer bestimmten Farbe steht dabei für den Stimmenanteil. Die Funktion akzeptiert als Argumente ausserdem mehrere Wahrheitswerte, welche angeben, ob die jeweiligen Kategorien beschriftet werden sollen, oder ob in den Textfeldern die Anzahl der Stimmen angezeigt werden sollen usw. Die Leiste hat auch ein Textfeld für einen Titel, welcher ebenfalls per Argument angegeben werden muss.

1.13.26 MainParties()

Liest die vierte Zeile eines Datensatz-Dokuments, auf welchem die beiden Haupt-Parteien angegeben sind. Diese beiden sind immer eine «Teilmenge» der Kategorien auf Zeile drei, und werden benötigt, um herauszufinden, in welche Richtung bspw. eine Verschiebung stattfinden sollte.

1.13.27 DrawKeyBox()

Erstellt die Legende aller Kategorien und der für sie benutzen Farben. Im Falle dessen, dass ein Vergleich angestellt wird und neue Vergleichs-Kategorien erstellt wurden, werden diese unter einem separaten Titel gelistet. Die Kategorien, welche in beiden Datensätzen identisch sind, erscheinen dann unter dem Titel «Übereinstimmend», alle anderen, wo sich ein Datenpunkt in einem Datensatz also von dem im anderen Datensatz unterscheidet, befinden sich unter dem Titel «Verglichen mit Historsch:».

1.13.28 MetaLabeling()

Verändert die durch das .xaml definierten Titel mit den korrekten Informationen.

1.13.29 TitleShiftModifier()

Fügt dem Titel, falls der Benutzer eine Veränderung der Resultate angegeben hat, die Angabe hinzu, um wie viele Prozentpunkte das Resultat modifiziert wurde.

1.13.30 CreateMultipleBars()

Ruft mehrere Male DrawElectoralVotes() auf, wird für die Darstellung von StatVis-Dokumenten benutzt.

1.13.31 CreateStatDictionary()

Liest für eine StatVis-Darstellung die entsprechenden Daten ab der fünften Zeile eines Datensatz-Dokuments aus. Dabei enthält jede Zeile die Informationen für eine Ausführung von DrawElectoralVotes(). In erster Position steht die gestellte Frage, welche zum Titel des dargestellten Balkens wird. Darauf folgen die Anteile der auf Zeile drei definierten Kategorien.

1.13.32 DrawTable()

Stellt eine Tabelle von 13-Keys Resultaten dar, welche pro Key aus einem Textfeld und einem entsprechend eingefärbten Frame besteht. Ausserdem werden die Anzahlen der Keys gezählt und mit DrawElectoralVotes() als Balken angezeigt.

1.13.33 CreateTableDictionary()

Liest für eine TableVis-Darstellung alle benötigten Daten aus dem Datensatz-Dokument, wobei die relevanten Daten auf der fünften Zeile beginnen. Dabei enthält jede Zeile genau zwei Informationen: Den Namen des Keys, und ob dieser als wahr oder falsch bewertet wurde. Diese Informationen werden als Dictionary zurückgegeben.

1.13.34 IncumbentModifizier()

Wird für die 13-Keys-Darstellung benötigt. Anhand von den beiden Haupt-Parteien auf Zeile vier wird bestimmt, welche Seite die des Amtsinhabers ist. Bei dem entsprechenden Kandidaten wird diese Information im Titel hinzugefügt.

1.13.35 ColorsExtract()

Als Argument wird eine Bitmap, also ein Bild, angegeben. Diese Funktion liest für jeden Pixel die Farbe aus und legt diese Farben in einer Liste ab.

1.13.36 DictionaryValueList()

Erstellt eine Liste mit allen Werten (Partei-Kategorien) in einem Dictionary aus CreateComparedDictionary(). Wird in DrawKeyBox() benutzt, um herauszufinden, ob es sich bei einer Kategorie um eine «neue» Vergleichs-Kategorie handelt.


1.13.37 DictionaryValueString()


Fast identisch mit DictionaryValueList(), nur werden alle Werte in einem String zusammengefügt. So kann schnell überprüft werden, ob durch CreateComparedDictionary() überhaupt neue Kategorien erstellt wurden.


2. Benutzungsanleitung

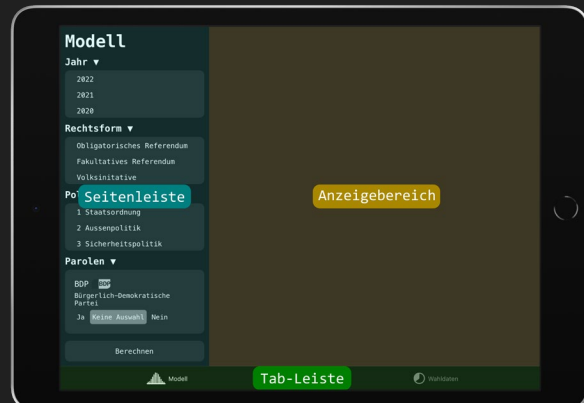
2.1 Übersicht

Die App besteht aus drei Funktionsbereichen, auf welche durch Antippen der einzelnen Symbole auf der Tab-Leiste zugegriffen werden kann. Die Bereiche sind:

 Modell

 Partei-Parolen

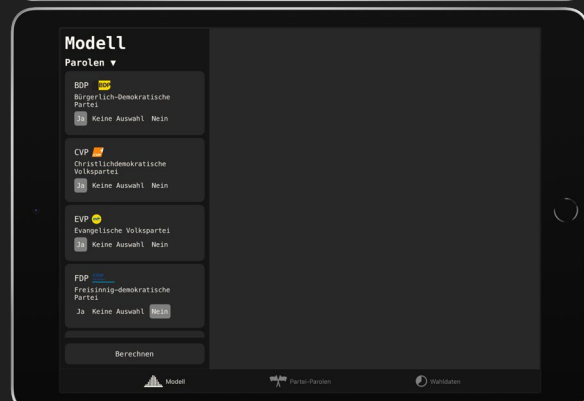
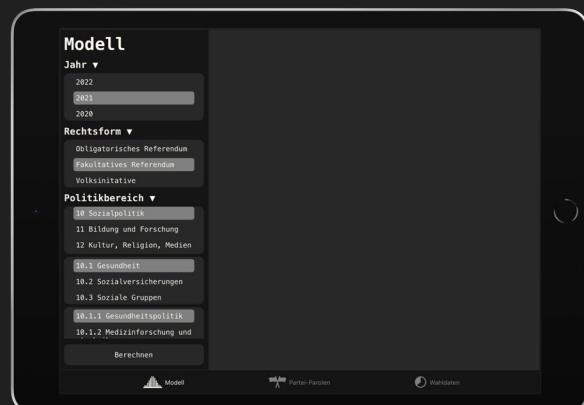
 Wahldaten



Alle diese Funktionsbereichsseiten sind im Grundsatz gleich aufgebaut. Sie bestehen aus einer Seitenleiste auf der linken Bildschirmseite und einem Anzeigebereich auf der rechten Bildschirmseite. Allgemein werden auf der Seitenleiste Einstellungen gesetzt und Auswahlen getroffen, und im Anzeigebereich erscheinen die zugehörigen Resultate. Einzig im Bereich Partei-Parolen befinden sich auch im Anzeigebereich Elemente, welche durch den Benutzer verändert werden können und so Resultate verändern. Allgemein ist auch festzuhalten, dass auf der Seitenleiste immer Titel hat, welche durch ein Dreieck markiert sind (▲ wenn die zugehörigen Elemente unsichtbar sind oder ▼ wenn die zugehörigen Elemente sichtbar sind). Diese Titel können angetippt werden, wodurch alle Elemente bis zum nächsten solchen Titel ihre Sichtbarkeit ändern.

2.2 Die Funktionen des Modell-Abschnitts

Alle relevanten Auswahlen werden auf der Seitenleiste getroffen. Dort gibt es vier mögliche Auswahlelemente, welche auf den nebenstehenden Bildern sichtbar sind. In den hellgrauen Listen kann jeweils ein Element ausgewählt werden. In diesen Bildern wurde das Jahr 2021 und die Rechtsform Fakultatives Referendum ausgewählt. Der Politikbereich besteht aus mehreren Auswahlen. Wenn ein erster Bereich, hier «10 Sozialpolitik» gemacht wurde, erscheint ein zweites Auswahlfeld, in welchem eine präzisere Auswahl gemacht werden kann, hier «10.1 Gesundheit». Dies setzt sich weiter fort. Zuletzt kann unter Parolen eingestellt werden, welche Parolen die grossen Parteien herausgeben. Hier sichtbar ist ein Ja von BDP, CVP und EVP und ein Nein der FDP. Mit dem



The screenshot displays the 'Ergebnisse' (Results) page for the Canton of Aargau. It is divided into two main sections: 'Modell' (Model) on the left and 'Resultate' (Results) on the right.

Modell

- Jahr:** 2022
- Rechtsform:** Obligatorisches Referendum
- Politikbereich:** Sozialpolitik
- Berechnen**

Resultate

Verteilung der Volksmehrheiten

Kategorie	Anzahl Stimmberechtigte	Anteil (%)
Nein	~100,000	~50%
Ja	~100,000	~50%

Kenndaten

- Durchschnittl. JA-Anteil: 64.69%
- Anteil der Simulationen mit zugewonnener Vorlage: 79.13%
- Ausletzte gefundene Vorlagen:
 - Parlamentarismus, 2017
 - Verfassungspakt für Komplementärmedizin, 2009
 - Wettbewerbs- und Steuerreform, 2015
 - Energiegesetz, 2017

Kantone

Aargau

- Kürzel: AG
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Basel-Stadt

- Kürzel: BS
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Basel-Landschaft

- Kürzel: BL
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Soleure

- Kürzel: SO
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Zürich

- Kürzel: ZH
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Glarus

- Kürzel: GL
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Uri

- Kürzel: UR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Schaffhausen

- Kürzel: SH
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Appenzell Auss. Rhod.

- Kürzel: AR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Appenzell Inn. Rhod.

- Kürzel: AI
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Fribourg

- Kürzel: FR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Valais

- Kürzel: VS
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Vaud

- Kürzel: VD
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Nevche

- Kürzel: NE
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Genève

- Kürzel: GE
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Luzern

- Kürzel: LU
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Unterwalden

- Kürzel: UW
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Obwalden

- Kürzel: OW
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Sankt Gallen

- Kürzel: SG
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Tessin

- Kürzel: TI
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Grigien

- Kürzel: GR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Glarus

- Kürzel: GL
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Appenzell Auss. Rhod.

- Kürzel: AR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Appenzell Inn. Rhod.

- Kürzel: AI
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Fribourg

- Kürzel: FR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Valais

- Kürzel: VS
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Vaud

- Kürzel: VD
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Nevche

- Kürzel: NE
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Genève

- Kürzel: GE
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Luzern

- Kürzel: LU
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Unterwalden

- Kürzel: UW
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Obwalden

- Kürzel: OW
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Sankt Gallen

- Kürzel: SG
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Tessin

- Kürzel: TI
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Grigien

- Kürzel: GR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Glarus

- Kürzel: GL
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Appenzell Auss. Rhod.

- Kürzel: AR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Appenzell Inn. Rhod.

- Kürzel: AI
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Fribourg

- Kürzel: FR
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Valais

- Kürzel: VS
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Vaud

- Kürzel: VD
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Nevche

- Kürzel: NE
- Stimmanteile: 1
- Stammwahlteilr.: 66.76%
- Anteil Simulation mit über 50%-JA: 59.56%

Genève

- Kürzel: GE
- Stimmanteile: 1
- Stammwahlteilr.: 66.76

Parolen

Ständerat

Parteien ▼

Autopartei

Bürgerlich-Demokratische Partei

Christlich-Demokratische Volkspartei

Die Mitte

Evangelisch-Demokratische Union

Evangelische Volkspartei

Grüne Partei der Schweiz

Grünliberale Partei

Länderding der Unabhängigen

Legge dei Ticinesi

94%
Nein: 74%
Beizitat

24%
Schweizerinnen
für die Freiheit

79%
Anzahl der abgegebenen Abstimmungen

(Zahlen basieren auf 121 Abstimmungen von 1984 bis 2011)

63%
Durchschnittliche Wahlbeteiligung der "Ja"-Wähler

46%
Durchschnittliche Wahlbeteiligung der "Nein"-Wähler

SPS 1984

Sozialdemokratische Partei

☒ Neutral ☐ Nein

1981: Einführung der Volksinitiative zur Teilrevision der Bundesverfassung

1983: Bundeskompetenz

1984: Bundeskompetenz für das Gewerbe

1984: Initiative «für Verwirklichung des Rechts

1985: Zivilisierungsmodell

1986: Gesetz über das Rechtssystem der Eisenbahn

FDP 1984

Freiwillig-demokratische Partei

☒ Neutral ☐ Nein

Erhöhung der Schweizerischen Bundesrat

1987: Bundeskompetenz bei der Wasserbau- und Forstwirtschaft

1987: Bundeskompetenz für die Eisenbahntechnik

1988: Gesetz über die Schweizerischen

1988: Vereinfachung des Zivilrechts

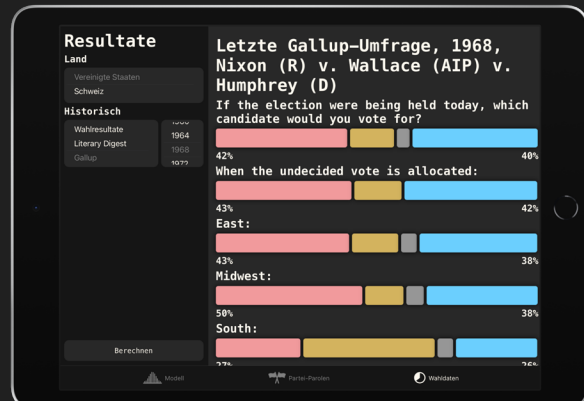
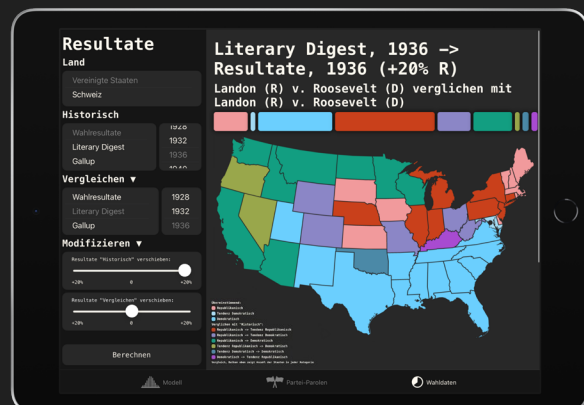
1988: Vereinfachung des Strafrechts

1988: Gesetz zur Kranken-

Parteikürzel, Logo und Parteiname in den dunklen Elementen im Anzeigebereich). Um eine Organisation aus der Berechnung zu entfernen, wird der X-Knopf in der oberen rechten Ecke benutzt. Die oben im Anzeigebereich vorhandenen Zahlen geben Kennwerte an. In dem gezeigten Beispiel sind das also die Werte für alle Vorlagen, zu denen sowohl SP und FDP eine Ja-Parole herausgegeben haben. Von diesen Vorlagen wurden, wie die mittlere Prozentzahl angibt, 79% angenommen. Die Zahlen daneben geben einige andere Informationen an. So war das beste Resultat einer Abstimmung mit Ja-Parolen dieser beiden Parteien 94% Ja, das schlechteste 24% Ja. Der durchschnittliche Anteil von Ja-Stimmen lag bei diesen Vorlagen bei 63% Ja, und der durchschnittliche Wähleranteil der Parteien, die eine Ja-Parole herausgegeben haben, lag bei 46% (Die SP und die FDP haben also zusammen durchschnittlich über alle die analysierten Vorlagen (in welchen sie also beide eine Ja-Parole herausgaben) einen Wähleranteil von 46%). Analysiert wurden 229 Abstimmungen von 1894 bis 2021.

2.4 Die Funktionen des Wahldaten-Abschnitts

Dieser Bereich dient zur Betrachtung von Daten. Es gibt drei Typen von Datensatz, welche jeweils durch ein Bild gezeigt werden. Alle Auswahlen finden auf der Seitenleiste statt. Erst muss das Land ausgewählt werden. In den Beispielen sind dies immer die Vereinigten Staaten, die Funktionsweise ist aber für die Schweizer Daten analog. Dann muss unter dem Titel «Historisch» ein Datensatz gewählt werden. Der häufigste Datensatztyp ist der, bei welchem Daten für alle Staaten/Kantone verfügbar sind. Dies ist der Fall für alle Abstimmungen in der Schweiz, alle Wahlergebnisse in den USA, alle Daten des Literary Digest, sowie die Gallup-Daten bis und mit 1948. Ein Beispiel eines solchen Datensatzes ist im oberen Bild sichtbar. Bei diesem Datentyp sind einige weitere Optionen verfügbar. Einerseits kann unter «Vergleichen» ein zweiter Datensatz ausgewählt werden, mit welchem der erste verglichen wird, andererseits können die Resultate für beide Datensätze unter «Modifizieren» verändert werden, es kann also gewissermaßen ein was-wäre-wenn-Szenario erstellt werden. Im Beispiel sind beide diese Funktionen benutzt worden; das Beispiel zeigt den Vergleich zwischen der Vorhersage des Literary Digest und den echten Resultaten für das Jahr 1936, wenn bei den echten



Resultaten die Republikaner 20 Prozentpunkte besser abgeschlossen hätten. Dabei stehen alle Farben für ein bestimmtes Resultat, welches in der Legende unten links im Anzeigebereich erklärt ist (Diese Legende erscheint eigentlich tiefer unten auf der Anzeige und wurde im Beispiel zu Anschauungszwecken nach oben verschoben). Beispielsweise gilt für alle im Beispiel Hellblau markierten Staaten, dass sie sowohl gemäss Literary Digest als auch in den echten Resultaten, falls die Republikaner 20% besser abgeschnitten hätten, für die Demokraten stimmen würden. Die beiden anderen Anzeigetypen sind simpler: Die im zweiten Bild ersichtliche Darstellung wird für Umfragen benutzt, für welche keine Daten aus Staatenebene verfügbar sind. Dies ist der Fall für alle Gallup-Umfragen ab 1952. Das Format ist, dass jeweils eine Frage, welche gestellt wurde mit den zugehörigen Antwortprozenten in einem Balken dargestellt werden. Auch hier folgt ganz unten eine Farblegende. Im Beispiel des zweiten Bildes ist die erste Frage nach dem bevorzugten Daten, wobei 42% den Republikaner (in Rot) und 40% den Demokraten (in Blau) als Präferenz angaben. Der dritte Datentyp, welcher im letzten Bild sichtbar ist, ist eine Art Tabelle, welcher nur für die «13 Keys»-Daten benutzt wird. Es wird der Name des Schlüssels angegeben, und dann mit der Farbe der Partei markiert, der er zugutekommt. Oben ist ein Balken sichtbar, in welchem die Anzahl Schlüssel zusammengezählt werden. Hier sind 11 Schlüssel wahr, was für den Amtsinhaber spricht. Dieser ist in diesem Beispiel Republikaner, weswegen dieser Teil des Balkens in Rot gehalten ist.