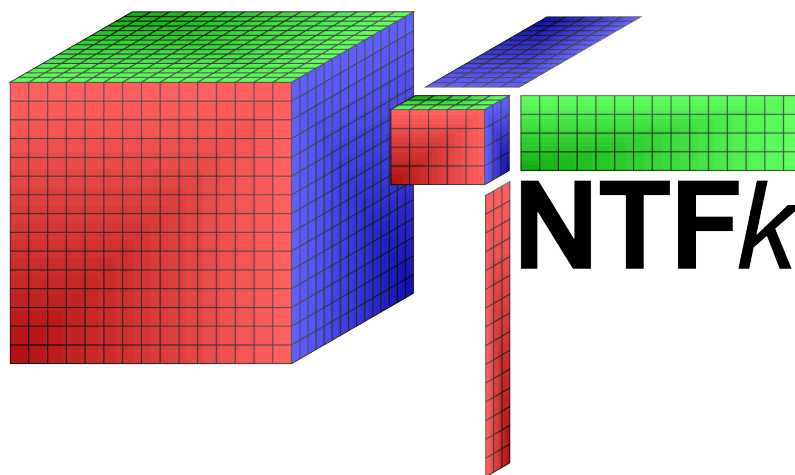




IT DIPLOMUDDANNELSEN
AFGANGSPROJEKT - EFTERÅR 2019

TEST AF NMFk ALGORITMEN: UNSUPERVISED LEARNING TIL IDENTIFICERING AF UKENDTE FORURENINGSKILDER I ET INDVINDINGSOPLAND



DATO: 18. december 2019
STUDERENDE: Allan Kötter
STUDIENUMMER: s140381
VEJLEDER: John Aasted Sørensen,
ANSLAG: 81.983 tegn (12.380 ord)

Indhold

1	Indledning	4
1.1	Problemformulering og projektdefinition	5
1.2	Machine Learning inden for Jordforureningsområdet	6
2	Teori	8
2.1	Linær algebra og tensor dekomposition	8
2.2	Højere ordens dekompositioner	10
2.3	Blind Source Separation (BSS)	10
2.4	NMFk og NTFk	11
3	Systemdesign	14
3.1	Etablering af cloud-baserede analysemiljø	14
3.2	Dataservice - opbygning af containeren <i>mssql</i>	16
3.3	Analysemiljøet - opbygning af containeren <i>lab</i>	18
3.4	Orkestrering af miljøet med Docker-Compose	20
4	Datagrundlag for test	21
4.1	Datakilder til vandanalyser i dansk kontekst	22
4.2	Generisk databaseudtræk til test af NTFk	23
4.3	Geografisk afgrænsning af et datasæt	24
4.4	Efterfølgende transformation af datasæt til 3 dimensionel tensor	25
5	Test af NTFk	26
5.1	Metode	26
5.2	Test 1	28
5.3	Test 2	31
5.4	Test 3	34
5.5	Test 4	37
5.6	Test 5	40
5.7	Test 6	46
5.8	Gennemgang af testresultater	46
6	Diskussion	50
6.1	Test af NTFk algoritmen	50
6.2	"Cloud native - Container first"	55
6.3	Julia - et nyt lovende sprog	57
7	Konklusion og perspektivering	59
7.1	NTFk	59
7.2	Azure og Docker	59
7.3	Perspektivering	60

Litteratur	61
Bilag	64
A Anvendte værktøjer	65
B Testresultater - eksempel for test 3	66
C Udskrift af kode - Dockerfile mssql	82
D Udskrift af koden - Scripts på mssql	84
E Udskrift af koden: Dockerfile lab	91
F Kodeudskrift: Docker-Compose YAML	99
G Kodeudskrift - SQL til dataudtræk	101
H Notebooks og script - eksempel for test 3	105
I Projektlog	170

Resumé

En vigtig opgave for de 5 danske regioner, er kortlægning, undersøgelser og afværgende foranstaltninger over for gamle jordforureninger mhp. bl.a. at beskytte drikkevandsforsyningen. Identifikation og opsporingen af de kilder, der udgør en risiko for drikkevandsforsyning er en dyr og tidskrævende opgave.

Jeg har i dette projekt testet en ny unsupervised machine learning algoritme (*NTFk*), der er udviklet og publiceret af Velimir Valentinov Vesselinov og hans kollegaer fra Los Alamos National Laboratory i USA. Algoritmen skulle - på baggrund af et større antal kemiske analyser af grundvandet i et område - kunne identificere de ukendte forureningskilder, der har givet anledning til de målte koncentrationer i vandprøverne. Hvis dette kan lade sig gøre, ville regionerne kunne spare både tid og penge på opsporingen af forureningskilder og i stedet fokusere på at lave de nødvendige afværgende foranstaltninger til sikring af drikkevandsressourcen.

Jeg har ved hjælp af Docker-containere oprettet et cloud-baseret analysemiljø i Azure, og har implementeret og testet *NTFk* algoritmen her i. Forsøget er gennemført som 5 strukturerede tests, hvor der er genererede forskellige datasæt fra Birkerød indvindingsopland. Dette opland er valgt, fordi her har myndighederne gennemført forureningsundersøgelser i mange år, og der findes et stort datagrundlag. Samtidig er der også et godt kendskab til de potentielt truende forureningskilder, der findes i oplandet, hvilket vil kunne bruges til at verificere de modellerede resultater.

De 5 tests omfatter 2 tests med naturligt forekomne kemiske komponenter, der ikke kan relateres til specifikke forureningskilder, og 3 tests med chlorerede opløsningsmidler og en sidste test, der omfatter alle de kemiske komponenter, der optræder i et større antal i datasættet. Testene har givet følgende resultater:

- Med data i test 1, kan jeg genskabe de resultater, som Vesselinov får med samme datasæt.
- Test 2 viser, at normalisering af datasættet fra test 1 giver en dårligere modelperformance.
- Test 3 - 5 viser at modellen identificerer en "enkelt-komponent" kilde for hver forureningskomponent i datasættet. Endvidere påvirker alle kilderne alle borer i modellen. Dette er et usansynligt scenarie og er ikke i overensstemmelse med den gængse forståelse af forureningsspredning i grundvandet.

Konklusionen på de gennemførte tests er, at algoritmen ikke er anvendelig til regions arbejde på nuværende tidspunkt. Forklaringen på dette kan være "sparsity" - dvs. at de anvendte datasæt er meget "sparsomt" besatte. Endvidere udviser de chlorerede komponenter i datasættene et meget ujævnt og varierende signal, der vurderes at kunne være problematisk for algoritmen at udlede klare kilde-relaterede tendenser

fra. Endelig er algoritmen og den software-pakke, som den er implementeret i, under stadig udvikling.

Jeg skylder Vesselinov en stor tak, for al den hjælp han har givet mig i løbet af projektet. Ud fra hans demo-kørsel på mit dataset (test1) har jeg kunne verificerer, at min installation har fungeret og giver sammenlignlige resultater med hans kørsel. Der findes endvidere ikke dokumentation til hans software-pakke, og uden hans vejledning, kunne jeg ikke have gennemført projektet.

1. Indledning

Regionerne - og Amterne før kommunalreformen - har igennem mange år indsamlet data om forurenede og potentielt forurenede arealer. Disse data danner grundlaget for at gennemføre den offentlige indsats for beskyttelse af drikkevandsforsyningen i et område. Kortlægningen af forurenede arealer sker ud fra lister med grænseværdier for udvalgte miljøfremmede stoffer som fastsættes af Miljøstyrelsen. Listerne med stoffer udgør således rammen for, hvilke data der bliver indsamlet, og efterfølgende er til rådighed, når risikoen fra forureningskilder over for et vandværk skal vurderes.

Da Miljøstyrelsen den 1. juni 2018 fastlagde, at en række nye pesticider skulle indgå i vandværkernes kontrolprogrammer, viste vandværkernes analyser at disse nye stoffer er et langt større problem end hidtil antaget. Problemet for regionerne er, at der ikke bagudrettet er blevet analyseret for disse nye stoffer i kortlægningsarbejdet. Derfor indeholder regionernes databaser ikke information, der kan bidrage til at identificere, hvilke punktkilder der er årsag til den pesticidforurening, der nu bliver fundet i vandværkernes boringer. Regionerne skal derfor igang med at opspore og genkortlægge disse punktkilder.

Den måde man traditionelt vil løse denne udfordring på, er ved anvendelsen af numeriske 1D-, 2D- og 3D-modeller, som vha. partielle differentiaalligninger beregner grundvandets strømning og transport af forureningsstoffer som partikelbaner inden for et modelområde. Disse modeller kan også bruges til "baglæns simulering af partikelbaner" og på den måde beregne beliggenheden af en ukendt forureningskilde ud fra fund af forurening i f.eks. en vandforsyningsboring.

Udfordringen ved numeriske modeller er, at det ofte kræver en stor mængde data og viden at kunne sætte modellen op. Dette gør grundvandsmodeller til en dyr løsning. Mange modeller er desuden præget af utilstrækkelig viden om lokale variationer i geologien, hvilket kan have stor betydning for grundvandsstømning og forurenings-transport. Dette resulterer i store usikkerheder i modelresultaterne. Desuden tager det lang tid at opstille og træne modellen samt at køre simuleringerne.

NTFk

I 2018 og 2019 har Vesselinov og hans kollegaer publiceret to artikler om en ny unsupervised Machine Learning algoritme kaldet *NTFk*, der jf. Vesselinov kan identificere antallet af ukendte forureningskilder, der påvirker et grundvandsmagasin. Algoritmen kan også identificere stofkoncentrationer i punktkilderne, samt spatielle og

temporærer dynamikker i disse [35], [36]. Algoritmen kan fastlægge disse informationer udelukkende ud fra målinger i påvirkede observationsboringer, uden information om geologi, geokemi og lignende.

Potentialet i metoden rammer således direkte ned i regionernes udfordring som beskrevet ovenfor - dvs. behovet for at kunne identificere potentielle forureningskilder der påvirker en række vandforsyningsboringer, udelukkende ud fra analyser i vandforsyningsboringerne og uden at have data om selve forureningskilden. Selvom selve kortlægningen af punktkilderne ikke ville kunne ske på baggrund af algoritmens resultater, ville regionerne kunne spare værdifuld tid i forbindelse med den indledende kildeopsporing. Samtidig ville man hurtigere kunne komme i gang med afværgeforanstaltninger, hvis algoritmen kunne udpege beliggenheden af de punktkilder, der påvirker en indvindingsboring.

Vesselinov beskriver i sin artikel, at algoritmen både er testet på et syntetisk datasæt, og på et "rigtigt" datasæt fra den virkelige verden. Algoritmen baserer sig på *Blind Source Separation (BSS)* og forudsætter, at den sammenblanding af forurening, der sker i grundvandsmagasinet er linær og kræver samtidigt et "informativt" datasæt [36]. I de fleste grundvandsmagasiner er linær sammenblanding af forureninger ikke forventelig, men i danmark har vi typisk lange tidsserier med mange analyseparametre. Det kan ikke ud fra artiklen afgøres, om de datasæt der typisk er til rådighed fra danske vandværksboringer, lever op til kravene om et "informativt" datasæt. Vesselinov skriver i sin artikel, at han vil stille en *Open Source* implementering af algoritmen til rådighed i programmeringssproget *Julia*.

1.1 Problemformulering og projektdefinition

Med dette projekt ønsker jeg at undersøge, om *NTFk* kan anvendes i en dansk kontekst og om algoritmen dermed kan bruges som beslutningsstøttværktøj for de 5 danske regioner i forbindelse med kildeopsporing af ukendte forureningskilder herunder pesticider. Projektet kommer til at bestå af følgende elementer:

Infrastruktur Der etableres en prototype på en analyseplatform i Regionens IT-arkitektur, hvor *NTFk* algoritmen kan implementeres. Platformen skal stille regionens databaser med analyseresultater analyseresultaterne til rådighed, så disse dynamisk kan trækkes ind i analysemiljøet og analyseres, Miljøet skal stille almindelige faciliteter til rådighed bl.a. adgang til R, Python og Julia, samt mulighed for interaktivt at kunne analysere og visualisere data. Miljøet skal endvidere kunne skaleres, således at det på sigt kan udvides til også at omfatte de 4 øvrige regioners data.

NTFk installation Vesselinovs implementering af *NTFk* downloades, installeres og testes. Det skal undersøges om jeg kan køre algoritmen i analysemiljøet, herunder lære at anvende algoritmens output og opnå validerede resultater. Til dette formål skal jeg lære programmeringssproget *Julia* og sætte mig ind i algoritmens funktionalitet.

Dansk test datasæt Der genereres et datasæt vha. tilgængelige datakilder fra et indvindingsopland, med kendte og vel-undersøgte forureningskilder, samt omfattende kontrolanalyser i oplandets indvindingsboringer. Det undersøges hvorvidt resultaterne fra algoritmen påviser de samme forureningskilder som regionens undersøgelser.

Pesticid datasæt Såfremt forsøget med det danske datasæt viser et positivt resultat, forsøges algoritmen anvendt på et opland med pesticid-problemer, hvor regionen mangler viden om de punktkilder, som påvirker vandindvindingen. Dette gøres i samarbejde med de sagsbehandlere, der arbejder med denne problemstilling, og det evalueres om resultaterne kan bruges som beslutningsstøtte i regionens videre arbejde.

Det teoretiske studie vil fokusere på at undersøge *decompositions metoder*, herunder begrebet *Blind Source Separation (BSS)*. Projektet vil desuden undersøge *Containere* som en central byggesten i en moderne cloud-baseret IT-arkitektur - dette vil tage afsæt i *Azure* og *Docker*.

1.2 Machine Learning inden for Jordforureningsområdet

Anvendelsen af Machine Learning inden for jordforureningsområdet er ikke særligt udbredt. Der har gennem tiden været anvendt simple sandsynlighedsmodeller som beslutningsstøtte til forskellige formål inden for jordforureningsområdet. Bl.a. har der gennem mange år været et værktøj i Miljøstyrelsens risikovurderings-værktøj *JAGG*, der kunne beregne antallet af jordprøver der skulle tages i et område, for at opnå en ønsket sandsynlighed for at en punktkilde blev fundet. Der har også i de senere år været eksempler på anvendelsen af variogrammer i kombination med Monte Carlo simuleringer, for at beregne sandsynligheden for udbredelsen af en forurening.

Miljøstyrelsen har i 2013 gennemført *Miljøprojekt nr. 1502* [30], der fokuserer på at afklare mulighederne for at skelne mellem punkt- og fladekilder ud fra pesticidanalyser i en grundvandsprøve. I rapporten indgår et studie fortaget af GEUS samme år, hvor der er gennemført en PCA analyse på 5 datasæt med pesticidanalyser, og denne analyse konkluderer, at "*man ikke kan lave en multivariat datamodel, der ud fra en vandprøves samtidige indhold af specifikke stoffer kan afgøre om analysen stammer fra en punktkilde eller en fladekilde.*".

Studiet omfatter også en række statistiske analyser af de 5 datasæt, og Tuxen et. al. vurderer bl.a. på denne baggrund, at "*såfremt der er forskelle i koncentrationerne af specifikke pesticider over tid, skyldes det formentlig en punktkilde. Årsagen hertil er at dispersion i grundvandet vil udviske eventuelle variationer i inputkoncentrationerne fra en fladekilde (fx som følge af varierende sprøjteperioder), mens små forskelle i grundvandsstrømninger vil kunne medføre store variationer i om en boring fanger fanen fra en punktkilde og dermed koncentrationniveauet*" [30]

Miljøprojektet resulterer i at anviser en række nøgleparametre, der tilsammen indikerer, om en vandprøve repræsenterer en punktkilde eller en fladekilde eller evt. et mix. Nøgleparametrene omfatter bl.a. tilstedeværelsen af mange eller få stoffer, fund af høje eller lave koncentrationer, varierende koncentrationer af et stof over tid - dvs. vurderede, afledte parametre.

I to nyere projekter fra 2019 har GEUS brugt random forest algoritmen på pejledata fra den nationale boringsdatabase *Jupiter*. I det ene projekt bliver dybden til det terrænnære grundvand modelleret som et højopløseligt kort for Region Midtjylland [?]. Denne model er udviklet til kommunernes klimatilpasning, men ville også kunne

bruges i forbindelse med forureningsundersøgelser. I det andet projekt bliver Redox-grænsen modelleret som et landsdækkende kort med høj opløsning. Redox-grænsen har relevans for beskyttelsen af indvindingsboringer mod nedsivende nitrat fra landbruget, og er således heller ikke udviklet til jordforureningsområdet [?].

2. Teori

I nedenstående kapitel vil teorien bag NMFk og NTFk blive gennemgået. Teorien vil blive beskrevet med henblik på en overordnet forståelse af algoritmerne, og ikke ud fra et matematisk perspektiv. Da NMFk og NTFk bygger på hhv. Non-negative Matrix Factorization (MNF) og Non-negative Tensor Factorization (MTF), vil disse også blive beskrevet med afsæt i matrice dekomposition.

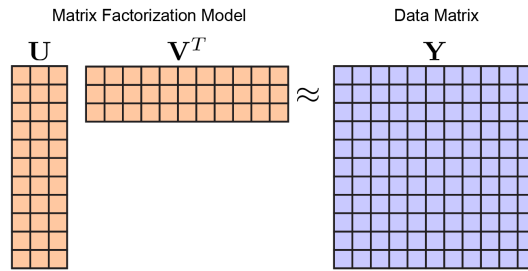
2.1 Linær algebra og tensor dekomposition

Matematikken bag NMFk og NTFk bygger - lige som mange andre algoritmer inden for Machine Learning - på linær algebra. Linær algebra er en matematisk disciplin, der beskæftiger sig med lineære funktioner og deres rumlige repræsentationer. De centrale elementer i linær algebra er en skalar, en vektor, en matrice og en tensor, og disse relaterer sig til elementets rumlige dimensioner. En skalar har 0 dimensioner, en vektor har én dimension, en matrice har to dimensioner og en tensor har tre eller flere dimensioner.

I linær algebra kan elementer *faktoriseres*, hvilket betyder at de kan repræsenteres af produktet mellem to andre elementer. Som et simpelt eksempel kan skalar 12 repræsenteres af produktet mellem to andre skalarer $2 \cdot 6$. Repræsentationen er ikke unik, da $3 \cdot 4$, $4 \cdot 3$ og $6 \cdot 2$ også er gyldige repræsentationer [25].

På samme måde kan højere ordens elementer også faktoriseres. Matricer kan således repræsenteres af produktet mellem to andre matricer og også her er der mange gyldige repræsentationer. Dette er udgangspunktet for matrice dekomposition, som generelt forsøger at finde den optimale repræsentation ud fra en eller flere givne rammebetingelser [26].

Forskellige optimeringsmetoder giver forskellige dekompositions algoritmer, hvor den mest kendte og anvendte nok er PCA. En generaliseret matematisk fremstilling af optimeringsmetoden baseret på *Least Sum of Squares* er vist i ligning 2.1 og matrice faktorisering er vist som en grafisk repræsentation på figur 2.1:



Figur 2.1: Matrix dekomposition (fra: <http://alexhwilliams.info>)

$$d(Y, Y^T) = \|Y - UV^T\|_F^2 + \lambda_u f_u(U) + \lambda_v f_v(V) \quad (2.1)$$

hvor $U \in \Omega_u, V \in \Omega_v$

Optimeringen i ligning 2.1 sker ved minimering af *Sum of Least Squares*, dvs. differencen mellem den modellerede matrice (UV^T) og den oprindelige matrix (Y) sættes i anden potens og opsummeres. *Sum of Least Squares* kendes også som en almindelig optimeringsmetode til lineær regression.

På denne *Sum of Least Squares* beregning lægges to lambda-funktioner til for hhv. U og V . Lambda-funktionerne kaldes for "*Regulatorer*" og kan betragtes som en form for "*handikap*", som lægges til, hvis metodens rammebetingelser ikke er overholdt.

Non-negative Matrix Factorization

Som nævnt ovenfor, er den mest almindelige algoritme inden for matrix dekomposition PCA analysen. PCA er karakteriseret ved, at der ikke sættes nogen rammebetingelser og de to lambda-funktioner i optimeringsligningen (2.1) sættes derfor til 0

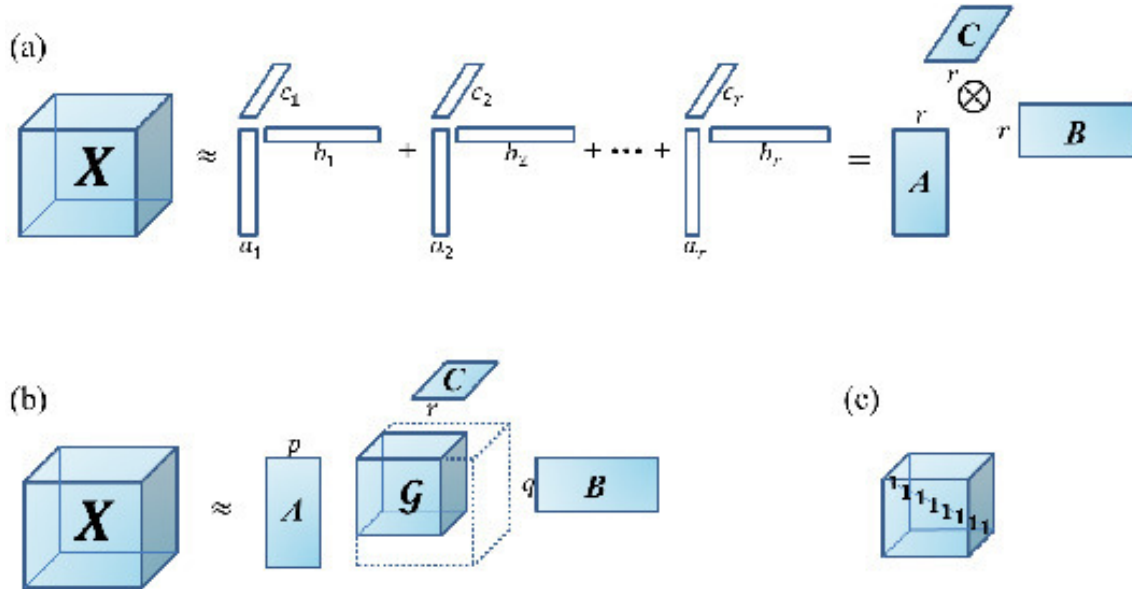
Det der kendetegner NMF algoritmen er, at hverken U eller V har nogen negative værdier, ligesom Y heller ikke har negative værdier. Denne "Non-negative" rammebetingelse er gældende inden for mange problemstillinger i naturvidenskab, eksempelvis målinger af kemiske stoffer i grundvand.

I et studie fra 1999, demonstrerede Lee og Seung [12], at - sammenlignet med andre decompositions-metoder bl.a. PCA - kan den non-negative rammebetingelse sikre, at delelementer af et signal kan identificeres i den faktorerede repræsentation. I studiet blev billeder af ansigter analyseret med hhv NMF, PCA, HQ, og alle 3 modeller kunne genskabe et af de originale billeder med stor nøjagtighed.

NMF algoritmen kunne dog, som den eneste af de tre algoritmer, separere de faktiske delelementer af et generaliseret "ansigtssignal" i form af et næsebor, et øje en mundvig osv., der efterfølgende kunne sammensættes til det modellerede ansigt. Til sammenligning var de faktorerede signaler i PCA modellen ikke genkendelige som ansigtselementer.

2.2 Højere ordens dekompositioner

Matematikken bag matrix dekompositioner kan generaliseres til højere ordens dimensioner, ved at betragte tensorer som serier af matricer. Dette princip kaldes for *Higher Order Singular Value Decomposition* (HOSVD), og en grafiske repræsentation af en 3 dimensionel tensor er vist på figur 2.2 [19], [27].



Figur 2.2: Higher Order Singular Value Decomposition (HOSVD)

De to vigtigste variationer af HOSVD modellen er CANDECOMP/PARAFAC ("CP") og *Tucker*. I CP er der den rammebetingelse, at kerne tensoren (benævnt G) skal være diagonal 2.2 (c), mens dette ikke er tilfældet for Tucker [19], [27].

2.3 Blind Source Separation (BSS)

Blind Source Separation er en af de problemer, som kan løses med Tensor Decomposition. Den klasiske problemstilling, som ofte benyttes til at illustrere BSS kaldes "*the cocktail-party problem*". Dette omfatter et lokale, hvor et antal mennesker går rundt mellem hinanden og snakker. I lokalet er der hængt et antal mikrofoner op, som optager lyden i rummet.

Hver mikrofon optager et mix af alle lyde i lokalet dvs. samtaler, klirende glas, baggrundstøj osv., og signalet fra en given mikrofon er påvirket af bl.a. mikrofonens afstand til hver lydkilde. For at opløse signalerne fra mikrofonerne og isolere en given samtale i lokalet, kan BSS anvendes. ([15] i [1]). De fleste decompositions-algoritmer kan bruges til BSS, og omfatter typisk en eller anden form for rammebetingelse som f.eks. *maximum variability*, *statistical independence*, *non-negativity*, *smoothness*, *sparsity*, *simplicity*.

Mange af metoderne er dog udfordret af det, der kaldes *underdetermination*, hvilket specielt gør sig gældende for sparsomme datasæt med mange manglende værdier.

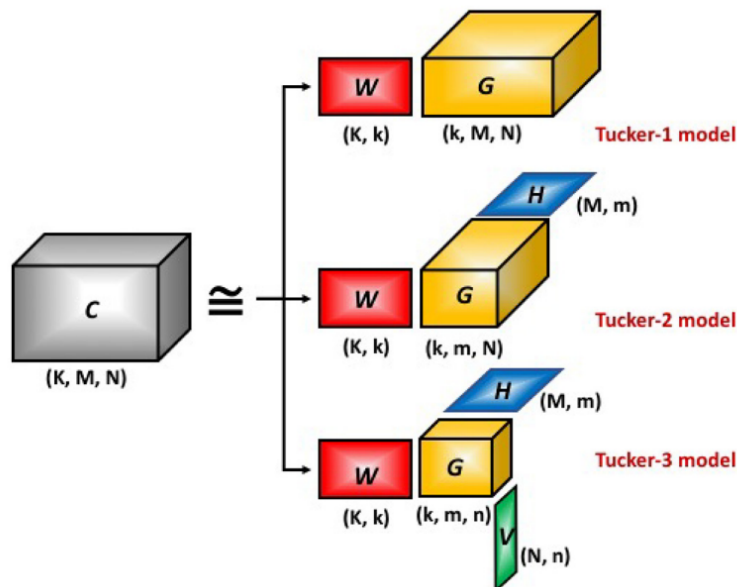
Netop fordi NMF- og NTF-algoritmerne er effektive til håndtering af sparsomme datasæt, er disse algoritmer også blandt de bedste til at løse problemet med underdetermination i forbindelse med BSS [1].

2.4 NMFk og NTFk

NMFk- og NTFk-algoritmerne er som navnet antyder baseret på NMF og NTF, og er endvidere koblet til *k-means*-algoritmen. NMFk og NTFk kan anvendes til feature extraction, blind source separation, detektion af anomalier og billede genkendelse m.m. Algoritmerne er udviklet af Velimir V Vesselinov som en del af en større softwarepakke til dataanalyse kaldes *Mads (Model Analysis & Decision Support)* fra *Los Alamos National Laboratory* [32]. Algoritmerne er publiceret i hhv. [1] og [36], og er tilgængelige som open source pakker fra [33] og [34].

NMFk- og NTFk-algoritmerne er baseret på *Tucker-decomposition* (se foregående afsnit), og NTFk er en generalisering af NMFk til datasæt med 3 eller flere dimensioner. Alt efter dimensionerne af datasættet, eksisterer der flere forskellige modeller for dekompositionen, hvor antallet af faktoreriserings-matricer og størrelsen af kerne-tensoren (G) kan variere [36]. På figur 2.3 kan den 3 dimensionelle tensor C opløses til en kerne-tensor G , samt 3 variationer af faktoreriserings-matricer. De 3 dimensioner i C har i Vesselinovs studie været borer, stoffer og tid - dvs. målte stofkoncentrationer i vandprøver fra borer over tid. Problemstillingen kan sammenlignes med cocktail-party eksemplet, ved at betragte lokalet som et område med et antal forureningskilder (personer) og et antal monitoringsboringer (mikrofoner) der måler et signal fra forureningskilderne.

I Vesselinovs studie er det Tucker-1 modellen der anvendes, og resultatet er en faktoreriserings matrice W , der repræsenterer de kemiske signaturer for hver "*grundvandstype*" dvs. forureningskildernes kemiske signaturer, samt kerne-tensoren G , der repræsenterer forureningskildernes påvirkning af borerne dvs. et "*opblandingsforhold*" for forureningskilderne i hver boring over tid.



Figur 2.3: NTFk

Den matematiske løsningen af ovenstående baseres på følgende rammebetingelser:

- Data i både G og W er positive (Non-negative)
- Forureningskildernes opblandingsforhold i hver boring skal kunne summeres til 0 for hvert tids-observation.

Løsningen af Tucker-1 modellen forudsætter endvidere, at der er en linær interaktion mellem datasættets features. Hvis interaktionen mellem features ikke er linær, kan NTFk ikke anvendes på nuværende tidspunkt [36].

Kørsel af NMFk/NTFk

Når det rette datasæt er genereret, kan NMFk-algoritmen køres på datasættet med kommandoen:

```
NMFk.execute(T, nkrange, nNMF, kwargs ...)
```

Hvor T er data-tensoren, der skal være udformet som et Julia array, med 2 eller flere dimensioner. Parameteren $nkrange$ er en talserie f.eks. $2:10$, som angiver det antal mulige kilder, som algoritmen skal teste løsninger for. Denne parameter bruges til k-means cluster-analysen, som er indbygget i NMFk/NTFk. Parametern $nNMF$ er antallet af NMF-gennemkørsler der skal bruges for at finde den optimale NMF-løsning. Defaultværdien her er 10.

Algoritmen fungerer ved at iterere over talserien i $nkrange$. For hvert tal (k) i $nkrange$, gennemføres en NMF-kørsel, med optimering over f.eks. 10 gentagelser (hvis $nNMF$ er sat til 10). På den optimale løsning bliver der derefter kørt en k-mean cluster analyse (initieret med k clustre). Når hele $nkrange$ er gennemført, evalueres cluster-løsningerne ved hjælp af parametrene Fit, Silhouette og AIC, og den optimale k -værdi - dvs. antal kilder findes. et output fra en NMFk-kørsel er vist på figur 2.4.

```

OF: min 0.0008701585972911654 max 0.0010514486959002983 mean 0.000915369415628309 std 6.295880873231395e-5
[ Info: Cluster assignments:
10x10 Array[Int64,2]:
1 4 3 1 8 8 1 1 6 1
2 8 1 6 5 2 7 6 2 6
3 6 6 8 2 1 9 3 4 5
4 1 8 5 1 3 4 8 1 3
5 3 2 2 3 4 3 7 5 7
6 2 7 7 6 7 10 9 7 2
7 7 4 4 7 9 2 2 9 9
8 9 9 3 4 10 5 5 8 10
9 5 5 9 9 5 6 10 10 4
10 10 10 10 10 6 8 4 3 8
[ Info: Cluster centroids:
10x7 Array[Float32,2]:
5.67299e-6 1.14118e-5 7.31087e-6 4.88909e-6 4.90072e-6 4.07769e-6 7.107e-6
0.000320023 0.000231585 0.00581133 0.000403551 0.000111972 0.000145181 4283.11
0.000188524 0.000146531 0.00961501 9.44522e-5 1.41717 6.39864e-5 0.000306756
0.00351114 0.00263093 3.69111e-5 0.00168229 0.00169471 0.695005 0.00163088
2.92893e-5 72.4422 0.10183 2.10236e-5 1.94708e-5 1.45515e-5 4.83978e-5
41.4382 0.00225334 0.0100708 0.000397241 0.000465406 0.000224043 0.00267729
5.67299e-6 1.14118e-5 7.31087e-6 4.88909e-6 4.90072e-6 4.07769e-6 7.107e-6
0.00231871 0.000574503 0.000112265 79.7869 0.00021436 0.000429226 0.00407643
5.67299e-6 1.14118e-5 7.31087e-6 4.88909e-6 4.90072e-6 4.07769e-6 7.107e-6
5.67299e-6 1.14118e-5 7.31087e-6 4.88909e-6 4.90072e-6 4.07769e-6 7.107e-6
[ Info: Silhouettes for each of the 10 clusters:
1x10 Array[Float32,2]:
-0.0999954 1.0 1.0 0.999999 0.999999 1.0 -0.0999954 1.0 -0.0999954 -0.0999954
Mean silhouette = 0.5600016415119171
Min silhouette = -0.09999537467956543
Objective function = 0.00087017973 Max error = 0.026631353 Min error = NaN
Signals: 10 Fit: 0.0008701797 Silhouette: -0.09999537 AIC: -4188.006
[ Info: Results
Signals: 2 Fit: 1554.75 Silhouette: 1 AIC: 4207.376
Signals: 3 Fit: 583.6803 Silhouette: 0.9999999 AIC: 4713.874
Signals: 4 Fit: 3.079072 Silhouette: 0.9999978 AIC: -2874.615
Signals: 5 Fit: 1.248511 Silhouette: 0.999848 AIC: -2221.896
Signals: 6 Fit: 0.7214782 Silhouette: 1 AIC: -896.7676
Signals: 7 Fit: 0.0008893694 Silhouette: 0.06241607 AIC: -11244.61
Signals: 8 Fit: 0.454227e-05 Silhouette: -0.7442139 AIC: -13132.91
Signals: 9 Fit: 0.0008834251 Silhouette: -0.09999713 AIC: -6525.334
Signals: 10 Fit: 0.0008701797 Silhouette: -0.09999537 AIC: -4188.006
[ Info: Optimal solution: 6 signals
ak@207b50f95e2:~/work/scripts$

```

Figur 2.4: Resultat af en NMFk kørsel med 2-10 clustre

Fit er et mål for modellens nøjagtighed baseret på ligning 2.1 - det er med andre ord det afstandsmål mellem model og virkelighed, som også kaldes "Frobenius normen". Siluetten er et normalt mål, som viser "robustheden" i det fundne antal klynger - dvs. antal punktkilder. AIC står for "Akaike Information Criterion", og er et almindeligt anvendt mål for modelresultatet, baseret på afvigelsen mellem datasættets sandsynlighedfordeling og modelresultatets sandsynlighedfordeling.

3. Systemdesign

3.1 Etablering af cloud-baserede analysemiljø

Ud over at afprøve NTFk-akgoritmen, er formålet med dette projekt også at komme i gang med at arbejde på regionens cloud-plattform *Azure*. Jeg ønsker derfor at undersøge, hvordan jeg kan etablere et analysemiljø i skyen, der kan bruges til at teste NTFk-algoritmen i dette projekt, men også bruges til fremtidige projekter. Samtidig skal jeg finde ud af, hvordan man tilgår og arbejder i dette miljø mest hensigtsmæssigt.

Docker er en ny container-baseret teknologi, som der er en masse ”hype” omkring - både inden for almindelig software udvikling og inden for data science. Stort set alle open source applikationer leveres i dag som et Docker-image. Jeg har meget begrænset erfaringer med Docker, men ønsker som udgangspunkt at forsøge at bygge systemet op med Docker. Dette er ikke strengt nødvendigt for at kunne gennemføre projektet, men projektet her er en fin mulighed for at få erfaring med container-teknologier, som formentlig er kommet for at blive.

Kravspekifikation

Analysemiljøet skal således bygges ved hjælp af en eller flere Docker-containere, og skal kunne køre i regionens Azure-tenant. Analysemiljøet skal kunne tilgås via en grafisk brugerflade, hvor jeg skal kunne arbejde interaktivt med de mest almindelige sprog inden for Data Science og lave grafiske visualiseringer af data. Adgang til Julia er en forudsætning, da Vesselinovs kode bliver stillet til rådighed som Julia pakker.

Fra miljøet skal der være adgang til opdaterede kopier af Regionens fagdatabaser, således at der løbende kan laves dynamiske dataudtræk. Der skal også være adgang til kopidatabaserne fra et specialiserede SQL-værktøj, som f.eks. *SQL Server Management Studio* (SSMS) eller *Azure Data Studio*, således at der kan arbejdes målrettet med opbygning af standardiserede views, rapporter og vedligeholdelse af dataset.

De scripts der opbygges i analysemiljøet skal sammen med resultater fra model-kørsler endvidere kunne lagres på en måde, så de kan eksporteres / downloades til det lokale miljø.

Systemdesign

Som udgangspunkt for miljøet har jeg sat en Virtuel Maskine op, med Ubuntu 18.04 som styresystem. Den virtuelle maskine har fået navnet ***** (herefter

kaldet lvm01) og opfylder dermed regionens standard for navngivning af ressourcer i Azure. På denne maskine har jeg installeret Docker, og det er her mine Docker-containere kører.

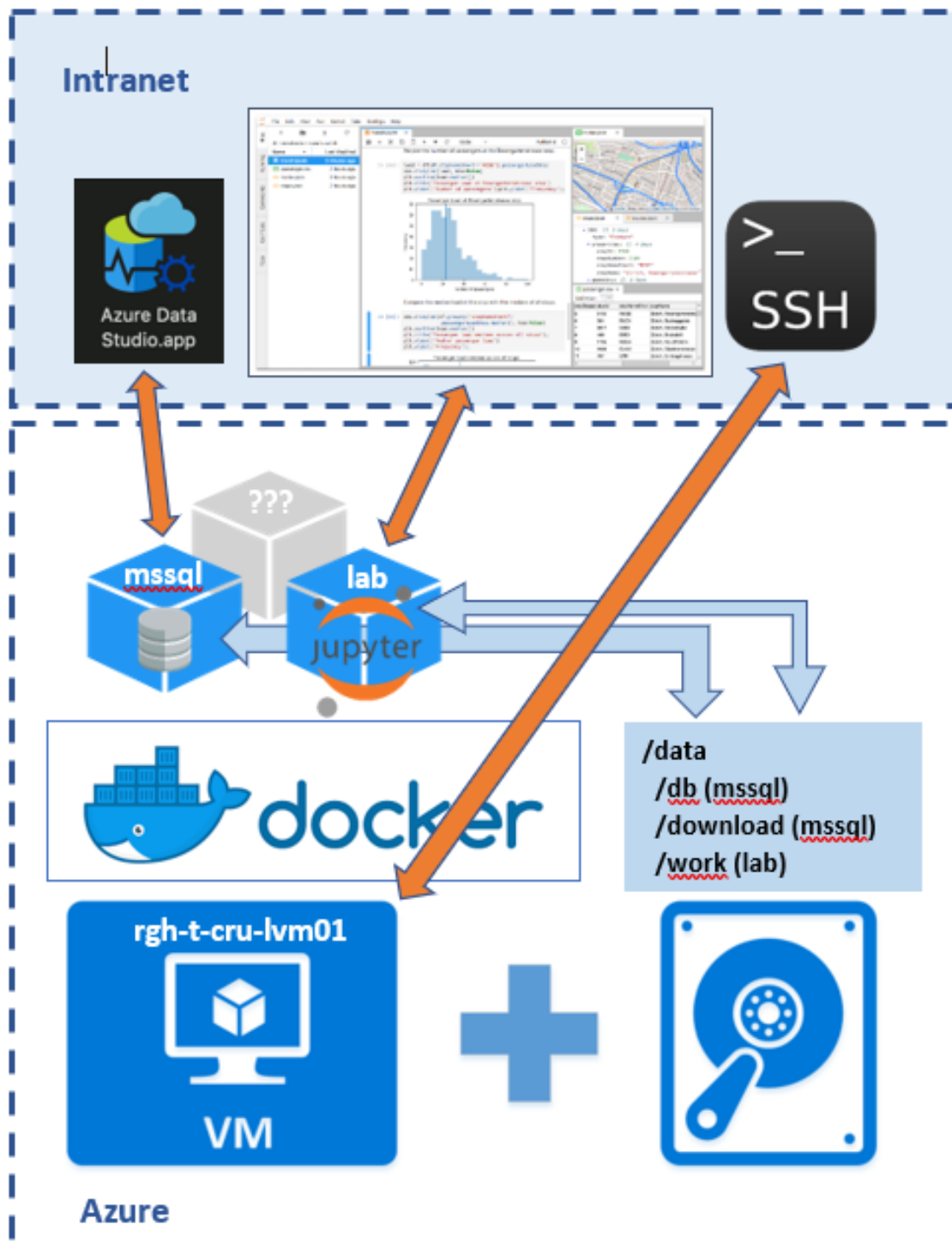
I Azure findes der faktisk forskellige ”container-services”, hvor jeg kunne have kørt docker-containere, men IT-afdelingen har lukket af for disse, så det har jeg ikke mulighed for. Endvidere giver det fin mening at have en virtuel maskine sat op som udviklings-miljø, hvor Docker images kan udvikles, konfigureres og afprøves inde de sættes i drift.

Den virtuelle maskine er blevet konfigureret med en harddisk, der giver mulighed for persistent datalagring. Harddisken er mountet på mappen */data* på den virtuelle maskine, og harddiske er opdelt mapperne */db*, */download* og */work*. På den virtuelle maskine er Docker installeret, hvor der kører flere Docker-containere. De to vigtigste for dette projekt er en MS SQL Server container med navnet **mssql** samt en Jupyter-Lab container med navnet **lab**. Begge disse containere har jeg udviklet, dvs. sat op og konfigureret, som en del af dette projekt, så de opfylder mine kravspecifikationer. Dette vil blive beskrevet nedenfor.

Containeren *mssql* anvender mappen */data/db* til persistent data-lagting, mens *lab* anvender mappen */data/work*. Adgangen til den virtuelle maskine styres med IP-filtrering på portnumre samt almindeligt logon. Der er åbnet for IP-adresser på regionsgården samt min IP-adresse derhjemme. Til eksamen vil jeg endvidere åbne, så jeg kan tilgå miljøet fra DTU. For de udvalgte IP-adresser er der åbnet for følgende portnumre:

- Port 22 - adgang til lvm01 via ssh
- Port 1433 - mappet til container *mssql*
- Port 8888 - mappet til container *lab*

De to docker containere er endvidere forbundet med et virtuelt netværk, således at de kan kommunikere med hinanden. En princip-skitse af infrastrukturen er vist på figur 3.1



Figur 3.1: Infrastruktur - systemdesign

3.2 Dataservice - opbygning af containeren *mssql*

(Udskrift af Dockerfile og de væsentligste scripts kan ses i bilag C og D)

Database containeren *mssql* kan ses som systemets "bagende", hvor de datasæt, der skal analyseres kan genereres fra og eksterne data og resultater evt. lagres til. Alle regionens fagsystemer er baseret på MS SQL Server, og natlige kopier kan downloades

som bak-filer, der kan restores. Microsoft stiller en grydeklar SQL Server til rådighed som docker-container, men jeg har i dette projekt valgt at lave et tilpasset image, der vha. et cron-job holder kopierne af vores fagdatabaser opdateret.

Databaserne ville også godt kunne opdateres fra host-maskinen eller andre steder fra, men en af de væsentlige pointer med containere er at have så få afhængigheder til eksterne systemer som muligt. Ved at etablere opdateringsjobbet i selve containeren, opnås det der kaldes et *self-contained system*, og containeren kan derfor uden videre flyttes til andre servere uden yderligere tilpasninger.

Base image og konfiguration

Microsofts Docker image til SQL Server (https://hub.docker.com/_/microsoft-mssql-server) er anvendt som base image. Dette billede etablerer en SQL Server 2017 Developer Edition på en Ubuntu 16.04. Developer Editions må anvendes til test og udviklingsformål, men ikke til systemer i produktion. Licensen vurderes dermed at være tilstrækkelig til dette projekts formål.

I containeren installeres programmerne zip, unzip og cron, der skal bruges til opdateringsjobbet. Den videre konfiguration omfatter at kopiere en .net fil med loginoplysninger til download af databaser, samt et entrypoint-script, der køres når containeren starter. Desuden sceduleres et cronjob vha crontab.

Scriptet entrypoint.sh

Entrypoint scriptet starter SQL serveren og kører derefter scriptet restore-data.sh. Restore-scriptet køres således når containeren startes, samt efter den scedulerede plan, som i dette tilfælde er sat til hver nat kl. 00:00.

Desuden startes cron med kommandoen *system cron start*, men denne sidste del af scriptet virker af en eller anden grund ikke. Som et led i at starte containeren, skal man derfor på nuværende tidspunkt åbne en bash-prompt i containeren, og manuelt starte cron. Når man har gjort dette fungerer opdateringsjobbet.

Scriptet restore-data.sh

Scriptet *restore-data.sh* gennemgår de 3 downloadede filer i mappen /data/download. Hvis en eller flere af filerne er mere end 20 timer gammel, downloades en ny version, og den pågældende database restores med programmet *sqlcmd*, samt et sql-script.

Docker run og parametre

Hvis billedet bygges med navnet local/mssql, kan containeren køres med kommandoen:

```
docker run -name=mssql \
```



```
-p 1433:1433 \  
-e ACCEPT_EULA=yes \  
-e SA_PASSWORD=XXXXXXX \  
-v /data/mssql:/var/opt/mssql \  
-v /data/downloads:/downloads \  
local/mssql
```

Bemærk at host-computerens port 1433 mappes til containerens port 1433 og at mapperne `/data/mssql` og `/data/downloads` på hostcomputeren stilles til rådighed for containeren som persistent datalagring.

Efter containeren er startet, vil der gå 30 sekunder for at sikre at SQL Serveren også er startet, hvorefter scriptet `restore-data.sh` eksekveres (ses med `print` på skærmen og evt. download af ny database).

3.3 Analysemiljøet - opbygning af containeren *lab*

(Udskrift af Dockerfile kan ses i bilag E)

Udgangspunktet for containeren *lab* var i starten docker billedet *jupyter/datascience-notebook* (<https://hub.docker.com/r/jupyter/datascience-notebook/>), der er en Jupyter-notebook / JupyterLab distribution, med adgang til Julia. Julia er som sagt en forudsætning for at kunne køre Vesselinovs kode, og jeg har fra tidligere projekter gode erfaringer med Jupyter som udviklingsplatform. Til at starte med kunne NMFk- og NTFk-pakkerne dog ikke køre på denne container, pga. afhængigheder til manglende biblioteker og programmer på styresystemet.

Vesselinov har publiceret et docker billede, som kan køre de to pakker uden fejl, og dette docker billede kunne også have været valgt, men det er ikke konfigureret med henblik på at levere en Jupyter-server. Ved at sammenligne de to billeder, vurderede jeg at konfigurationen af NMFk/NTFk var enklere, og ud fra Vesselinovs Docker image, lykkedes det at få NMFk/NTFk-koden til at køre i en *datascience-notebook*, ved at tilføje Vesselinovs konfiguration og genbygge billedet.

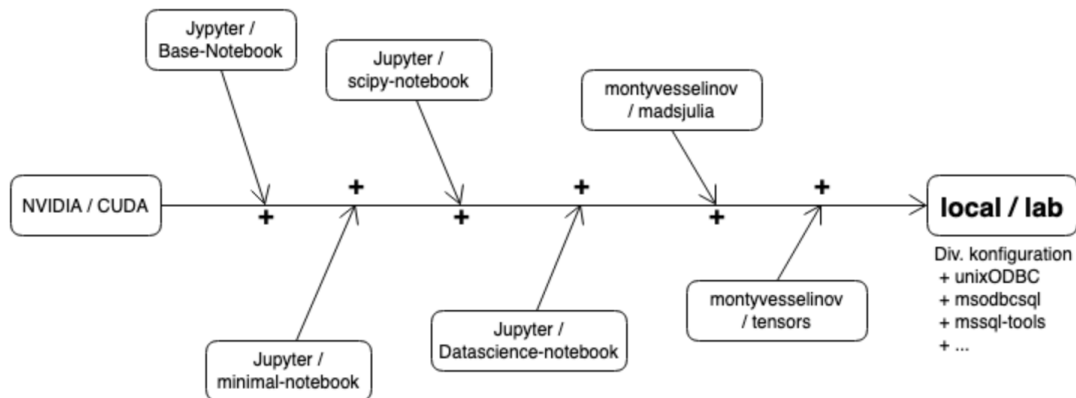
Ca. halvvejs inde i projektet besluttede jeg også at ville indbygge understøttelse af GPU-beregninger (CUDA) i *lab*-containeren. NMFk/NTFk tager lang tid at køre, hvis det er lidt større datasæt, og generelt er der flere og flere biblioteker der understøtter GPU-beregninger. Dette har ikke været nogen nem opgave, men det er lykkedes, og fremadrettet vil jeg også kunne bruge f.eks. TensorFlow eller andre GPU-understøttede software-pakker i analysemiljøet.

Base image og konfiguration

Det billede jeg er endt med at bygge, bruger *nvidia/cuda:latest* som base-image (<https://hub.docker.com/r/nvidia/cuda/>). Billedet er bygget op omkring en Ubuntu 18.04, hvor *CUDA toolkit* fra NVIDIA er blevet installeret. Med dette base image kan man give *docker run* kommandoen en parameter, der instruerer containeren i at gøre brug af en eller flere GPU'er (se nedenfor). En forudsætning for at dette virker er, at de tilgængelige GPU'er er NVIDIA-kompatible og at *NVIDIA Container Runtime* (<https://github.com/NVIDIA/nvidia-docker>) er installeret på host-computeren,

og dette er også gjort på **lvm01**.

Ved at tage udgangspunkt i dette billede, har det endvidere krævet, at jeg har overført de nødvendige installationer og konfigurationer fra hhv. *jupyter/datascience-notebook* billedet og fra Vesselinovs NMFk/NTFk-billeder. Datascience-notebook billedet er i sig selv baseret på en længere række af andre jupyter-billeder (kaldet Jupyter Docker Stacks - se <https://github.com/jupyter/docker-stacks>), så opbygningen af dette billede har været et ret omfattende stykke arbejde. En principskitse af dette og de billeder der er indgået i opbygningen af billedet er vist på figur 3.2.



Figur 3.2: Principskitse for opbygning af Docker billede til lab-containeren. (se bilag E)

Julia og NMFk/NTFk installation

Installationen af Julia er kopieret fra docker-billedet *jupyter/datascience-notebook*, og ikke fra Vesselinovs docker-billede. Dette har jeg valgt for at sikre integration med *jupyterLab*-servicen. Installationen af Julia kan ses i dockerfilen linje 254-277 (bilag E).

Endvidere er der installeret nogle basis-pakker i julia, som bliver prækompileret og dermed umiddelbart tilgængelige i analysemiljøet. Dette er bl.a. valgt for pakkerne NMFk og Mads, da dette er store pakker, der tager lang tid at kompilere. Installation af julia pakker kan ses i Dockerfilen linje 307-337 (bilag E).

Forbindelse til database-containeren *mssql*

For at kunne lave dataudtræk fra database-containeren i en given Julia-baseret notebook i analysemiljøer, kræver det at man bruger julia-pakke ODBC. Pakken ODBC kræver at programmerne *unixODBC*, *msodbcsql117* og *mssql-tools* blevet installeret i containeren, og dette er gjort i Dockerfilens linje 63-76 (bilag E).

Docker run og parametre

Hvis dockerfilen bygges til et billede med navnet *local/lab*, kan container køres med kommandoen:

```
docker run -name=lab \
    -gpus=all \
    -p 8888:8888 \
    -e JUPYTER_ENABLE_LAB=yes \
    -v /data/work/lab:/home/ak/work \
    local/lab
```

Bemærk parameteren `-gpus=all`, som stiller alle host-computerens GPU'er til rådighed for containeren. Endvidere mappes host-computerens port 8888 til samme port i containeren, hvorfra Jupyter-serveren kan tilgås. Mappen `/data/work/lab` på host-computeren mappes til den mappe på containeren, som man arbejder i fra JupyterLab.

3.4 Orkestrering af miljøet med Docker-Compose

(Se YAML-file i bilag F)

Docker-Compose er en "add-on" til Docker, som skal installeret separat. Docker-Compose giver mulighed for at orkestrere flere containere der tilsammen udgør et komplet miljø på samme til. Orkestreringen sker ved hjælp af en YAML-fil, hvor man både kan specificere hvilke containere der skal startes og evt. bygges, samt hvilke parametre der skal sættes når containerne startes og evt. om containerne skal bygges inden de startes.

Samtidig kan man i YAML-filen definere et fælles virtuelt netværk, så containere i miljøet kan kommunikere indbyrdes, samt definere fælles volumer m.m. Jeg har i dette projekt lavet en YAML-fil, hvor jeg starter containeren `mssql`, samt et par andre containere. I starten af projektet startede jeg også lab-containeren med `docker-compose`, men `docker-compose` understøtter ikke *NVIDIA Container Runtime* på nuværende tidspunkt, og parameteren `-gpus=all` kan ikke gives til lab-containeren via Docker-Compose.

På nuværende tidspunkt starter jeg derfor `mssql`-containeren og sammen med et par andre containere med kommandoen:

```
docker-compose up &
```

.. og derefter starte lab-containeren som vist i foregående afsnit. Når jeg starter lab-containeren, giver jeg også parameteren `-network=afgangsprojekt_default`, som er det virtuelle netværk der etableres med Docker-Compose.

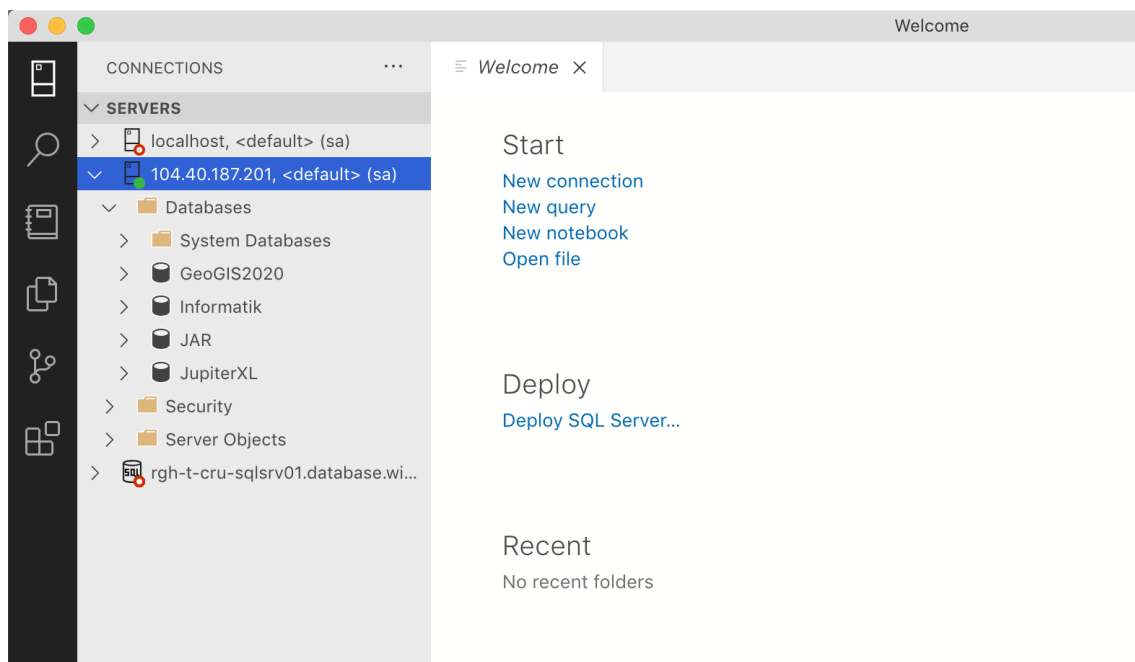
Miljøet kan efterfølgende stoppes med kommandoen:

```
docker-compose down
```

4. Datagrundlag for test

I dette kapitel vil det datagrundlag, der er lagt til grund for testen, blive beskrevet. Jeg har valgt at arbejde med et datasæt fra Birkerød indvindingsopland, da dette er et af de oplande i Region Hovedstaden, som er meget velundersøgt. Regionen har arbejdet i Birkerød indvindingsopland siden kommunalreformen i 2007 og regionen har på nuværende tidspunkt et godt overblik over de forureningskilder i oplandet, som udgør en risiko for drikkevandforsyningen. Som følge deraf bør der være mange data til rådighed, og der burde også kunne genereres nogle lange tidsserier.

Som beskrevet i kapiten 3, er mssql-containeren sat op, så den selv opdaterer kopierne af fagdatabaserne. Disse består af 3 databaser: Jupiter, GeoGIS og JAR, hvoraf det kun er Jupiter og GeoGIS der indeholder vandanalyser og målte stofkoncentrationer. JAR databasen indeholder udelukkende administrative informationer om lokaliteter f.eks. historiske anvendelser, gennemførte undersøgelser og afgørelser. JAR databasen er dog alligevel taget med for at have et fuldstændigt datagrundlag - også til fremtidige projekter. Data fra JAR har ikke indgået i dette projekt, og vil derfor ikke blive beskrevet yderligere.



Figur 4.1: 4 databaser på SQL Serveren i mssql-containeren

Ud over de 3 kopidatabaser er der også oprettet en ekstra database med navnet *Informatik*. Tanken med denne database er, at man her kan oprette SQL-views, der

trækker data ud fra de tre kopidatabaser. Man ville også kunne importere og lagre eksterne data her og gemme resultater fra evt. analyser og modelkørsler. Grunden til denne konstruktion er, at de tre kopidatabaser bliver overskrevet når der *restores* en ny kopi ind i databasen hver nat. Det bliver Informatik-databasen ikke, og data og views i denne database bliver dermed persistente. De 4 databaser er vist på figur 4.1.

4.1 Datakilder til vandanalyser i dansk kontekst

Herunder gives et overblik over de potentielle datakilder, som kan anvendes til at generere et datasæt som afspejler geokemiske sammensætning af grundvandet i et område. Overordnet set er det kemiske analyser af vandprøver fra grundvandet, som vi er interesseret i, og de bliver skabt af forskellige aktører og med forskelligt formål.

Vandværkernes analyseprogrammer

Vandværkerne har ansvar for at sikre drikkevandskvaliteten, og har derfor også ansvar for at lave regelmæssige analyser af det råvand - dvs. grundvand - som de indvinder fra deres borer. Analyserne foretages efter et kontrolprogram kommunerne fastlægger som drikkevandsmyndighed. Vandværkerne har også ansvar for, at analyserne bliver indberettet til den nationale borings og vandkemidatabase (Jupiter), således at de bliver tilgængelige for offentligheden.

Der er et fast antal parametre der altid indgår i vandværkernes råvandsprøver, men der ud over kan kommunerne fastlægge en udvidelse af kontrolprogrammet, hvis der f.eks. findes en forureningskilde i oplandet, der kan udgøre en risiko. Disse data omfatter i overvejende grad kun vandprøver fra vandværkernes indvindingsboringer.

Frivillige kontroller og analyseprogrammer

Vandværkerne kan desuden vælge, selv at lave yderligere undersøgelser eller monitoringsprogrammer i oplandet. De vandanalyser, der bliver lavet frivilligt, er ikke omfattet af kravet om indberetning, og disse data ville skulle indhentes hos vandværket, hvis de skal indgå i datasættet. Andre frivillige undersøgelser f.eks. borgere eller virksomheder, der får analyseret en vandprøve i forbindelse med et projekt, er heller ikke nødvendigvis omfattet af krav om indberetning.

Regionernes analyseprogrammer

I forbindelse med regionernes forureningsundersøgelser bliver der lavet rigtig mange analyser af grundvandet, og disse vandanalyser bliver også indberettet. Dog er der en forsinkelse på indberetningen af disse vandanalyser, fordi regionerne ikke ønsker at offentliggøre belastende data for grundejer før der er truffet afgørelse om kortlægning af ejendommen og grundejer er blevet orienteret om dette.

Desuden er regionens undersøgelser, og dermed analyseprogrammer, målrettet afklaringen af om specifikke historiske aktiviteter på ejendommen har givet anledning til forurening. I forbindelse med en forureningsundersøgelse bliver der således sjældent analyseret for uorganiske komponenter, som der typisk bliver gjort i vandværkernes kontrolprogrammer. De parametre, der indgår i en specifik undersøgelse vil ikke nødvendigvis være de samme, som indgår i undersøgelsen på andre lokaliteter i oplandet.

Fokus på chlorerede opløsningsmidler i Birkerød

I de fleste undersøgelser i Region Hovedstaden er det chlorerede opløsningsmidler, der har været i fokus. Chlorerede opløsningsmidler er særligt problematiske for grundvandet, fordi de er mobile og svært nedbrydelige. Grænseværdierne for chlorerede opløsningsmidler er relativt lave (dvs. de er relativt giftige), og når de bliver nedbrudt, nedbrydes de typisk til andre chlorerede forbindelser, der er lige så giftige eller i nogen tilfælde giftigere.

Min vurdering er derfor, at et godt datasæt at teste NMFk-algoritmen af på, skulle omfatte og evt. afgrænses til chlorerede opløsningsmidler og nedbrydningsprodukter. Chlorerede komponenter ville formodentligt kunne give et større datasæt end f.eks. oliekomponenter, og bør indeholde signaler fra de forureningskilder, som vi har kendskab til i oplandet.

4.2 Generisk databaseudtræk til test af NTFk

Som det fremgår af ovenstående, er der flere udfordringer i forhold til at generere et samlet datasæt med med geokemiske grundvandsanalyser for oplandet. En stor del af de tilgængelige data findes som indberettede data i Jupiter-databasen, men ikke alle, så det vil være nødvendigt at kombinere data fra Jupiter med data fra GeoGIS databasen for at opnå et komplet datasæt.

Dog er mange af regionens data også indberettet til Jupiter, så dataudtrækket skal kunne frasortere disse data, så de kun forekommer en gang i datasættet. En udfordring i denne forbindelse er, at de to databaser har forskellige datamodeller, hvilket vanskeliggøre sammenligningen af data på tværs. Samtidig er der udfordringer med datakvaliteten af gamle data i databaserne, da der tilbage i tiden ikke var lige så mange krav til indberetning og kvalitetskontrol af data.

For at oprette et generisk datasæt, der kan ligge til grund for dette projekt og evt. fremtidige projekter omkring grundvandskemi, har jeg valgt af oprette et basisview for hver database, der leverer et datasæt med samme tabelstruktur. De to views hedder hhv. *JupiterSpeciesAmount* og *GeoGISSpeciesAmount*. Tabelstrukturen for disse to views ser således ud:

Kolonne	Datatype	Bemærkninger
CatchmentId	GUID	Identifikation af indvindingsoplandet, som boringen ligger i.
Catchment	varchar(254)	Navn på indvindingsoplandet.
WellNo	varchar(50)	Boringsnummer - identifikation af boring.
IntakeNo	integer	Filternummer - der hvor grundvandet strømmer ind i boringen.
X	float	Boringens X-koordinat
Y	float	Boringens Y-koordinat
Z	float	Ca. niveau, hvor vandprøven er udtaget baseret på filter top og bund
Date	datetime	Tidspunktet hvor vandprøven er udtaget
SpeciesNo	integer	Identifikation af analyseparameteren (stoffet)
Species	varchar(50)	Navnet på analyseparameteren (stoffet)
Amount	float	Den målt koncentration af stoffet
Unit	varchar(50)	Enheden på den målte koncentration f.eks. $\mu\text{g/l}$

Tabel 4.1: Fælles datastruktur for de to basis-views

Med ensartede felter i de to datasæt, er det nu muligt at samle de to datasæt med en UNION operater kombineret med SELECT UNIQUE. Datasættet er endvidere aggregeret over prøvedatoens årstal med MAX, således at datasættet viser den højeste målte koncentration i en borings filter pr. år. Aggregeringen er primært gjort for at reducerer antallet af NULL forekomster i det endelige datasæt, dels for at lægge mig tæt op ad Vesselinovs anvisninger omkring input-data. Dette samlede dataudtræk er implementeret i Informatik-databasen i viewet *CatchmentSpeciesAmount*. SQL-scripts for de 3 views er vist i bilag G.

Der er i dette projekt ikke indhentet data fra frivillige undersøgelser, som ikke er indberettet.

4.3 Geografisk afgrænsning af et datasæt

Der er ikke informationer om indvindingsopland i hverken Jupiter eller GeoGIS databasen. Indvindingsoplandene udpeges af Miljøstyrelsen i forbindelse med Statens grundvandskortlægning, og oplandene er dermed dynamiske. Indvindingsoplandene er desuden et spatialt datasæt, med en kolonne der indeholder oplandets geometri (Geom) i form af et polygon. For at kunne identificere hvilke boringer der ligger inden for hvilke oplande, har jeg oploadet regionens datasæt med indvindingsoplande til Informatik-databasen. Tabellen hedder eOpland.

Begge de to basis-views tager således udgangspunkt i tabellen eOpland, og det første INNER JOIN i de to views etablerer en relation mellem boringer og opland med funktionen:

```
... eOpland.Geom.STIntersects(Boring.Geom)=1
```

Herefter kan data filtreres med en WHERE klausul på indvindingsoplande. Man skal i forbindelse med anvendelse af de to basis-views være opmærksom på, at ind-

vindingsoplande kan lappe ind over hinanden. Det gør at boringen kan ligge i flere oplande og analyserne bliver i de tilfælde redundante - dvs. vist flere gange - i de to basis-views. Dette er dog ikke noget problem i det endelige dataudtræk, da der her sættes et filter på udtrækket i forhold til oplandet og der ud over også anvendes en SELECT DISTINCT. Men hvis man vil generere et datasæt for flere oplande, vil det give problemer.

4.4 Efterfølgende transformation af datasæt til 3 dimensionel tensor

Det, der kommer ud af det samlede dataudtræk, er en lang tabel, hvor hver række repræsenterer en målt koncentration. Rækken indeholder desuden oplysninger om hvilken boring og hvilket år koncentrationen er målt, samt hvilket stof koncentrationen er målt for.

Input-formatet til NMFk er omvendt et 3 dimensional array udelukkende bestående af målte koncentrationer, men hvor de 3 dimensioner repræsenterer hhv. boringer, stoffer og tid. Da datasættet er velstruktureret og indeholder alle de nødvendige oplysninger, er det dog relativt simpelt at transformere datasættet til det rigtige format. Julia-koden til at gøre dette er vist på figur 4.2, hvor variabelen T indeholder det endelige inputformat.

```
years = unique(sort((df.Year)))
wells = unique(sort((df.Wellno)))
species = unique(sort((df.Species)))
T = Array{Float32}(undef, length(wells), length(species), length(years))
T .= NaN
for i in 1:size(df, 1)
    iy = indexin([df.Year[i]], years)[1]
    iw = indexin([df.Wellno[i]], wells)[1]
    is = indexin([df.Species[i]], species)[1]
    T[iw, is, iy] = df.MaxAmount[i]
end
```

Figur 4.2: Transformerings af dataudtræk til 3D array i Julia

Eksempel på Jupyter notebook og script, hvor datatransformationen indgår kan ses i bilag H.

5. Test af NTFk

Da jeg i starten af projektet henvendte mig til Vesselinov, og spurgte, om jeg kunne få en kopi af hans kode, gjorde han mig opmærksom på, at det ville blive svært for mig at bruge koden, fordi den ikke er dokumenteret og kun indeholder meget få bemærkninger. Set i bak-spejlet kan jeg kun give han ret - ud over at NTFk-pakken ikke er dokumenteret, er den også under konstant opdatering, og opleves som værende ret ustabil

Vesselinov tilbød mig i stedet at analysere datasættet for mig, men da en væsentlig del af projektet har omfattet at implementere og teste NTFk-algoritmen i regionens eget miljø, aftalte jeg i stedet med Vesselinov, at han kunne hjælpe mig med nogle demo-kørsler på et dansk datasæt, som jeg så kunne gentage og teste resultaterne af. Jeg skylder Vesselinov en stor tak for denne hjælp, da jeg ikke havde kunne gennemføre projektet uden.

Efter Vesselinovs anvisninger har jeg brugt NMFk-pakken til at udføre nedenstående tests, men formålet har været at teste NTFk-algoritmen som beskrevet i [?]. Jeg har undret mig en del over dette, og her derfor spurgt Vesselinov, hvorfor det er NMFk pakken han har vist mig sine demo-kørsler med. Han har svaret mig følgende [31]:

"The boundaries between NMFk and NTFk are somewhat fuzzy. Your problem is tensor based. But because it has mixing constraints, I coded it in NMFk where this approach was originally coded for matrices as well. I may move it in NTFk sometime in the future."

5.1 Metode

Jeg har med udgangspunkt i Vesselinovs demo-kørsler lært at bruge udvalgte funktioner i NTFk pakken og derved selv kunne gennemføre yderligere test-kørsler på andre datasæt. Ud over en masse indledende prøvekørsler med henblik på at lære at bruge pakken, har jeg til sidst i projektet lavet 6 formaliserede test-kørsler, med det formål at afklare, om NTFk-algoritmen er anvendelig på regionernes vandkemi-data. Tabel reftab:tests viser et overblik og disse, og hvad der er lagt til grund.

For hver test er der oprettet en mappe i JupiterLab, og i mappen har jeg oprettet 2 standard Jupyter-notebooks og et standard Julia-script. Første notesbog (*nbGetdata.ipynb*) genererer test-datasættet vha. nogle standardiserede data-udtræk fra SQL serveren, der tilpasses testen vha. parametre. Det genererede datasæt til den pågældende test, gemmes i mappen som en csv-fil (data.csv). Dette har den fordel at datagrundlaget for de 5 tests er ensartet og bliver samtidig dokumenteret. Datasæt-

Test	Stoffer	Boringer	Bemærkninger
Test1	5 uorg.	5 boringer	Datasæt magen til Vesselinovs bud på noget simpelt der virker. Omfatter 5 uorganiske stoffer i 5 udvalgte boringer. Denne test er brugt til verificering af, at min kørsel giver samme resultat som Vesselinovs kørsel.
Test2	5 uorg.	5 boringer	Magen til Test 1, men kørt med normaliserede data. Test af om normalisering giver et andet resultat
Test3	5 chl. opl.	14 udvalgte boringer	Test af algoritmen på afgrænset datasæt med chlorerede opløsningsmidler. Denne test kan verificeres i forhold til den viden regionen har om chlorerede punktkilder i oplandet
Test4	5 chl. opl.	Alle boringer	Som test 3, men alle boringer med data - test af datasættets densitet
Test5	11 chl. opl.	10 udvalgte boringer	Flere stoffer men færre boringer end test 3. Test af stof-dimensionen for modelresultatet
Test6	Alle	Alle	Indledende forsøg med et fuldt datasæt.

tet kan også efterfølgende bruges til visualisering og evt. downloads.

Julia-scriptet (*Run_NMFk.jl*) køres efterfølgende. Her indlæses det genererede datasæt og den Tensor, som skal bruges som inputdata til kørslen oprettes. Scriptet kører derefter funktionen *NMFk.execute()*, processerer resultatet og gemmer disse som CSV-filer. Til sidst skrives en opsummering af testen til en *README.md* fil i testmappen, så dette er tilgængeligt efterfølgende. Uddrag fra README-filerne er vist for hver test nedenfor.

Grunden til at selve model-kørslen er lavet som et script, og ikke en Jupyter Notebook er, at kørsler med store datasæt tager lang tid, og hvis man mister forbindelsen til den notesbog, hvor modellen køres - f.eks. hvis PC'en går i dvale eller slukkes - så går jobbet i stå!

Med et script kan man i stedet logge på den virtuelle maskine, og her starte en session med *tmux*. I *tmux* logges der ind i en terminal (*bash*) på lab-containeren (dvs. ikke Jupyter servicen men containerens styresystem) og her kan scriptet så køres med Julia. Herefter kan man koble ud af *tmux*-session og logge ud af den virtuelle maskine. Scriptet vil fortsætte med at eksekvere i den kørende *tmux*-session på den virtuelle maskine. Og man kan løbende koble sig ind i den kørende *tmux*-session og checke fremdriften af testen.

Den sidste notesbog (*nbInspect.ipynb*) anvendes til at inspicere og visualisere testresultaterne. I nedenstående er udvalgte visualiseringer vist, og disse er lavet med denne notesbog for hver af de 6 tests. Der er lavet visualiseringer for alle boringer mht. estimerede koncentrationer mod faktiske koncentrationer i alle tests, men dette er et ret omfattende materiale. Nedenfor har jeg valgt kun at vise visualiseringen for en enkelt boring (W975), da der her ses variationer i data for både de 5 uorganiske stoffer og

nogle af de chlorerede stoffer.

5.2 Test 1

5 uorganiske stoffer i 5 borer uden normalisering af data.

Dato: 2019-12-11

Dir: /home/ak/work/NMFk/Birkerod/Test1

Dataset:

Size T: (5, 5, 28)

Wells: 5

Size years: 28

Size species: 5

Species:

SpeciesNo; Species

380; Carbon,org,NVOC

2022; Fluorid

2041; Jern

2056; Kalium

2081; Magnesium

Execute:

NMFk.execute(T, 2:10; quiet=false, scale=false)

Testresultat:

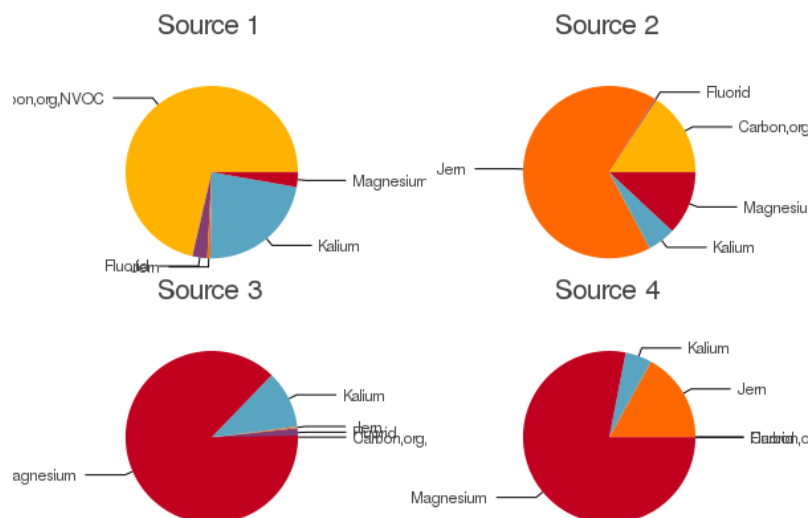
Tid: 3 minutes, 21 seconds, 134 milliseconds

Optimal solution: 4

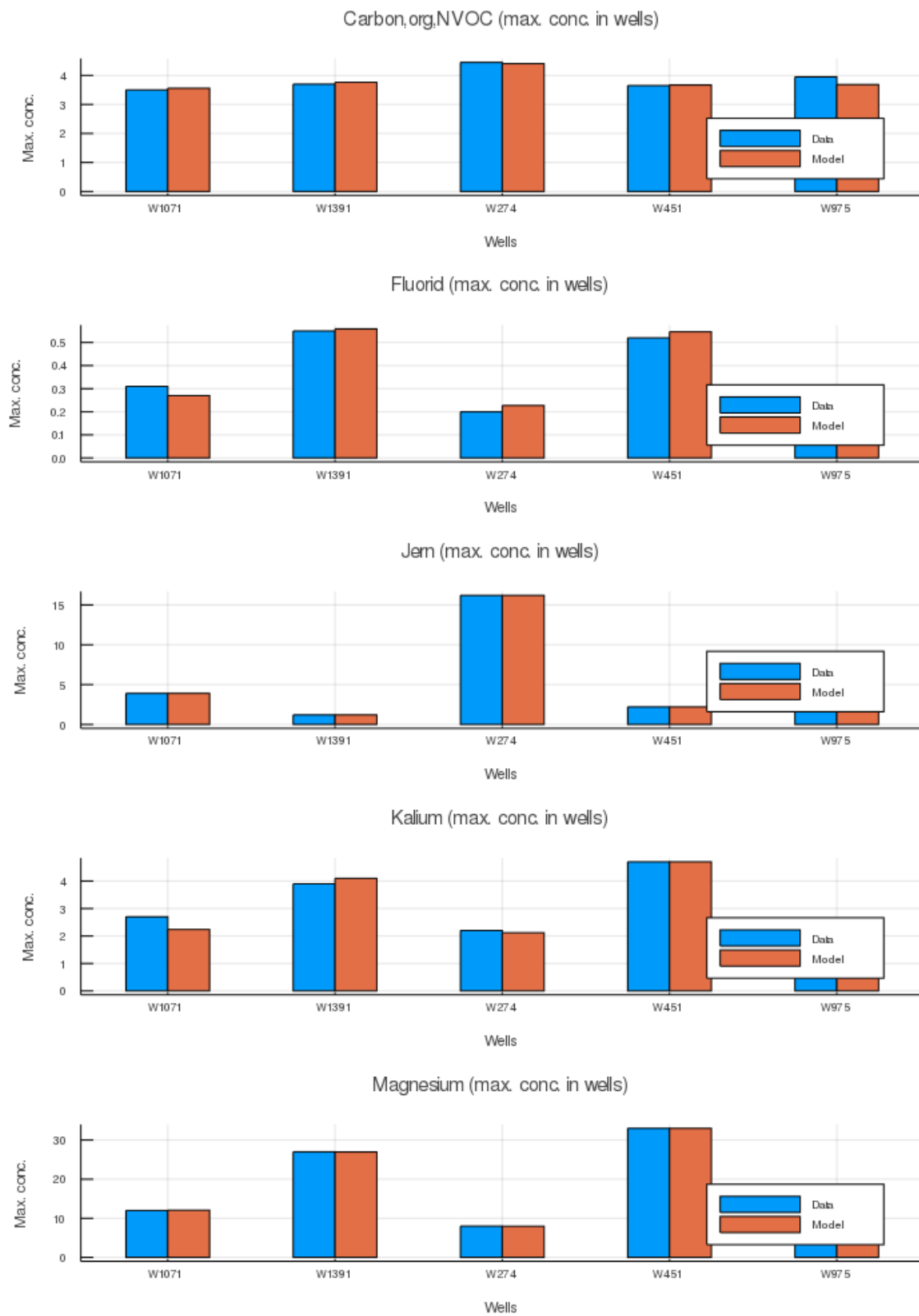
Fit: 4.887795

Silhouette: 0.99394673

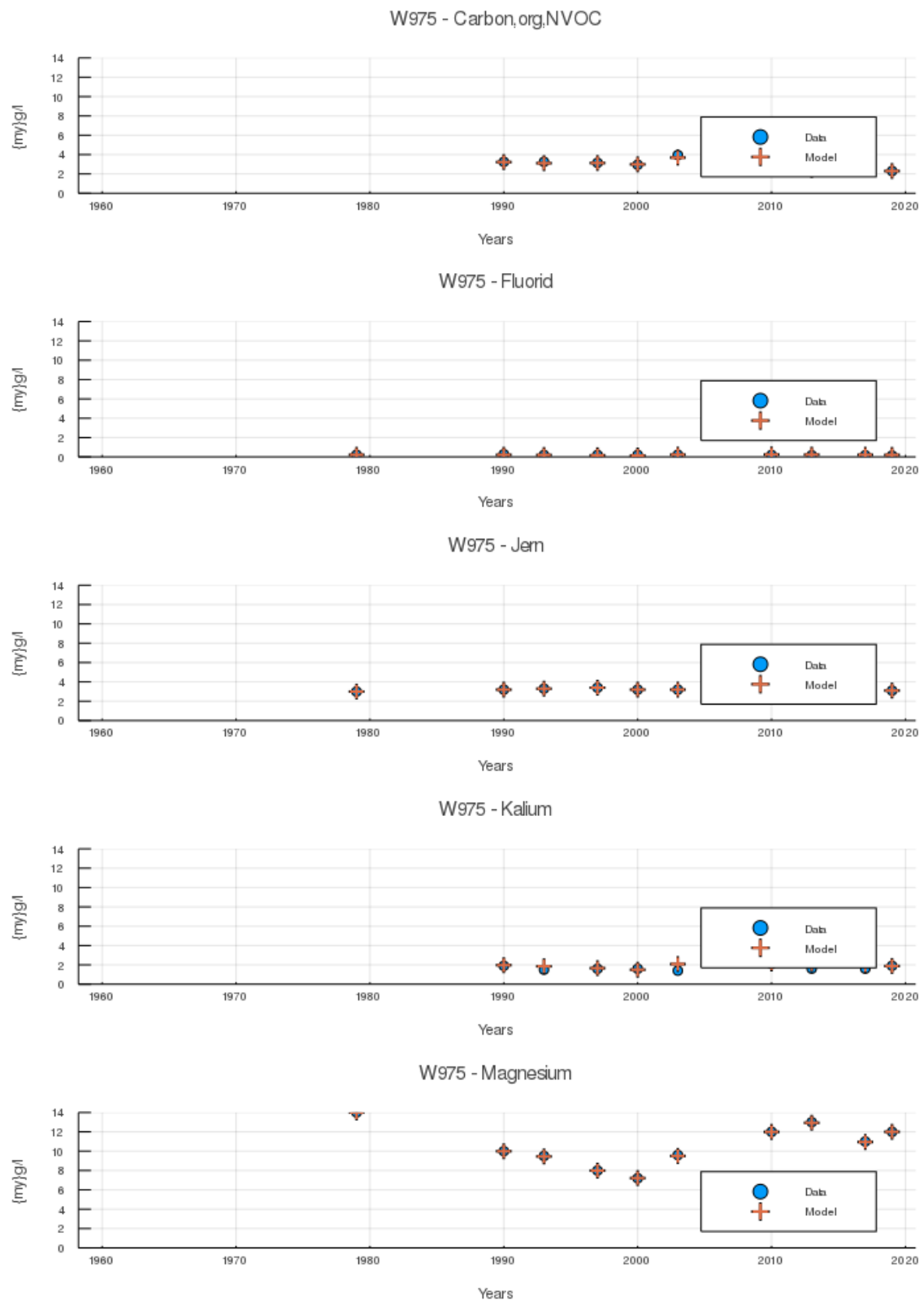
AIC: 246.86218



Figur 5.1: Test 1- den kemiske signatur for de 4 grundvandstyper



Figur 5.2: Test 1 - maks. konc. af stoffer i hver boring - estimeret vist mod faktisk



Figur 5.3: Test 1 - estimeret konc. plottet mod faktisk konc. over tid i boring W975 for 5 stoffer

5.3 Test 2

5 uorganiske stoffer i 5 boringer med normalisering af data.

Dato: 2019-12-11

Dir: /home/ak/work/NMFk/Birkerod/Test2

Dataset:

Size T: (5, 5, 28)

Wells: 5

Years: 28

Species: 5

Species:

SpeciesNo; Species

380; Carbon,org,NVOC

2022; Fluorid

2041; Jern

2056; Kalium

2081; Magnesium

Execute:

NMFk.execute(T, 2:10; quiet=false, scale=true)

Testresultat:

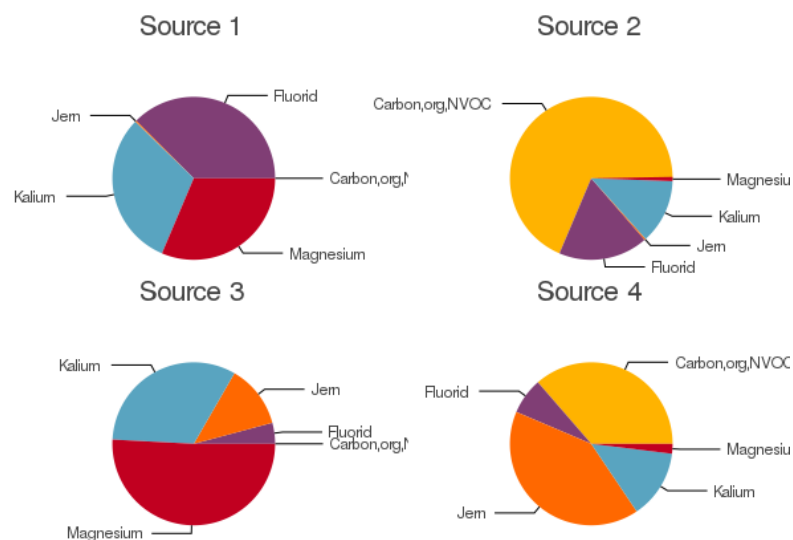
Tid: 2 minutes, 18 seconds, 841 milliseconds

Optimal solution: 4

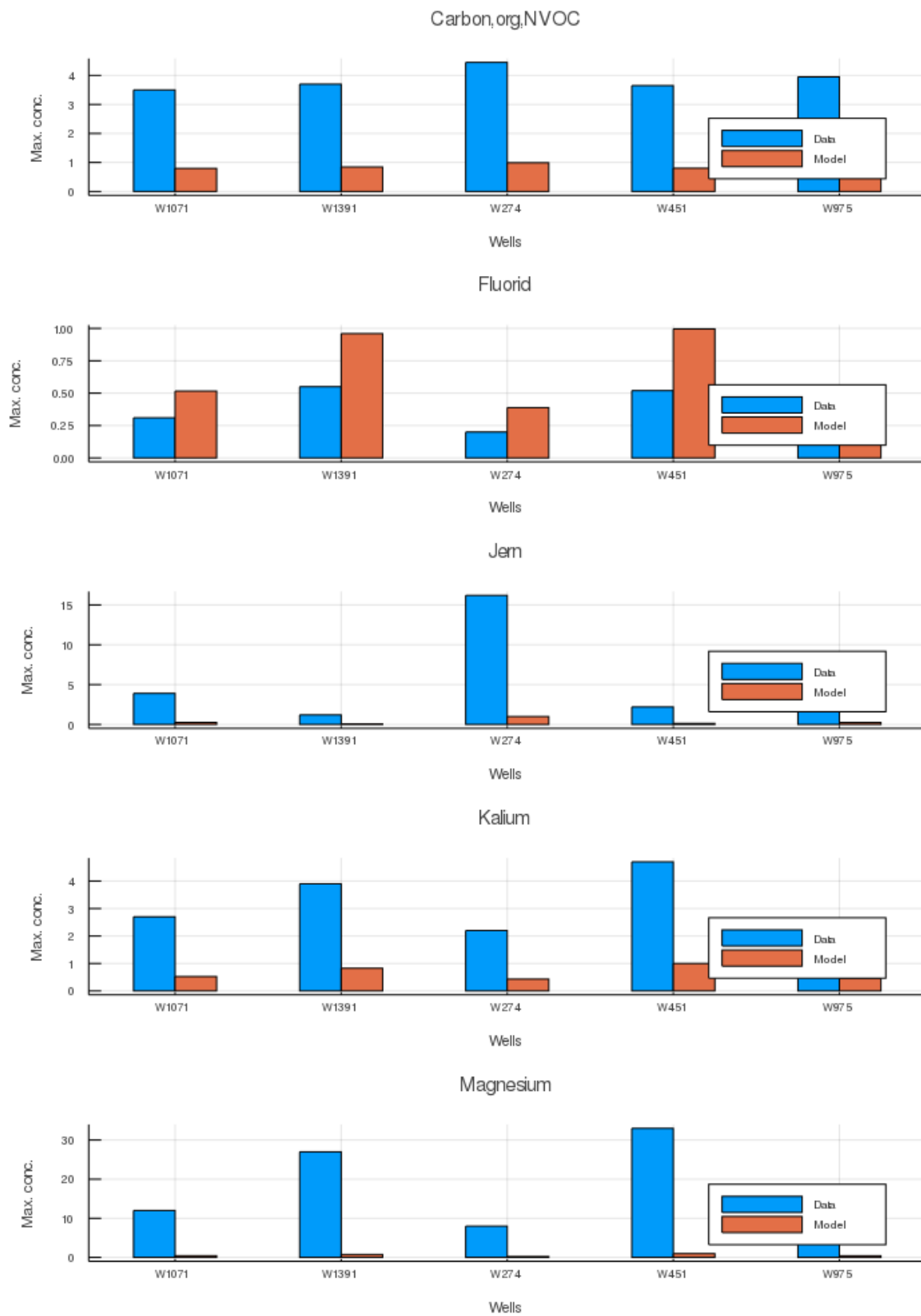
Fit: 0.27703032

Silhouette: 0.267032

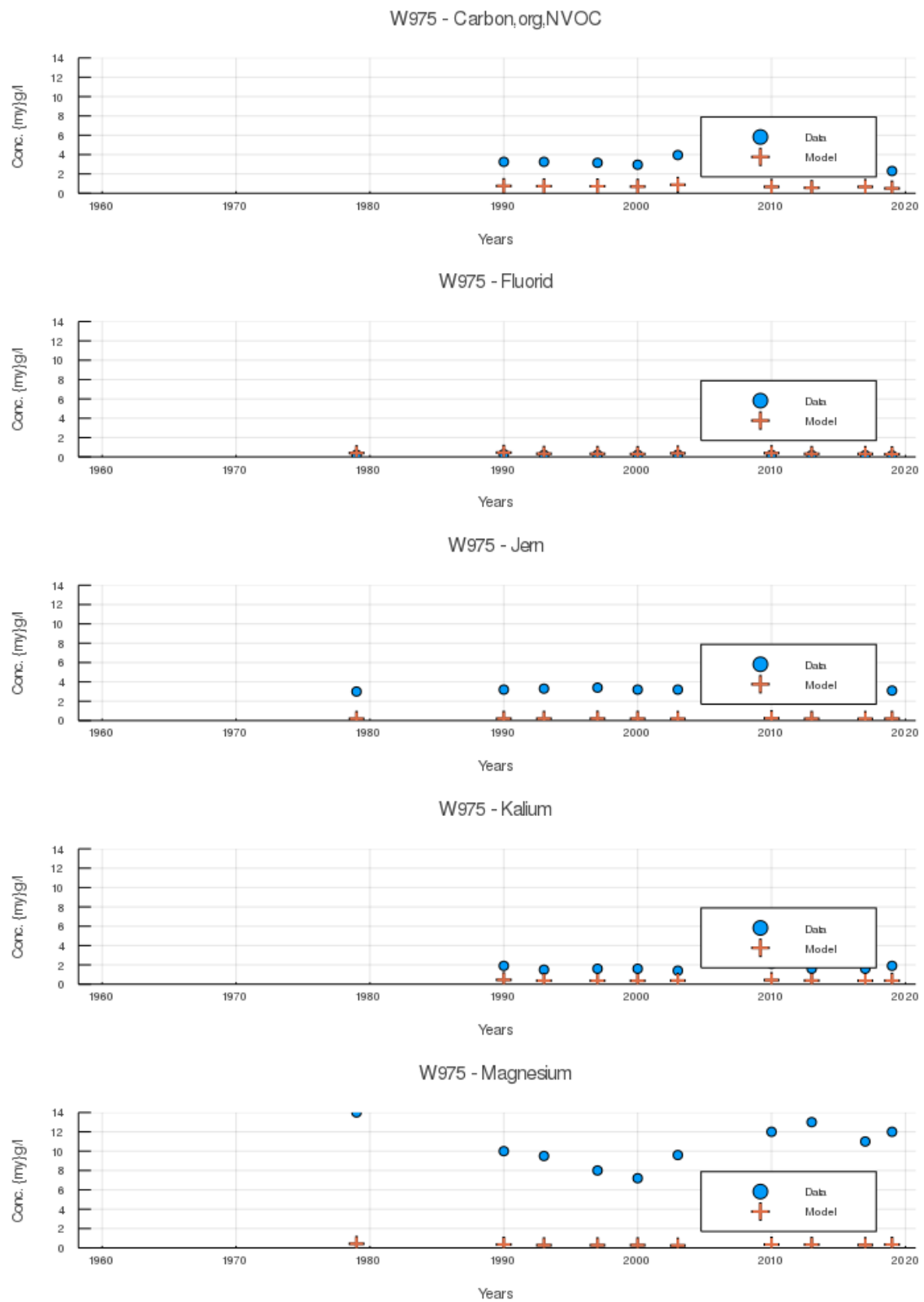
AIC: -390.35986



Figur 5.4: Test2 - den kemiske signatur for de 4 grundvandstyper



Figur 5.5: Test2 - maks. konc. af stoffer i hver boring - estimeret vist mod faktuel



Figur 5.6: Test 2 - estimeret konc. plottet mod faktisk konc. over tid i boring W975 for 5 stoffer

5.4 Test 3

5 chlorerede stoffer i alle boringer - ikke normalisering af data.

Dato: 2019-12-11

Dir: /home/ak/work/NMFk/Birkerod/Test3

Dataset:

Size T: (14, 5, 22)

Wells: 14

Years: 22

Species: 5

Species:

SpeciesNo;Species

2612; Chloroform

2616; Tetrachlormethan

2617; Tetrachlorethylen

2618; Trichlorethylen

2621; 1,1,1-trichlorethan

Execute:

NMFk.execute(T, 2:10; quiet=false, scale=false)

Testresultat:

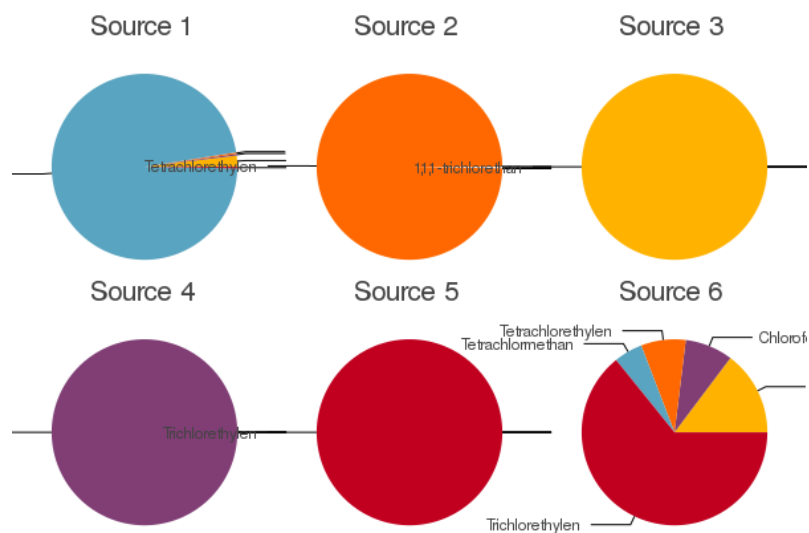
Tid: 16 minutes, 1 second, 878 milliseconds

Optimal solution: 6

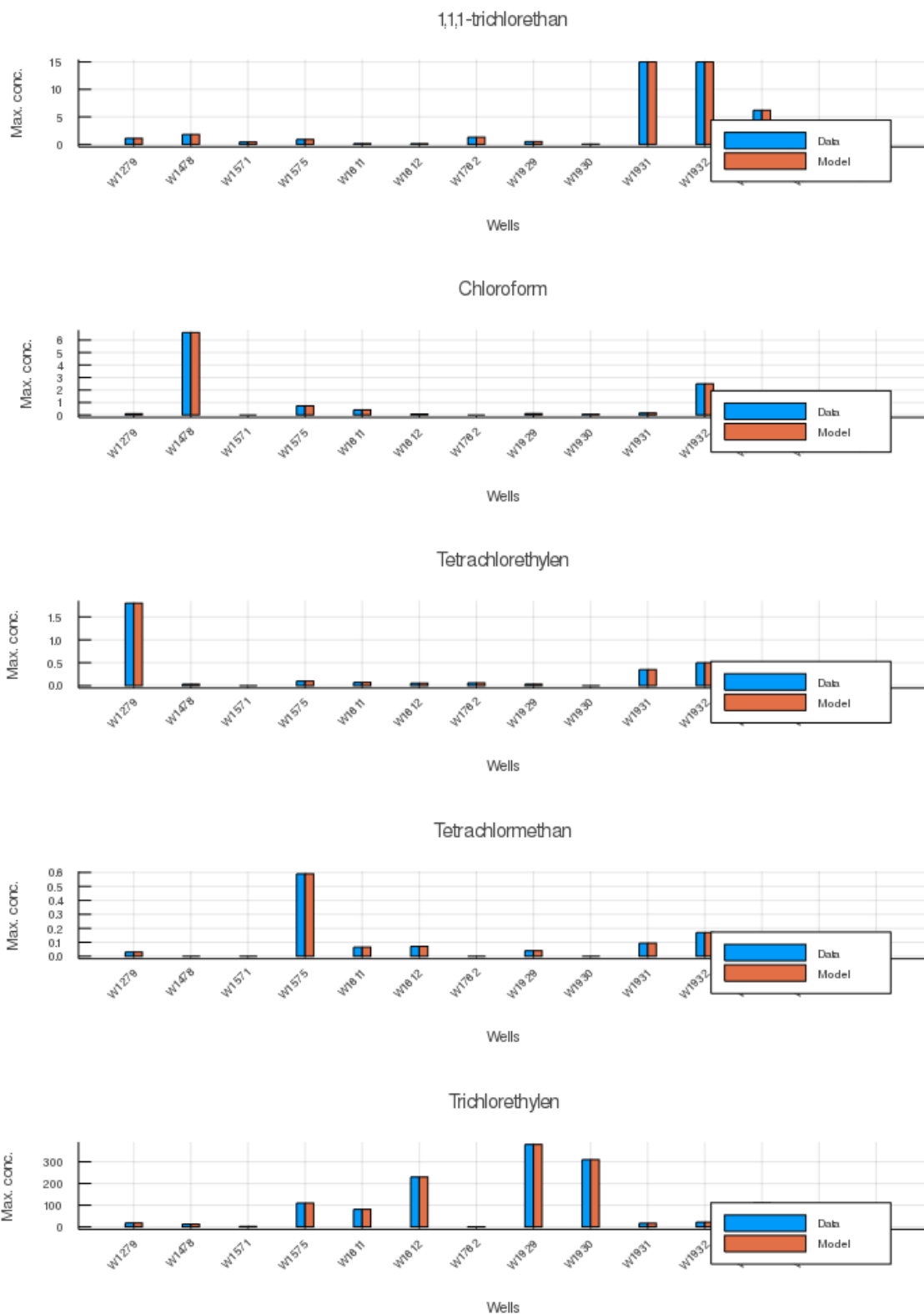
Fit: 7.269577e-6

Silhouette: 0.99997014

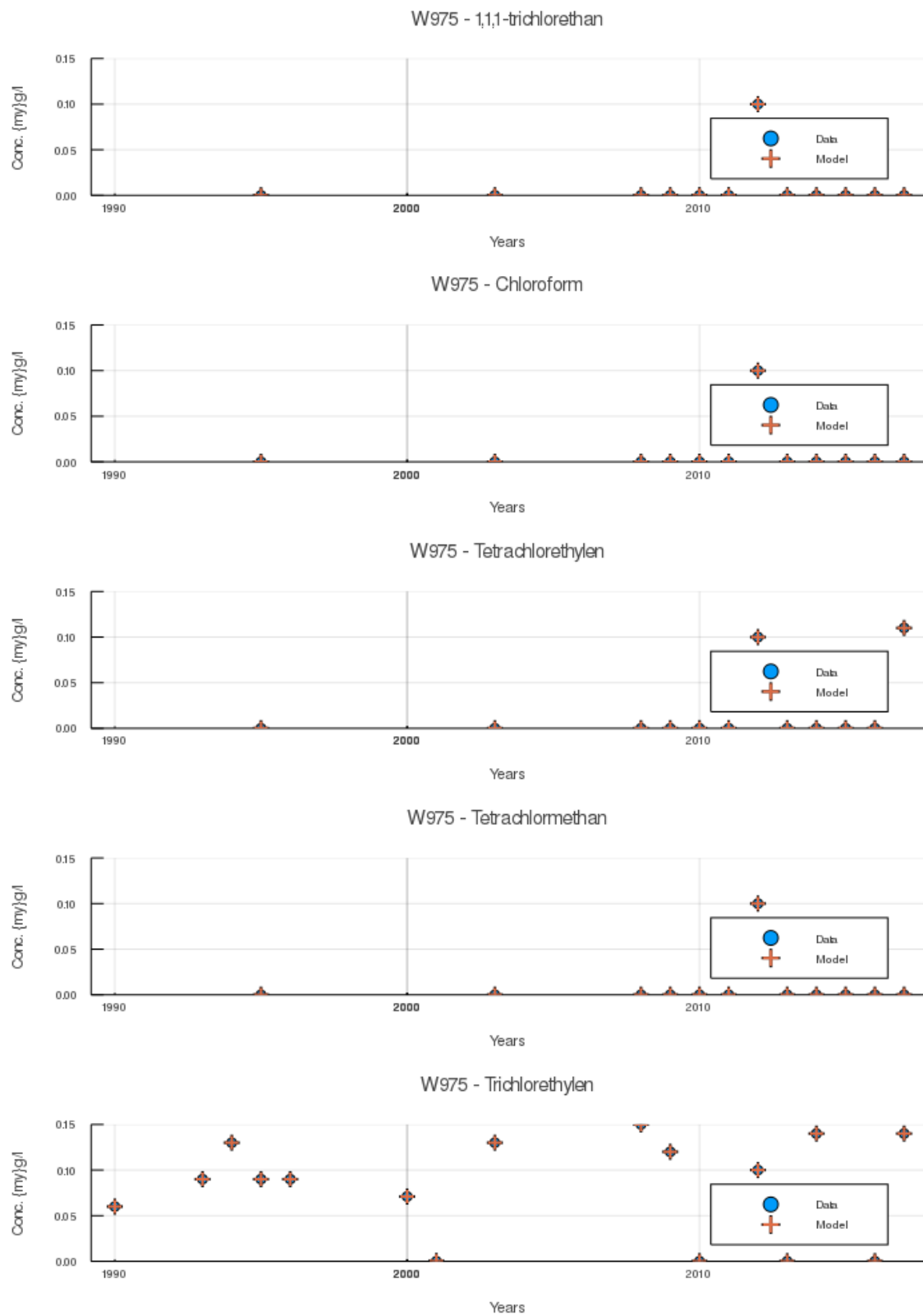
AIC: -9009.638



Figur 5.7: Test3 - den kemiske signatur for de 6 punktkilder



Figur 5.8: Test3 - maks. konc. af stoffer i 10 boringer - estimeret vist mod faktuel



Figur 5.9: Test 3 - estimeret konc. plottet mod faktuel konc. over tid i boring W975 for 5 stoffer

5.5 Test 4

5 chlorerede stoffer i alle boringer - ikke normalisering af data.

Dato: 2019-12-12

Dir: /home/ak/work/NMFk/Birkerod/Test4

Dataset:

Size T: (60, 5, 26)

Wells: 60

Years: 26

Species: 5

Species:

SpeciesNo;Species

2612; Chloroform

2616; Tetrachlormethan

2617; Tetrachlorethylen

2618; Trichlorethylen

2621; 1,1,1-trichlorethan

Execute:

NMFk.execute(T, 2:10; quiet=false, scale=true)

Testresultat:

Tid: 3 hours, 9 minutes, 834 milliseconds

Optimal solution: 6

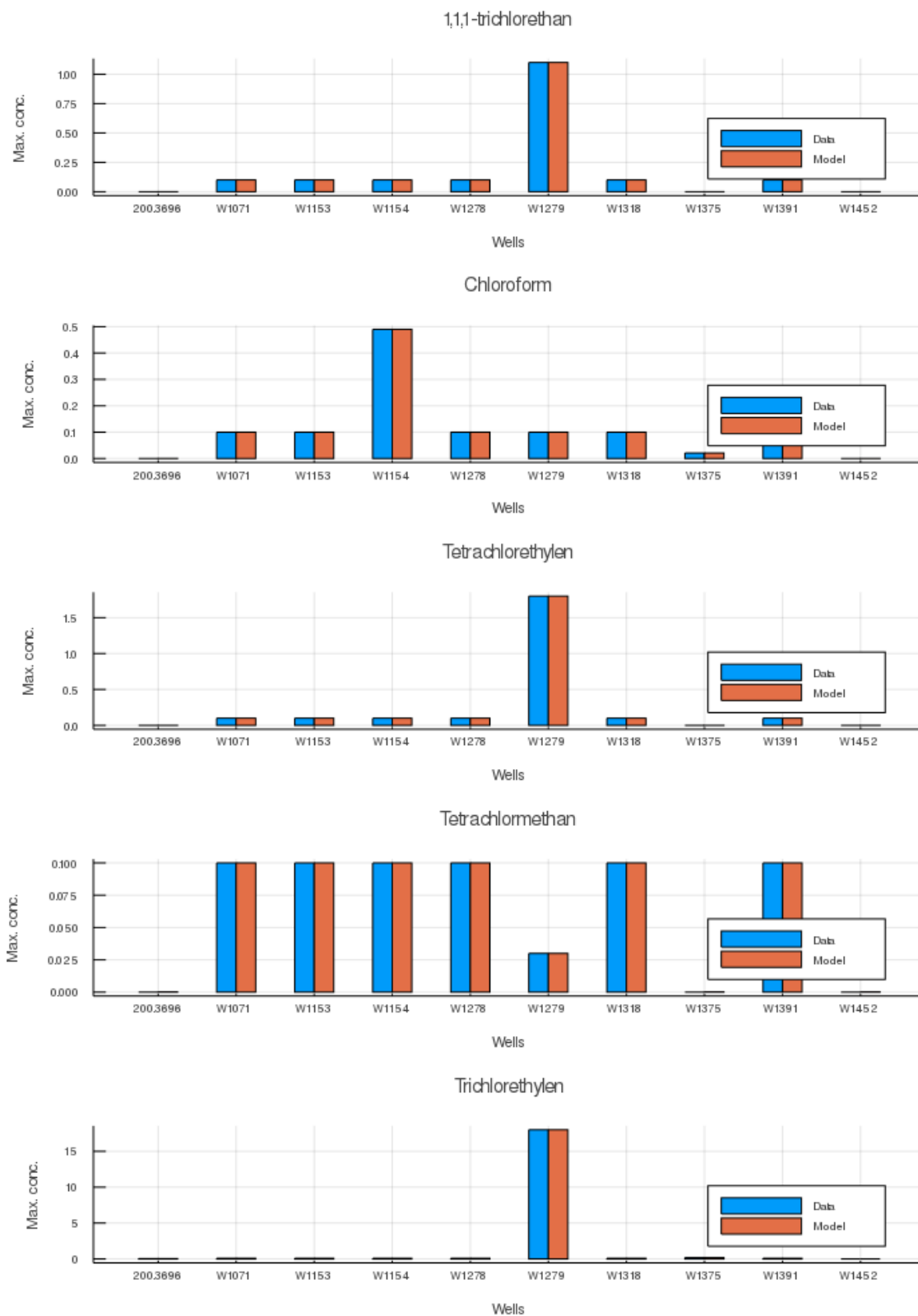
Fit: 6.1578874e-5

Silhouette: 0.9998682

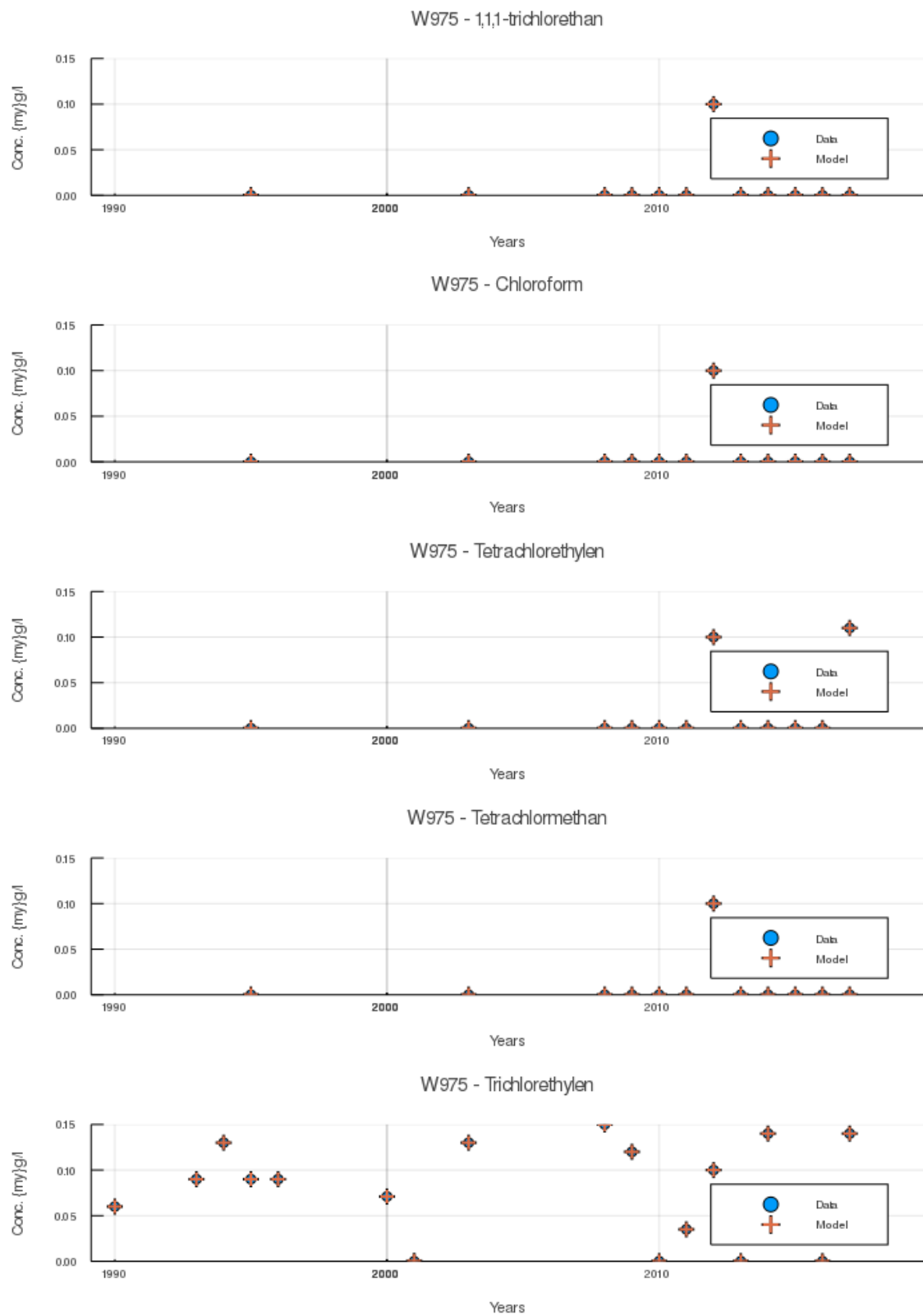
AIC: -9812.318



Figur 5.10: Test4 - den kemiske signatur for de 6 punktkilder



Figur 5.11: Test4 - maks. konc. af stoffer i 10 boringer - estimeret vist mod faktuel



Figur 5.12: Test 4 - estimeret konc. plottet mod faktisk konc. over tid i boring W975 for 5 stoffer

5.6 Test 5

11 chlorerede stoffer i 10 boringer - ikke normalisering af data.

Dato: 2019-12-12

Dir: /home/ak/work/NMFk/Birkerod/Test4

Dataset:

Size T: (10, 11, 22)

Wells: 10

Years: 22

Species: 11

Species:

SpeciesNo;Species

404; Cis-1,2-dichlorethylen

407; 1,1-Dichlorethylen

408; Trans-1,2-dichloreth

2612; Chloroform

2616; Tetrachlormethan

2617; Tetrachlorethylen

2618; Trichlorethylen

2621; 1,1,1-trichlorethan

4542; 1,1-dichlorethan

9422; 1,2-dichlorethan

9946; Vinylchlorid

```
## Execute: NMFk.execute(T, 2:10; quiet=false, scale=false)
```

```
## Testresultat:
```

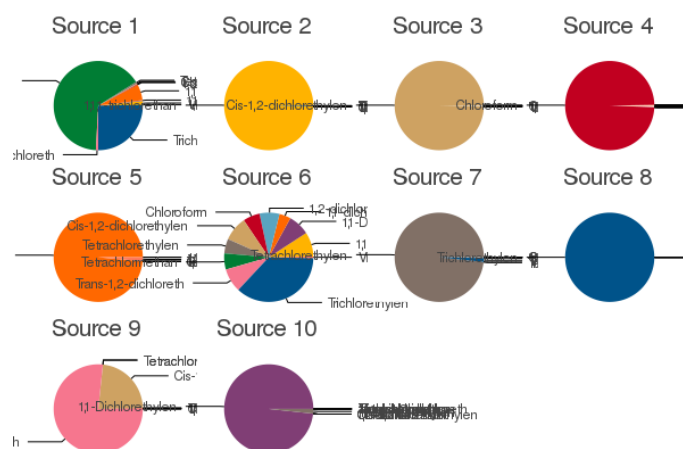
Tid: 2 hours, 12 minutes, 12 seconds, 490 milliseconds

Optimal solution: 10 ... (eller 12?)

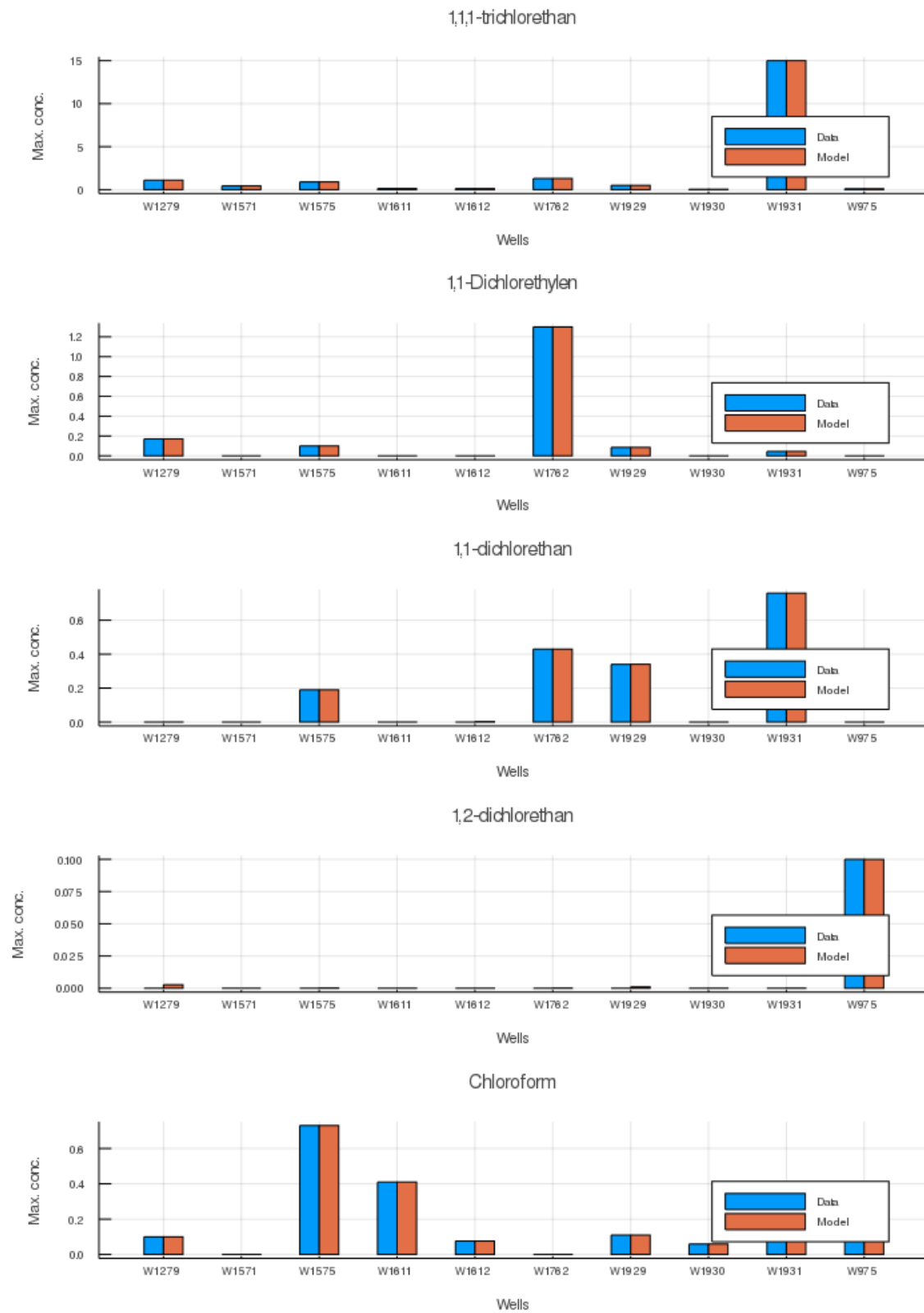
Fit: 0.0058599343

Silhouette: 0.7590346

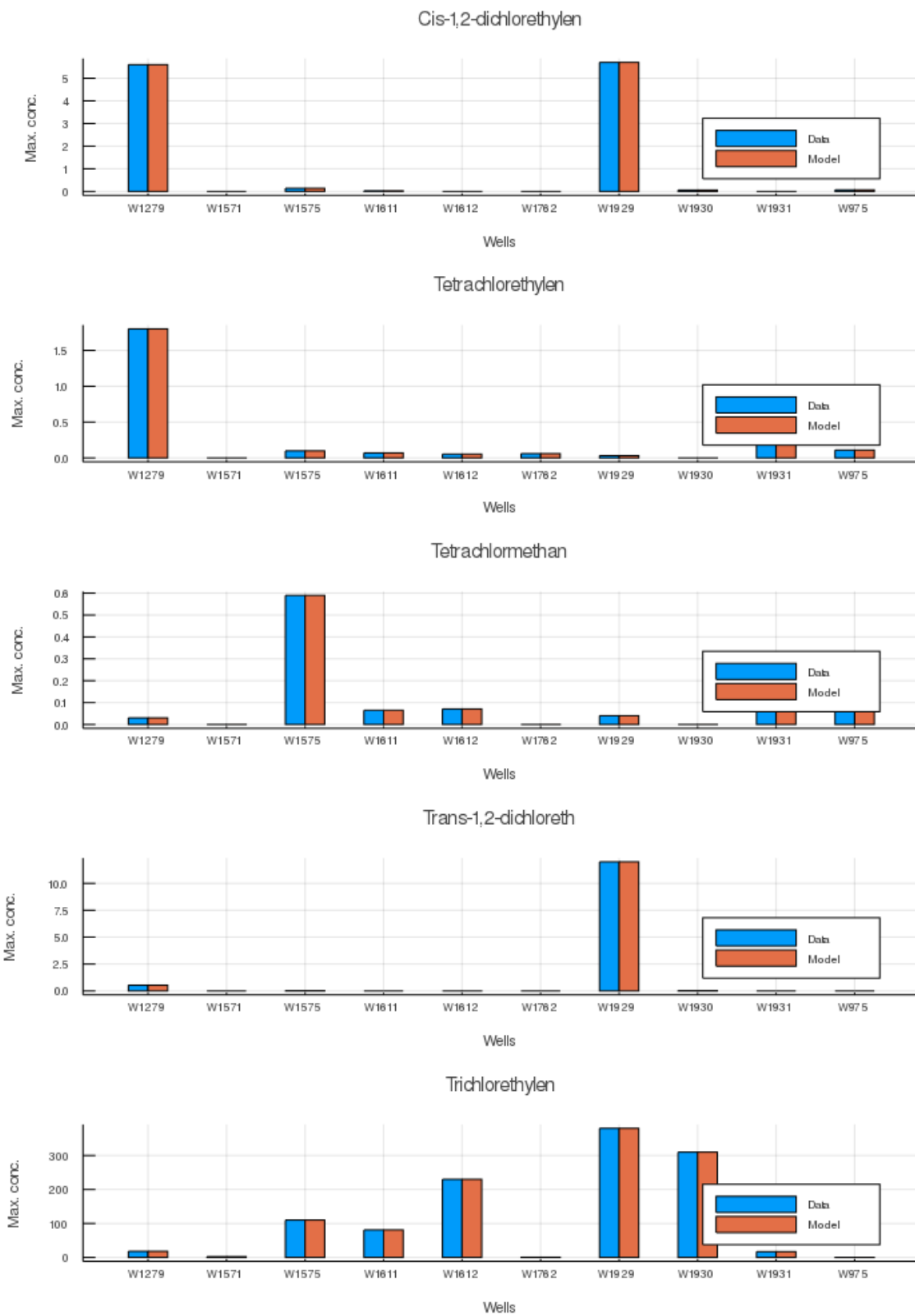
AIC: -2956.9404



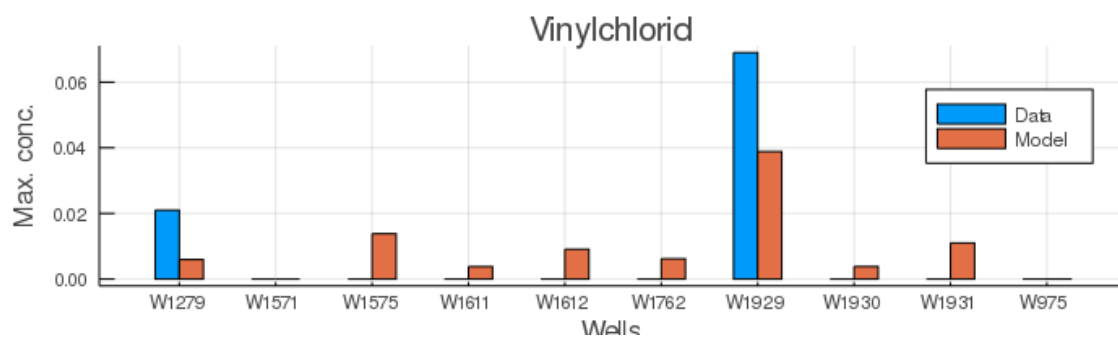
Figur 5.13: Test5 - den kemiske signatur for de 6 punktkilder



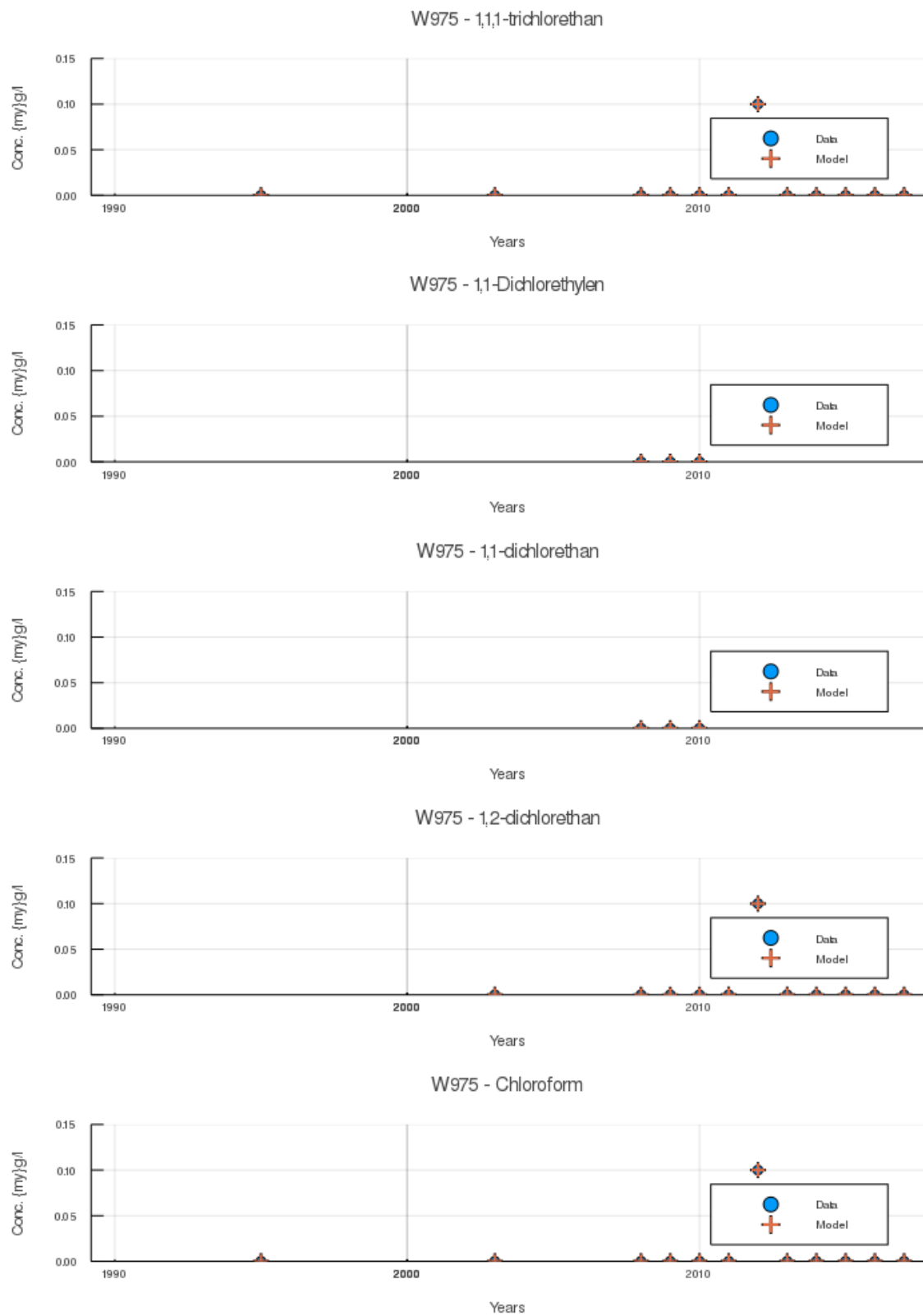
Figur 5.14: Test5 - maks. konc. af første 5 stoffer i 10 boringer - estimeret vist mod faktuel



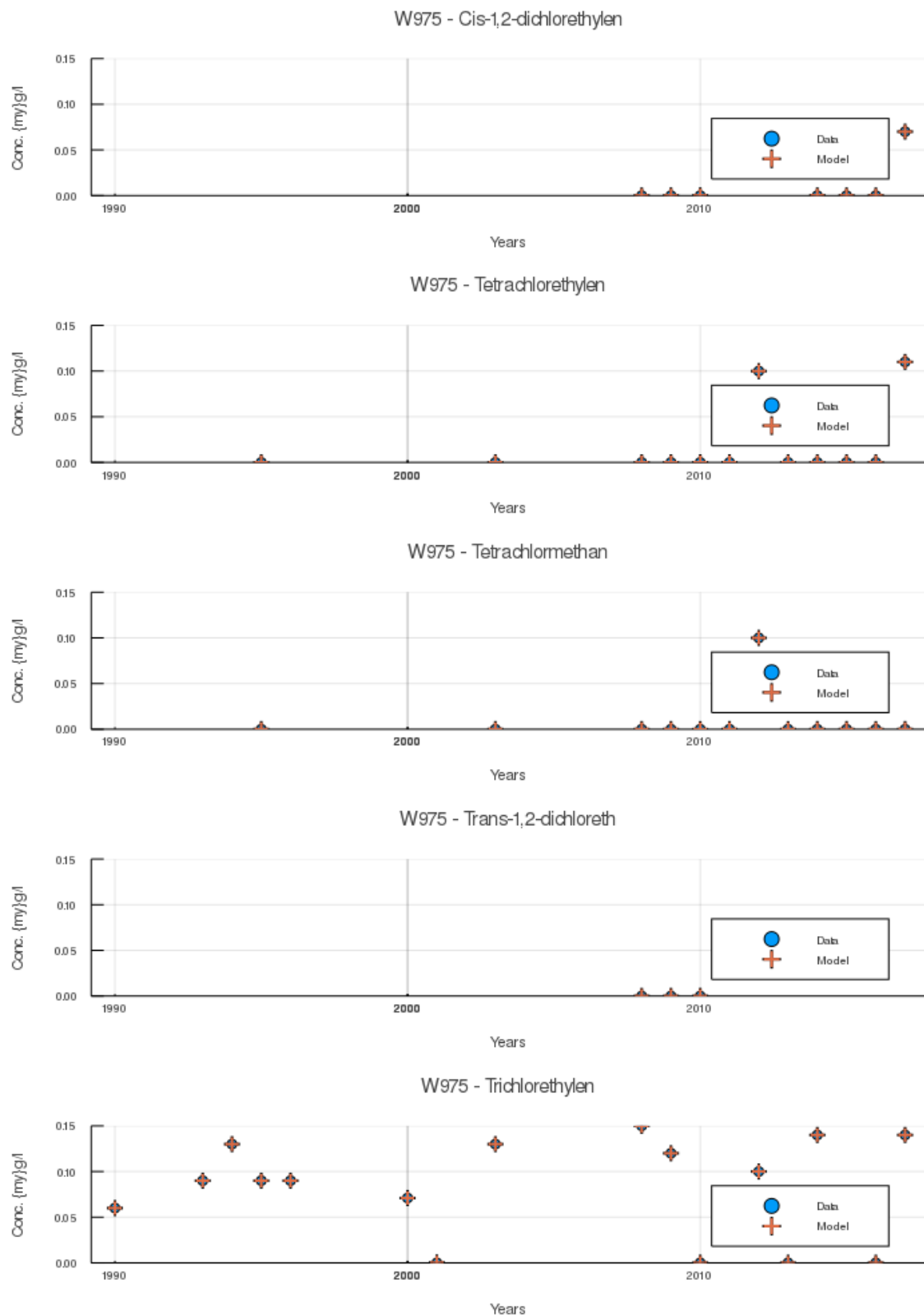
Figur 5.15: Test5 - maks. konc. af næste 5 stoffer i 10 boringer - estimeret vist mod faktuel



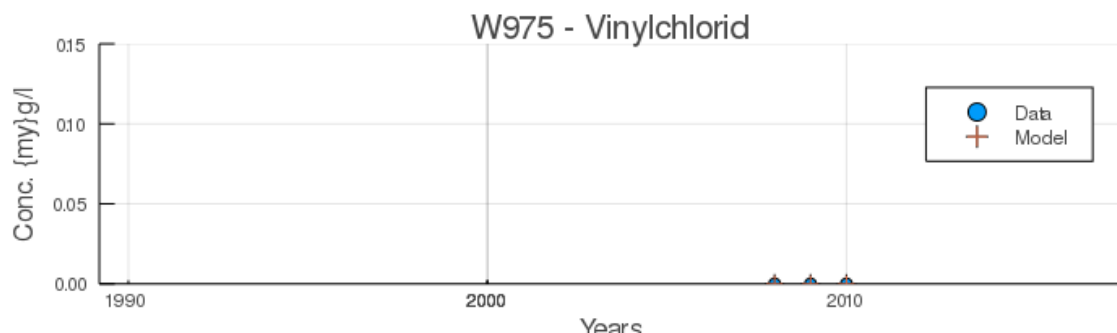
Figur 5.16: Test5 - maks. konc. af 11. stof i 10 borer - estimeret vist mod faktisk



Figur 5.17: Test 5 - estimeret konc. plottet mod faktuel konc. over tid i boring W975 for første 5 stoffer



Figur 5.18: Test 5 - estimeret konc. plottet mod faktisk konc. over tid i boring W975 for næste 5 stoffer



Figur 5.19: Test 5 - estimeret konc. plottet mod faktisk konc. over tid i boring W975 for 11. stof

5.7 Test 6

Test 6 er ikke færdig med at køre i skrivende skrivende stund, og har på nuværende tidspunkt kørt i over 5 døgn. Resultatet kommer dermed ikke til at indgå i denne rapport og test 6 vil ikke blive behandlet yderligere..

5.8 Gennemgang af testresultater

I nedenstående vil jeg fokusere på test 1-5, da disse er mest sammenlignelige, og fint kan illustrere nogle overordnede tendenser i resultaterne og underbygge projektets konklusioner. Test 6 kan primært illustrer, at større datasæt tager meget lang tid at analyserer med MTFk. Hvis større datasæt skal analyseres, vil det kræve at miljøet skales ud.

Overordnet model-performance i de 5 tests

Sammenlignes performance-målinger for de 5 tests kan følgende iagttages:

- Parameteren Fit er højest for test 1 og falder for de efterfølgende tests. Test 3 og 4 har det laveste Fit, mens Fit for test 5 også er lav, men ligger væsentligt højere end test 3 og 4.
- Normaliseringen af data i test 2 giver tilsyneladende et lavere Fit
- Parameteren Silhouette er væsentligt lavere for test 2 (0,27) en for de øvrige 4 tests, hvor den for test 1 og test 3 og 4 er ca. 1.0
- Parameteren AIC viser samme tendens for Fit i de 5 tests. Kun test 1 har en AIC der er større end 0

Test	Fit	Silhouette	AIC
Test 1	4.887795	0.99394673	246.86218
Test 2	0.27703032	0.267032	-390.35986
Test 3	7.269577e-6	0.99997014	-9009.638
Test 4	6.1578874e-5	0.9998682	-9812.318
Test 5	0.0058599343	0.7590346	-2956.9404

Tabel 5.1: Sammenligning af modelperformance for test 1-5

Grundvandstyper / punktkilde-signaturer

Figurerne 5.1, 5.4, 5.7, 5.10 og 5.13 viser den kemiske sammensætning af de grundvandstyper og punktkilder, som modellerne har isoleret fra det kemiske signal i boringerne. Overordnet set giver test 1 og test 2 forskellige grundvandstyper, mens test 3 og test 4 har resulteret i sammenlignelige grundvandstyper / punktkilder. Da det eneste der adskiller test 1 og 2 er, at test 2 er lavet med normaliserede data, kan det konstateres af normalisering har betydning for, hvilke grundvandstyper der isoleres fra datasætte

I test 1 ses 4 forskellige grundvandstyper, hvor type 1 er domineret af NVOC (Non Volatile Organic Carbon) og Kalium, type 2 er domineret af Jern og de to sidste typer er domineret af Magnesium - type 3 med mere kalium og type 4 med mere jern.

I test 2 er type 1 og 3 domineret af Magnesium og Kalium, men type 2 og type 4 er domineret af NVOC. I type 1 ses også et større indhold af Fluorid, mens type 3 har et indslag af Jern. Type 2 har et indslag af Kalium og Fluorid, mens type 4 har et væsentligt indhold af jern og et mindre indhold af kalium.

I test 3 og 4 har modellen identificeret en punktkilde for hver af de 5 chlorerede opløsningsmidler i datasættet, samt en "blandings-kilde", der består af ca. lige store andele af hver af de 5 stoffer. Noget lignende ses i test 5, hvor der er identificeret 10 punktkilder, og hvor 8 af dem består af et enkelt stof. De sidste 2 er blandinger af alle stoffer.

Estimerede koncentrationer

Figurerne 5.2, 5.5, 5.8 og 5.11 viser den maksimale stof-koncentration i de 5 boringer for hver af de 5 stoffer for hhv. de faktiske data og de estimerede.

De 5 figurer underbygger overordnet set opsummeringen vist i tabel 5.1, men kan give et indblik i hvilke stoffer eller boringer der påvirker performance. I test 1 ses de estimerede maks. koncentrationer at ramme tæt på de faktiske maks. koncentrationer i boringerne. I test 2 ses at modellen generelt underestimerer maks. koncentrationerne for NVOC, jern, kalium og magnesium, mens koncentrationerne af fluorid overestimeres.

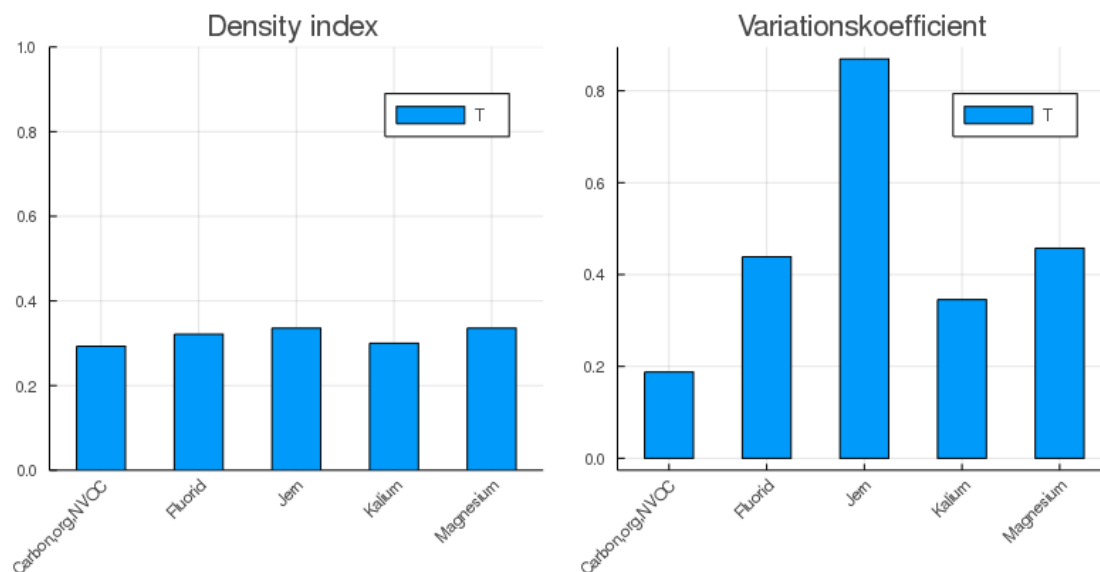
Figurerne 5.3, 5.6, 5.9, 5.12 og 5.17 - 5.19 viser modellerede koncentrationer af stoffer plottet mod faktiske koncentrationer i en enkelt boring over tid. Her ses at modellen i test 1 rammer koncentrationer ganske nøjagtigt, mens modellen i test 2 underestimerer koncentrationerne af stoffer.

I test 3, 4 og 5 ses et særdeles godt fit mellem model og fakta. Dette stemmer fint overens med at modellerne har en meget lave værdier for Fit, men nøjagtigheden virker lidt påfaldende. Jeg har fejlsøgt min kode, men har ikke kunne finde fejl - de plottede værdier kommer fra hhv. datasættet (T) og de estimerede værdier (Te), så jeg vurderer at der er tale om "overfitting". Samme tendens gør sig gældende i søjlediagrammerne for max-koncentrationerne i de enkelte boringer. Den eneste afvigelse her er at vinylchlorid i Test 5 viser afvigelser mellem data og model. Her ses også et

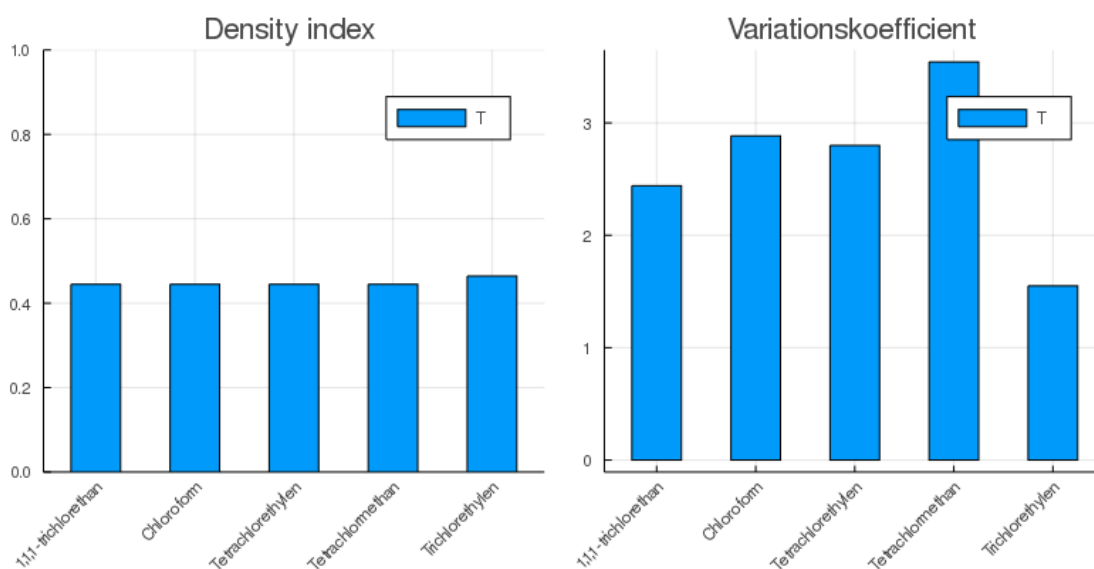
højere fit end for Test 3 og 4.

Variation og densitet i data

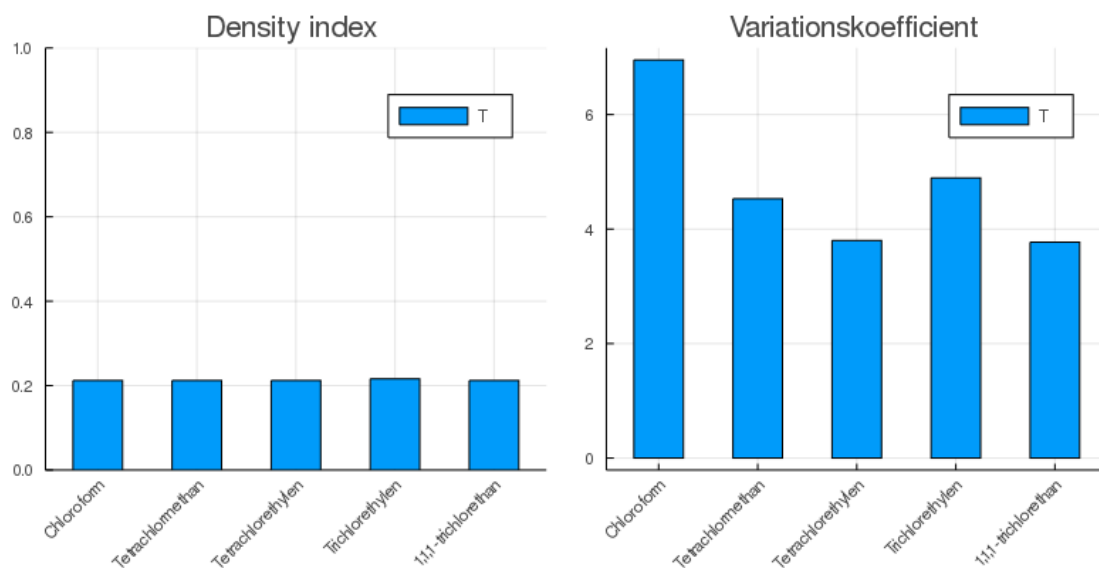
For at få en fornemmelse af variationen og densiteten i datasættet, har jeg beregnet densitets-index og variationskoefficient for stofferne i de 5 tests. Densitetsindexet er beregnet som antallet af datapunkter med værdier (ikke NaN), i forhold til det samlede antal datapunkter.



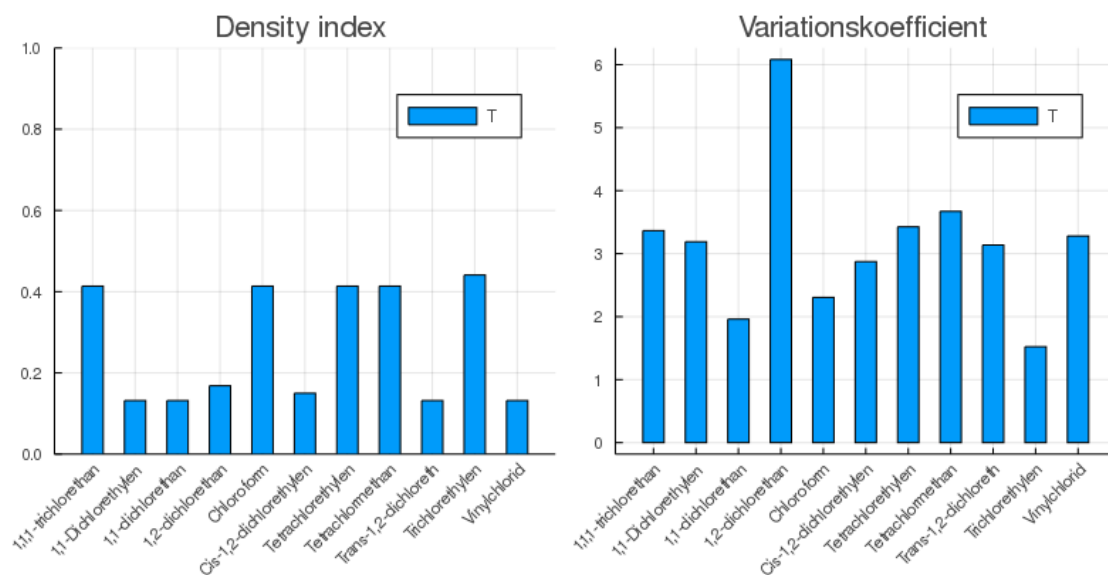
Figur 5.20: Densitet og variationskoefficient i data for test 1 og 2



Figur 5.21: Densitet og variationskoefficient i data for test 3



Figur 5.22: Densitet og variationskoefficient i data for test 4



Figur 5.23: Densitet og variationskoefficient i data for test 5

6. Diskussion

Jeg har valgt ikke at gennemføre yderligere kørsler af f.eks. et pesticid-datasæt, da de gennemførte tests for Birkerød indvindingsopland viser, at resultaterne er tvivlsomme. I nedenstående vil jeg forsøge at underbygge dette og samtidig diskutere mulige årsager.

Nederst giver jeg også en kort opsamling på de øvrige elementer af dette projekt - dvs. etablering af en analysemiljø i Azure, anvendelsen af containere som byggeklodser i et analysemiljø og Julia som værktøj til dataanalyse.

6.1 Test af NTFk algoritmen

På baggrund af de gennemførte tests, har jeg haft følgende refleksioner og overvejelser:

Grundvandtyper og punktkilder

I test 1, der er den test der bedst estimerer de faktiske koncentrationer, identificeres der 4 ”grundvandstyper”. Jeg skriver her grundvandstyper, og ikke ”punktkilder”, fordi test 1 bruger udelukkende naturligt forekommende komponenter, som kan findes i alt grundvand i Danmark. De uorganiske stoffer er et udtryk for forvitring og udvaskning af mineraler fra de geologiske lag, som grundvandsmagasinerne og de overliggende lag består af, og NVOC er ofte afsmittning fra f.eks. begravede organiske aflejringer fra istiden. Stofferne vil givetvis forekomme i varierende koncentrationer, men variationen i de geologiske lag på f.eks. Sjælland vurderes ikke at være tilstrækkelig stor, til at den kemiske signatur som er resultatet i test 1 kan forekomme.

I test 1 består f.eks. kilde 3 og kilde 4 næsten udelukkende af Magnesium, mens variationskoefficienten for magnesium i datasættet er relativ lav (se figur 5.20). Rent fagligt vurderer jeg umiddelbart, at den kemiske signatur for grundvandstyperne i test 2 er mere sandsynlig at finde i virkeligheden - dvs. en mere varieret blanding af alle 5 uorganiske komponenter. Et højt indhold af NVOC kan f.eks. godt forekomme, hvis grundvandet er i kontakt med begravede tørvemoser fra istiden. På samme måde kan et højere indhold af jern forekomme fra de dybe grundvandsmagasiner, hvor der er anaerobe forhold så jernmineraler bliver opløst.

I Vesselinovs artikel [36] skelnes der ikke mellem begreberne ”grundvandstyper” og ”punktkilder”, men jeg vurderer, at dette er en vigtig skelnen, da de uorganiske komponenter i mine dataset ikke kan betragtes som have oprindelse i en punktkilde, men mere som et eller flere baggrunds-signaler. En udfordring i forhold til at verificere den kemiske sammensætning af grundvandstyper er dog, at den er svær at gå ud i

naturen og måle, da det netop ikke er en lokaliseret punktkilde, men et mere diffust baggrundssignal.

I test 3 og 4 er der identificeret 6 punktkilder, og dette virkede umiddelbart meget lovende, da det svarer nogen lunde til antallet af prioriterede lokaliteter, som regionen har fokus på i oplandet. Men når man kikker lidt nærmere på de identificerede kilder, er de tydeligvis sammenfaldende med de enkelte forureningskomponenter, som datasættet består af. Og her er der faktisk tale om egentlige punktkilder, da det er miljøfremmede stoffer, der forekommer som følge af menneskelige aktiviteter på en specifik lokalitet. Disse vil derfor også lettere kunne verificeres i virkeligheden, men her virker modellens identifikation af kildernes kemiske signatur altså sandsynlig.

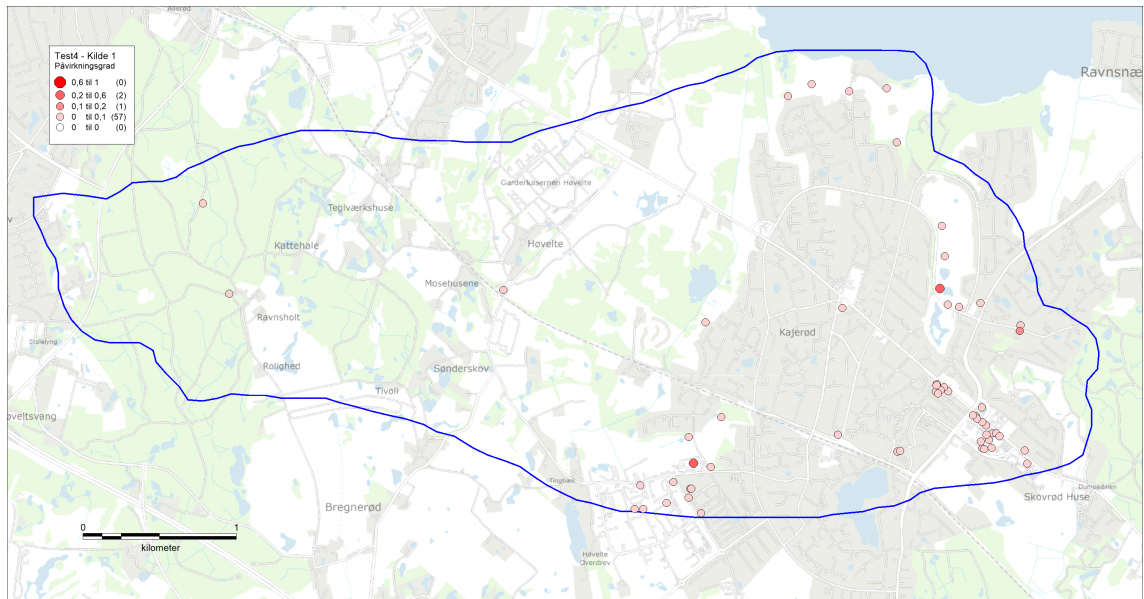
De påviste stoffer kan sagtens forekomme i meget varierende koncentrationer, og specifikke stoffer kan også sagtens mangle helt fra en punktkilde, men typisk vil en punktkilde stadig optræde som en blandingsforurening, og ikke udelukkende bestå af et enkelt stof. I forhold til kildesignaturen i test 5 er der kun er testet for mellem 2 og 10 kilder, og testen resulterede i en løsning på 10 kilder. Løsningen har et lavt Fit og en lav silouette ($<0,8$), hvilket indikerer at løsningen ikke er robust. Det er derfor ikke usandsynligt, at den optimale løsning ligger over 10, f.eks. på 12, med en kilde for hver af stofferne samt en blandingskilde, lige som det ses for test 3 og 4. Dette antal er ikke i overensstemmelse med regionens viden om punktkilder i oplandet.

Samlet set vurderes kildesignaturen for test 1 samt test 3-5 ikke at være sandsynlige kemiske signaturer for grundvandstyper / forureningskilder, men test 2 er svær at verificere. Robustheden af test 2 er endvidere meget lav (0,28).

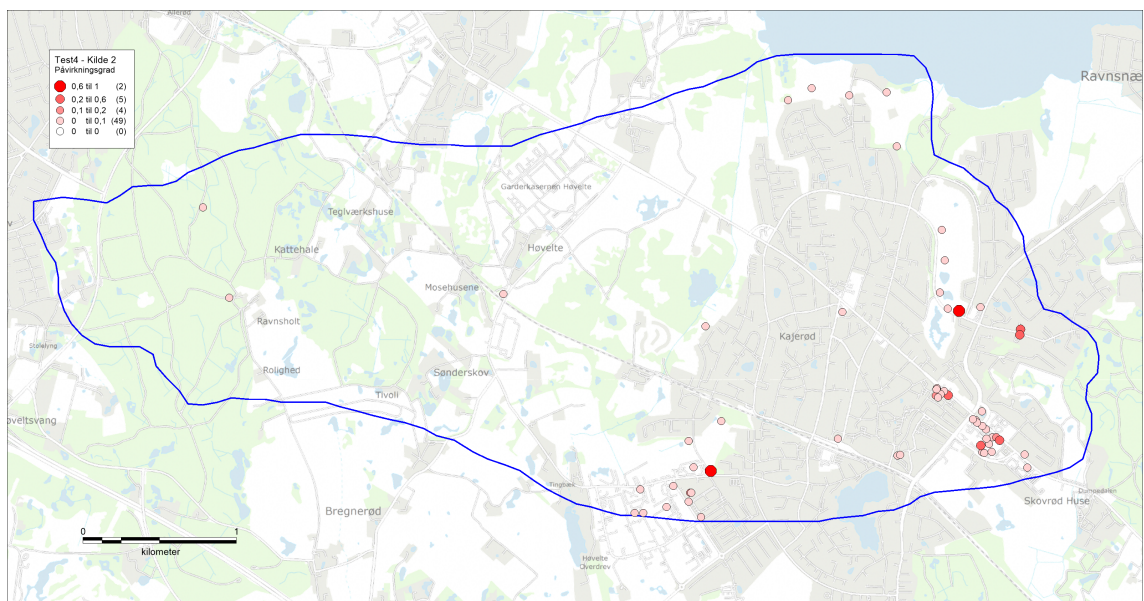
Kildernes påvirkning af boringer i oplandet

Generelt synes alle kilder at påvirke alle boringer i modellerne. Påvirkningen varierer dog således, at kilderne påvirker nogen boringer meget og andre boringer lidt. Da modellerne med uorganiske komponenter kan betragtes som påvirkning fra forskellige grundvandstyper, der bliver blandet i boringerne, er den meget brede påvirkning måske sandsynlig, men for de chlorerede opløsningsmidler som repræsentere egentlige punktkilder, er dette scenarie ikke sandsynligt.

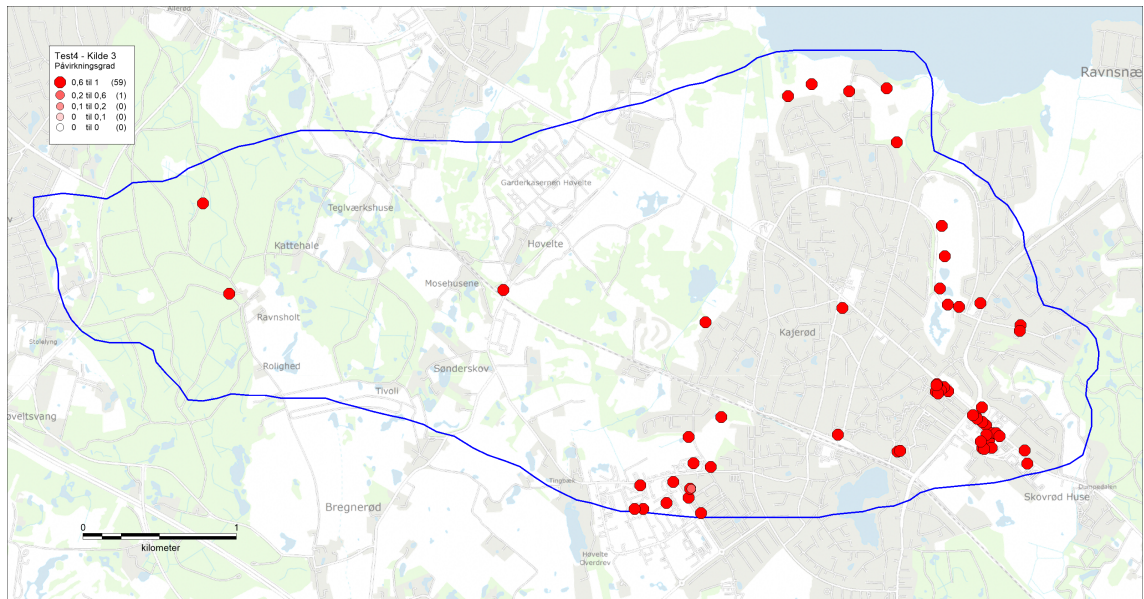
For at illustrere problemet, har jeg lavet kortetene på figur 6.1 - 6.6, der viser, at der ingen steder ses en påvirkningsgrad på 0 for nogen af kilderne. Nogen kilder har endvidere en kraftig påvirkning i hele indvindingsoplandet (se f.eks 6.3)



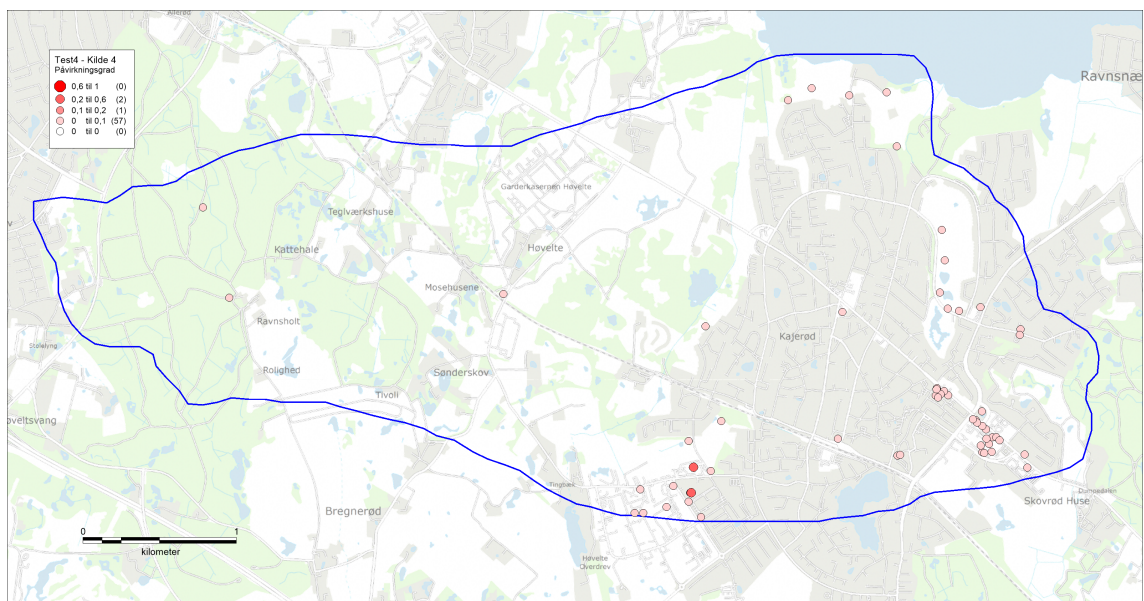
Figur 6.1: Kort over påvirkningsgraden af kilde 1 i datasættets boringer (Test 4)



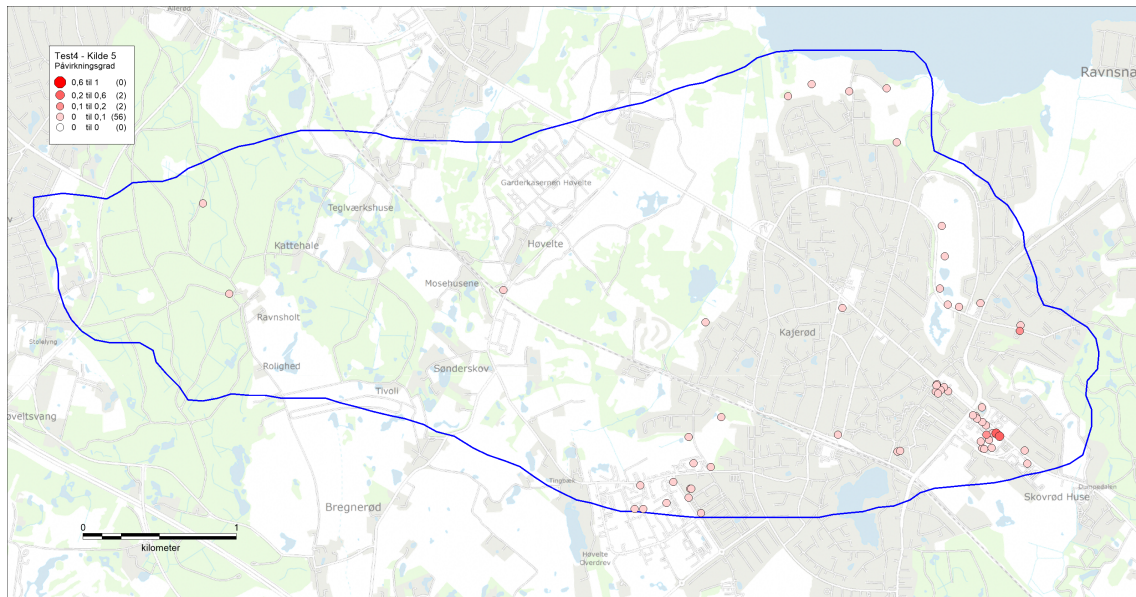
Figur 6.2: Kort over påvirkningsgraden af kilde 2 i datasættets boringer (Test 4)



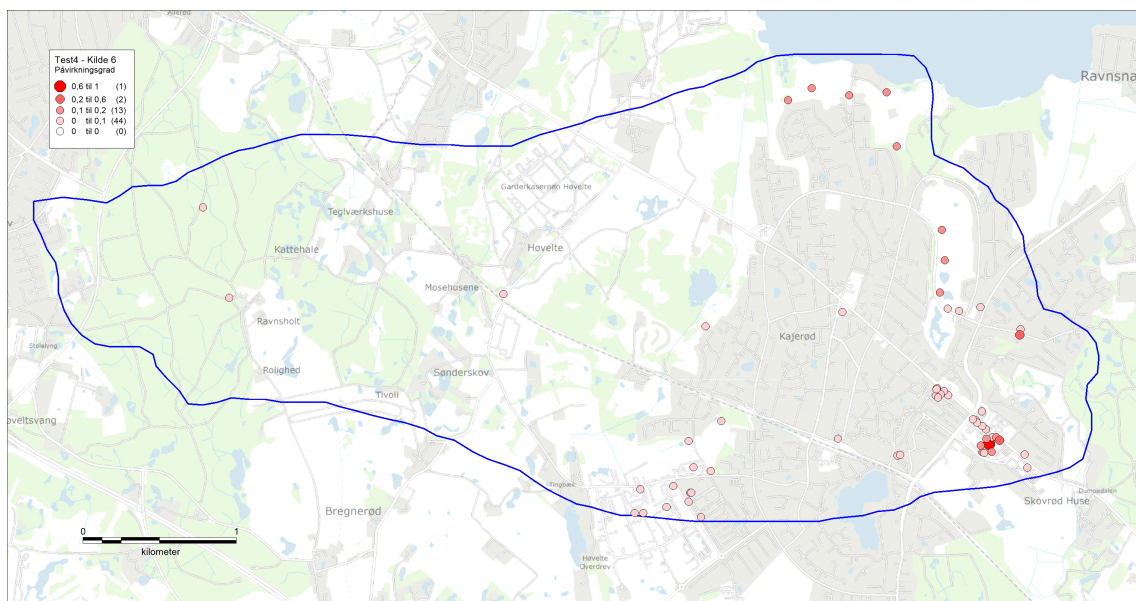
Figur 6.3: Kort over påvirkningsgraden af kilde 3 i datasættets boringer (Test 4)



Figur 6.4: Kort over påvirkningsgraden af kilde 4 i datasættets boringer (Test 4)



Figur 6.5: Kort over påvirkningsgraden af kilde 5 i datasættets boringer (Test 4)



Figur 6.6: Kort over påvirkningsgraden af kilde 6 i datasættets boringer (Test 4)

Mulige årsager

De mulige årsager til de mærkelige resultater er svære at identificere. Jeg har drøftet resultaterne med Vesselinov [31], som peger på "sparsity" som den primære årsag til, at modellen har svært ved at identificere punktkilderne i oplandet. Og data-sættene er da også meget "sparsomme". I figur 5.20 - 5.23 er søjlediagrammerne for *densitet-sindexet* et udtryk for antallet af målinger i datasættet i forhold til det samlede antal datapunkter i datasættet. Som figurerne viser, er der generelt mellem 60% og 85% af datapunkterne, som mangler data, så dette er en meget sandsynlig årsag.

Så selv om NMF / NTF jo netop er kendetegnet ved at kunne håndtere sparsomt besatte datasæt [16], og Vesselinov også i anden sammenhæng har givet udtryk for, at

NTFk-algoritmen fungerer fint med mange manglende værdier i datasættet, er mine datasæt i dette projekt formodentligt for sparsommeligt besatte.

En anden mulig årsag - eller medvirkende årsag - kunne være stor variation i koncentrationerne over tid. Hvis man sammenligner de målte koncentrationer i test 1 og 2, med de målte koncentrationer i test 3, 4 og 5 (figur 5.3, 5.6, 5.9, 5.12 og 5.17 - 5.19), er det tydeligt at koncentrationerne af de uorganiske komponenter ikke varierer nær så meget over tid, som koncentrationerne for de chlorerede komponenter gør.

Dette bekræftes af variationskoefficienterne for de 5 datasæt, der viser, at variationen er ca. 10 gange så stor i de chlorerede datasæt som i det uorganiske datasæt. Dette hænger formodentligt sammen med, at koncentrationerne for de chlorerede komponenter er meget lave og at transporten af komponenterne fra kildeområder påvirkes af fluktuationer i grundvandsstrømningen. Dette kombineret med at komponenterne er flygtige med en deraf følgende analyse- og prøvetagningsusikkerheder.

Jeg mener således, at de meget "savsakkede" kurver med lave koncentrationer af chlorerede komponenter i mange orskellige borer, let kan resultere i samme "kildesignatur". Vesselinov har bekræftet dette, og har endvidere også bekræftet, at NTFk på nuværende tidspunkt ikke kan håndtere boringernes beliggenhed som en del af analysen [31]. Så længe dette er tilfældet, og algoritmen som udgangspunkt modellerer, at samme punktkilde påvirker hele indvindingsoplandet, bliver det svært at bruge algoritmen i regionens arbejde.

Det kan ikke udelukkes, at der i mine tests forekommer fejl. Der har dog ikke været tid til at foretage yderligere tests, som kunne afdække dette. Det kan heller ikke udelukkes at algoritmen ville kunne anvendes på et mere velafgrænset datasæt - f.eks. et afgrænset område med mere strukturerede prøvetagning. Eksempelvis hvis man var i tvivl om borerne på en forurenede lokalitet også var påvirket af en ukendt punktkilde opstrøms lokaliteten. Men det kræver, at problematikken med, at algoritmen resulterer i en kilde for hver komponent med homogen "enkeltstof"-signatur bliver løst.

6.2 "Cloud native - Container first"

'Cloud native - Container first' er et begreb, som man oftere og oftere støder på, når man læser og hører om emner inden for Data Science disciplinen. I dette projekt har jeg valgt at arbejde på en virtuel maskine i Azure, og med Docker-containere som byggeklodser i min systemarkitektur. Hele systemet kunne også godt have været sat op på den virtuelle maskine, og det havde muligvis været en bedre udnyttelse af den virtuelle maskines ressourcer. Alligevel har jeg valgt at bygge systemet i Docker, fordi jeg tror, at containere bliver en fuldstændig central teknologi for fremtidens cloud-baserede analyse-miljøer og en viden om dette vil være vigtig inden for denne disciplin.

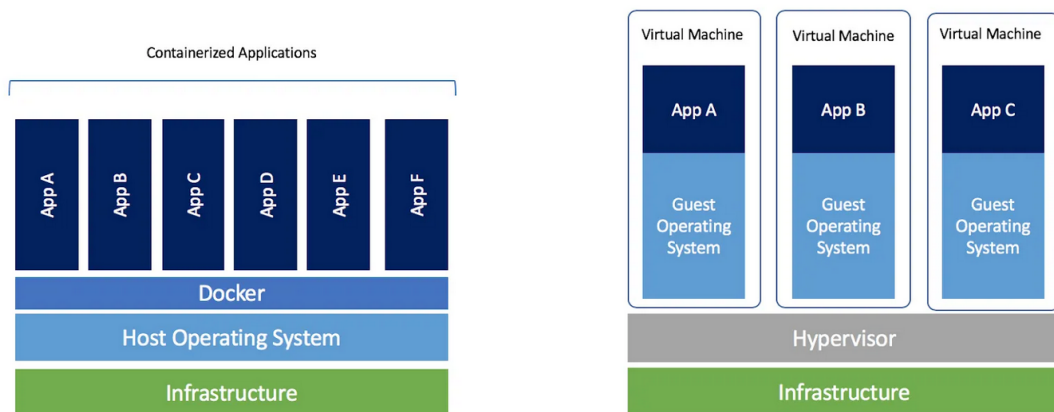
Virtualisering - VM og containere

Virtualisering har eksisteret i mange år, og containere er i virkeligheden bare en ny virtualiserings-teknologi. Og virtualisering er en løsning på rigtig mange behov, som

software-udviklere, IT-arkitekter og systemansvarlige står overfor bl.a. :

- Portability - at kunne flytte applikationer og systemer
- Forvaltning af system ressourcer
- Sikkerhed
- Driftssikkerhed
- Fleksibilitet

Både VM'er og containere kan løse disse behov for os, men containere gør det med færre ressourcer. Som det fremgår affigur 6.7, kræver Docker og andre container-teknologier ikke, at der bliver installeret et fuldt styresystem i hver container - i containeren er styresystemet også virtualiseret. Containeren er lige så konfigurer-bar, og der med opnå man samme fordele som i en VM med et væsentligt lavere ressourceforbrug. Forskellen mellem VM'er og containere kan på den måde godt sammenlignes med IAAS-laget og PAAS-laget i skyen.



Figur 6.7: Sammenligning af arkitektur - virtuel maskine og container

Min erfaring fra dette projekter, at Docker er superlet at komme i gang med. Man tager udgangspunkt i et eksisterende container-billede, og skriver konfigurationen in i en tekstfil (Dockerfile). Konfigurationen består af simple nøgleord som f.eks. FROM, RUN, ADD, COPY og CMD, og almindelige unix/linux commandoer, og tekstfilen kan efterfølgende kompileres til et containerbillede.

Billederne er som udgangspunkt self-contained og de billeder jeg har bygget i dette projekt - og som i dag kører i Azure - kan flyttes til Amazon eller til min kollegas computer på en halv time, og jeg har ingen bekymringer for om de vil kører. Kort sagt - jeg er super begejstret over mit første rigtige bekendtskab med Docker og dette bliver ikke de sidste containere jeg har bygget.

6.2.1 Skalering

Den VM og JupyterLab-containeren som jeg har brugt i dette projekt er sat op med 4 GPU'er og CUDA-understøttelse. Jeg har desværre ikke fået dette i spil i projektet - NTFk-algoritmen understøtter efter sigende GPU-beregninger, men jeg har ikke

haft tid til at finde ud af hvordan dette gøres. Men flere og flere software-pakker og moduler til både R, Python og Julia understøtter GPU-beregninger, og der er en massiv effektivisering, ved at gøre brug af dette.

At sætte en eller flere GPU'er i maskinen og flytte beregninger over på disse, og dette er at "opskalere". Systemer der er baseret på Docker giver dog også mulighed for at "udskalere" - dvs. distribuere beregningerne på flere contianere. Der er to muligheder for at gøre dette - Docker Swarm og Kubernetes. Docker Swarm er Dockers egen udskalerings-teknologi og mere simpel at anvende end Kubernetes. Kubernetes er langt mere konfigurerbar end Docker Swarm og det er den der bliver talt og skrevet rigtig meget om i øjeblikket. Jeg har desværre ikke prøvet at udskalere i dette projekt, men det bliver det næste jeg vil kaste mig over i 2020.

6.2.2 Sikkerhed

Sikkerheden i systemet er først og fremmest baseret på Azures netværks-sikkerhed for ressourcer. Den overordnet sikkerhed består af IP-filtrering, som sættes op på det virtuelle netværk til den virtuelle maskine. Adgangen til maskinen kan dermed kun ske fra nogle udvalgte IP-adresser og gennem nogle udvalgte porte.

Næste led i sikkerheds-kæden er det virtuelle styresystems adgangskontrol. Som bruger kan man logge på den virtuelle maskine via ssh, og dette kan ske via en krypteret adgang med RSA private-key public-key konceptet. Sidste led i sikkerhedskæden er applikationernes adgangskontrol, hvor adgang til SQL-serveren sker med en SQL bruger login og adgang til Jupyter Lab sker med en token - dvs. en lang tekststreng, der indsættes i URL'en.

Ved at bygge systemet med containere, vil jeg påstå at man tilføjer et ekstra sikkerhedsniveau. Containere fungerer på den måde, at de kan sættes op til kun at bruge en begrænset del af host-systemets ressourcer. Hvis der ikke er sat begrænsninger på, kan de kun bruge de tilgængelige ressourcer på host-systemet. Dette forhindrer et evt. DDoS angreb på en af containerne at lægge styresystemet og de øvrige containere ned.

De containere er et isoleret miljø, vil en virus-angreb heller ikke kunne sprede sig til de andre containere eller host-systemet - eller i hverfald have væsentlig sværere ved det.

6.3 Julia - et nyt lovende sprog

Julia er et nyt og meget lovende sprog, der er lavet med Data science for øje. Det er et JIT-kompileret og eksekverer derfor kode og beregninger mere effektivt end både R, Python og matlab. Jeg har ikke haft nogen forudgående erfaring med Julia, og mit umiddelbare indtryk har været positivt.

Syntaksen ligner meget syntaksen i R og Python, som jeg har erfaring med fra tidligere. F.eks. arbejdet med fler-dimensionelle array's og DataFrames er meget sammenligneligt med den måde man gør dette i Python. Og Julias udviklere har selvfølgelig fundet en masse inspiration i de andre sprog, og har formodentligt taget de bedste

ideer fra de andre sprog. Julia's standart faciliteter til plot og grafer ligner og meget det man finder i de andre analyse-miljøer, og i Julia findes der også pakker, der gør at man i plot-miljøet kan anvende Pythons pyplot-modul og R's ggplot2-pakke. Alt dette gør det nemt at komme i gang med Julia.

En udfordring ved sproget er dels at det er nyt, og det udvikler sig hurtigt. Jeg har ofte oplevet at søge efter løsningen på en fejl på et nettet, men hvor de løsninger jeg fandt var forældet, fordi de var skrevet til en tidligere version. Desuden er der et væsentligt mindre "community" omkring sproget end man finder i R og Python. Det kan derfor være sværere at finde gode eksempler på, hvordan man løser en specifik udfordring eller retter en fejl. Endeligt kan den velkendte syntaks også godt snyde, for der måske er indbygget en lidt anden logik eller sat en ekstra parameter i på en funktion.

En sidste udfordring jeg vil nævne er geospatielle analyser og visualiseringer, som Python excellerer i. Python har i mange år været det fortrukne sprog for open source og kommercielle applikationer. Python er derfor meget veludviklet på dette område. I Julia har jeg fundet 'en enkelt pakke (GMT.jl) der stiller geospatielle faciliteter til rådighed. Jeg har forsøgt at anvende pakken i dette projekt plotte data ud som kort, men er ikke kommet i mål med det. I mit arbejde er geospatielle analyser og visualiseringer en meget væsentlig disciplin, så det vil være en forudsætning at kunne dette, hvis jeg skal basere mit analyse-arbejde på Julia.

7. Konklusion og perspektivering

7.1 NTFk

På baggrund af de gennemførte forsøg i dette projekt, vurderes NTFk-algoritmen ikke at kunne bruges i forbindelse med regionernes arbejde inden for jordforureningsområdet på nuværende tidspunkt. De gennemførte forsøg med algoritmen har ikke kunne identificere forureningskilder, der har kunne verificeres i virkeligheden. De forureningskilder som algoritmen har isoleret og identificeret i forsøgene, afspejler ikke regionens faktiske viden om forureningskilder i Birkerød indvindingsopland.

Årsagen til at algoritmen ikke kan identificere de kendte forureningskilder i Birkerød, kan ikke fastslås endeligt på baggrund af dette projekt. Men en mulig forklaring er, at de datasæt jeg har arbejdet med i disse forsøg har være meget sparsomt besatte - kun ca. 20%-40% af datapunkterne i datasættene har indeholdt målinger. Grunden til dette er, at forureningsundersøgelser bliver gennemført over en relativ kort periode og har til formål at undersøge og beskrive en given forurening på et givent tidspunkt - ikke at monitorere alle kilder i en sammenhæng over en længere periode.

En anden medvirkende årsag vurderes at være "stof-signaturen" i de enkelte tidsserier. Koncentrationerne er meget lave og ustabile, hvilket vurderes at give algoritmen problemer med at isolere tendenser, der kan identificeres som separate bidrag fra punktkilder. Det kan ikke udelukkes at man ville kunne generere et datasæt - f.eks. i forbindelse med et langvarigt monitorings-projekt - hvor densiteten af data var højere og koncentrationerne mere konstante, og som derfor ville resultere nogle mere retvisende model-resultater.

Endelig er Vesselinovs arbejde med algoritmen under stadig udvikling. På trods af at mine forsøg med algoritmen ikke har været positive, vurderes det arbejde der foregår inden for dette område at være særdeles interessant, og noget vi som fagpersoner bør holde et vågent øje med.

7.2 Azure og Docker

En væsentlig del af dette projekt har været at komme i gang med at bruge Azure som analyseplatform og at undersøge mulighederne for at bygge et analysemiljø op vha. Docker-containere. Denne del af projektet er lykkedes, og selv om det miljø jeg har opbygget i projektet her skal betragtes som en prototype, har miljøet vist sig at fungerer upåklageligt.

At bygge systemet modulært af en række 'autonome' enheder med veldefineret ansvar og funktionalitet, opnås et system der er lettere at vedligeholde og implementere på andre platforme. Konfiguration og opbygning af nye containere ud fra eksisterende ved hjælp af en *Dockerfile*, er en stærk og fleksibel mulighed at have i værktøjskassen, og samtidig nemt at komme igang med.

Med denne modulære opbygning, vil det desuden være muligt både at skalere løsningen og at udbygge løsningen til f.eks. også at omfatte et REST-API til modelresultater eller en webbaseret brugerflade, til grafisk præsentation af data og resultater. Endelig kan det nævnes, at Azure også giver mulighed for at scripte ressourcer vha. f.eks. ARM-templates og Powershell-scripts. Dette er ikke gjort i dette projekt, men kunne - sammen med containere-teknologien - give et fuldstændigt scriptet miljø, der kunne flyttes og "re-deployes" på ganske kort tid.

7.3 Perspektivering

Jeg mener at jeg med dette projekt har fået etableret et cloud-baseret analysemiljø, der kan imødekomme alle de behov, jeg har som dataanalytiker i Region Hovedstaden. I miljøet har jeg mulighed for at arbejde interaktivt med både R, Python og Julia via Jupyter Notebooks, og har adgang til opdaterede kopier af regionens fagdatabaser. Databasemiljøet skal selvfølgelig udbygges med mange flere eksterne data og evt. tilføres noget datawarehouse-funktionalitet, men miljøet er umiddelbart anvendeligt til nye analyse-projekter.

Og selv om jeg med dette projekt afslutter min IT-diplom uddannelse på DTU, er min egen udvikling inden for dette område langt fra slut. Projektet og uddannelsen har tvært imod givet mig lyst til at lære mere og fordybe mig yderligere. Lige nu har jeg meget stor lyst til at komme et spadestik dybere ned i multivariable metoder, herunder forskellige dekompositions-metoder. Desuden har jeg netop købt en bog om datavisualisering som disciplin, hvor jeg også tænker, at der ligget et stort potentiale for mig i forhold til formidling. Og så skal jeg selvfølgelig også prøve at udskalerer beregninger i containere vha. Kubernetes. Endelig skal jeg have kikket lidt mere på tilgængelige spatielle værktøjer i Julia, da dette vil være en forudsætning for jeg kan satse på dette nye og lovende sprog, som min foretrukne værktøjskasse.

Litteratur

- [1] Boian S. Alexandrov and Velimir V. Vesselinov. Blind source separation for groundwater pressure analysis based on nonnegative matrix factorization. *Water Resources Research*, 50(9):7332–7347, sep 2014.
- [2] François Bachoc. Spatial Blind Source Separation. 2019.
- [3] Merle Behr, Chris Holmes, and Axel Munk. Multiscale blind source separation. *The Annals of Statistics*, 46(2):711–744, apr 2018.
- [4] Nicolas Gillis. The Why and How of Nonnegative Matrix Factorization. 2014.
- [5] Luc T. Ikelle. Nonnegative Tensor Factorization. In *Handbook of Geophysical Exploration: Seismic Exploration*, volume 39, pages 569–577. 2010.
- [6] Filip L. Iliev, Valentin G. Stanev, Velimir V. Vesselinov, and Boian S. Alexandrov. Nonnegative Matrix Factorization for identification of unknown number of sources emitting delayed signals. *PLOS ONE*, 13(3):e0193974, mar 2018.
- [7] Mina Kemiha and Abdellah Kacha. Complex Blind Source Separation. *Circuits, Systems, and Signal Processing*, 36(11):4670–4687, nov 2017.
- [8] Mina Kemiha and Abdellah Kacha. Complex Blind Source Separation. *Circuits, Systems, and Signal Processing*, 36(11):4670–4687, nov 2017.
- [9] Julian Koch, Helen Berger, Hans Jørgen Henriksen, and Torben Obel Sonnenborg. Modelling of the shallow water table at high spatial resolution using Random Forests. *Hydrology and Earth System Sciences Discussions*, pages 1–26, may 2019.
- [10] Tamara G. Kolda. Tensor Decompositions and Applications. 2010.
- [11] S. Santosh Kumar, S. H. Bharathi, and M. Archana. Non-negative matrix based optimization scheme for blind source separation in automatic speech recognition system. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pages 1–6. IEEE, oct 2016.
- [12] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, oct 1999.
- [13] Chunteng Li, Yuzhong Jiang, Fangjun Liu, and Yali Xiang. Blind Source Separation Algorithm Based on Improved Particle Swarm Optimization Under Noisy Condition. In *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 398–401. IEEE, may 2018.
- [14] Ye Liu. Tucker Decomposition Network: Expressive Power and Comparison. 2019.
- [15] Josh H McDermott. The cocktail party problem. *Current Biology*, 19(22):R1024–R1027, dec 2009.

- [16] Morten Mørup, Lars Kai Hansen, and Sidse M. Arnfred. Algorithms for Sparse Nonnegative Tucker Decompositions. *Neural Computation*, 20(8):2112–2131, aug 2008.
- [17] Chaitanya Narisetty. Bayesian Non-Parametric Multi-Source Modelling Based Determined Blind Source Separation. 2019.
- [18] Klaus Nordhausen, Hannu Oja, Peter Filzmoser, and Clemens Reimann. Blind Source Separation for Spatial Compositional Data. *Mathematical Geosciences*, 47(7):753–770, oct 2015.
- [19] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. Introduction to Tensor Decompositions and their Applications in Machine Learning. nov 2017.
- [20] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *comput. appl. math.* 20, 53-65. *Journal of Computational and Applied Mathematics*, 20:53–65, 11 1987.
- [21] Mark F. Snodgrass and Peter K. Kitanidis. A geostatistical approach to contaminant source identification. *WATER RESOURCES RESEARCH*, 33(4):537–546, 1997.
- [22] Valentin G. Stanev, Filip L. Iliev, Scott Hansen, Velimir V. Vesselinov, and Boian S. Alexandrov. Identification of release sources in advection–diffusion system by machine learning combined with Green’s function inverse method. *Applied Mathematical Modelling*, 60:64–76, aug 2018.
- [23] Y-h. Taguchi. Application of TD Based Unsupervised FE to Bioinformatics. In *Unsupervised Feature Extraction Applied To Bioinformatics*, pages 213–296. 2020.
- [24] Y-h. Taguchi. Applications of PCA Based Unsupervised FE to Bioinformatics. In *Unsupervised Feature Extraction Applied To Bioinformatics*, pages 119–211. 2020.
- [25] Y-h. Taguchi. Introduction to Linear Algebra. In *Unsupervised Feature Extraction Applied To Bioinformatics*, pages 3–22. 2020.
- [26] Y-h. Taguchi. Matrix Factorization. In *Unsupervised Feature Extraction Applied To Bioinformatics*, pages 23–45. 2020.
- [27] Y-h. Taguchi. Tensor Decomposition. In *Unsupervised Feature Extraction Applied To Bioinformatics*, pages 47–78. 2020.
- [28] Y-h. Taguchi. *Unsupervised Feature Extraction Applied to Bioinformatics*. Unsupervised and Semi-Supervised Learning. Springer International Publishing, Cham, 2020.
- [29] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearsons, first edit edition, 2014.
- [30] Nina Tuxen, Sandra Roost, Julie L. L. Kofoed, Philip John Binning, Julie Chambon, and Poul L. Bjerg. Skelnen mellem pesticidkilder. Miljøprojekt nr. 1502. Technical report, Miljøstyrelsen, 2013.
- [31] Velimir V. Vesselinov. Private Communication, 2019.
- [32] Velimir V. Vesselinov. Mads: Model analysis & decision support, 2019.
- [33] Velimir V. Vesselinov. Nmfk: Nonnegative matrix factorization using k-means clustering, 2019.
- [34] Velimir V. Vesselinov. Ntfk: Nonnegative tensor factorization using k-means clustering, 2019.

- [35] Velimir V. Vesselinov, Boian S. Alexandrov, and Daniel O'Malley. Contaminant source identification using semi-supervised machine learning. *Journal of Contaminant Hydrology*, 212(Sp. Iss. SI):134–142, may 2018.
- [36] Velimir V. Vesselinov, Boian S. Alexandrov, and Daniel O'Malley. Nonnegative tensor factorization for contaminant source identification. *Journal of Contaminant Hydrology*, 220:66–97, jan 2019.
- [37] V.V. Vesselinov, M.K. Mudunuru, S. Karra, D. O'Malley, and B.S. Alexandrov. Unsupervised machine learning based on non-negative tensor factorization for analyzing reactive-mixing. *Journal of Computational Physics*, 395:85–104, oct 2019.
- [38] Yong Xiang, Dezhong Peng, and Zuyuan Yang. *Blind Source Separation*. SpringerBriefs in Electrical and Computer Engineering. Springer Singapore, Singapore, 2015.

Bilag

A. Anvendte værktøjer

Systemet er udviklet og konfigureret ved brug af følgende værktøjer:

- Azure Virtual Machine:
 - * Operativsystem: Ubuntu 18.04
 - * Størrelse: Standard_NC24_Promo
 - * Harddisk: 512 Gb
- Docker 19.03.4
 - * Image: Microsoft SQL Server (https://hub.docker.com/_/microsoft-mssql-server)
 - * Image: nvidia/cuda (<https://hub.docker.com/r/nvidia/cuda/>)
 - * Images from <https://github.com/jupyter/docker-stacks>
- Docker compose 1.24.1
- Julia 1.2 (<https://julialang.org/>)
- Julia pakker:
 - * NMFk (<https://github.com/TensorDecompositions/NmFk.jl>)
 - * MTFk (<https://github.com/TensorDecompositions/NTFk.jl>)
 - * Mads (<https://madsjulia.github.io/Mads.jl/>)
 - * m.fl. ...
- SQL Server Management Studio 18.4 og Azure Data Studio 1.13.1
- Visual studio Code 1.40
- Diverse kommandolinjeværktøjer bl.a.
 - * tmux (<https://github.com/tmux/tmux>)
 - * vim og neovim

Rapporten er skrevet med LaTeX med håndtering af referencer i BibTeX.

B. Testresultater - eksempel for test 3

NMFk Test 3

Dato: 2019-12-12 Dir: /home/ak/work/NMFk/Birkerod/Test3

TEST:NMFk.execute(T, 2:10; quiet=false, scale=false)

Testresultat:

Tid: 16 minutes, 1 second, 878 milliseconds Optimal solution: 6 Fit: 7.269577e-6 Silhouette: 0.99997014 AIC: -9009.638

Dataset:

Size T: (14, 5, 22) Size wells: 14 Size years: 22 Size species: 5

Years:

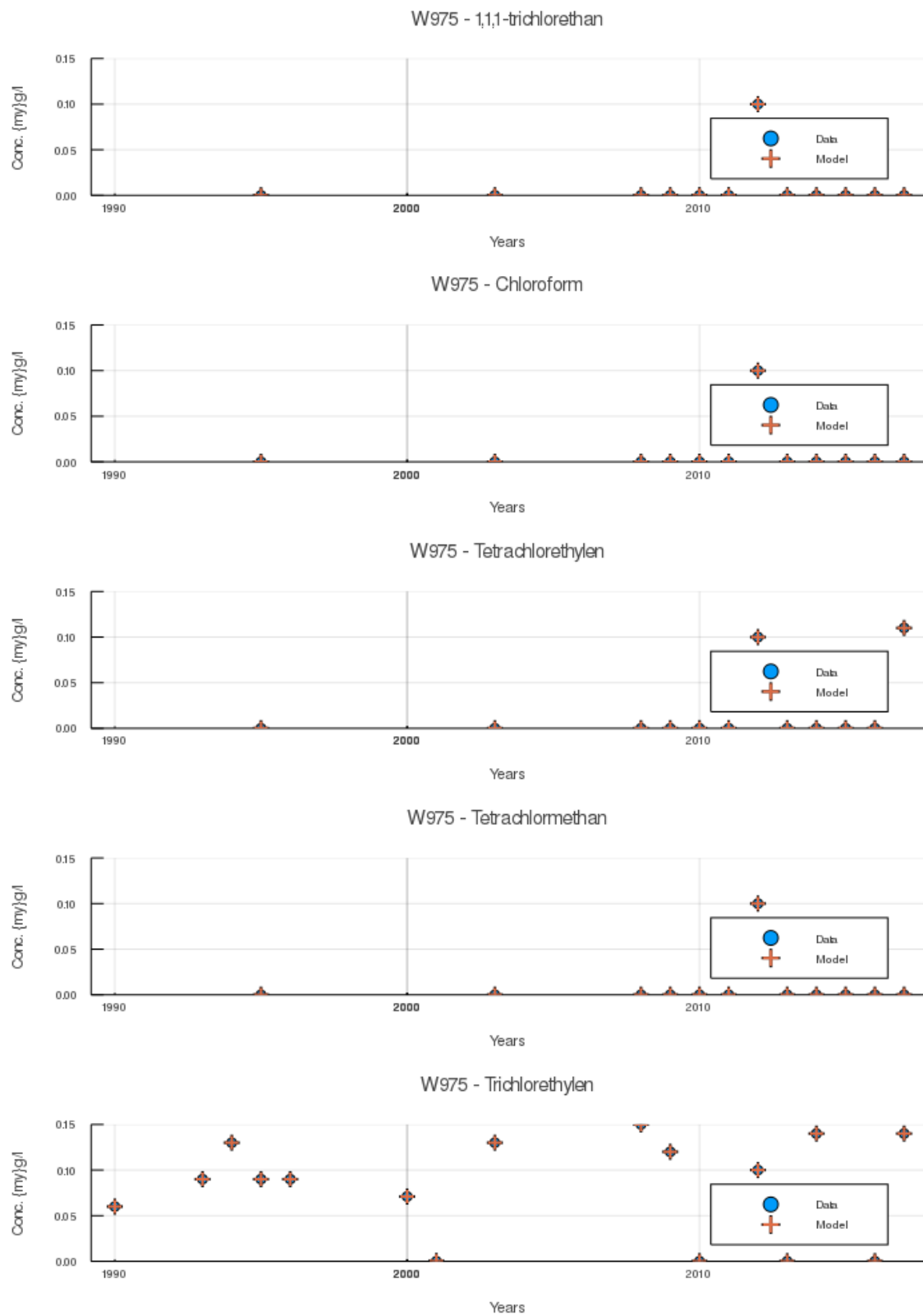
[1990, 1993, 1994, 1995, 1996, 2000, 2001, 2002, 2003, 2004, 2005, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]

Wells:

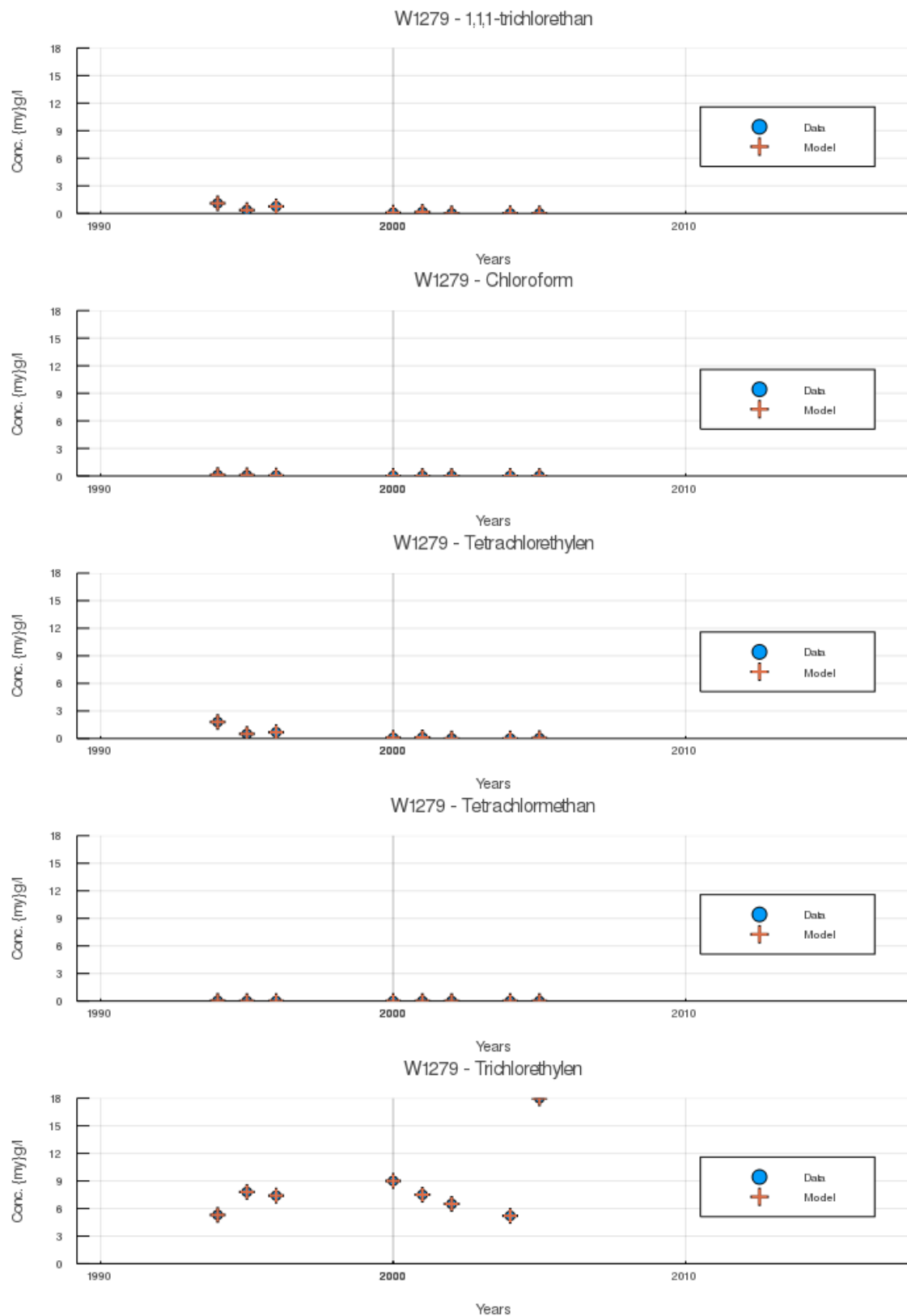
Wellno;X;Y W1279; 715184.3; 6.1945128e6 W1478; 715362.3; 6.1937388e6 W1571; 715631.3; 6.1934918e6 W1575; 715381.3; 6.1936438e6 W1611; 715337.3; 6.1935898e6 W1612; 715397.3; 6.1935938e6 W1762; 713321.3; 6.1933698e6 W1929; 715403.3; 6.1936888e6 W1930; 715348.3; 6.1935868e6 W1931; 715427.3; 6.1936878e6 W1932; 715450.3; 6.1936688e6 W1933; 715363.3; 6.1936788e6 W1934; 715326.3; 6.1936358e6 W975; 715582.3; 6.1943568e6

Species:

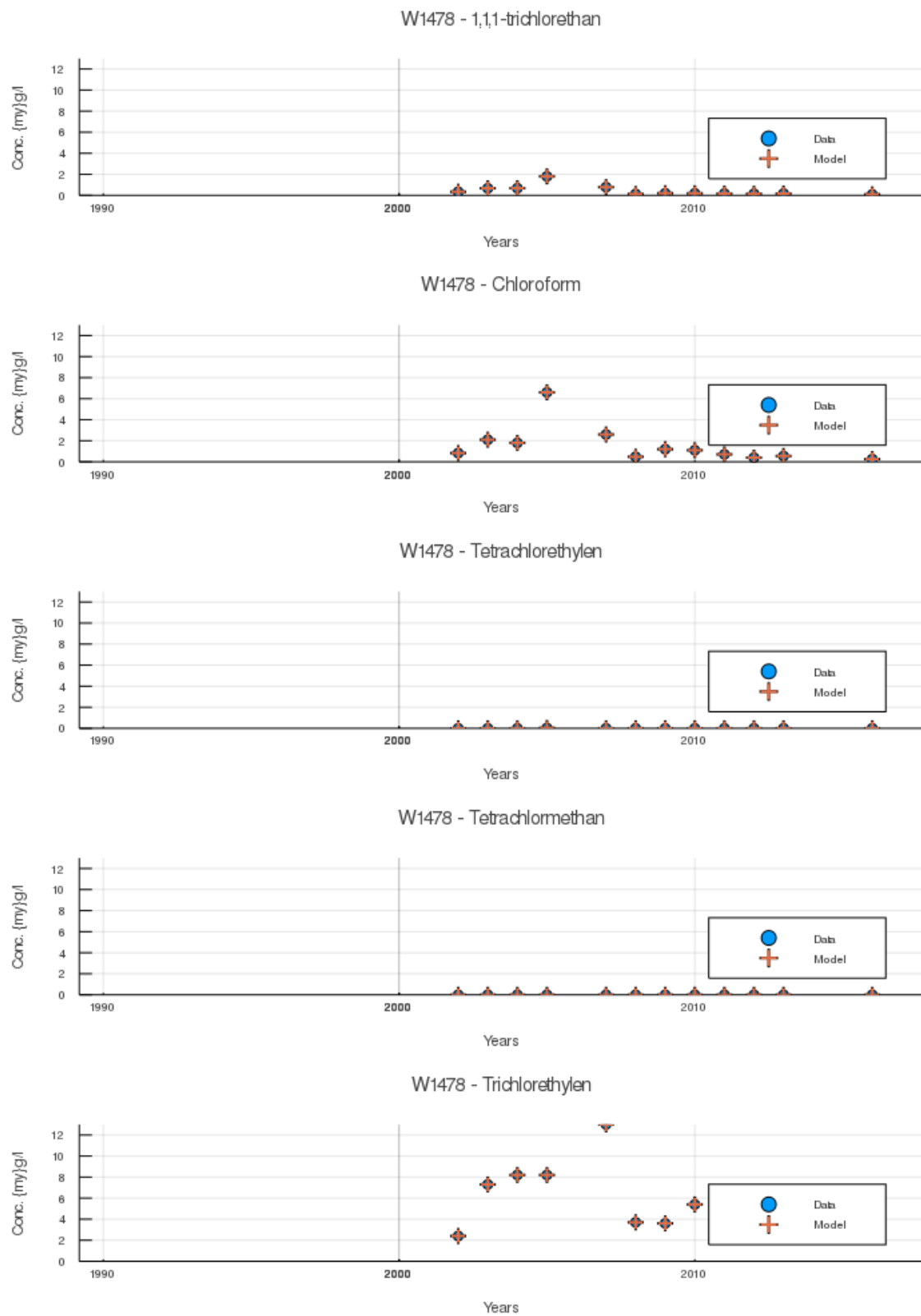
SpeciesNo;Species 2612; Chloroform 2616; Tetrachlormethan 2617; Tetrachlorethylen 2618; Trichlorethylen 2621; 1,1,1-trichlorethan



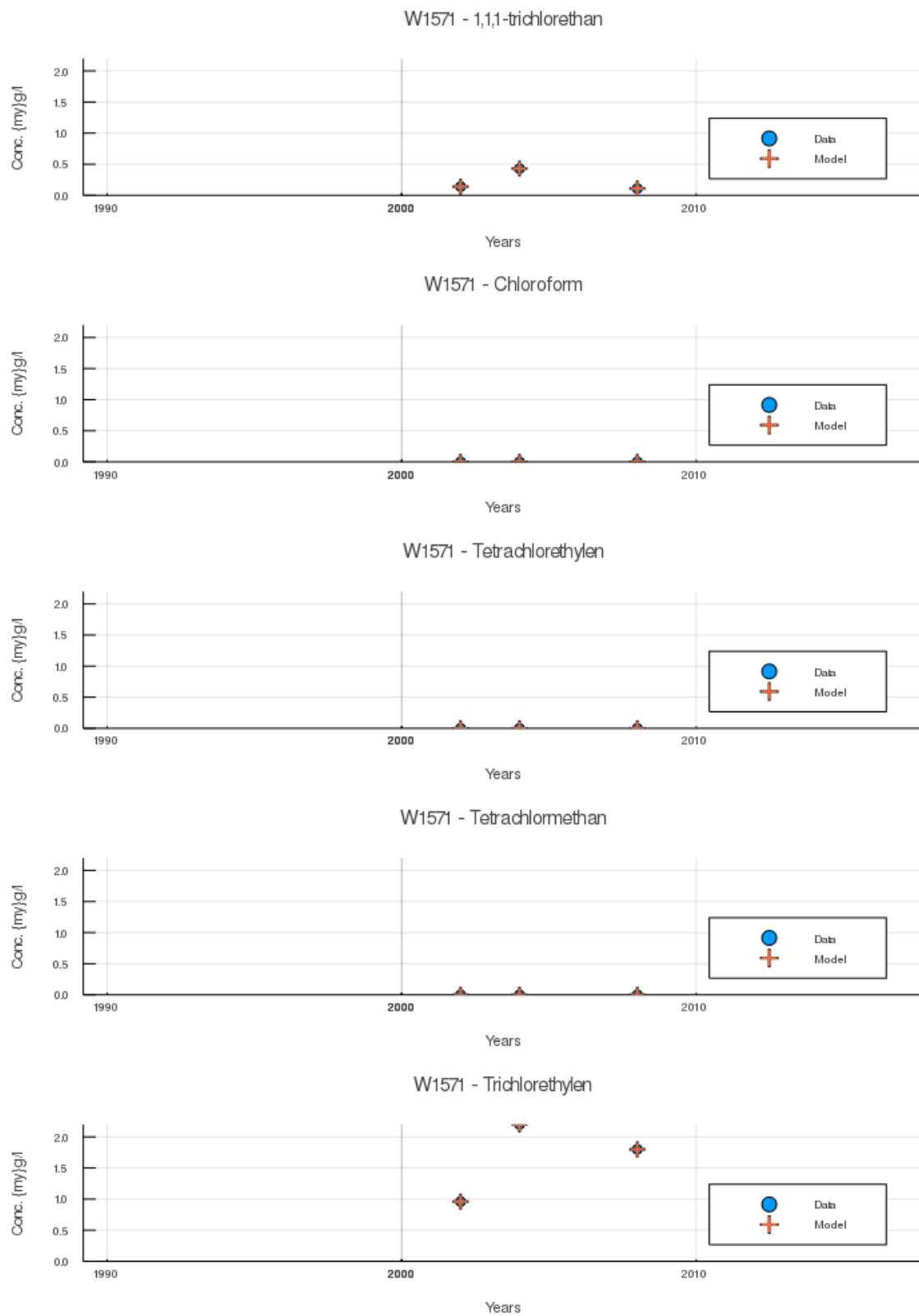
Figur B.1: Test 3 - estimerede og faktiske koncentrationer i boring W975



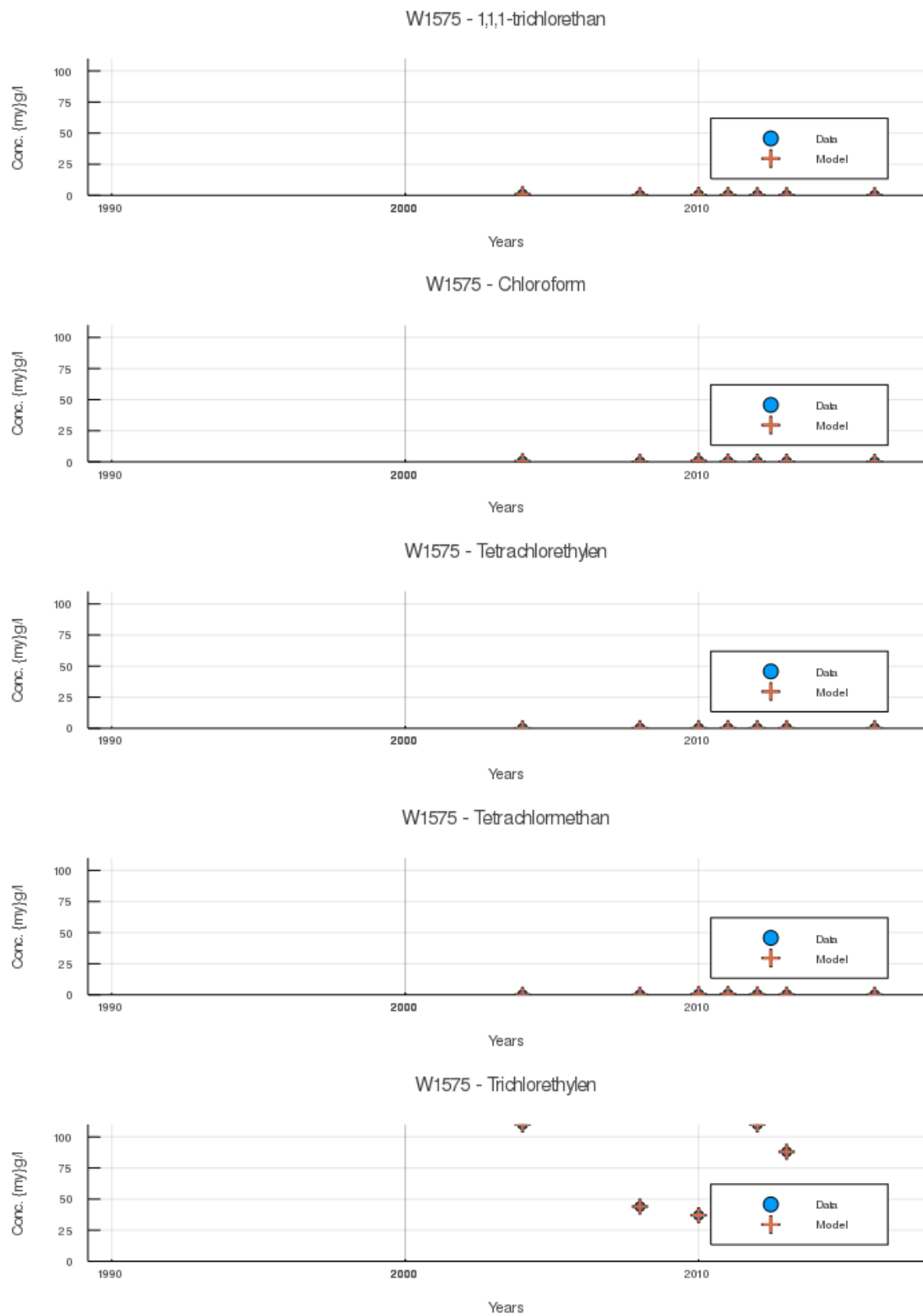
Figur B.2: Test 3 - estimerede og faktiske koncentrationer i boring W1279



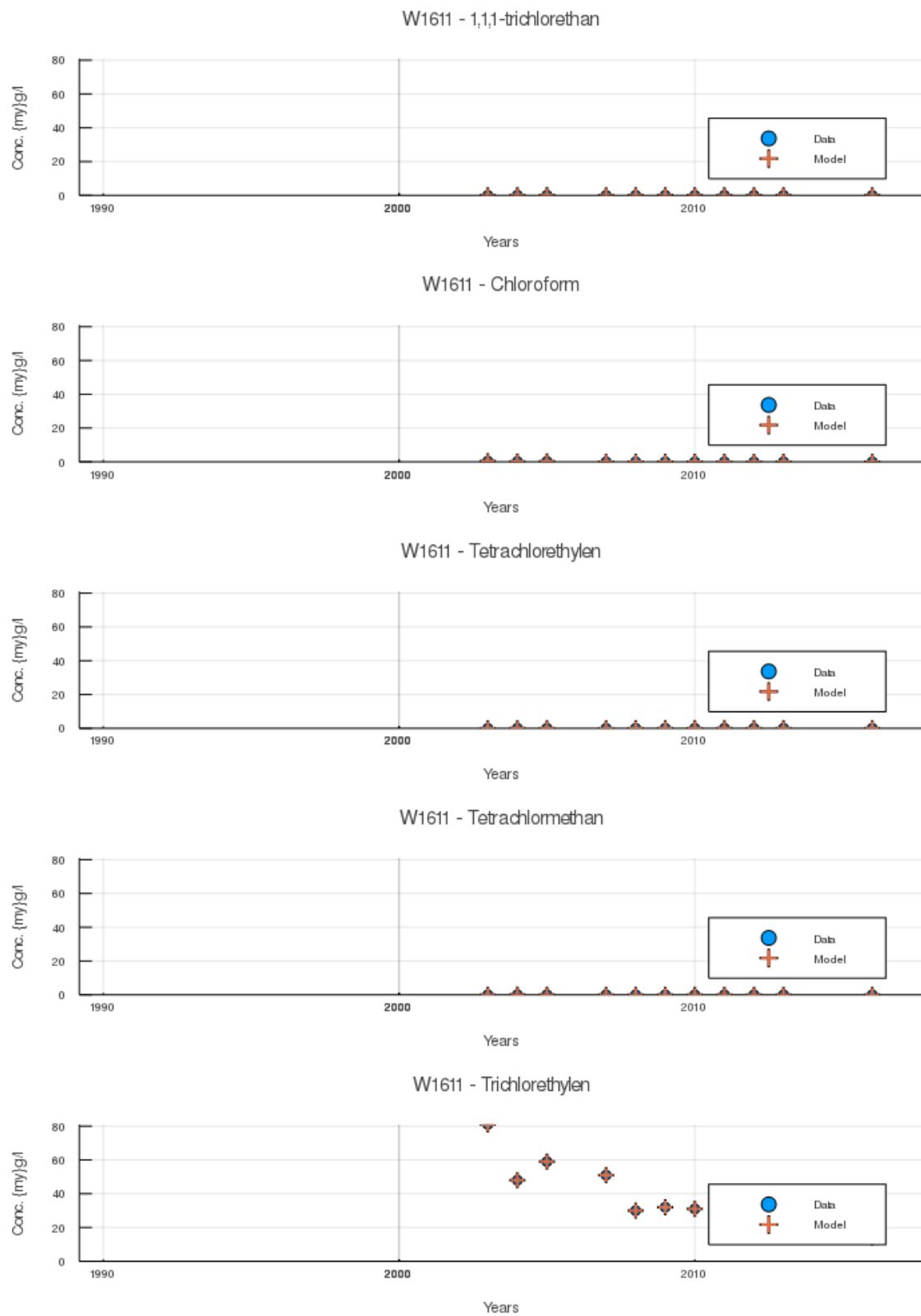
Figur B.3: Test 3 - estimerede og faktiske koncentrationer i boring W1478



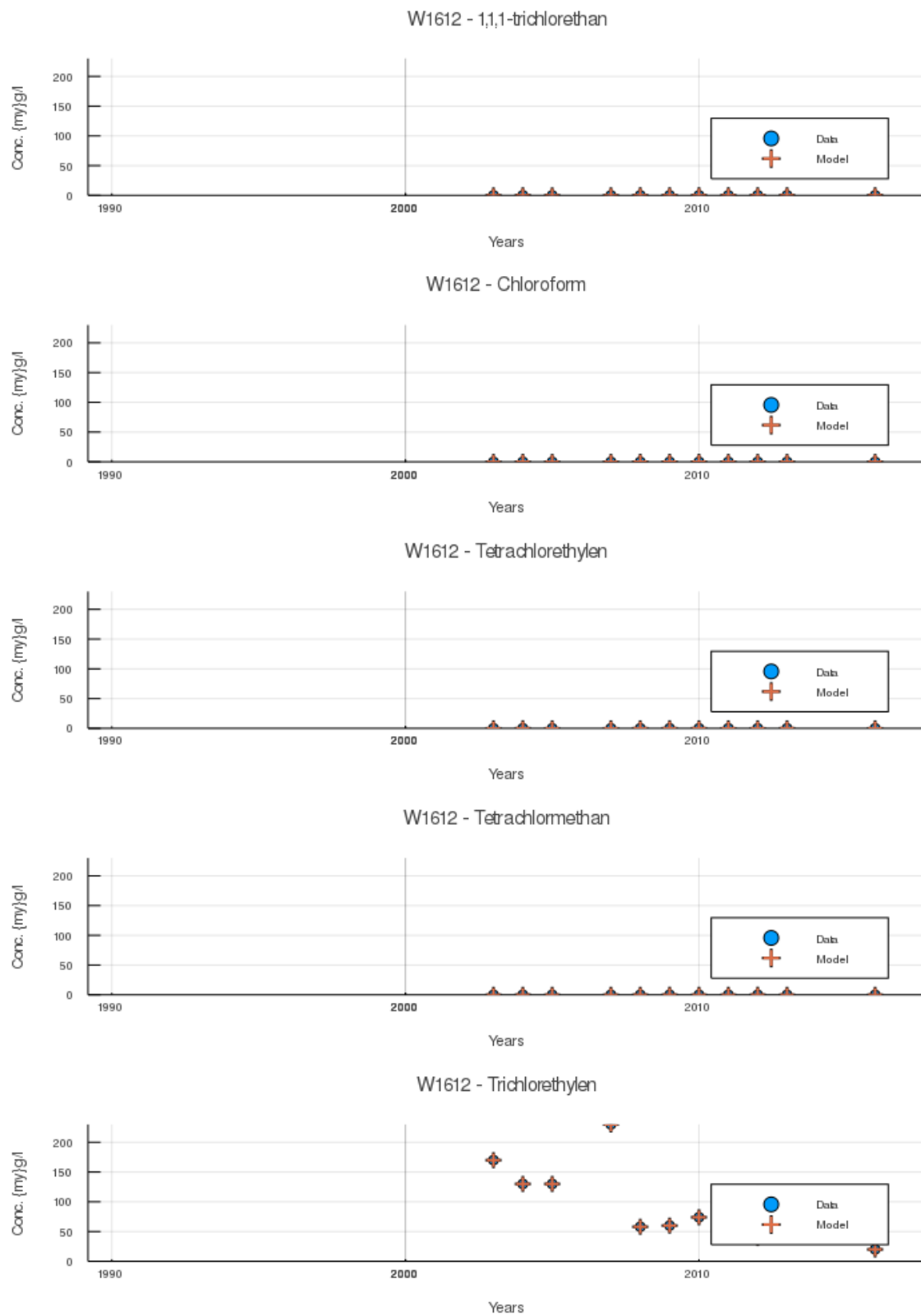
Figur B.4: Test 3 - estimerede og faktiske koncentrationer i boring W1571



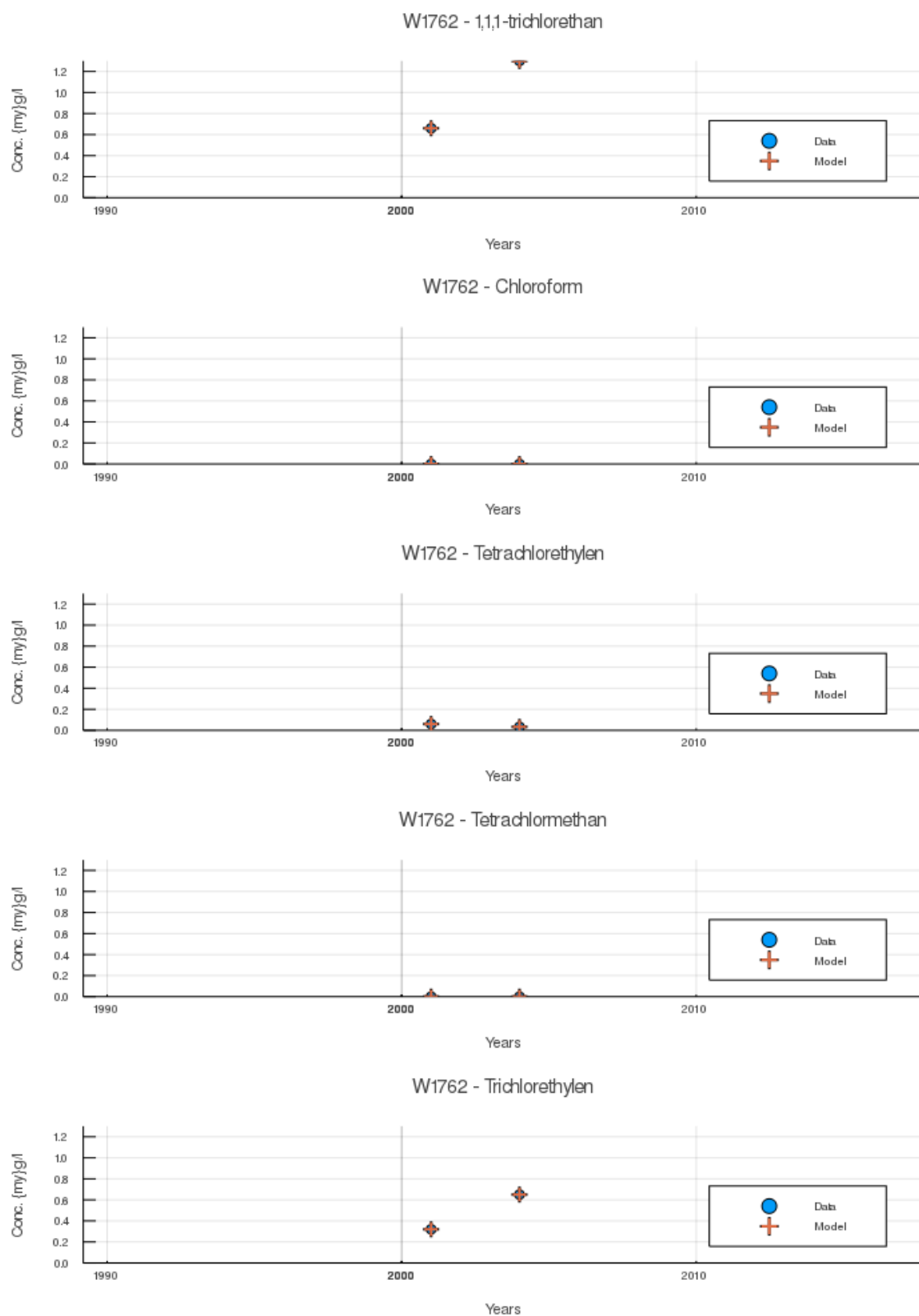
Figur B.5: Test 3 - estimerede og faktiske koncentrationer i boring W1575



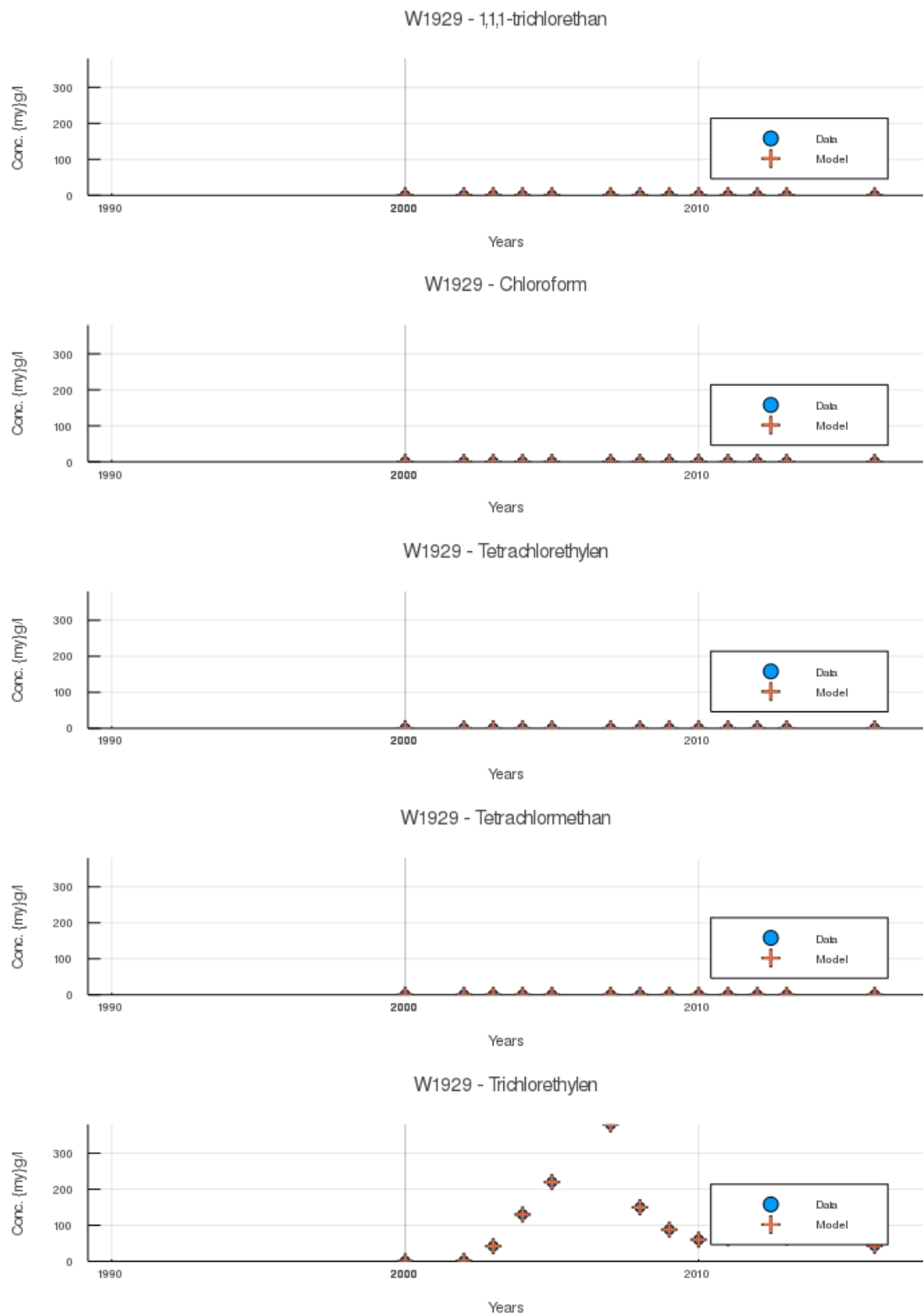
Figur B.6: Test 3 - estimerede og faktiske koncentrationer i boring W1611



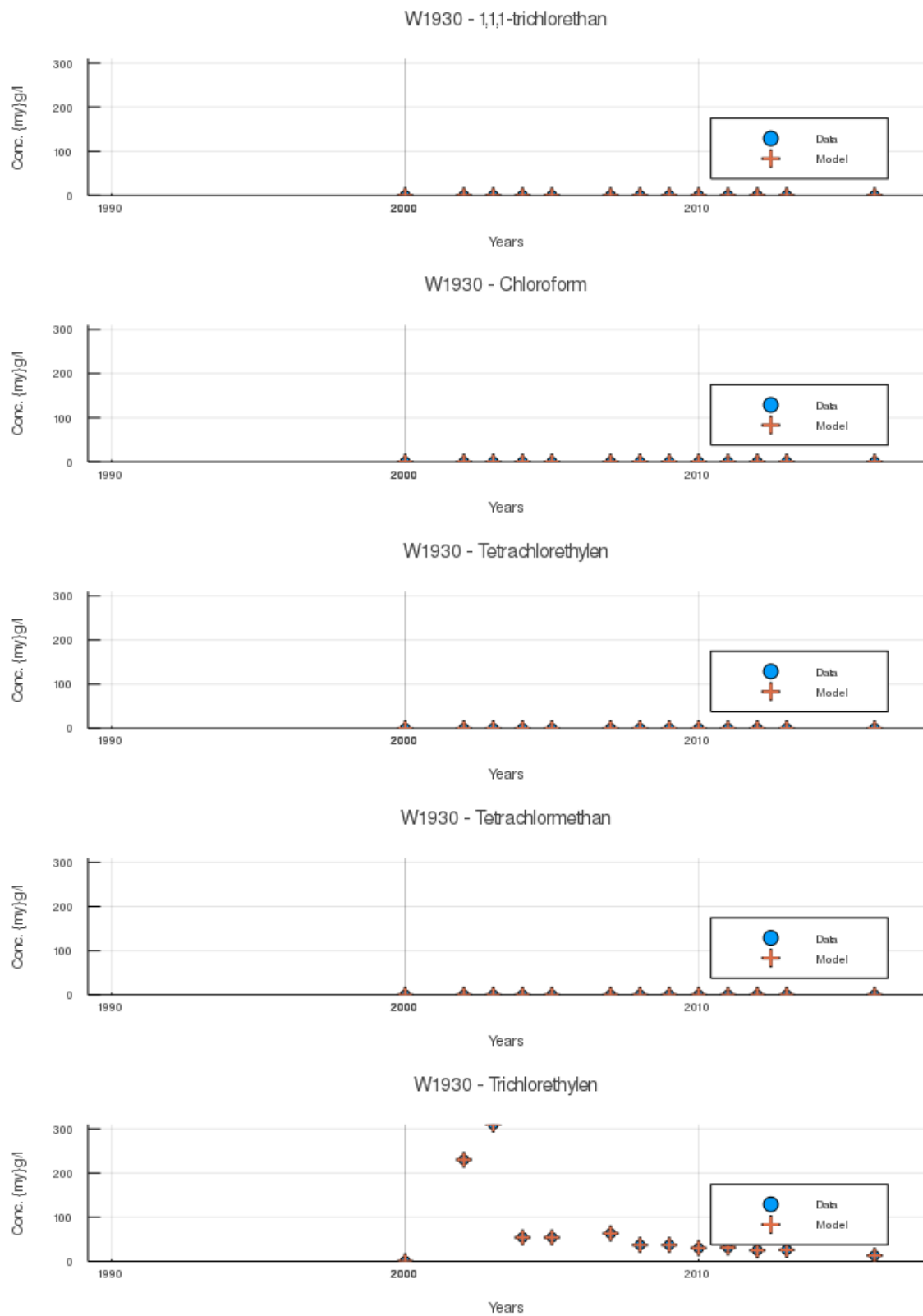
Figur B.7: Test 3 - estimerede og faktiske koncentrationer i boring W1612



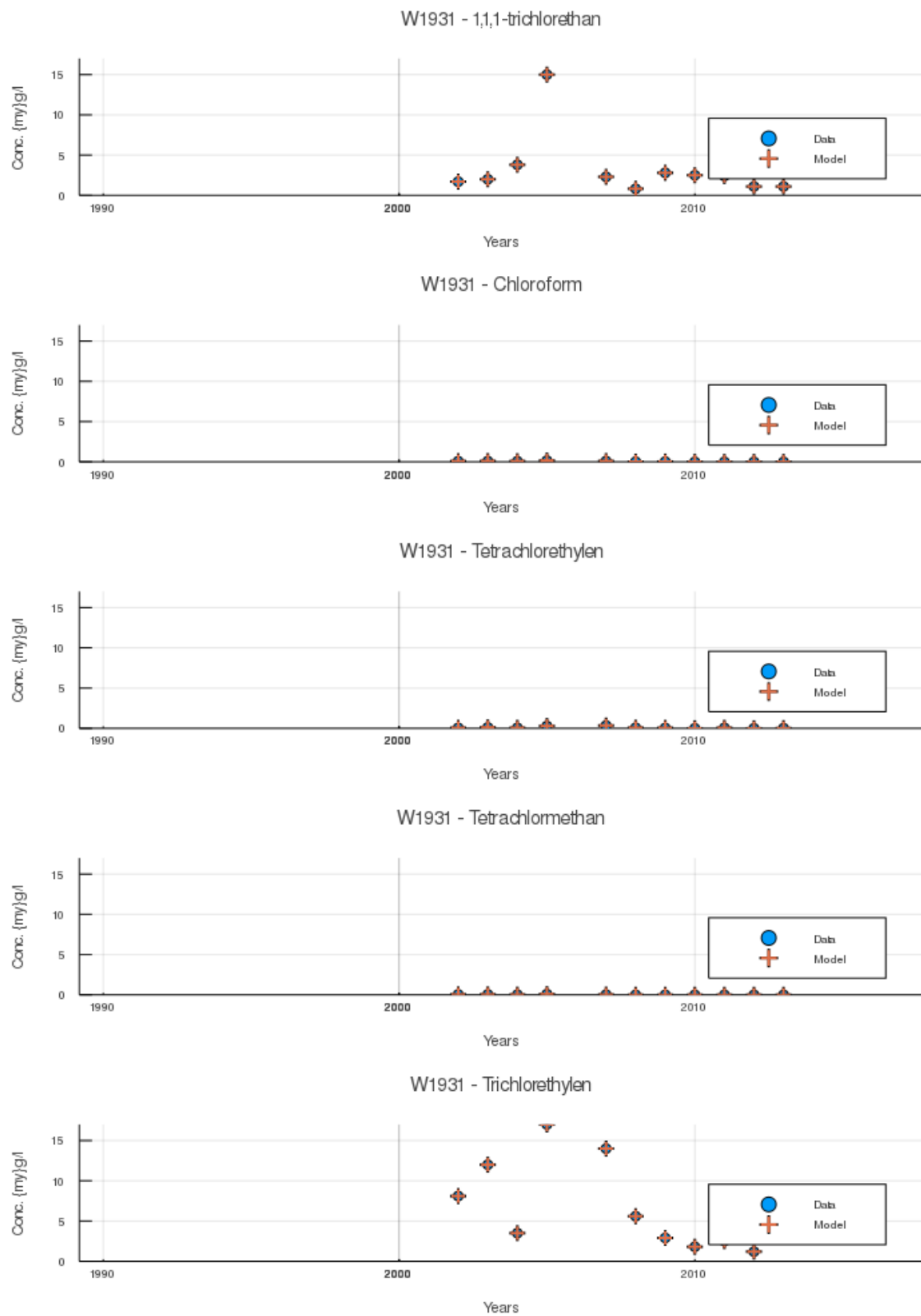
Figur B.8: Test 3 - estimerede og faktiske koncentrationer i boring W1762



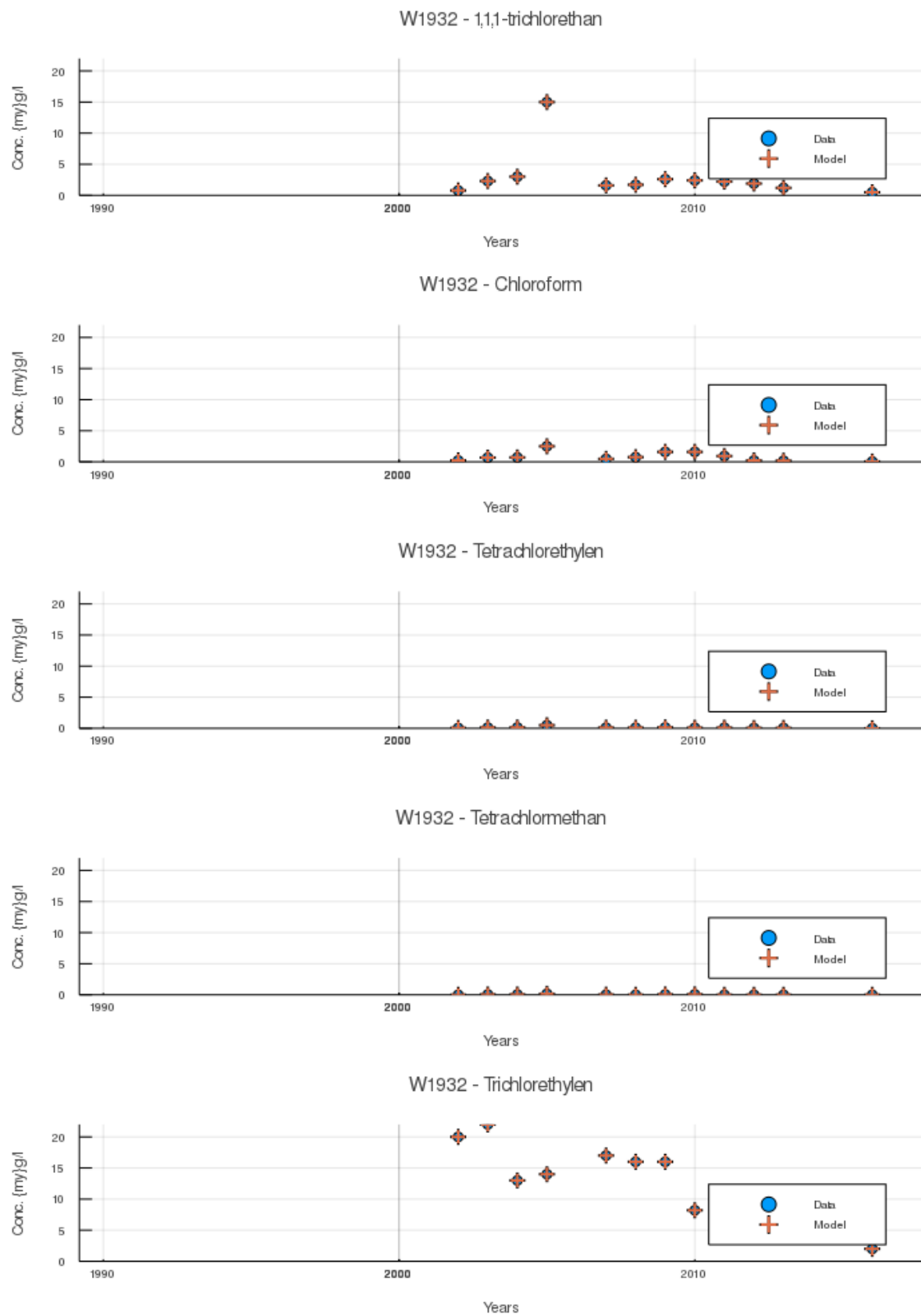
Figur B.9: Test 3 - estimerede og faktiske koncentrationer i boring W1929



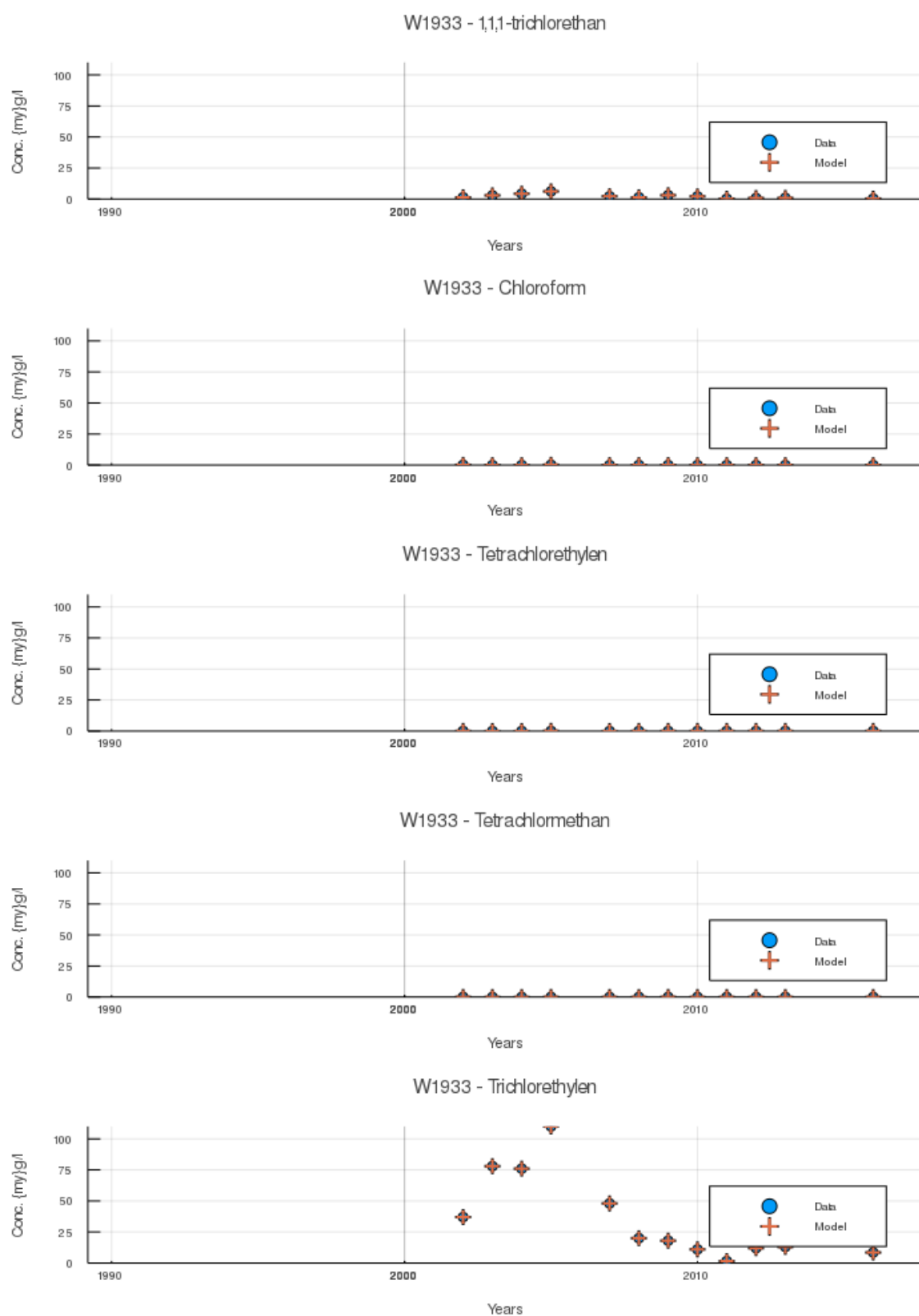
Figur B.10: Test 3 - estimerede og faktiske koncentrationer i boring W1930



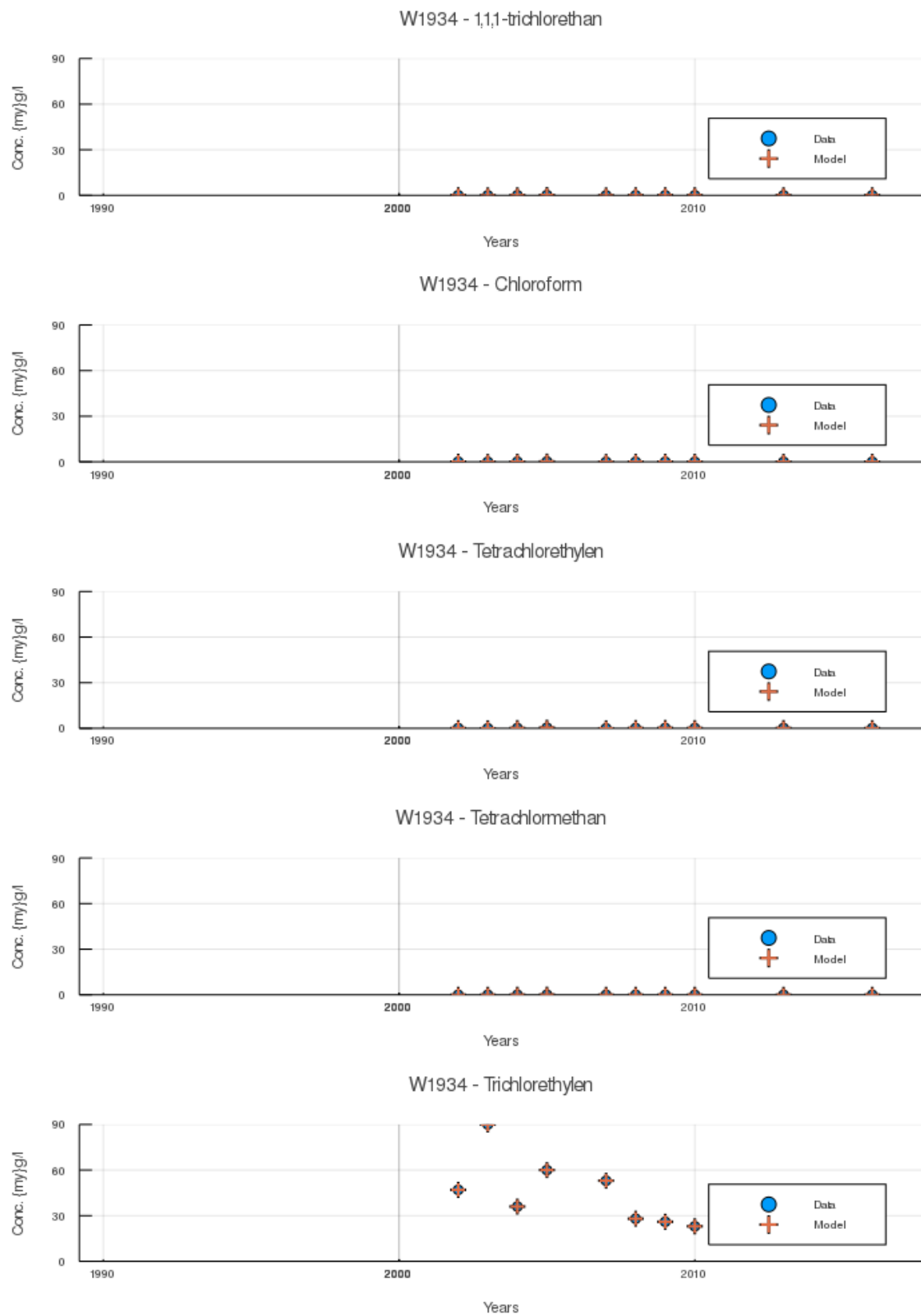
Figur B.11: Test 3 - estimerede og faktiske koncentrationer i boring W1931



Figur B.12: Test 3 - estimerede og faktiske koncentrationer i boring W1932



Figur B.13: Test 3 - estimerede og faktiske koncentrationer i boring W1933



Figur B.14: Test 3 - estimerede og faktiske koncentrationer i boring W1934

C. Udskrift af kode - Dockerfile mssql

```
1 FROM mcr.microsoft.com/mssql/server:latest
2
3 # Update packages and install
4 RUN apt-get -y update && \
5     apt-get install -y apt-utils \
6         zip \
7         unzip \
8         cron
9
10
11 # Create scripts directory
12 RUN mkdir -p /usr/local/scripts
13
14 # Copy scripts and set scripts to be executable
15 COPY .netrc /root/
16 COPY entrypoint.sh /root/
17 RUN chmod +x /root/entrypoint.sh
18
19
20 # Create the log file to be able to run tail
21 RUN touch /var/log/cron.log
22
23 # Add crontab file in the cron directory
24 ADD dbrestore-schedule /etc/cron.d/dbrestore-schedule
25 RUN chmod 0644 /etc/cron.d/dbrestore-schedule
26 RUN crontab /etc/cron.d/dbrestore-schedule
27
28
29 # Run entrypoint.sh
30 CMD exec /bin/bash /root/entrypoint.sh
31
```

D. Udskrift af koden - Scripts på mssql

```
1 #start SQL Server, start the script to restore the databases
2 echo ""
3 echo "Running entrypoint"
4 echo ""
5
6 /scripts/restore-data.sh &
7 exec /opt/mssql/bin/sqlservr
8
9
10 system cron start
11 cron -f
12
```

```

1 #!/bin/bash
2 #wait for the SQL Server to come up
3 sleep 30s
4
5 echo ""
6 echo ""
7 echo ""
8 echo "*****"
9 echo "* RESTORING DATABASES                                *"
10 echo "*"                                                  *"
11 echo "*****"
12 echo ""
13 echo "Date: $(date)"
14 echo ""
15
16 export HOME=/root
17 cd /root
18
19 # JupiterXL download
20 download=0
21 file=/downloads/jupiterxl_rh_XL.zip
22 if [ -f "$file" ]; then
23     echo "$file exist - checking date ..."
24     if [ "$((($(date +%s) - $(date -r $file +%s))/3600))" -gt "20" ]; then
25         echo "$file is to old"
26         download=1
27     else
28         echo "$file is up to date"
29     fi
30 else
31     echo "$file does not exist"
32     echo "Starting download ..."
33     download=1
34 fi
35
36 if [ "$download" -eq "1" ]; then
37     wget http://gerda.geus.dk/Gerda/exportfiles
38     /?filename=jupiterxl_rh_XL.zip -O $file
39     unzip -o $file -d /downloads/
40     /opt/mssql-tools/bin/sqlcmd -i /scripts/restore-jupiterxl.sql
41     -S"localhost" -U"sa" -P"*****"
42 fi
43
44 ## GeoGIS Download
45 download=0
46 file=/downloads/GeoGIS2020RegionH.bak
47 if [ -f "$file" ]; then
48     echo "$file exist - checking date ..."
49     if [ "$((($(date +%s) - $(date -r $file +%s))/3600))" -gt "20" ]; then
50         echo "$file is to old"
51         download=1
52     else
53         echo "$file is up to date"
54     fi
55 else
56     echo "$file does not exist"

```

```

55     echo "Starting download ..."
56     download=1
57 fi
58
59 if [ "$download" -eq "1" ]; then
60     wget --no-check-certificate ftps://***** /GeoGIS2020RegionH.bak
61     -O $file
62     /opt/mssql-tools/bin/sqlcmd -i /scripts/restore-geogis2020.sql
63     -S"localhost" -U"sa" -P"*****"
64 fi
65
66 # JAR download
67 download=0
68 file=/downloads/RH_jar_drift_eksport.bak
69 if [ -f "$file" ]; then
70     echo "$file exist - checking date ..."
71     if [ "$((($(date +%s) - $(date -r $file +%s))/3600))" -gt "20" ]; then
72         echo "$file is to old"
73         download=1
74     else
75         echo "$file is up to date"
76     fi
77 else
78     echo "$file does not exist"
79     echo "Starting download ..."
80     download=1
81 fi
82
83 if [ "$download" -eq "1" ]; then
84     wget ftp://***** /RH_jar_drift_daglig_db_eksport
85     /RH_jar_drift_eksport.bak -O $file
86     /opt/mssql-tools/bin/sqlcmd -i /scripts/restore-jar.sql -S"localhost"
87     -U"sa" -P"*****"
88 fi
89
90 echo ""
91 echo ""
92
93 #run the setup script to create the DB and the schema in the DB
94 #/opt/mssql-tools/bin/sqlcmd -S localhost -U sa -P Yukon900 -d master -i
95 setup.sql
96
97 #import the data from the csv file
98 #/opt/mssql-tools/bin/bcp DemoData.dbo.Products in "/usr/src
99 /app/Products.csv" -c -t',' -S localhost -U sa -P Yukon900
100

```

```
1 USE [master]
2 GO
3
4 DROP DATABASE JupiterXL
5 GO
6
7 RESTORE DATABASE [JupiterXL]
8 FROM DISK = N'/downloads/jupiterxl_rh_xl.bak'
9 WITH FILE = 1
10 ,MOVE N'jupiterxl_rh_xl' TO N'/var/opt/mssql/data/JupiterXL.mdf'
11 ,MOVE N'jupiterxl_rh_xl_log' TO N'/var/opt/mssql/data/JupiterXL_log.LDF'
12 ,NOUNLOAD
13 ,STATS = 5
14
15
```

```
USE [master]
GO
DROP DATABASE GeoGIS2020
GO
RESTORE DATABASE [GeoGIS2020]
FROM DISK = N'/downloads/GeoGIS2020RegionH.bak'
WITH FILE = 1
    ,MOVE N'GeoGIS2020RegionHovedstaden' TO N'/var/opt/mssql
/data/GeoGIS2020.mdf'
    ,MOVE N'GeoGIS2020RegionHovedstaden_log' TO N'/var/opt/mssql
/data/GeoGIS2020.ldf'
    ,NOUNLOAD
    ,STATS = 5
```



```
1 USE [master]
2 GO
3
4 DROP DATABASE JAR
5 GO
6
7
8 RESTORE DATABASE JAR
9 FROM DISK='/downloads/RH_jar_drift_eksport.bak'
10 WITH FILE = 1
11     ,MOVE 'JAR_Default_dat' TO '/var/opt/mssql/data/JAR.mdf'
12     ,MOVE 'JAR_Default_dat3' TO '/var/opt/mssql/data/JAR2.mdf'
13     ,MOVE 'jar_log' TO '/var/opt/mssql/data/JAR.ldf'
14     ,NOUNLOAD
15     ,STATS = 5
16 GO
17
```

E. Udskrift af koden: Dockerfile lab

```
1 ARG BASE_CONTAINER=nvidia/cuda:latest
2 FROM $BASE_CONTAINER
3
4
5 LABEL maintainer="Allan Kötter <***** >"
6
7 USER root
8 RUN set -eux
9
10 # Set when building on Travis so that certain long-running build steps can
11 # be skipped to shorten build time.
12 ARG TEST_ONLY_BUILD
13
14 ARG NB_USER="ak"
15 ARG NB_UID="1000"
16 ARG NB_GID="100"
17
18 # Install all OS dependencies for notebook server that starts but lacks all
19 # features (e.g., download as all possible file formats)
20 ENV DEBIAN_FRONTEND noninteractive
21
22 RUN apt-get update && apt-get -yq dist-upgrade \
23 && apt-get install -yq --no-install-recommends \
24     libcurl4 \
25     curl \
26     wget \
27     bzip2 \
28     ca-certificates \
29     debconf \
30     sudo \
31     locales \
32     fonts-liberation \
33     run-one \
34     build-essential \
35     neovim \
36     git \
37     inkscape \
38     jed \
39     libsm6 \
40     libc6 \
41     libstdc++6 \
42     libkrb5-3 \
43     libxext-dev \
44     libxrender1 \
45     lmodern \
46     netcat \
47     openssl \
48     pandoc \
49     python-dev \
50     texlive-fonts-extra \
51     texlive-fonts-recommended \
52     texlive-generic-recommended \
53     texlive-latex-base \
54     texlive-latex-extra \
55     texlive-xetex \
56     tzdata \
```

```
57     unzip \
58     ffmpeg \
59     fonts-dejavu \
60     gfortran \
61     gcc
62
63 ## ODBC
64
65 RUN curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key
add -
66 RUN curl https://packages.microsoft.com/config/ubuntu/18.04/prod.list >
/etc/apt/sources.list.d/mssql-release.list
67
68 ARG ACCEPT_EULA=Y
69 RUN apt-get update && apt-get -yq dist-upgrade \
70     && apt-get install -yq --no-install-recommends \
71     msodbcsql17 \
72     mssql-tools
73
74 RUN apt-get install -yq --no-install-recommends \
75     unixodbc-dev \
76     unixodbc
77
78
79 # libraries fra Mads
80 RUN apt-get -y install make g++ vim gettext-base libcroco3 libglib2.0-0
libglib2.0-data libgpm2 libicu60 libncurses5 libxml2 sgml-base shared-mime-
info xdg-user-dirs xml-core fontconfig-config fonts-dejavu-core libexpat1
libfontconfig1 libfreetype6 libbsd0 libcairo2 libpixmap-1-0 libx11-6
libx11-data libxau6 libxcb-render0 libxcb-shm0 libxcb1 libxdmcp6 libxext6
libxrender1 libpng16-16 ucf fontconfig libdatatriel libgraphite2-3
libharfbuzz0b libpango-1.0-0 libpangol1.0-0 libpangocairo-1.0-0
libpangoft2-1.0-0 libpangox-1.0-0 libpangoxft-1.0-0 libthai-data libthai0
libxft2 python3 libpython3.5
81
82
83 RUN rm -rf /var/lib/apt/lists/*
84
85
86 RUN echo "en_US.UTF-8 UTF-8" > /etc/locale.gen && \
87     locale-gen
88
89
90 # Add local files as late as possible to avoid cache busting
91 COPY start.sh /usr/local/bin/
92 COPY start-notebook.sh /usr/local/bin/
93 COPY start-singleuser.sh /usr/local/bin/
94 COPY jupyter_notebook_config.py /etc/jupyter/
95
96 # Add a script that we will use to correct permissions after running
certain commands
97 ADD fix-permissions /usr/local/bin/fix-permissions
98
99 RUN chmod +x /usr/local/bin/*.sh
100 RUN chmod +x /usr/local/bin/fix-permissions
101
102 # Configure environment
103 ENV CONDA_DIR=/opt/conda \
```

```

104     SHELL=/bin/bash \
105     NB_USER=$NB_USER \
106     NB_UID=$NB_UID \
107     NB_GID=$NB_GID \
108     LC_ALL=en_US.UTF-8 \
109     LANG=en_US.UTF-8 \
110     LANGUAGE=en_US.UTF-8
111 ENV PATH=$CONDA_DIR/bin:$PATH \
112     HOME=/home/$NB_USER
113
114 # Enable prompt color in the skeleton .bashrc before creating the default
NB_USER
115 RUN sed -i 's/^#force_color_prompt=yes/force_color_prompt=yes/' /etc/skel
/.bashrc
116
117 # Create NB_USER with name jovyan user with UID=1000 and in the 'users'
group
118 # and make sure these dirs are writable by the `users` group.
119 RUN echo "auth requisite pam_deny.so" >> /etc/pam.d/su && \
120     sed -i.bak -e 's/^%admin/#%admin/' /etc/sudoers && \
121     sed -i.bak -e 's/^%sudo/#%sudo/' /etc/sudoers && \
122     useradd -m -s /bin/bash -N -u $NB_UID $NB_USER && \
123     mkdir -p $CONDA_DIR
124
125 RUN chown $NB_USER:$NB_GID $CONDA_DIR && \
126     chmod g+w /etc/passwd
127
128
129 RUN fix-permissions $HOME
130 RUN fix-permissions "$(dirname $CONDA_DIR)"
131
132 USER $NB_UID
133 WORKDIR $HOME
134
135 RUN echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> $HOME/.bash_profile
136 RUN echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> $HOME/.bashrc
137
138
139 # Setup work directory for backward-compatibility
140 RUN mkdir /home/$NB_USER/work && \
141     fix-permissions /home/$NB_USER
142
143 # Install conda as jovyan and check the md5 sum provided on the download
site
144 ENV MINICONDA_VERSION=4.6.14 \
145     CONDA_VERSION=4.7.10
146
147 RUN cd /tmp && \
148     wget --quiet https://repo.continuum.io/miniconda
/Miniconda3-{$MINICONDA_VERSION}-Linux-x86_64.sh && \
149     echo "718259965f234088d785cad1fbd7de03
*Miniconda3-{$MINICONDA_VERSION}-Linux-x86_64.sh" | md5sum -c - && \
150     /bin/bash Miniconda3-{$MINICONDA_VERSION}-Linux-x86_64.sh -f -b -p
$CONDA_DIR && \
151     rm Miniconda3-{$MINICONDA_VERSION}-Linux-x86_64.sh && \
152     echo "conda {$CONDA_VERSION}" >> $CONDA_DIR/conda-meta/pinned && \
153     $CONDA_DIR/bin/conda config --system --prepend channels conda-forge &&
\

```

```

154     $CONDA_DIR/bin/conda config --system --set auto_update_conda false && \
155     $CONDA_DIR/bin/conda config --system --set show_channel_urls true && \
156     $CONDA_DIR/bin/conda install --quiet --yes conda && \
157     $CONDA_DIR/bin/conda update --all --quiet --yes && \
158     conda list python | grep '^python ' | tr -s ' ' | cut -d '.' -f 1,2 |
sed 's/$/.*/' >> $CONDA_DIR/conda-meta/pinned && \
159     conda clean --all -f -y && \
160     rm -rf /home/$NB_USER/.cache/yarn && \
161     fix-permissions $CONDA_DIR && \
162     fix-permissions /home/$NB_USER
163
164 # Install Tini
165 RUN conda install --quiet --yes 'tini=0.18.0' && \
166     conda list tini | grep tini | tr -s ' ' | cut -d ' ' -f 1,2 >>
$CONDA_DIR/conda-meta/pinned && \
167     conda clean --all -f -y && \
168     fix-permissions $CONDA_DIR && \
169     fix-permissions /home/$NB_USER
170
171
172 # Install Jupyter Notebook, Lab, and Hub
173 # Generate a notebook server config
174 # Cleanup temporary files
175 # Correct permissions
176 # Do all this in a single RUN command to avoid duplicating all of the
177 # files across image layers when the permissions change
178 RUN conda install --quiet --yes \
179     'notebook' \
180     'jupyterhub' \
181     'jupyterlab' && \
182     conda clean --all -f -y && \
183     npm cache clean --force && \
184     jupyter notebook --generate-config && \
185     rm -rf $CONDA_DIR/share/jupyter/lab/staging && \
186     rm -rf /home/$NB_USER/.cache/yarn && \
187     fix-permissions $CONDA_DIR && \
188     fix-permissions /home/$NB_USER
189
190 # Install Python 3 packages
191 RUN conda install --quiet --yes \
192     'beautifulsoup4=4.8.*' \
193     'conda-forge::blas*=openblas' \
194     'bokeh=1.3.*' \
195     'cloudpickle=1.2.*' \
196     'cython=0.29.*' \
197     'dask=2.2.*' \
198     'dill=0.3.*' \
199     'h5py=2.9.*' \
200     'hdf5=1.10.*' \
201     'ipywidgets=7.5.*' \
202     'matplotlib-base=3.1.*' \
203     'numba=0.45.*' \
204     'numexpr=2.6.*' \
205     'pandas=0.25.*' \
206     'patsy=0.5.*' \
207     'plotly' \
208     'jupyterlab-plotly-extension' \

```

```

209     'protobuf=3.9.*' \
210     'pyodbc' \
211     'scikit-image=0.15*' \
212     'scikit-learn=0.21*' \
213     'scipy=1.3*' \
214     'seaborn=0.9*' \
215     'sqlalchemy=1.3*' \
216     'statsmodels=0.10*' \
217     'sympy=1.4*' \
218     'vincent=0.4.*' \
219     'xlrd' \
220     && conda clean --all -f -y && \
221     # Activate ipywidgets extension in the environment that runs the
notebook server
222     jupyter nbextension enable --py widgetsnbextension --sys-prefix && \
223     # Also activate ipywidgets extension for JupyterLab
224     # Check this URL for most recent compatibilities
225     # https://github.com/jupyter-widgets/ipywidgets/tree/master/packages
/jupyterlab-manager
226     jupyter labextension install @jupyter-widgets/jupyterlab-manager --no-
build && \
227     jupyter labextension install jupyterlab_bokeh --no-build && \
228     jupyter lab build --dev-build=False && \
229     npm cache clean --force && \
230     rm -rf $CONDA_DIR/share/jupyter/lab/staging && \
231     rm -rf /home/$NB_USER/.cache/yarn && \
232     rm -rf /home/$NB_USER/.node-gyp && \
233     fix-permissions $CONDA_DIR && \
234     fix-permissions /home/$NB_USER
235
236
237 # Install facets which does not have a pip or conda package at the moment
238 RUN cd /tmp && \
239     git clone https://github.com/PAIR-code/facets.git && \
240     cd facets && \
241     jupyter nbextension install facets-dist/ --sys-prefix && \
242     cd && \
243     rm -rf /tmp/facets && \
244     fix-permissions $CONDA_DIR && \
245     fix-permissions /home/$NB_USER
246
247 # Import matplotlib the first time to build the font cache.
248 ENV XDG_CACHE_HOME /home/$NB_USER/.cache/
249 RUN MPLBACKEND=Agg python -c "import matplotlib.pyplot" && \
250     fix-permissions /home/$NB_USER
251
252
253
254 ### INSTALL JULIA
255 # Julia dependencies
256 # install Julia packages in /opt/julia instead of $HOME
257 USER root
258
259 ENV JULIA_DEPOT_PATH=/opt/julia
260 ENV JULIA_PKGDIR=/opt/julia
261 ENV JULIA_VERSION=1.2.0
262

```

```

263 RUN mkdir /opt/julia-${JULIA_VERSION} && \
264     cd /tmp && \
265     wget -q https://julialang-s3.julialang.org/bin/linux/x64/`echo
${JULIA_VERSION} | cut -d. -f 1,2`/julia-${JULIA_VERSION}-linux-
x86_64.tar.gz && \
266     echo "926ced5dec5d726ed0d2919e849ff084a320882fb67ab048385849f9483afc47
*julia-${JULIA_VERSION}-linux-x86_64.tar.gz" | sha256sum -c - && \
267     tar xzf julia-${JULIA_VERSION}-linux-x86_64.tar.gz -C
/opt/julia-${JULIA_VERSION} --strip-components=1 && \
268     rm /tmp/julia-${JULIA_VERSION}-linux-x86_64.tar.gz
269 RUN ln -fs /opt/julia-*/bin/julia /usr/local/bin/julia
270
271 # Show Julia where conda libraries are \
272 RUN mkdir /etc/julia && \
273     echo "push!(Libdl.DL_LOAD_PATH, \"\$CONDA_DIR/lib\")" >> /etc/julia
/juliarc.jl && \
274     # Create JULIA_PKGDIR \
275     mkdir $JULIA_PKGDIR && \
276     chown $NB_USER $JULIA_PKGDIR && \
277     fix-permissions $JULIA_PKGDIR
278
279 USER $NB_UID
280
281 # R packages including IRKernel which gets installed globally.
282 RUN conda install --quiet --yes \
283     'r-base=3.6.1' \
284     'r-caret=6.0*' \
285     'r-crayon=1.3*' \
286     'r-devtools=2.1*' \
287     'r-forecast=8.7*' \
288     'r-hexbin=1.27*' \
289     'r-htmltools=0.3*' \
290     'r-htmlwidgets=1.3*' \
291     'r-irkernel=1.0*' \
292     'r-nycflights13=1.0*' \
293     'r-plyr=1.8*' \
294     'r-randomforest=4.6*' \
295     'r-rcurl=1.95*' \
296     'r-reshape2=1.4*' \
297     'r-rmarkdown=1.14*' \
298     'r-rsqlite=2.1*' \
299     'r-shiny=1.3*' \
300     'r-sparklyr=1.0*' \
301     'r-tidyverse=1.2*' \
302     'rpy2=2.9*' \
303     && conda clean --all -f -y && \
304     fix-permissions $CONDA_DIR && \
305     fix-permissions /home/$NB_USER
306
307 # Add Julia packages. Only add HDF5 if this is not a test-only build since
308 # it takes roughly half the entire build time of all of the images on
Travis
309 # to add this one package and often causes Travis to timeout.
310 #
311 # Install IJulia as jovyan and then move the kernelspec out
312 # to the system share location. Avoids problems with runtime UID change not
313 # taking effect properly on the .local folder in the jovyan home dir.

```



```
314 RUN julia -e 'import Pkg; Pkg.update()' && \  
315 (test $TEST_ONLY_BUILD || julia -e 'import Pkg; Pkg.add("HDF5")') && \  
316 julia -e "using Pkg; pkg\"add IJulia\"; pkg\"precompile\"" && \  
317 # move kernelspec out of home \  
318 mv $HOME/.local/share/jupyter/kernels/julia* $CONDA_DIR/share/jupyter  
/kernels/ && \  
319 chmod -R go+rx $CONDA_DIR/share/jupyter && \  
320 rm -rf $HOME/.local && \  
321 fix-permissions $JULIA_PKGDIR $CONDA_DIR/share/jupyter  
322  
323 RUN julia -e 'import Pkg; ENV["PYTHON"]=""; Pkg.REPLMode.pkgstr("add  
PyCall; precompile"); import PyCall'  
324 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add Conda; precompile");  
import Conda; Conda.add("pyyaml")'  
325 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add Revise; precompile");  
import Revise'  
326 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add SVR; precompile");  
import SVR'  
327  
328 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add FileIO; precompile");  
import FileIO'  
329 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add JLD2; precompile");  
import JLD2'  
330 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add WebIO; precompile");  
import WebIO'  
331 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add Plotly; precompile");  
import Plotly'  
332  
333 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("add ODBC; precompile");  
import ODBC'  
334  
335 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("develop Mads; precompile");  
import Mads'  
336 RUN julia -e 'import Pkg; Pkg.REPLMode.pkgstr("develop NMFk; precompile");  
import NMFk'  
337  
338 EXPOSE 8888  
339  
340 # Configure container startup  
341 ENTRYPOINT ["tini", "-g", "--"]  
342 CMD ["start-notebook.sh"]  
343  
344  
345 # Fix permissions on /etc/jupyter as root  
346 USER root  
347 RUN fix-permissions /etc/jupyter/  
348  
349 # Switch back to jovyan to avoid accidental container runs as root  
350 USER $NB_UID  
351  
352  
353  
354  
355
```

F. Kodeudskrift: Docker-Compose YAML

```
1 version: '3.7'
2 services:
3   mssql:
4     build: ./mssql
5     image: "local/mssql"
6     container_name: mssql
7     ports:
8       - "1433:1433"
9     volumes:
10      - "/data/mssql:/var/opt/mssql"
11      - "/data/downloads:/downloads"
12      - "/data/scripts:/scripts"
13     environment:
14       - ACCEPT_EULA=yes
15       - SA_PASSWORD=*****
16 #   jupyter-notebook:
17 #     image: jupyter/datascience-notebook
18 #     container_name: jupyter-notebook
19 #     ports:
20 #       - "8888:8888"
21 #     volumes:
22 #       - "/data/jupyter:/home/jovyan/work"
23 #     environment:
24 #       - JUPYTER_ENABLE_LAB=yes
25 #       - RESTARTABLE=yes
26   madsen:
27     build: ./Madsen
28     image: local/madsen
29     container_name: madsen
30     ports:
31       - "8080:80"
32   redis:
33     image: "redis:alpine"
34     container_name: redis
35     ports:
36       - "6379:6379"
37     volumes:
38       - "/data/redis:/data"
39
```

G. Kodeudskrift - SQL til dataudtræk

```
1 SET ANSI_NULLS ON
2 GO
3 SET QUOTED_IDENTIFIER ON
4 GO
5
6
7 CREATE VIEW [dbo].[JupiterSpeciesAmount] AS
8 SELECT o.oplandid AS 'catchmentid'
9         ,o.navn AS 'catchment'
10        ,b.BOREHOLENO AS 'Wellno'
11        ,ROUND(b.XUTM32EUREF89,1) AS 'X'
12        ,ROUND(b.YUTM32EUREF89,1) AS 'Y'
13        ,ROUND(ISNULL(f.Z,b.ELEVATION - b.DRILLDEPTH),1) AS 'Z'
14        ,s.SAMPLEDATE AS 'Date'
15        ,stof.CODE AS 'SpeciesNo'
16        ,stof.LONGTEXT AS 'Species'
17        ,CASE WHEN a.ATTRIBUTE IS NOT NULL THEN 0
18              ELSE a.AMOUNT END AS 'Amount'
19        ,enhed.LONGTEXT AS 'Unit'
20        ,b.GEOM AS 'geom'
21 FROM e_opland o
22     INNER JOIN [JupiterXL].[dbo].[BOREHOLE] b ON b.GEOM.STIntersects(o.geom)=1
23     INNER JOIN [JupiterXL].[dbo].[GRWCHEMSAMPLE] s ON s.BOREHOLEID =
24     b.BOREHOLEID
25     INNER JOIN (
26         SELECT b1.BOREHOLEID
27               ,intakeid
28               ,b1.ELEVATION - min([TOP]) - (max(BOTTOM)-
29 min([TOP])/2) AS 'Z'
30         FROM [JupiterXL].[dbo].[BOREHOLE] b1
31         INNER JOIN [JupiterXL].[dbo].[SCREEN] sc1 ON
32 sc1.BOREHOLEID = b1.BOREHOLEID
33         GROUP BY b1.BOREHOLEID, ELEVATION, sc1.INTAKEID) f ON
34 f.BOREHOLEID = s.BOREHOLEID
35         AND f.INTAKEID = s.INTAKEID
36     INNER JOIN [JupiterXL].[dbo].[GRWCHEMANALYSIS] a ON a.SAMPLEID =
37 s.SAMPLEID
38     INNER JOIN [JupiterXL].[dbo].[CODE] enhed ON enhed.CODE = a.UNIT
39     AND enhed.CODETYPE = 752
40     INNER JOIN [JupiterXL].[dbo].[CODE] stof ON stof.CODE = a.COMPOUNDNO
41     AND stof.CODETYPE = 221
42 GO
43
```

```

1 SET ANSI_NULLS ON
2 GO
3 SET QUOTED_IDENTIFIER ON
4 GO
5
6
7 CREATE VIEW [dbo].[GeoGISSpeciesAmount] AS
8 SELECT o.oplandid AS 'catchmentid'
9         ,o.navn AS 'catchment'
10        ,ISNULL(PublicNo, PointNo) AS 'WellNo'
11        ,i.IntakeNo
12        ,ROUND(X1,1) AS 'X'
13        ,ROUND(Y1,1) AS 'Y'
14        ,ROUND(i.Z,1) AS 'Z'
15        ,s.SampleDate AS 'Date'
16        ,sp.ParameterId AS 'SpeciesNo'
17        ,sp.Parameter AS 'Species'
18        ,CASE WHEN a.Attribute = '<' THEN 0
19              WHEN a.Attribute IS NULL THEN a.[Value]
20              ELSE NULL END AS 'Amount'
21        ,u.Unit
22 FROM e_opland o
23     INNER JOIN [GeoGIS2020].[dbo].[Points] p ON
24     p.Geom.STIntersects(geometry::STPointFromText(('POINT ( ' + ltrim(str(X1)) +
25     ' ' + ltrim(str(Y1)) + ' ')),25832))=1
26     AND X1 > 0 AND Y1 > 0
27     INNER JOIN (
28         SELECT TOP (1000) [PointId]
29         ,[IntakeNo]
30         ,ReferenceLevel - min([Depth1]) - ((max([Depth2]) -
31         min([Depth1]))/2) AS 'Z'
32         FROM [GeoGIS2020].[dbo].[Intakes]
33         GROUP BY PointId, IntakeNo, ReferenceLevel
34         HAVING min(Depth1) IS NOT NULL AND max(Depth2) IS NOT NULL
35         AND ReferenceLevel IS NOT NULL) i ON i.PointId =
36     p.PointId
37     INNER JOIN [GeoGIS2020].[dbo].[AnalysisSamples] s ON s.PointId =
38     p.PointId
39     AND s.IntakeNo = i.IntakeNo
40     INNER JOIN [GeoGIS2020].[dbo].[AnalysisValues] a ON a.SampleId =
41     s.SampleId
42     INNER JOIN [GeoGIS2020].[dbo].[AnalysisParameters] sp ON
43     sp.ParameterId = a.ParameterId
44     INNER JOIN [GeoGIS2020].[dbo].[AnalysisUnits] u ON u.UnitId =
45     a.UnitId
46
47 GO

```

```
1 SET ANSI_NULLS ON
2 GO
3 SET QUOTED_IDENTIFIER ON
4 GO
5
6
7 CREATE VIEW [dbo].[CatchmentSpeciesAmount] AS
8
9 WITH species AS (
10 SELECT catchmentid
11        ,[Wellno]
12        ,[X]
13        ,[Y]
14        ,[Z]
15        ,YEAR([Date]) AS 'Year'
16        ,[SpeciesNo]
17        ,[Species]
18        ,MAX([Amount]) AS 'MaxAmount'
19        ,[Unit]
20 FROM [Informatik].[dbo].[JupiterSpeciesAmount]
21 GROUP BY catchmentid, [Wellno], [X], [Y], [Z], YEAR([Date]), [SpeciesNo],
[Species], [Unit]
22
23
24 UNION ALL
25
26 SELECT catchmentid
27        ,[Wellno]
28        ,[X]
29        ,[Y]
30        ,[Z]
31        ,YEAR([Date]) AS 'Year'
32        ,[SpeciesNo]
33        ,[Species]
34        ,MAX([Amount]) AS 'MaxAmount'
35        ,[Unit]
36 FROM [Informatik].[dbo].[GeoGISSpeciesAmount]
37 GROUP BY catchmentid, [Wellno], [X], [Y], [Z], YEAR([Date]), [SpeciesNo],
[Species], [Unit]
38 )
39
40 SELECT DISTINCT *
41 FROM species
42
43 GO
44
```

H. Notebooks og script - eksempel for test 3

nbGetdata

December 14, 2019

```
[1]: # TEST 3
      # 5 chlorerede forbindelser - udvalgte borer - alle år
      #
      using Pkg
      Pkg.add("DataFrames")
      Pkg.add("CSV")
      using ODBC
      using DataFrames
      using CSV
      using FileIO
      using Dates
      using NMFk
```

```
Updating registry at `~/opt/julia/registries/General`
Updating git-repo
`https://github.com/JuliaRegistries/General.git`
Resolving package versions...

Warning: julia version requirement for package Mads not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225
Warning: julia version requirement for package NMFk not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225
```

```
Updating `~/opt/julia/environments/v1.2/Project.toml`
[no changes]
Updating `~/opt/julia/environments/v1.2/Manifest.toml`
[no changes]
Resolving package versions...

Warning: julia version requirement for package Mads not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225
Warning: julia version requirement for package NMFk not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225
```

HTML{String}("<script>\n// Immediately-invoked-function-expression to avoid global variables.\n")

```

Updating `~/opt/julia/environments/v1.2/Project.toml`
[no changes]
Updating `~/opt/julia/environments/v1.2/Manifest.toml`
[no changes]

Info: Loading DataFrames support into Gadfly.jl
@ Gadfly /opt/julia/packages/Gadfly/09PWZ/src/mapping.jl:228

```

```

[30]: omr = "Birkerød"
stofgrp = "chlorerede"
slist = "2612, 2616, 2617, 2618, 2621"
wlist = "'193. 975', '193. 1279', '193. 1478', '193. 1571', '193. 1575', '193. 1611', '193. 1612'"
wlist = string(wlist, ", '193. 1762', '193. 1929', '193. 1930', '193. 1931', '193. 1932', '193. 1933', '193. 1934'")
println(slist)
println(wlist)
cd("/home/ak/work/NMFk/Birkerod/Test3")
pwd()

```

```

2612, 2616, 2617, 2618, 2621
'193. 975', '193. 1279', '193. 1478', '193. 1571', '193. 1575', '193. 1611',
'193. 1612', '193. 1762', '193. 1929', '193. 1930', '193. 1931', '193. 1932',
'193. 1933', '193. 1934'

```

```

[30]: "/home/ak/work/NMFk/Birkerod/Test3"

```

```

[31]: # ODBC connection
dsn = ODBC.DSN("Driver={ODBC Driver 17 for SQL Server};Server=mssql;
↳Database=Informatik;Uid=sa;
Pwd=***** ;")

```

```

[31]: ODBC.DSN(DRIVER=ODBC Driver 17 for SQL Server;SERVER=mssql;UID=sa;PWD=***** ;A
PP=julia;WSID=d19c580d8b97;DATABASE=Informatik;)

```

```

[32]: # Hent ID for Birkerød opland fra SQL Serveren
sql = "SELECT oplandid
      ,navn
      FROM e_opland
      WHERE navn like '$(omr)%"
catchment = ODBC.query(dsn, sql)
catchmentid = catchment.oplandid[1]

```

```

[32]: "F892B55B-1272-4EF6-A4DB-8E0888139559"

```

```

[5]: # Tæl hvor mange analyser der findes for hvert af disse stoffer i Birkerød
# medtag kun stoffer med over 100 analyser
sql = "SELECT [Species]

```

```

        ,[SpeciesNo]
        ,COUNT(*) AS 'Antal'
    FROM [Informatik].[dbo].[CatchmentSpeciesAmount]
    GROUP BY [catchmentid], [Species], [SpeciesNo]
    HAVING catchmentid = '$(catchmentid)'
        AND SpeciesNo IN ($(slist))
        AND COUNT(*) > 100"
    spcount = ODBC.query(dsn, sql)

```

[5]:

	Species	SpeciesNo	Antal
	String	Int32	Int32
1	Chloroform	2612	331
2	Tetrachlormethan	2616	331
3	Tetrachlorethylen	2617	331
4	Trichlorethylen	2618	337
5	1,1,1-trichlorethan	2621	331

[6]:

```

speciesnolist = join(spcount.SpeciesNo," ")
specieslist = join(spcount.Species," ")

```

[6]: "Chloroform, Tetrachlormethan, Tetrachlorethylen, Trichlorethylen,
1,1,1-trichlorethan"

[23]:

```

# Hent alle data for disse stoffer fra SQL serveren
# Anonymiser boringsnummeret
sql = "SELECT Year
        ,REPLACE(REPLACE([Wellno], '193.', 'W'),' ', '') AS 'Wellno'
        ,X
        ,Y
        ,[SpeciesNo]
        ,[Species]
        ,[MaxAmount]
    FROM [Informatik].[dbo].[CatchmentSpeciesAmount]
    WHERE catchmentid = '$(catchmentid)'
        AND SpeciesNo IN ($(slist))
        AND WellNo IN ($(wlist))"
df = ODBC.query(dsn, sql)
df = unique(df)

```

[23]:

	Year	Wellno	X	Y	SpeciesNo	Species	MaxAmount
	Int32	String	Float64	Float64	Int32	String	Float64
1	1990	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.06
2	1993	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.09
3	1994	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.13
4	1995	W975	7.15582e5	6.19436e6	2612	Chloroform	0.0
5	1995	W975	7.15582e5	6.19436e6	2616	Tetrachlormethan	0.0
6	1995	W975	7.15582e5	6.19436e6	2617	Tetrachlorethylen	0.0
7	1995	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.09
8	1995	W975	7.15582e5	6.19436e6	2621	1,1,1-trichlorethan	0.0
9	1996	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.09
10	2000	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.071
11	2001	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.0
12	2003	W975	7.15582e5	6.19436e6	2612	Chloroform	0.0
13	2003	W975	7.15582e5	6.19436e6	2616	Tetrachlormethan	0.0
14	2003	W975	7.15582e5	6.19436e6	2617	Tetrachlorethylen	0.0
15	2003	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.13
16	2003	W975	7.15582e5	6.19436e6	2621	1,1,1-trichlorethan	0.0
17	2008	W975	7.15582e5	6.19436e6	2612	Chloroform	0.0
18	2008	W975	7.15582e5	6.19436e6	2616	Tetrachlormethan	0.0
19	2008	W975	7.15582e5	6.19436e6	2617	Tetrachlorethylen	0.0
20	2008	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.15
21	2008	W975	7.15582e5	6.19436e6	2621	1,1,1-trichlorethan	0.0
22	2009	W975	7.15582e5	6.19436e6	2612	Chloroform	0.0
23	2009	W975	7.15582e5	6.19436e6	2616	Tetrachlormethan	0.0
24	2009	W975	7.15582e5	6.19436e6	2617	Tetrachlorethylen	0.0
25	2009	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.12
26	2009	W975	7.15582e5	6.19436e6	2621	1,1,1-trichlorethan	0.0
27	2010	W975	7.15582e5	6.19436e6	2612	Chloroform	0.0
28	2010	W975	7.15582e5	6.19436e6	2616	Tetrachlormethan	0.0
29	2010	W975	7.15582e5	6.19436e6	2617	Tetrachlorethylen	0.0
30	2010	W975	7.15582e5	6.19436e6	2618	Trichlorethylen	0.0
...	...	...	...	...	...	...	...

```
[33]: filename = "data.csv"
      CSV.write(filename, df, delim=';')
      pwd()
```

```
[33]: "/home/ak/work/NMFk/Birkerod/Test3"
```

```
[25]: years = unique(sort((df.Year)))
      wells = unique(sort((df.Wellno)))
      species = unique(sort((df.Species)))
      T = Array{Float32}(undef, length(wells), length(species), length(years))
      T .= NaN
      for i in 1:size(df, 1)
          iy = indexin([df.Year[i]], years)[1]
```

```
iw = indexin([df.Wellno[i]], wells)[1]
is = indexin([df.Species[i]], species)[1]
T[iw, is, iy] = df.MaxAmount[i]
end
```

```
[26]: println(size(T))
      NMFk.checkarrayentries(T)
```

(14, 5, 22)

```
Info: Bad indices in dimension 1: Int64[]
@ NMFk /opt/julia/dev/NMFk/src/NMFkChecks.jl:62
Info: Bad indices in dimension 2: Int64[]
@ NMFk /opt/julia/dev/NMFk/src/NMFkChecks.jl:62
Info: Bad indices in dimension 3: Int64[]
@ NMFk /opt/julia/dev/NMFk/src/NMFkChecks.jl:62
```

[26]: true

```

1 import Pkg
2 Pkg.develop("NMFk")
3 Pkg.add("CSV")
4 Pkg.add("DataFrames")
5 using CSV
6 using DataFrames
7 using Dates
8 using NMFk
9
10 #####
11 # TEST 3
12 # 5 chl. udvalgte boringer
13 # scale=true
14 #####
15
16
17 filename = "./data.csv"
18 df = CSV.read(filename, delim=';', header=true)
19
20 years = unique(sort((df.Year)))
21 wells = unique(sort((df.Wellno)))
22 species = unique(sort((df.Species)))
23 T = Array{Float32}(undef, length(wells), length(species), length(years))
24 T .= NaN
25 for i in 1:size(df, 1)
26     iy = indexin([df.Year[i]], years)[1]
27     iw = indexin([df.Wellno[i]], wells)[1]
28     is = indexin([df.Species[i]], species)[1]
29     T[iw, is, iy] = df.MaxAmount[i]
30 end
31
32 start = Dates.now()
33 We, He, fitquality, robustness, aic, kopt = NMFk.execute(T, 2:10;
    quiet=false, scale=false)
34 slut = Dates.now()
35 duration = Dates.canonicalize(Dates.CompoundPeriod(Dates.DateTime(slut) -
    Dates.DateTime(start)))
36
37 Wopt = We[kopt]
38 Hopt = He[kopt]
39 fit = fitquality[kopt]
40 sil = robustness[kopt]
41 aicopt = aic[kopt]
42
43 Te = NMFk.mixmatchcompute(T, Wopt, Hopt)
44 NMFk.setbadmixerelements!(T, Wopt, Hopt)
45 NMFk.normnan(T.-Te)
46
47 for i in 1:size(Wopt)[2]
48     CSV.write("export/Source$(i).csv", DataFrame(Wopt[:,i,:]), delim=";")
49 end
50 wells = unique(sort(df[:,2:4]))
51 warr = convert(Matrix, wells)
52 species = unique(sort(df[:,5:6]))
53 sarr = convert(Matrix, species)
54 CSV.write("export/Wells.csv", wells, delim=";")

```

```

55 CSV.write("export/Species.csv", species, delim=";")
56 CSV.write("export/He.csv", DataFrame(Hopt), delim=";")
57
58 open("README.md", "w") do f
59     write(f, "# NMFk Test 3\n")
60     write(f, "Dato: $(Dates.today())\n")
61     write(f, "Dir: $(pwd())\n")
62     write(f, "\n")
63     write(f, "## TEST:")
64     write(f, "NMFk.execute(T, 2:10; quiet=false, scale=false)\n")
65     write(f, "\n")
66     write(f, "## Testresultat:\n")
67     write(f, "Tid: $(duration)\n")
68     write(f, "Optimal solution: $(kopt)\n")
69     write(f, "Fit: $(fit)\n")
70     write(f, "Silhouette: $(sil)\n")
71     write(f, "AIC: $(aicopt)\n")
72     write(f, "\n\n")
73     write(f, "## Dataset:\n")
74     write(f, "Size T: $(size(T))\n")
75     write(f, "Size wells: $(size(wells)[1])\n")
76     write(f, "Size years: $(size(years)[1])\n")
77     write(f, "Size species: $(size(species)[1])\n")
78     write(f, "\n\n")
79     write(f, "### Years:\n")
80     write(f, "[$(join(years, ", " ))]\n")
81     write(f, "\n")
82     write(f, "### Wells: \n")
83     write(f, "[$(join(names(wells), ";"))]\n")
84     for i in 1:size(warr)[1]
85         write(f, "[$(join(warr[i,:], "; " ))]\n")
86     end
87     write(f, "\n")
88     write(f, "### Species:\n")
89     write(f, "[$(join(names(species), ";"))]\n")
90     for i in 1:size(sarr)[1]
91         write(f, "[$(join(sarr[i,:], "; " ))]\n")
92     end
93     write(f, "\n\n\n")
94 end
95
96 println("kopt: $(kopt)")

```

nbInspect

December 14, 2019

```
[4]: import Pkg
      Pkg.add("CSV")
      Pkg.add("DataFrames")
      Pkg.add("Plots")
      Pkg.add("StatsPlots")
      import CSV
      import DataFrames
      import Plots
      import StatsPlots
      import FileIO
      using Statistics
      import Cairo
      import NMFk
      import Mads
      import JLD2
      gr()
```

Resolving package versions...

```
Warning: julia version requirement for package Mads not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/stdlib/v1.2/Pkg/src/Operations.jl:225
Warning: julia version requirement for package NMFk not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/stdlib/v1.2/Pkg/src/Operations.jl:225
```

Updating `/opt/julia/environments/v1.2/Project.toml`

[no changes]

Updating `/opt/julia/environments/v1.2/Manifest.toml`

[no changes]

Resolving package versions...

```
Warning: julia version requirement for package Mads not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/stdlib/v1.2/Pkg/src/Operations.jl:225
Warning: julia version requirement for package NMFk not satisfied
 @ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/stdlib/v1.2/Pkg/src/Operations.jl:225
```

Updating `/opt/julia/environments/v1.2/Project.toml`


```

[no changes]
Updating `/opt/julia/environments/v1.2/Manifest.toml`
[no changes]
Resolving package versions...

Warning: julia version requirement for package Mads not satisfied
@ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225
Warning: julia version requirement for package NMFk not satisfied
@ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225

Updating `/opt/julia/environments/v1.2/Project.toml`
[no changes]
Updating `/opt/julia/environments/v1.2/Manifest.toml`
[no changes]
Resolving package versions...

Warning: julia version requirement for package Mads not satisfied
@ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225
Warning: julia version requirement for package NMFk not satisfied
@ Pkg.Operations /buildworker/worker/package_linux64/build/usr/share/julia/std
lib/v1.2/Pkg/src/Operations.jl:225

Updating `/opt/julia/environments/v1.2/Project.toml`
[no changes]
Updating `/opt/julia/environments/v1.2/Manifest.toml`
[no changes]

```

```

UndefVarError: gr not defined

```

```

Stacktrace:

```

```

[1] top-level scope at In[4]:16

```

```

[5]: filename = "data.csv"
df = CSV.read(filename, delim=';', header=true)
println(df[1:5,:])
println("  ...")
pwd()

```

```

5×7 DataFrames.DataFrame
  Row  Year  Wellno  X          Y          SpeciesNo  Species
MaxAmount
      Int64  String  Float64

```

```
Float64      Int64      String
Float64
```

```
  1    1990    W975    7.15582e5    6.19436e6    2618    Trichlorethylen
0.06
  2    1993    W975    7.15582e5    6.19436e6    2618    Trichlorethylen
0.09
  3    1994    W975    7.15582e5    6.19436e6    2618    Trichlorethylen
0.13
  4    1995    W975    7.15582e5    6.19436e6    2612    Chloroform
0.0
  5    1995    W975    7.15582e5    6.19436e6    2616    Tetrachlormethan
0.0
...
```

```
[5]: "/home/ak/work/NMFk/Birkerod/Test3"
```

```
[6]: wells = unique(sort(df[:,2:4]))
species = unique(sort(df[:,5:6]))
years = unique(sort(df.Year))
T = Array{Float32}(undef, size(wells, 1), size(species, 1), length(years))
T .= NaN
for i in 1:size(df, 1)
    iw = indexin([df.Wellno[i]], wells[:,1])[1]
    is = indexin([df.Species[i]], species[:,2])[1]
    iy = indexin([df.Year[i]], years)[1]
    T[iw, is, iy] = df.MaxAmount[i]
end
```

```
[7]: NMFk.load(2:10, 10, 3);
```

```
Signals: 2 Fit:      620.7581 Silhouette: 0.9999999 AIC:      1105.926
Signals: 3 Fit:      68.9193 Silhouette:      1 AIC:      213.1042
Signals: 4 Fit:      4.830698 Silhouette: 0.9999994 AIC:     -997.5359
Signals: 5 Fit:      0.7245538 Silhouette: 0.9999998 AIC:     -1682.494
Signals: 6 Fit: 7.269577e-06 Silhouette: 0.9999701 AIC:     -9009.638
Signals: 7 Fit: 7.84002e-06 Silhouette: -0.07106656 AIC:     -8331.437
Signals: 8 Fit: 8.210143e-06 Silhouette: -0.06335305 AIC:     -7673.562
Signals: 9 Fit: 2.263081e-05 Silhouette: -0.09931815 AIC:     -6346.928
Signals: 10 Fit: 8.330557e-06 Silhouette: -0.09913092 AIC:     -6411.501
```

```
[8]: We, He, fit, sil, aic = NMFk.load(4)
Te = NMFk.mixmatchcompute(T, We, He)
NMFk.setbadmixerelements!(T, We, He)
NMFk.normnan(T.-Te)
```

```
Signals: 4 Fit:      4.830698 Silhouette: 0.9999994 AIC:     -997.5359
```

[illegible]

[illegible]

[illegible]

```

Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.09, 0.0]
Xe[w, :, t] = Float32[1.7941458e-5, 0.00042286524, 0.05504119, 0.011934755,
0.09000749]
W[w, :, t] = Float32[0.99970174, 8.750275e-8, 0.00023680199, 6.13901e-5]
X[w, :, t] = Float32[0.04, 0.0, 0.68, 7.4, 0.76]
Xe[w, :, t] = Float32[0.7736279, 0.03491329, 0.07115365, 0.017423447, 7.3999104]
W[w, :, t] = Float32[0.9710538, 0.0043523563, 0.019473273, 0.005120588]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]

```

```

X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, 0.09, NaN]
Xe[w, :, t] = Float32[59.087868, 2.2675185, 1.3695996, 0.3938739, 0.090000264]
W[w, :, t] = Float32[0.334744, 0.33243304, 0.00023366374, 0.33258927]
X[w, :, t] = Float32[0.0, 0.0, 0.07, 9.0, 0.07]
Xe[w, :, t] = Float32[0.07025075, 8.575573e-5, 0.05533748, 0.012926619,
8.999997]
W[w, :, t] = Float32[0.97590894, 0.00039528107, 0.023684055, 1.1742543e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.0, 0.0]
Xe[w, :, t] = Float32[1.791646e-5, 0.00042295502, 0.055054203, 0.011929136,
0.00016505449]
W[w, :, t] = Float32[0.99993813, 8.7485546e-8, 3.7540553e-7, 6.14034e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.0, 0.0]

```

```

Xe[w, :, t] = Float32[1.791646e-5, 0.00042295502, 0.055054203, 0.011929136,
0.00016505449]
W[w, :, t] = Float32[0.99993813, 8.7485546e-8, 3.7540553e-7, 6.14034e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, 0.071, NaN]
Xe[w, :, t] = Float32[59.05092, 2.2663896, 1.3687779, 0.3936357, 0.07100033]
W[w, :, t] = Float32[0.33516747, 0.33222514, 0.00018366586, 0.33242372]
X[w, :, t] = Float32[0.0, 0.0, 0.12, 7.5, 0.16]
Xe[w, :, t] = Float32[0.16132441, 4.3296557e-5, 0.057609886, 0.013398767,
7.49999]
W[w, :, t] = Float32[0.97935027, 0.0009077532, 0.01973668, 5.296232e-6]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.06, 0.32, 0.66]
Xe[w, :, t] = Float32[0.659677, 0.00014966747, 0.06989271, 0.016048836,
0.3200005]
W[w, :, t] = Float32[0.9954263, 0.003711978, 0.00084202667, 1.9721874e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]

```



```

Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, 0.0, NaN]
Xe[w, :, t] = Float32[7.8935223, 0.32371408, 0.23049815, 0.06308333,
0.0002682625]
W[w, :, t] = Float32[0.90810907, 0.044409096, 2.1674171e-7, 0.047481634]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 6.5, 0.0]
Xe[w, :, t] = Float32[1.9715977e-5, 0.00041397804, 0.054112542, 0.0123356795,
6.5000067]
W[w, :, t] = Float32[0.9828347, 8.873769e-8, 0.017105158, 6.007695e-5]
X[w, :, t] = Float32[0.83, 0.0, 0.0, 2.4, 0.34]
Xe[w, :, t] = Float32[0.33862513, 0.83032876, 0.055563856, 0.019514032,
2.400007]
W[w, :, t] = Float32[0.86995935, 0.0018860808, 0.006315165, 0.12183941]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.96, 0.14]
Xe[w, :, t] = Float32[0.13861078, 0.0004445726, 0.058041755, 0.012851021,
0.9600076]
W[w, :, t] = Float32[0.99662954, 0.000779944, 0.0025262637, 6.424235e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 1.1, 0.13]
Xe[w, :, t] = Float32[0.12861666, 0.00044306938, 0.057796005, 0.012797629,
1.1000077]
W[w, :, t] = Float32[0.99631757, 0.0007237072, 0.0028946837, 6.40455e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 230.0, 0.0]
Xe[w, :, t] = Float32[8.777633e-5, 0.00015812174, 0.021732751, 0.026315242,
230.0]
W[w, :, t] = Float32[0.39471677, 1.6535495e-7, 0.60526085, 2.2217211e-5]
X[w, :, t] = Float32[0.08, 0.05, 0.06, 8.1, 1.7]

```

```

Xe[w, :, t] = Float32[1.6994524, 0.08042833, 0.09156785, 0.02351587, 8.100007]
W[w, :, t] = Float32[0.9573264, 0.009560885, 0.021315569, 0.011797162]
X[w, :, t] = Float32[0.22, 0.0, 0.05, 20.0, 0.78]
Xe[w, :, t] = Float32[0.7794742, 0.22002195, 0.067947194, 0.01943671, 20.000002]
W[w, :, t] = Float32[0.91070485, 0.0043809228, 0.052631162, 0.032283053]
X[w, :, t] = Float32[0.2, 0.06, 0.06, 37.0, 1.4]
Xe[w, :, t] = Float32[1.3997792, 0.20039031, 0.0796392, 0.02423082, 37.000004]
W[w, :, t] = Float32[0.86535954, 0.007871796, 0.097367845, 0.029400833]
X[w, :, t] = Float32[0.08, 0.0, 0.1, 47.0, 0.05]
Xe[w, :, t] = Float32[0.051059134, 0.079480685, 0.04874978, 0.015694197,
46.999992]
W[w, :, t] = Float32[0.864369, 0.0002853873, 0.12368362, 0.011661997]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[2.1, 0.0, 0.0, 7.3, 0.67]
Xe[w, :, t] = Float32[0.66864187, 2.1002314, 0.0519218, 0.030027477, 7.3000064]
W[w, :, t] = Float32[0.6688958, 0.0037134802, 0.019209, 0.30818173]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.41, 0.0, 0.0, 81.0, 0.0]
Xe[w, :, t] = Float32[0.0017258946, 0.41018948, 0.040007886, 0.01963804,
81.00001]
W[w, :, t] = Float32[0.7266535, 3.7284796e-8, 0.21315679, 0.060189657]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 170.0, 0.0]

```

```

Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad

```



```

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad

```



```

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
Xe[w, :, t] = Float32[6.777709e-5, 0.00021217945, 0.030425392, 0.022562236,
170.00002]
W[w, :, t] = Float32[0.55260295, 1.3540394e-7, 0.4473667, 3.023434e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.035, 0.0, 0.0, 42.0, 0.5]
Xe[w, :, t] = Float32[0.49853677, 0.035375576, 0.059930235, 0.017880399,
42.000008]
W[w, :, t] = Float32[0.88148075, 0.0028043666, 0.11052583, 0.0051890323]

```



```

X[w, :, t] = Float32[0.0, 0.0, 0.0, 310.0, 0.0]
Xe[w, :, t] = Float32[0.00011888524, 0.000108394976, 0.010142479, 0.031319425,
310.0]
W[w, :, t] = Float32[0.18419857, 2.2977089e-7, 0.81578636, 1.4807323e-5]
X[w, :, t] = Float32[0.091, 0.055, 0.12, 12.0, 2.0]
Xe[w, :, t] = Float32[2.0006847, 0.09100627, 0.09771246, 0.025700679, 11.999998]
W[w, :, t] = Float32[0.9438171, 0.01125566, 0.031578653, 0.013348622]
X[w, :, t] = Float32[0.68, 0.054, 0.12, 22.0, 2.3]
Xe[w, :, t] = Float32[2.3006282, 0.6799635, 0.09821763, 0.03197328, 21.999996]
W[w, :, t] = Float32[0.829406, 0.012929697, 0.057893943, 0.09977037]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 78.0, 3.0]
Xe[w, :, t] = Float32[2.9972668, 0.0007173442, 0.11137135, 0.035445694,
78.00001]
W[w, :, t] = Float32[0.777775, 0.016865412, 0.20526229, 9.731526e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 90.0, 0.0]
Xe[w, :, t] = Float32[4.3450316e-5, 0.00030353433, 0.042015478, 0.017558364,
90.00001]
W[w, :, t] = Float32[0.763115, 1.0818097e-7, 0.23684119, 4.3752643e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.13, 0.0]
Xe[w, :, t] = Float32[1.7952512e-5, 0.0004228097, 0.055035394, 0.011937257,
0.1300074]
W[w, :, t] = Float32[0.9995965, 8.751036e-8, 0.0003420645, 6.138189e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 5.2, 0.0]
Xe[w, :, t] = Float32[1.9355633e-5, 0.00041577674, 0.05430088, 0.0122543685,
5.200007]
W[w, :, t] = Float32[0.98625547, 8.848475e-8, 0.013684119, 6.034272e-5]
X[w, :, t] = Float32[1.8, 0.0, 0.0, 8.2, 0.68]
Xe[w, :, t] = Float32[0.6785957, 1.8002639, 0.05446729, 0.028219339, 8.2000065]
W[w, :, t] = Float32[0.71048087, 0.003776478, 0.02157761, 0.26416504]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 2.2, 0.43]
Xe[w, :, t] = Float32[0.4284555, 0.00048091495, 0.064401, 0.014730938,
2.2000084]
W[w, :, t] = Float32[0.9917308, 0.00241089, 0.005789404, 6.8879446e-5]
X[w, :, t] = Float32[0.5, 0.0, 0.0, 110.0, 0.89]
Xe[w, :, t] = Float32[0.88858616, 0.5002732, 0.055078365, 0.027544677,
110.000015]
W[w, :, t] = Float32[0.63213336, 0.004988237, 0.28947222, 0.07340621]
X[w, :, t] = Float32[0.2, 0.065, 0.061, 48.0, 0.11]
Xe[w, :, t] = Float32[0.11057087, 0.2002106, 0.04896093, 0.016902069, 48.000004]
W[w, :, t] = Float32[0.8436899, 0.0006174438, 0.12631513, 0.029377535]
X[w, :, t] = Float32[0.076, 0.029, 0.054, 130.0, 0.093]
Xe[w, :, t] = Float32[0.09341676, 0.07591773, 0.03770847, 0.021126645,
129.99998]
W[w, :, t] = Float32[0.64623344, 0.00052369956, 0.34210387, 0.011138957]
X[w, :, t] = Float32[0.0, 0.0, 0.033, 0.65, 1.3]
Xe[w, :, t] = Float32[1.2987188, 0.0003594243, 0.08426064, 0.020044087,
0.6500062]
W[w, :, t] = Float32[0.99093276, 0.007307844, 0.0017104419, 4.8970724e-5]

```

```

X[w, :, t] = Float32[0.0, 0.0, 0.0, 130.0, 0.25]
Xe[w, :, t] = Float32[0.24892189, 0.00028659616, 0.04183489, 0.021607816, 130.0]
W[w, :, t] = Float32[0.656455, 0.001400485, 0.34210393, 4.0614796e-5]
X[w, :, t] = Float32[0.033, 0.0, 0.0, 54.0, 0.0]
Xe[w, :, t] = Float32[0.00016751883, 0.033279344, 0.04696492, 0.01551896,
54.000004]
W[w, :, t] = Float32[0.8530126, 8.8265026e-8, 0.14210467, 0.004882612]
X[w, :, t] = Float32[0.079, 0.024, 0.065, 3.5, 3.8]
Xe[w, :, t] = Float32[3.7982407, 0.07953106, 0.13959259, 0.03627163, 3.50001]
W[w, :, t] = Float32[0.9577585, 0.021370739, 0.009210303, 0.011660479]
X[w, :, t] = Float32[0.71, 0.049, 0.14, 13.0, 3.0]
Xe[w, :, t] = Float32[3.0006433, 0.70988005, 0.115070276, 0.035954777,
12.999997]
W[w, :, t] = Float32[0.84476364, 0.016867978, 0.03420978, 0.1041586]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 76.0, 4.3]
Xe[w, :, t] = Float32[4.29655, 0.00090129813, 0.14097285, 0.043400094,
76.000015]
W[w, :, t] = Float32[0.77570325, 0.024176441, 0.19999912, 0.00012120036]
X[w, :, t] = Float32[0.12, 0.031, 0.15, 36.0, 0.068]
Xe[w, :, t] = Float32[0.070343316, 0.11928699, 0.05045337, 0.015381631,
35.999985]
W[w, :, t] = Float32[0.8873676, 0.00039298675, 0.094736315, 0.01750307]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.048, 18.0, 0.0]
Xe[w, :, t] = Float32[8.9218556e-5, 0.00013235163, 0.052450147, 0.0130536705,
18.0]
W[w, :, t] = Float32[0.9526126, 4.7031884e-7, 0.047368187, 1.8735207e-5]
X[w, :, t] = Float32[6.6, 0.0, 0.031, 8.2, 1.8]
Xe[w, :, t] = Float32[1.7993716, 6.5995846, 0.04053327, 0.06600801, 8.199996]
W[w, :, t] = Float32[4.8762795e-5, 0.009971228, 0.021574367, 0.96840566]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.28, 0.055, 0.056, 59.0, 0.13]
Xe[w, :, t] = Float32[0.1304282, 0.28016403, 0.047161743, 0.018227069,
59.000004]
W[w, :, t] = Float32[0.8029007, 0.00072730216, 0.15526234, 0.041109685]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 130.0, 0.0]
Xe[w, :, t] = Float32[5.536018e-5, 0.00025554321, 0.036220446, 0.020060282,
130.00002]
W[w, :, t] = Float32[0.65785927, 1.2041978e-7, 0.34210396, 3.6653986e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]

```

```

W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 220.0, 0.18]
Xe[w, :, t] = Float32[0.17921929, 0.00018330028, 0.027222866, 0.026803622,
220.0]
W[w, :, t] = Float32[0.4200212, 0.001008147, 0.57894516, 2.5497107e-5]
X[w, :, t] = Float32[0.059, 0.0, 0.0, 54.0, 0.041]
Xe[w, :, t] = Float32[0.039826553, 0.059287436, 0.047647133, 0.01593256,
54.000008]
W[w, :, t] = Float32[0.8489738, 0.0002226426, 0.14210466, 0.008698896]
X[w, :, t] = Float32[0.16, 0.094, 0.29, 17.0, 15.0]
Xe[w, :, t] = Float32[14.997669, 0.1607292, 0.38964403, 0.1072703, 17.000013]
W[w, :, t] = Float32[0.84732765, 0.08438771, 0.044736136, 0.02354851]
X[w, :, t] = Float32[2.5, 0.17, 0.5, 14.0, 15.0]
Xe[w, :, t] = Float32[15.003205, 2.499254, 0.37109327, 0.122134574, 13.999984]
W[w, :, t] = Float32[0.5120966, 0.084364384, 0.036839798, 0.36669922]
X[w, :, t] = Float32[0.25, 0.075, 0.09, 110.0, 6.2]
Xe[w, :, t] = Float32[6.198139, 0.25081334, 0.17690611, 0.05895499, 110.000015]
W[w, :, t] = Float32[0.63886875, 0.034870762, 0.28947225, 0.03678822]
X[w, :, t] = Float32[0.22, 0.083, 0.41, 60.0, 0.2]
Xe[w, :, t] = Float32[0.2085399, 0.21746743, 0.0492915, 0.018372659, 59.999954]
W[w, :, t] = Float32[0.8090284, 0.0011682941, 0.15789382, 0.031909518]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[2.6, 0.0, 0.0, 13.0, 0.78]
Xe[w, :, t] = Float32[0.7786703, 2.6001842, 0.049492672, 0.03427873, 13.000006]
W[w, :, t] = Float32[0.57992685, 0.00432095, 0.034208607, 0.3815436]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 51.0, 0.0]
Xe[w, :, t] = Float32[3.2206455e-5, 0.0003537195, 0.04766561, 0.015119015,
51.000008]
W[w, :, t] = Float32[0.86573875, 9.824117e-8, 0.13420999, 5.1171835e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 230.0, 0.0]
Xe[w, :, t] = Float32[8.777633e-5, 0.00015812174, 0.021732751, 0.026315242,
230.0]
W[w, :, t] = Float32[0.39471677, 1.6535495e-7, 0.60526085, 2.2217211e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 380.0, 0.0]

```

```

Xe[w, :, t] = Float32[0.000146285, 1.9530828e-5, 1.3717519e-6, 0.03569779,
379.99997]
W[w, :, t] = Float32[1.9059457e-6, 2.882015e-7, 0.9999961, 1.6686212e-6]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 63.0, 0.0]
Xe[w, :, t] = Float32[3.563293e-5, 0.00033797757, 0.04592711, 0.015869582,
63.000008]
W[w, :, t] = Float32[0.83416224, 1.0111991e-7, 0.16578881, 4.8844933e-5]
X[w, :, t] = Float32[0.1, 0.03, 0.35, 14.0, 2.3]
Xe[w, :, t] = Float32[2.3055584, 0.09800626, 0.10424375, 0.027766295, 13.999965]
W[w, :, t] = Float32[0.93581223, 0.012971009, 0.036841687, 0.014375055]
X[w, :, t] = Float32[0.48, 0.021, 0.11, 17.0, 1.6]
Xe[w, :, t] = Float32[1.6005377, 0.4797617, 0.084783345, 0.026022026, 16.999994]
W[w, :, t] = Float32[0.875874, 0.008994977, 0.044736244, 0.070394814]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 48.0, 2.3]
Xe[w, :, t] = Float32[2.2975636, 0.0006660684, 0.09993249, 0.029218325,
48.00001]
W[w, :, t] = Float32[0.860665, 0.012928244, 0.12631524, 9.150876e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 53.0, 0.0]
Xe[w, :, t] = Float32[3.2775646e-5, 0.000351079, 0.04737586, 0.0152441105,
53.000008]
W[w, :, t] = Float32[0.860476, 9.871073e-8, 0.13947313, 5.078155e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.48, 0.0, 0.0, 3.7, 0.13]
Xe[w, :, t] = Float32[0.12869343, 0.4803342, 0.053499356, 0.016042506, 3.700007]
W[w, :, t] = Float32[0.9190682, 0.00071295386, 0.009736443, 0.070482366]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 1.8, 0.11]
Xe[w, :, t] = Float32[0.10862966, 0.0004394731, 0.057243682, 0.012717125,
1.8000075]
W[w, :, t] = Float32[0.99458843, 0.00061124, 0.004736782, 6.356465e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.022, 44.0, 0.02]
Xe[w, :, t] = Float32[0.01929742, 0.00022954868, 0.049115513, 0.014800123, 44.0]
W[w, :, t] = Float32[0.88406956, 0.000108515764, 0.11578899, 3.2915006e-5]
X[w, :, t] = Float32[0.037, 0.0, 0.038, 30.0, 0.043]
Xe[w, :, t] = Float32[0.04260997, 0.03701801, 0.051369328, 0.014305563,
30.000002]
W[w, :, t] = Float32[0.915383, 0.00023885726, 0.07894699, 0.005431148]
X[w, :, t] = Float32[0.04, 0.026, 0.026, 58.0, 0.076]
Xe[w, :, t] = Float32[0.07556371, 0.040242996, 0.04802946, 0.016282672,
58.000008]
W[w, :, t] = Float32[0.8410406, 0.0004241728, 0.15263094, 0.0059042587]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]

```

```

X[w, :, t] = Float32[0.0, 0.0, 0.0, 150.0, 0.0]
Xe[w, :, t] = Float32[6.149549e-5, 0.00023321016, 0.033322923, 0.021311253,
150.00002]
W[w, :, t] = Float32[0.6052312, 1.2751548e-7, 0.39473534, 3.334861e-5]
X[w, :, t] = Float32[0.051, 0.0, 0.0, 37.0, 0.024]
Xe[w, :, t] = Float32[0.022786213, 0.051312856, 0.04979102, 0.0147119835,
37.000008]
W[w, :, t] = Float32[0.89497626, 0.00012696662, 0.09736798, 0.007528785]
X[w, :, t] = Float32[0.022, 0.0, 0.079, 5.6, 0.82]
Xe[w, :, t] = Float32[0.82003623, 0.021836016, 0.072568335, 0.017515423,
5.5999975]
W[w, :, t] = Float32[0.977448, 0.004613802, 0.014736686, 0.0032015445]
X[w, :, t] = Float32[0.75, 0.0, 0.1, 16.0, 1.7]
Xe[w, :, t] = Float32[1.7001631, 0.74969536, 0.084970005, 0.028312353,
15.999996]
W[w, :, t] = Float32[0.8383421, 0.009549279, 0.042104494, 0.11000414]
X[w, :, t] = Float32[0.059, 0.0, 0.026, 20.0, 1.4]
Xe[w, :, t] = Float32[1.3985704, 0.05932514, 0.083227865, 0.022253998,
20.000008]
W[w, :, t] = Float32[0.9307992, 0.007868305, 0.052631266, 0.008701223]
X[w, :, t] = Float32[0.027, 0.0, 0.15, 28.0, 0.038]
Xe[w, :, t] = Float32[0.040130865, 0.026106996, 0.05169235, 0.014094972,
27.999985]
W[w, :, t] = Float32[0.92226094, 0.0002251644, 0.07368382, 0.003830096]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.15, 0.0]
Xe[w, :, t] = Float32[1.795804e-5, 0.0004227819, 0.055032495, 0.011938508,
0.1500074]
W[w, :, t] = Float32[0.99954385, 8.751417e-8, 0.00039469588, 6.137778e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[1.2, 0.0, 0.0, 3.6, 0.2]
Xe[w, :, t] = Float32[0.19875869, 1.2002668, 0.049211323, 0.021095136,
3.6000056]
W[w, :, t] = Float32[0.813313, 0.0010904346, 0.009472803, 0.17612375]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.042, 0.0, 0.045, 32.0, 0.052]
Xe[w, :, t] = Float32[0.051769257, 0.041958343, 0.051245812, 0.014519344, 32.0]
W[w, :, t] = Float32[0.90934354, 0.0002902784, 0.08421013, 0.0061560594]
X[w, :, t] = Float32[0.061, 0.071, 0.037, 60.0, 0.084]
Xe[w, :, t] = Float32[0.08409579, 0.061437655, 0.047758967, 0.016596936,
60.000004]
W[w, :, t] = Float32[0.83261997, 0.00047168593, 0.15789406, 0.0090143]

```

```

X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 88.0, 0.0]
Xe[w, :, t] = Float32[4.28657e-5, 0.00030603478, 0.042305224, 0.017433267, 88.0]
W[w, :, t] = Float32[0.7683777, 1.0762789e-7, 0.23157804, 4.4122382e-5]
X[w, :, t] = Float32[0.044, 0.0, 0.0, 37.0, 0.0]
Xe[w, :, t] = Float32[0.00020786538, 0.044304583, 0.0493389, 0.014526596,
37.000004]
W[w, :, t] = Float32[0.89613146, 8.2156305e-8, 0.09736799, 0.0065004583]
X[w, :, t] = Float32[0.032, 0.0, 0.047, 2.9, 2.8]
Xe[w, :, t] = Float32[2.7982244, 0.03238605, 0.11750337, 0.029713718, 2.9000084]
W[w, :, t] = Float32[0.9718789, 0.015744774, 0.0076314164, 0.004744904]
X[w, :, t] = Float32[1.6, 0.055, 0.14, 16.0, 2.6]
Xe[w, :, t] = Float32[2.6010368, 1.5997599, 0.09834797, 0.039372493, 15.999995]
W[w, :, t] = Float32[0.70855856, 0.014598664, 0.0421039, 0.23473887]
X[w, :, t] = Float32[0.042, 0.025, 0.04, 18.0, 3.1]
Xe[w, :, t] = Float32[3.0981028, 0.04262189, 0.121997416, 0.032588515,
18.000011]
W[w, :, t] = Float32[0.9289538, 0.01743192, 0.0473681, 0.006246153]
X[w, :, t] = Float32[0.038, 0.0, 0.19, 26.0, 0.038]
Xe[w, :, t] = Float32[0.04102858, 0.036782008, 0.051915154, 0.014043998,
25.999977]
W[w, :, t] = Float32[0.92595285, 0.00022996991, 0.068420656, 0.0053965277]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.12, 0.0]
Xe[w, :, t] = Float32[1.794975e-5, 0.0004228236, 0.055036843, 0.011936631,
0.120007426]
W[w, :, t] = Float32[0.99962276, 8.7508454e-8, 0.00031574888, 6.1383944e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[1.1, 0.0, 0.0, 5.4, 0.19]
Xe[w, :, t] = Float32[0.1887549, 1.1002731, 0.04954197, 0.020503396, 5.400006]
W[w, :, t] = Float32[0.8233029, 0.001036471, 0.014209695, 0.16145091]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.73, 0.55, 0.058, 37.0, 0.33]
Xe[w, :, t] = Float32[0.33344182, 0.7333417, 0.051226217, 0.021023339,
37.000034]
W[w, :, t] = Float32[0.7931656, 0.0018591259, 0.09736759, 0.10760767]
X[w, :, t] = Float32[0.036, 0.0, 0.03, 31.0, 0.044]
Xe[w, :, t] = Float32[0.04343177, 0.03608203, 0.051250625, 0.014367215,
31.000004]
W[w, :, t] = Float32[0.9128841, 0.00024350188, 0.08157857, 0.0052938014]
X[w, :, t] = Float32[0.045, 0.0, 0.025, 74.0, 0.042]
Xe[w, :, t] = Float32[0.04144521, 0.045053322, 0.044902127, 0.017102292, 74.0]
W[w, :, t] = Float32[0.79842174, 0.00023205446, 0.19473603, 0.0066101756]

```

```

X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.11, 0.04, 0.032, 60.0, 0.31]
Xe[w, :, t] = Float32[0.30967355, 0.110306576, 0.052448887, 0.018313296, 60.0]
W[w, :, t] = Float32[0.82418144, 0.0017398674, 0.15789402, 0.016184693]
X[w, :, t] = Float32[0.034, 0.0, 0.0, 30.0, 0.0]
Xe[w, :, t] = Float32[0.00016502039, 0.034316998, 0.050433796, 0.014024315,
30.000004]
W[w, :, t] = Float32[0.916018, 8.3566164e-8, 0.07894701, 0.005034909]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 1.8, 2.5]
Xe[w, :, t] = Float32[2.4973214, 0.000763301, 0.111132465, 0.027570961,
1.8000144]
W[w, :, t] = Float32[0.98110557, 0.014052337, 0.004736735, 0.000105363484]
X[w, :, t] = Float32[1.6, 0.07, 0.076, 8.2, 2.4]
Xe[w, :, t] = Float32[2.399774, 1.6003625, 0.0949325, 0.03763709, 8.200004]
W[w, :, t] = Float32[0.73012835, 0.013466161, 0.021577692, 0.23482779]
X[w, :, t] = Float32[0.0, 0.0, 0.025, 11.0, 2.2]
Xe[w, :, t] = Float32[2.1980755, 0.00051747763, 0.103050165, 0.02628431,
11.00001]
W[w, :, t] = Float32[0.95861435, 0.012368483, 0.028947167, 6.9995294e-5]
X[w, :, t] = Float32[0.034, 0.0, 0.13, 23.0, 0.032]
Xe[w, :, t] = Float32[0.033669934, 0.03327647, 0.052212458, 0.013788073,
22.999989]
W[w, :, t] = Float32[0.93440324, 0.00018864885, 0.060525976, 0.004882153]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.0, 0.0]
Xe[w, :, t] = Float32[1.791646e-5, 0.00042295502, 0.055054203, 0.011929136,
0.00016505449]
W[w, :, t] = Float32[0.99993813, 8.7485546e-8, 3.7540553e-7, 6.14034e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.71, 0.0, 0.0, 6.2, 0.17]
Xe[w, :, t] = Float32[0.16871189, 0.7103112, 0.05216062, 0.01792457, 6.2000065]
W[w, :, t] = Float32[0.87852335, 0.0009327746, 0.01631521, 0.10422865]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.24, 0.59, 0.099, 42.0, 0.23]
Xe[w, :, t] = Float32[0.2346133, 0.24329424, 0.052276734, 0.017574659, 42.00003]
W[w, :, t] = Float32[0.85246056, 0.0013144318, 0.11052576, 0.03569926]
X[w, :, t] = Float32[0.029, 0.0, 0.071, 25.0, 0.045]
Xe[w, :, t] = Float32[0.045337364, 0.028757725, 0.05222282, 0.013956705,
24.999996]
W[w, :, t] = Float32[0.9297374, 0.00025440368, 0.06578914, 0.0042190505]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 53.0, 0.041]
Xe[w, :, t] = Float32[0.03981516, 0.00036018365, 0.048273318, 0.015491516,
53.000008]

```

```

W[w, :, t] = Float32[0.8602509, 0.00022395294, 0.13947313, 5.2022297e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 64.0, 0.26]
Xe[w, :, t] = Float32[0.25872645, 0.00037276399, 0.051618274, 0.017540775,
64.00001]
W[w, :, t] = Float32[0.83007056, 0.0014557453, 0.16842037, 5.3328655e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 31.0, 0.0]
Xe[w, :, t] = Float32[2.6553385e-5, 0.00038046515, 0.05056311, 0.013868076,
31.000008]
W[w, :, t] = Float32[0.9183662, 9.375625e-8, 0.081578605, 5.5124707e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.063, 2.5, 2.4]
Xe[w, :, t] = Float32[2.3987987, 0.00029348562, 0.10881211, 0.026999164,
2.500005]
W[w, :, t] = Float32[0.9798866, 0.013497963, 0.0065788114, 3.6658555e-5]
X[w, :, t] = Float32[0.93, 0.0, 0.097, 7.2, 2.2]
Xe[w, :, t] = Float32[2.1998224, 0.92978656, 0.0960462, 0.03202503, 7.199998]
W[w, :, t] = Float32[0.8322676, 0.012356664, 0.018946556, 0.1364292]
X[w, :, t] = Float32[0.0, 0.0, 0.031, 1.6, 0.27]
Xe[w, :, t] = Float32[0.26924035, 0.00025155832, 0.06089775, 0.013702009,
1.6000036]
W[w, :, t] = Float32[0.994239, 0.0015149969, 0.004210454, 3.560613e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.035, 0.0]
Xe[w, :, t] = Float32[1.7926255e-5, 0.00042294164, 0.05504916, 0.011931315,
0.03500789]
W[w, :, t] = Float32[0.99984646, 8.749228e-8, 9.206674e-5, 6.1401384e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.4, 0.0, 0.0, 4.9, 0.15]
Xe[w, :, t] = Float32[0.14867437, 0.40034387, 0.054429956, 0.015728122,
4.900007]
W[w, :, t] = Float32[0.9275337, 0.0008272482, 0.012894379, 0.058744684]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.12, 0.22, 0.051, 110.0, 0.15]
Xe[w, :, t] = Float32[0.15145367, 0.12120389, 0.041545615, 0.020527234,
110.00001]
W[w, :, t] = Float32[0.69189423, 0.00084924453, 0.2894725, 0.017784053]
X[w, :, t] = Float32[0.0, 0.0, 0.035, 26.0, 0.032]
Xe[w, :, t] = Float32[0.031531703, 0.00018162683, 0.051999986, 0.013749912,
26.000002]
W[w, :, t] = Float32[0.931376, 0.00017738402, 0.068420745, 2.5879179e-5]

```



```

X[w, :, t] = Float32[0.0, 0.037, 0.0, 39.0, 0.12]
Xe[w, :, t] = Float32[0.11896119, 0.00060485804, 0.05208543, 0.015109426,
39.00001]
W[w, :, t] = Float32[0.89661175, 0.0006693194, 0.10263116, 8.775564e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.04, 0.0, 0.0, 76.0, 0.18]
Xe[w, :, t] = Float32[0.1788106, 0.040274862, 0.047750503, 0.018050786,
76.00001]
W[w, :, t] = Float32[0.79308707, 0.0010051144, 0.19999918, 0.0059086625]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 25.0, 0.0]
Xe[w, :, t] = Float32[2.4870464e-5, 0.00038860118, 0.05143236, 0.013492794,
25.000006]
W[w, :, t] = Float32[0.9341544, 9.248131e-8, 0.065789185, 5.632705e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 1.2, 1.1]
Xe[w, :, t] = Float32[1.0980859, 0.00057202956, 0.07965272, 0.018832415,
1.2000104]
W[w, :, t] = Float32[0.99058264, 0.0061788815, 0.0031578215, 8.064761e-5]
X[w, :, t] = Float32[0.22, 0.0, 0.046, 5.1, 1.9]
Xe[w, :, t] = Float32[1.8987281, 0.22023992, 0.09535587, 0.02546505, 5.1000056]
W[w, :, t] = Float32[0.9435879, 0.010678951, 0.013420761, 0.03231238]
X[w, :, t] = Float32[0.0, 0.0, 0.028, 12.0, 0.81]
Xe[w, :, t] = Float32[0.80890703, 0.00031901684, 0.07156577, 0.017708395,
12.000007]
W[w, :, t] = Float32[0.96382535, 0.0045516705, 0.031578764, 4.4198096e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.1, 0.1, 0.1, 0.1, 0.1]
Xe[w, :, t] = Float32[0.10152095, 0.10020223, 0.05651436, 0.013207239,
0.09999936]
W[w, :, t] = Float32[0.9844655, 0.0005689173, 0.00026302726, 0.014702584]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.55, 0.0, 0.0, 5.5, 0.17]
Xe[w, :, t] = Float32[0.16868676, 0.5503296, 0.053568844, 0.016853258,
5.5000067]
W[w, :, t] = Float32[0.90383714, 0.0009363619, 0.014473219, 0.08075326]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.11, 0.13, 0.069, 88.0, 0.15]
Xe[w, :, t] = Float32[0.15123542, 0.11051487, 0.04481567, 0.019081049, 88.0]
W[w, :, t] = Float32[0.75135815, 0.0008482962, 0.23157796, 0.016215598]
X[w, :, t] = Float32[0.021, 0.0, 0.056, 27.0, 0.031]
Xe[w, :, t] = Float32[0.031011654, 0.020876216, 0.051674243, 0.013942129,

```

```

26.999998]
W[w, :, t] = Float32[0.92571115, 0.00017397413, 0.07105229, 0.0030625644]
X[w, :, t] = Float32[0.025, 0.0, 0.028, 43.0, 0.094]
Xe[w, :, t] = Float32[0.09339197, 0.025088033, 0.050728917, 0.01535789, 43.0]
W[w, :, t] = Float32[0.8826373, 0.000524866, 0.1131574, 0.0036804262]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.039, 0.0, 0.0, 66.0, 0.19]
Xe[w, :, t] = Float32[0.18877617, 0.039292283, 0.049432285, 0.017480899,
66.00001]
W[w, :, t] = Float32[0.8194908, 0.0010612273, 0.1736835, 0.005764471]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 26.0, 0.021]
Xe[w, :, t] = Float32[0.019748986, 0.00039428627, 0.051732417, 0.013678019,
26.000006]
W[w, :, t] = Float32[0.931411, 0.00011107803, 0.06842075, 5.7112633e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.022, 6.3, 1.1]
Xe[w, :, t] = Float32[1.0985967, 0.00040294242, 0.078926675, 0.019153528,
6.3000073]
W[w, :, t] = Float32[0.9771836, 0.0061817523, 0.016578816, 5.5827648e-5]
X[w, :, t] = Float32[0.18, 0.0, 0.053, 5.3, 1.2]
Xe[w, :, t] = Float32[1.1992639, 0.18008572, 0.07987437, 0.020870762, 5.3000026]
W[w, :, t] = Float32[0.95288694, 0.0067440197, 0.0139471125, 0.0264219]
X[w, :, t] = Float32[0.0, 0.0, 0.05, 13.0, 0.93]
Xe[w, :, t] = Float32[0.92934024, 0.00020385883, 0.07413892, 0.018519005,
13.000002]
W[w, :, t] = Float32[0.9605333, 0.0052293465, 0.03421032, 2.701027e-5]
X[w, :, t] = Float32[0.035, 0.0, 0.18, 23.0, 0.029]
Xe[w, :, t] = Float32[0.031799134, 0.033867523, 0.05216542, 0.013780236,
22.99998]
W[w, :, t] = Float32[0.93432707, 0.00017810814, 0.060525957, 0.0049688877]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.0, 0.0]
Xe[w, :, t] = Float32[1.791646e-5, 0.00042295502, 0.055054203, 0.011929136,
0.00016505449]
W[w, :, t] = Float32[0.99993813, 8.7485546e-8, 3.7540553e-7, 6.14034e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]

```

```

Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.14, 0.0]
Xe[w, :, t] = Float32[1.7955277e-5, 0.00042279574, 0.055033945, 0.011937882,
0.14000739]
W[w, :, t] = Float32[0.99957013, 8.751226e-8, 0.0003683802, 6.137983e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]

```

```

X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]

```

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138


```
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
```

```

Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

```

[illegible]


```

Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

```

```

Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.03, 0.0]
Xe[w, :, t] = Float32[1.7924873e-5, 0.00042294856, 0.05504988, 0.011931002,
0.030008]
W[w, :, t] = Float32[0.9998596, 8.749133e-8, 7.890918e-5, 6.140241e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.26, 0.0, 0.0, 4.3, 0.087]
Xe[w, :, t] = Float32[0.08568703, 0.26035085, 0.054239854, 0.014399959,
4.300007]
W[w, :, t] = Float32[0.9500059, 0.00047608285, 0.011315532, 0.038202524]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.02, 33.0, 0.036]
Xe[w, :, t] = Float32[0.035211366, 0.0002551436, 0.051068153, 0.01421114, 33.0]
W[w, :, t] = Float32[0.9129235, 0.00019807788, 0.08684173, 3.6648195e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.037, 14.0, 0.0]
Xe[w, :, t] = Float32[5.0102735e-5, 0.00017913306, 0.053028453, 0.012803513,
14.000002]
W[w, :, t] = Float32[0.9631322, 2.547147e-7, 0.036841914, 2.560557e-5]

```

```

X[w, :, t] = Float32[0.0, 0.0, 0.0, 20.0, 0.053]
Xe[w, :, t] = Float32[0.051713478, 0.00040665726, 0.05332282, 0.013501516,
20.000006]
W[w, :, t] = Float32[0.94701886, 0.00029094948, 0.05263134, 5.8859878e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 43.0, 0.047]
Xe[w, :, t] = Float32[0.045783017, 0.0003743912, 0.04985677, 0.014903175,
43.000004]
W[w, :, t] = Float32[0.88653094, 0.00025754754, 0.11315743, 5.4106924e-5]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 13.0, 0.0]
Xe[w, :, t] = Float32[2.1521284e-5, 0.0004050141, 0.053170852, 0.012742232,
13.000007]
W[w, :, t] = Float32[0.9657308, 9.0021885e-8, 0.034210358, 5.8752415e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.051, 0.0, 0.0, 2.0, 0.5]
Xe[w, :, t] = Float32[0.49842396, 0.05143705, 0.06559213, 0.015480688,
2.0000086]
W[w, :, t] = Float32[0.98438764, 0.0028034137, 0.0052630543, 0.0075459196]
X[w, :, t] = Float32[0.0, 0.0, 0.031, 8.5, 0.37]
Xe[w, :, t] = Float32[0.36920545, 0.00025455013, 0.062153287, 0.014755185,
8.500004]
W[w, :, t] = Float32[0.97551847, 0.002077488, 0.022368278, 3.579606e-5]
X[w, :, t] = Float32[0.033, 0.0, 0.095, 21.0, 0.026]
Xe[w, :, t] = Float32[0.026878325, 0.032564763, 0.052354828, 0.013616169,
20.999992]
W[w, :, t] = Float32[0.93980896, 0.00015045203, 0.055262852, 0.0047777384]
X[w, :, t] = Float32[0.0, 0.0, 0.0, 0.0, 0.0]
Xe[w, :, t] = Float32[1.791646e-5, 0.00042295502, 0.055054203, 0.011929136,
0.00016505449]
W[w, :, t] = Float32[0.99993813, 8.7485546e-8, 3.7540553e-7, 6.14034e-5]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]

```

```

W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[NaN, NaN, NaN, NaN, NaN]
Xe[w, :, t] = Float32[44.43594, 1.7044458, 1.0298858, 0.30509624, 95.00127]
W[w, :, t] = Float32[0.25, 0.25, 0.25, 0.25]
X[w, :, t] = Float32[0.0, 0.0, 0.11, 0.14, 0.0]
Xe[w, :, t] = Float32[0.0011846613, 4.4954762e-5, 0.055063356, 0.011942709,
0.13999146]
W[w, :, t] = Float32[0.99961907, 6.66134e-6, 0.0003683385, 5.93336e-6]

```

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: good

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

Info: bad

@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134

```

Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: bad
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:134
Info: good
@ NMFk /opt/julia/dev/NMFk/src/NMFkMixTensor.jl:138

```

[8]: 1158.2831f0

```

[9]: println("Tensor T: ", size(T))
println("Antal boringer: ", size(species, 1))
println("Antal stoffer: ", size(species, 1))
println("Antal år: ", length(years))
println()
print("Size We ", size(We), ": ")
println("$(size(We)[1]) boringers påvirkning af $(size(We)[2]) kilder i
↪$(size(We)[3]) år")
print("Size He: ", size(He), ": ")
println("$(size(He)[1]) kilders signatur for $(size(We)[2]) stoffer")

```

```

Tensor T: (14, 5, 22)
Antal boringer: 5
Antal stoffer: 5
Antal år: 22

```

```

Size We (14, 4, 22): 14 boringers påvirkning af 4 kilder i 22 år
Size He: (4, 5): 4 kilders signatur for 4 stoffer

```

```

[10]: println("Boringer:")
print(wells)

```

```

Boringer:
14×3 DataFrames.DataFrame
  Row  Wellno  X          Y
      String  Float64  Float64

 1   W1279    7.15184e5  6.19451e6
 2   W1478    7.15362e5  6.19374e6
 3   W1571    7.15631e5  6.19349e6
 4   W1575    7.15381e5  6.19364e6

```

5	W1611	7.15337e5	6.19359e6
6	W1612	7.15397e5	6.19359e6
7	W1762	7.13321e5	6.19337e6
8	W1929	7.15403e5	6.19369e6
9	W1930	7.15348e5	6.19359e6
10	W1931	7.15427e5	6.19369e6
11	W1932	7.1545e5	6.19367e6
12	W1933	7.15363e5	6.19368e6
13	W1934	7.15326e5	6.19364e6
14	W975	7.15582e5	6.19436e6

```
[11]: println("Stoffer:")
      print(species)
```

Stoffer:

5×2 DataFrames.DataFrame

Row	SpeciesNo	Species
	Int64	String
1	2612	Chloroform
2	2616	Tetrachlormethan
3	2617	Tetrachlorethylen
4	2618	Trichlorethylen
5	2621	1,1,1-trichlorethan

```
[12]: print("År: ")
      print(join(years, ", "))
```

År: 1990, 1993, 1994, 1995, 1996, 2000, 2001, 2002, 2003, 2004, 2005, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017

```
[13]: We
```

```
[13]: 14×4×22 Array{Float32,3}:
```

```
[:, :, 1] =
```

NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN

0.335533	0.332038	0.00015472	0.332274
----------	----------	------------	----------

[:, :, 2] =

NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN

0.334744	0.332433	0.000233664	0.332589
----------	----------	-------------	----------

[:, :, 3] =

0.96703	0.00640646	0.0139465	0.0126166
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN

0.334245	0.332654	0.000338924	0.332762
----------	----------	-------------	----------

...

[:, :, 20] =

NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN

NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
0.99986	8.74913e-8	7.89092e-5	6.14024e-5


```
[:, :, 21] =
```

NaN	NaN	NaN	NaN
0.950006	0.000476083	0.0113155	0.0382025
NaN	NaN	NaN	NaN
0.912924	0.000198078	0.0868417	3.66482e-5
0.963132	2.54715e-7	0.0368419	2.56056e-5
0.947019	0.000290949	0.0526313	5.88599e-5
NaN	NaN	NaN	NaN
0.886531	0.000257548	0.113157	5.41069e-5
0.965731	9.00219e-8	0.0342104	5.87524e-5
NaN	NaN	NaN	NaN
0.984388	0.00280341	0.00526305	0.00754592
0.975518	0.00207749	0.0223683	3.57961e-5
0.939809	0.000150452	0.0552629	0.00477774
0.999938	8.74855e-8	3.75406e-7	6.14034e-5


```
[:, :, 22] =
```

NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN
0.999619	6.66134e-6	0.000368338	5.93336e-6

```
[14]: println(size(We))
```

```
(14, 4, 22)
```

```
[15]: using DataFrames
smax = deepcopy(wells)
println(smax)
for si in 1:size(We,2)
    cname = "S$(si)"
    col = Vector{Float64}()
```



```

for wi in 1:size(wells,1)
    push!(col, maximum(filter(!isnan, We[wi,si,:])))
end
smax.new = col
rename!(smax, Symbol("new")=>Symbol(cname))
end
CSV.write("export/well_source_max.csv", smax, delim=";")
print(smax)

```

14×3 DataFrame

Row	Wellno	X	Y
	String	Float64	Float64
1	W1279	7.15184e5	6.19451e6
2	W1478	7.15362e5	6.19374e6
3	W1571	7.15631e5	6.19349e6
4	W1575	7.15381e5	6.19364e6
5	W1611	7.15337e5	6.19359e6
6	W1612	7.15397e5	6.19359e6
7	W1762	7.13321e5	6.19337e6
8	W1929	7.15403e5	6.19369e6
9	W1930	7.15348e5	6.19359e6
10	W1931	7.15427e5	6.19369e6
11	W1932	7.1545e5	6.19367e6
12	W1933	7.15363e5	6.19368e6
13	W1934	7.15326e5	6.19364e6
14	W975	7.15582e5	6.19436e6

14×7 DataFrame

Row	Wellno	X	Y	S1	S2	S3
	String	Float64	Float64			
1	W1279	7.15184e5	6.19451e6	0.986255	0.00640646	0.0473682
2	W1478	7.15362e5	6.19374e6	0.950006	0.00997123	0.0342086
3	W1571	7.15631e5	6.19349e6	0.99663	0.00241089	0.0057894
4	W1575	7.15381e5	6.19364e6	0.912924	0.00498824	0.289472
5	W1611	7.15337e5	6.19359e6	0.963132	0.000727302	0.213157
6	W1612	7.15397e5	6.19359e6	0.947019	0.000669319	0.605261

```

7      W1762      7.13321e5      6.19337e6      0.995426      0.00730784      0.00171044
4.89707e-5
8      W1929      7.15403e5      6.19369e6      0.999938      0.00280437      0.999996
0.0161847
9      W1930      7.15348e5      6.19359e6      0.999938      0.000222643      0.815786
0.0086989
10     W1931      7.15427e5      6.19369e6      0.990583      0.0843877      0.0447361
0.0235485
11     W1932      7.1545e5      6.19367e6      0.984388      0.0843644      0.0578939
0.366699
12     W1933      7.15363e5      6.19368e6      0.994239      0.0348708      0.289472
0.0367882
13     W1934      7.15326e5      6.19364e6      0.939809      0.00116829      0.236841
0.0319095
14     W975      7.15582e5      6.19436e6      0.999938      0.332654      0.000394696
0.332762

```

```
[16]: He[:,1]
```

```

[16]: 4-element Array{Float32,1}:
      6.3612833e-7
     177.71544
      9.502043e-5
      0.028219998

```

```

[17]: ssign = DataFrame(Species = species[:,2])
      for s in 1:size(He,1)
          sname = "S$(s)"
          ssign.new = He[s,:]
          rename!(ssign, Symbol("new")=>Symbol(sname))
      end
      CSV.write("export/source_signature.csv", ssign, delim=";")
      ssign

```

```

[17]:

```

	Species	S1	S2	S3	S4
	String	Float32	Float32	Float32	Float32
1	Chloroform	6.36128e-7	177.715	9.50204e-5	0.02822
2	Tetrachlormethan	4.49907e-6	0.00290416	8.15858e-6	6.81487
3	Tetrachlorethylen	0.0550573	4.06448	9.54213e-8	2.78071e-6
4	Trichlorethylen	0.0119263	1.1169	0.0356975	0.0558655
5	1,1,1-trichlorethan	2.22941e-5	0.00186761	380.001	0.00174207

```

[18]: using Plots
      gr()

      plotarr = Any[]
      for s in 1:size(He,1)
          p = pie(labels, He[s,:], title="Source $(s)")

```

```

push!(plotarr, p)
end
plot(plotarr...)
png("export/soruce_signature.phg")

```

UndefVarError: labels not defined

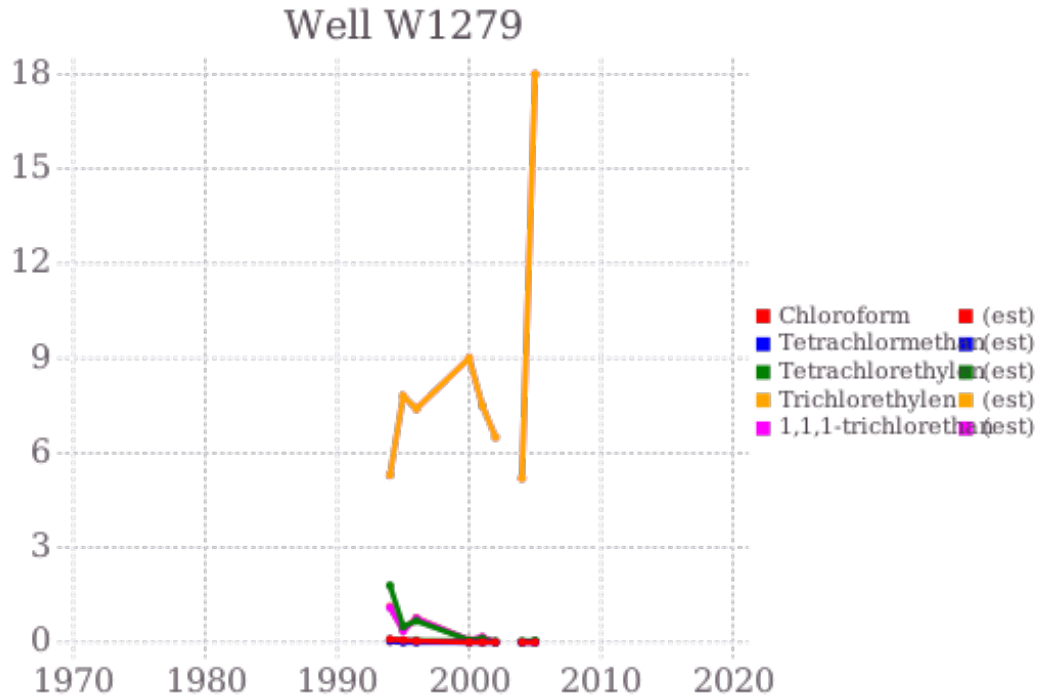
Stacktrace:

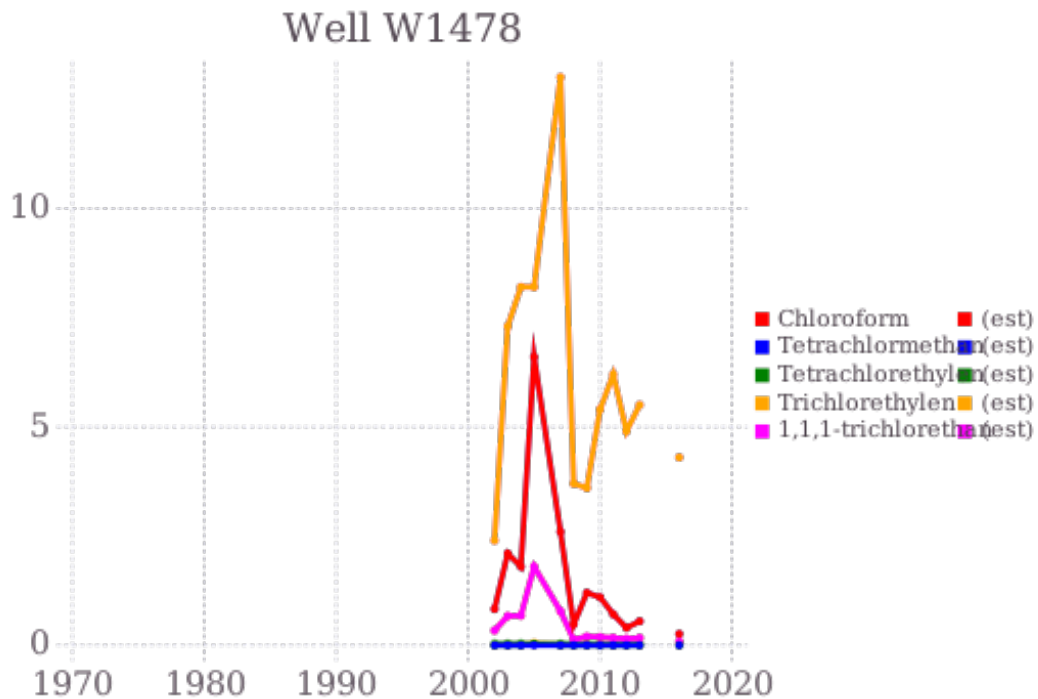
[1] top-level scope at ./In[18]:6

```

[19]: for i=1:size(wells, 1)
    gm = Mads.plotseries(permutedims(Te[i,:,:]); title="Well $(wells[i,1])",
    xmin=1970, xmax=2020, xaxis=years, ymin=0, plotdots=true, names=["(est)" for
    s in species[:,2]], quiet=true, code=true, opacity=0.5)
    Mads.plotseries(permutedims(T[i,:,:]); xmin=1970, xmax=2020, xaxis=years,
    ymin=0, plotdots=true, names=["$s" for s in species[:,2]], gm=gm)
end

```

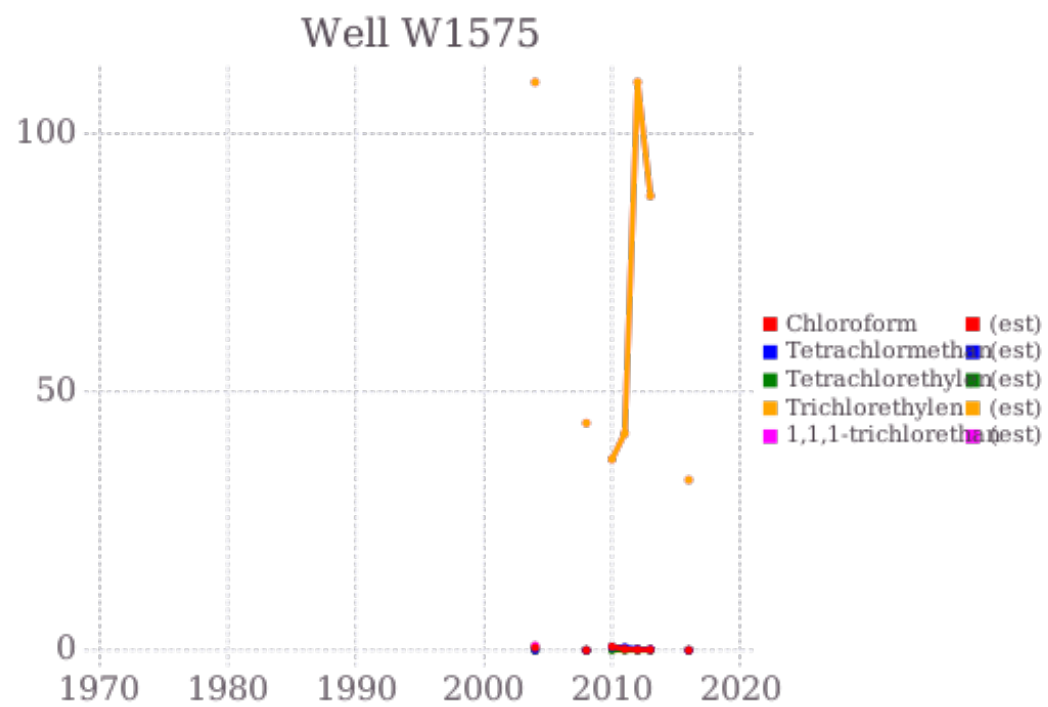
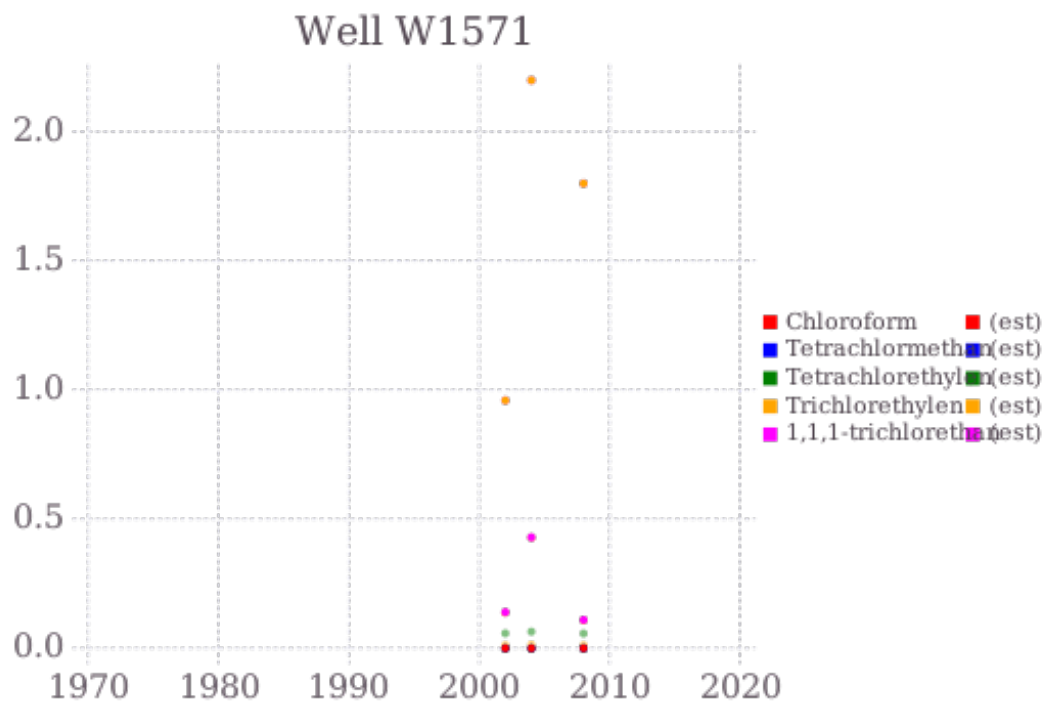


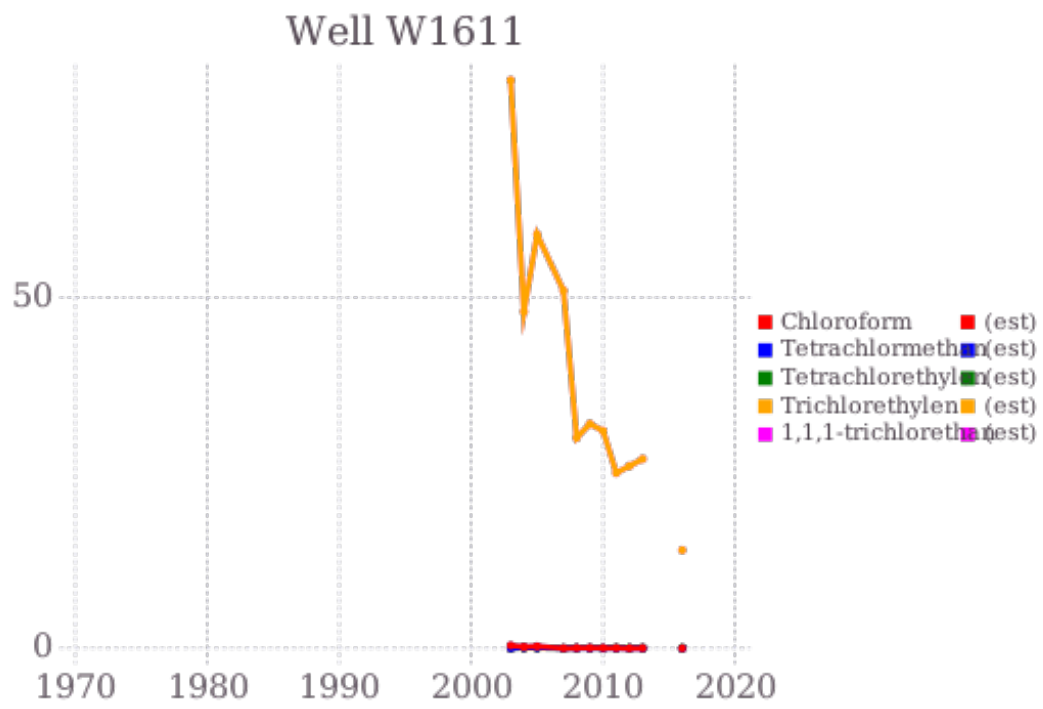


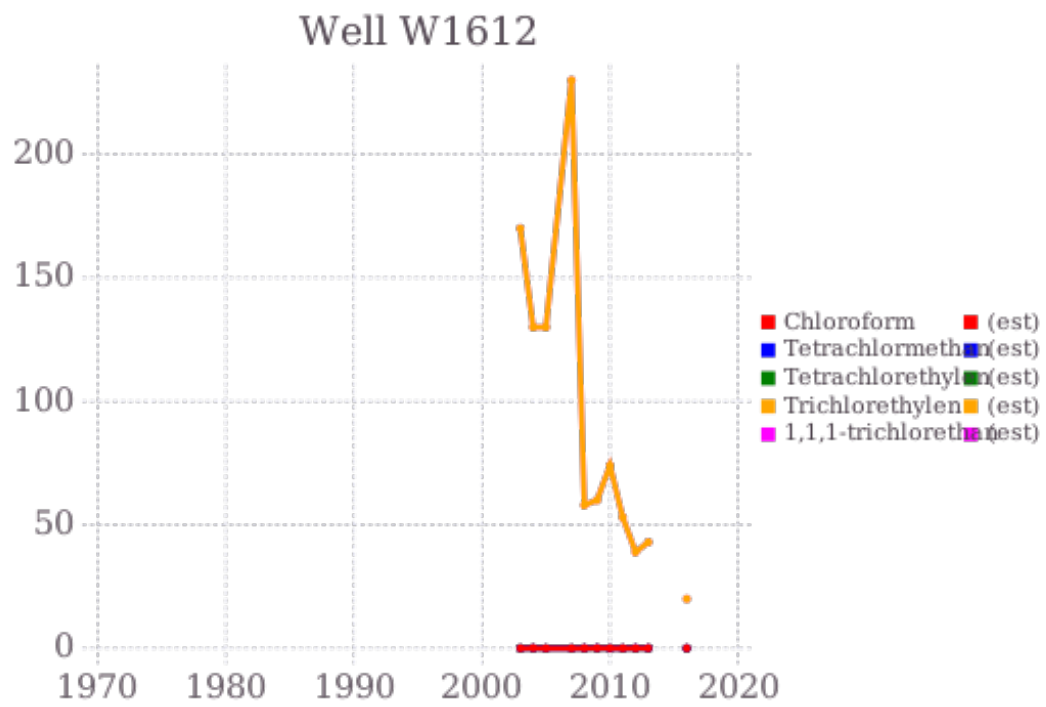
```

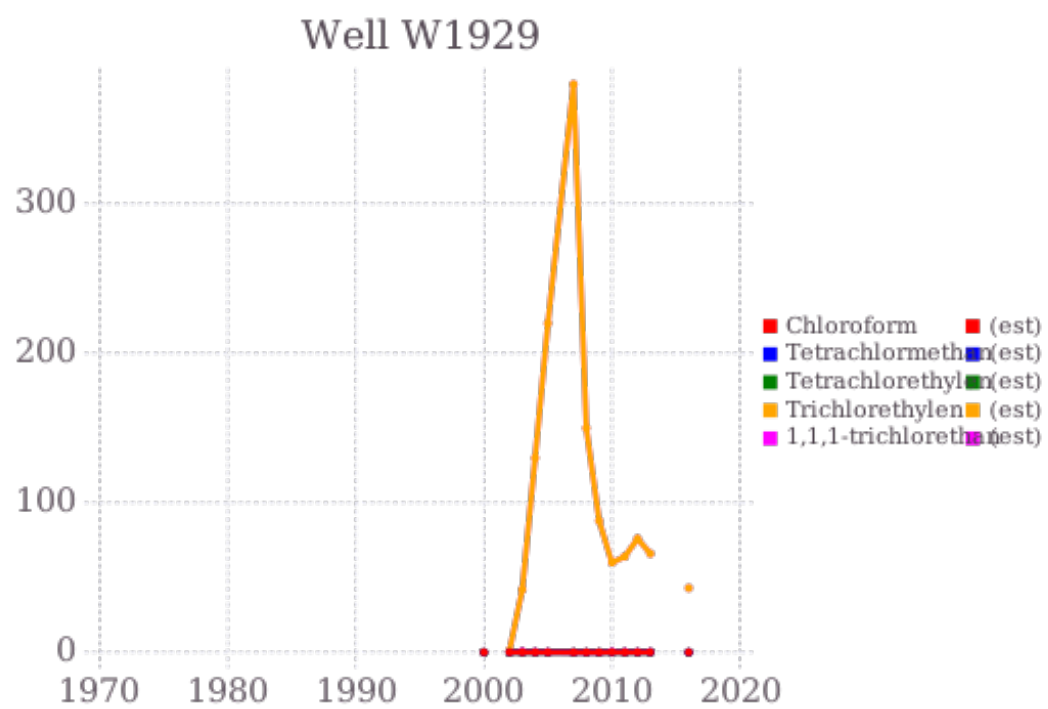
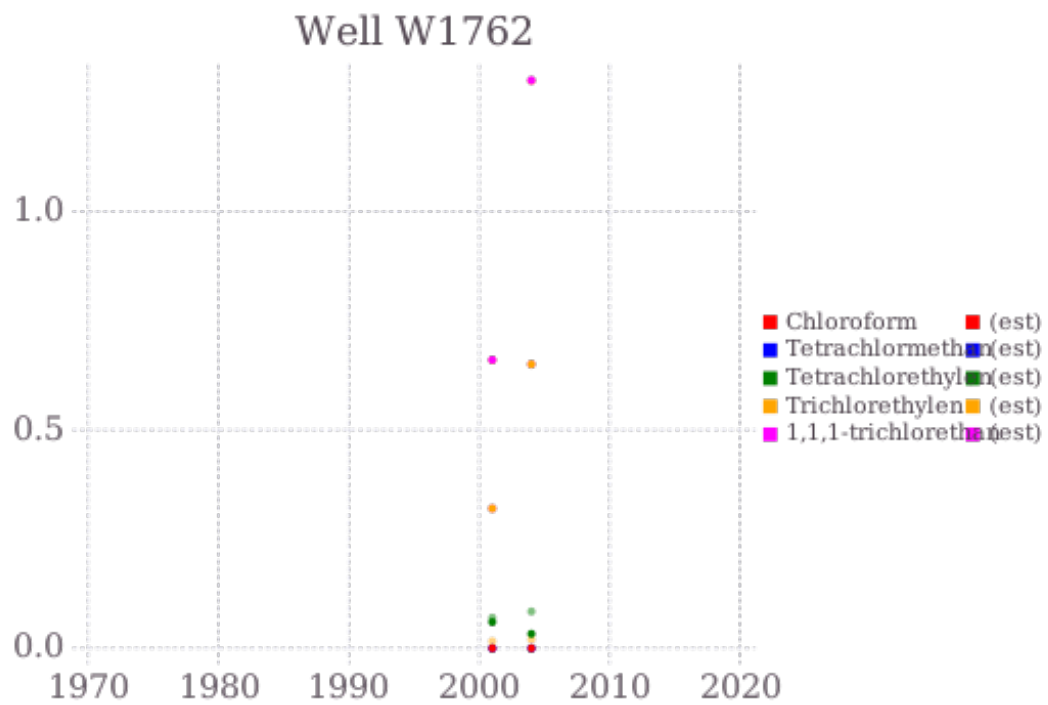
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
in apply_property at /opt/julia/packages/Compose/aVZwV/src/cairo_backends.jl
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
in apply_property at /opt/julia/packages/Compose/aVZwV/src/cairo_backends.jl
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
in apply_property at /opt/julia/packages/Compose/aVZwV/src/cairo_backends.jl
WARNING: Cairo._jl_libpango is deprecated, use Cairo.libpango instead.
  likely near /opt/julia/packages/IJulia/F1GUo/src/kernel.jl:52
in apply_property at /opt/julia/packages/Compose/aVZwV/src/cairo_backends.jl

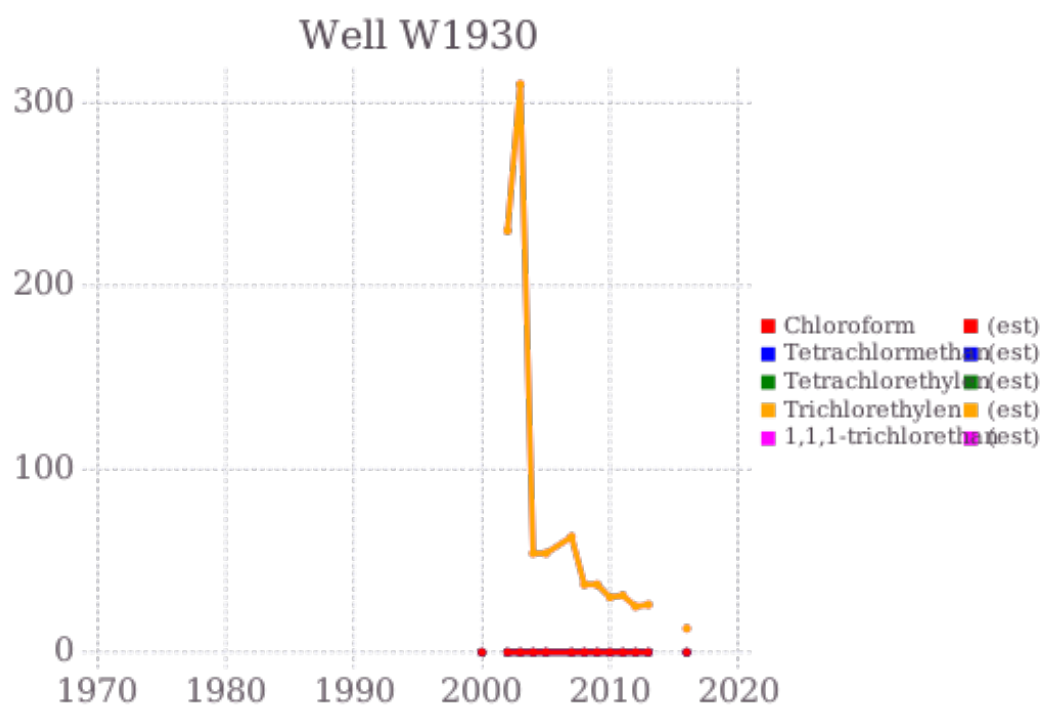
```

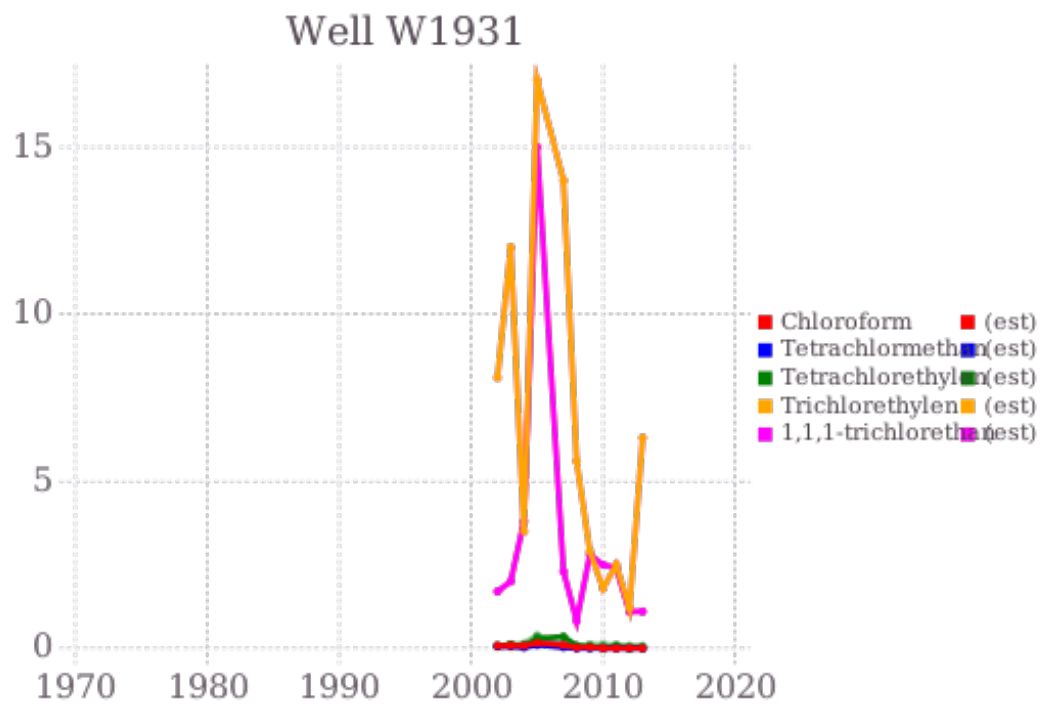


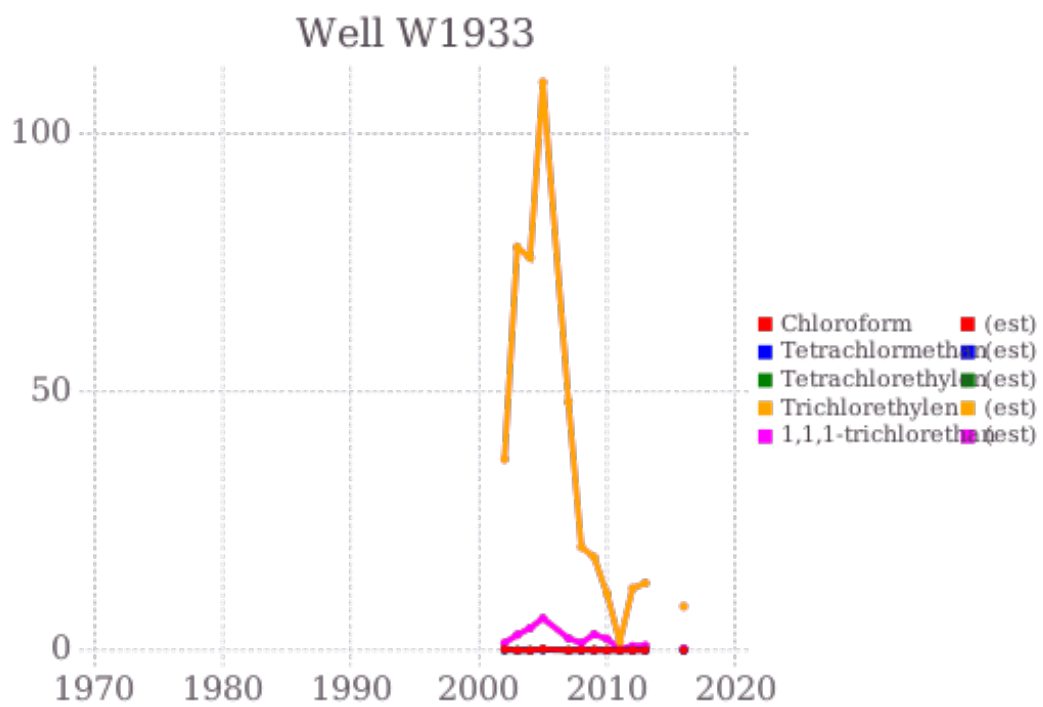
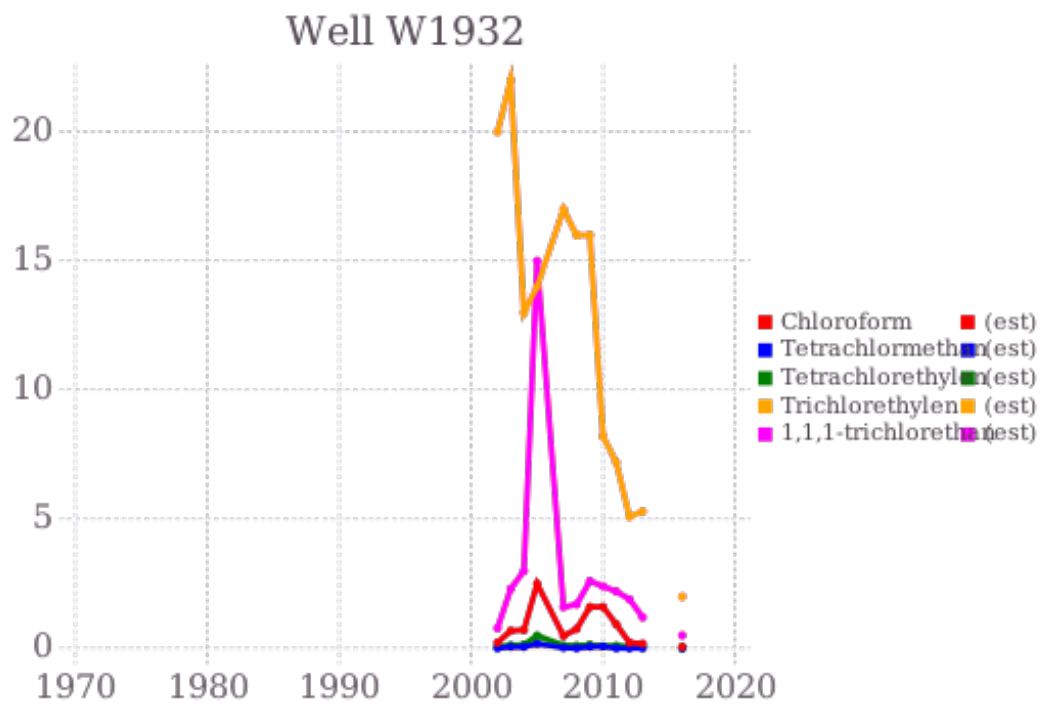


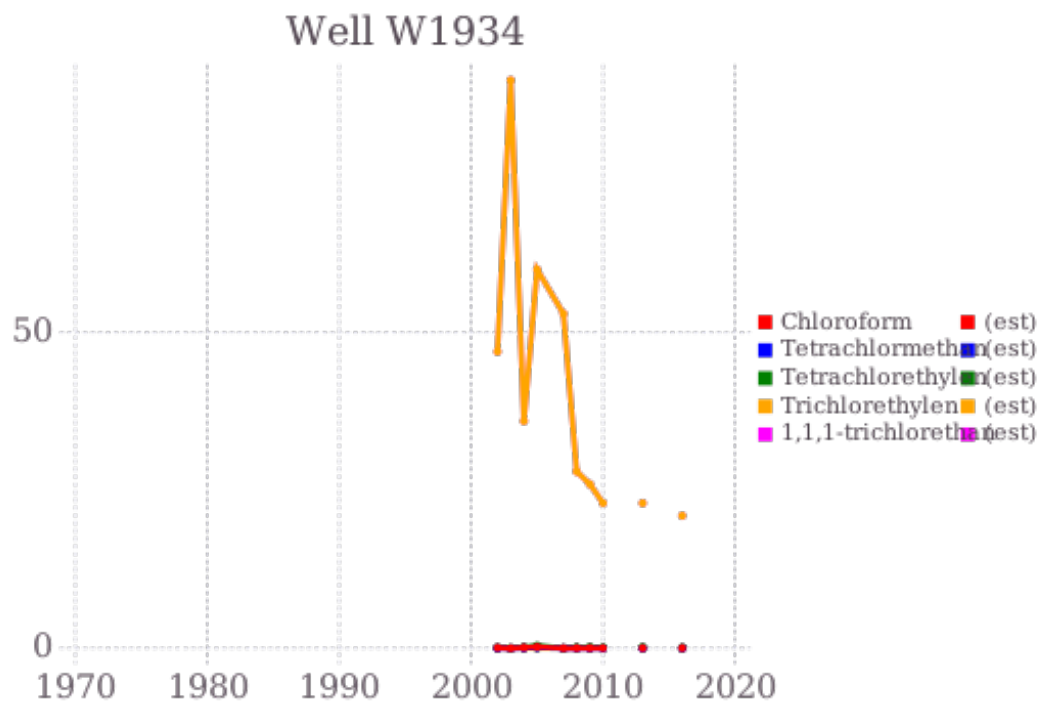


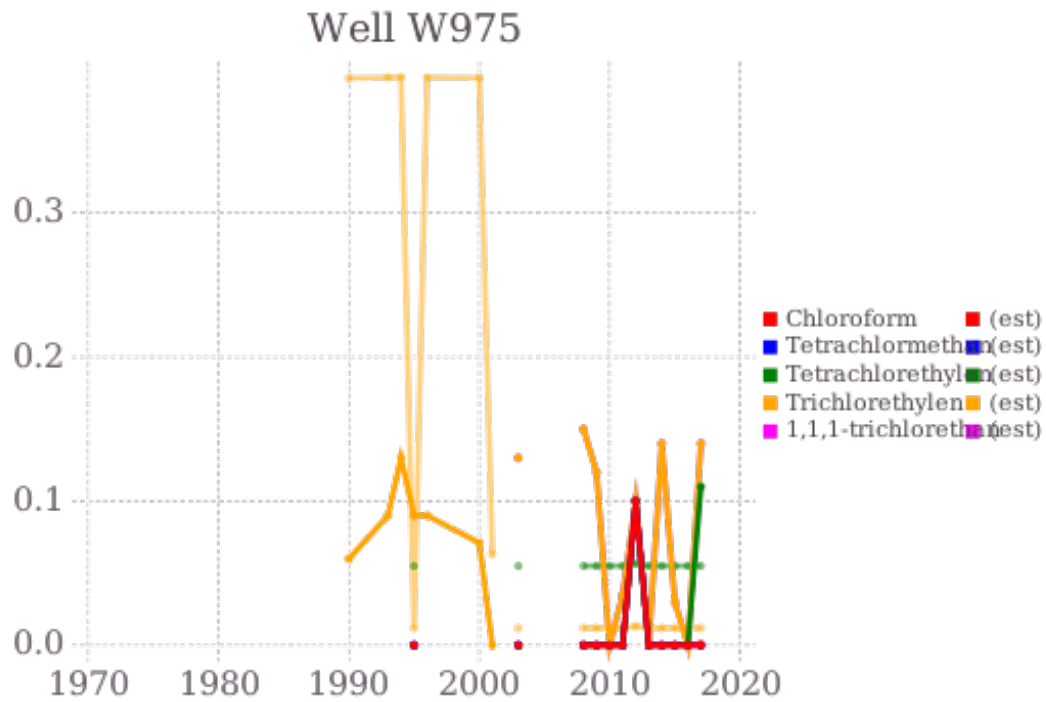












[20]: using Plots

```
for w in 1:size(wells,1)
    filepath="export/$(wells[w,1]).png"
    plotarr = Any[]
    ymax = maximum(filter(!isnan, T[w,:,:]))
    for s in 1:size(species,1)
        #println(" - ", species[s,2])
        #println(T[w,s,:])
        #println(Te[w,s,:])
        #println(years)
        p = plot(years, T[w,s,:], seriestype=:scatter, title="$(wells[w,1]) - $(species[s,2])", xlabel="Years", ylabel="Conc. {my}g/l", ylim=(0,ymax), label="Data")
        scatter!(years, Te[w,s,:], marker = (:cross,5), label="Model")
        push!(plotarr,p)
    end
    plot(plotarr..., layout=(5,1))
    plot!(size=(700,1000))
    png(filepath)
```

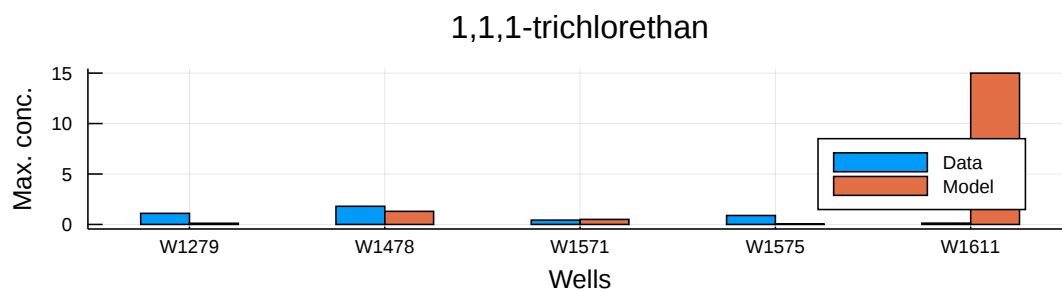
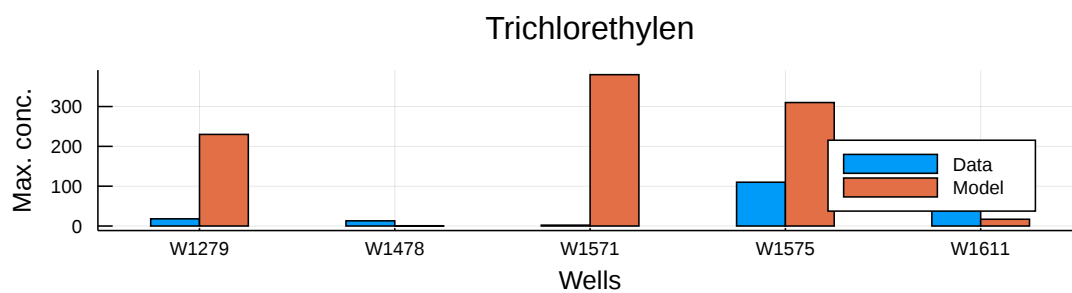
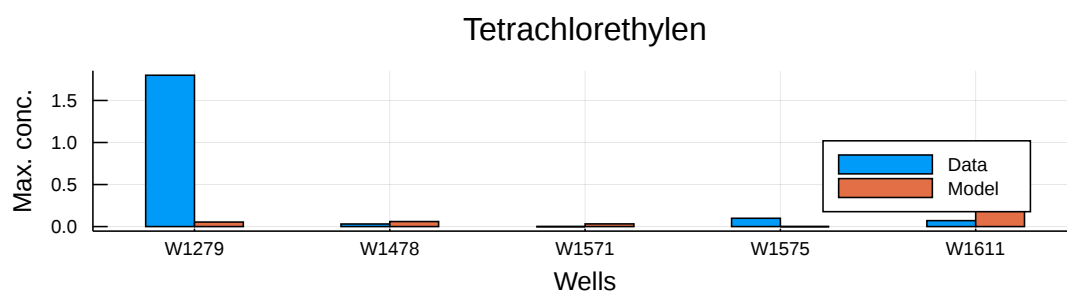
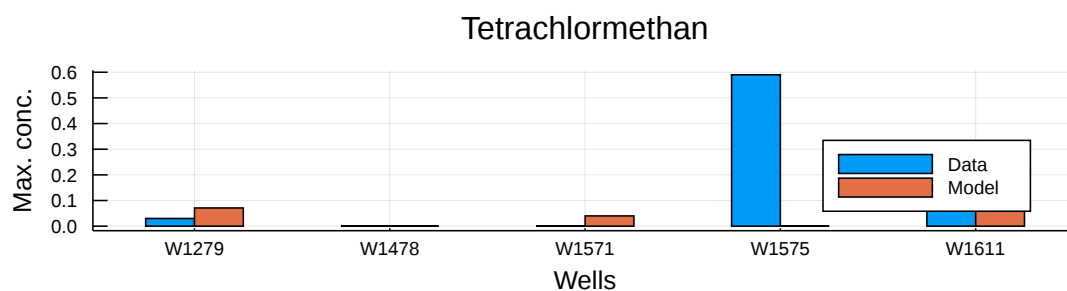
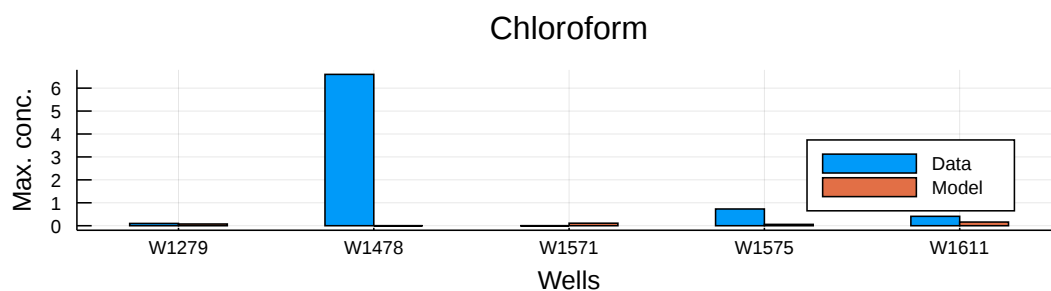
```
println(filepath)
end
```

```
export/W1279.png
export/W1478.png
export/W1571.png
export/W1575.png
export/W1611.png
export/W1612.png
export/W1762.png
export/W1929.png
export/W1930.png
export/W1931.png
export/W1932.png
export/W1933.png
export/W1934.png
export/W975.png
```

```
[21]: using StatsPlots

wellarr = wells[:,1]
ctg = repeat(["Data", "Model"], inner = 5)
plotarr = Any[]
for si in 1:size(species,1)
    Twmax = Float32[]
    Tewmax = Float32[]
    for wi in 1:size(wells,1)
        push!(Twmax, maximum(filter(!isnan, T[wi,si,:])))
        push!(Tewmax, maximum(filter(!isnan, Te[wi,si,:])))
    end
    p = groupedbar([Twmax Tewmax], group = ctg, title = "$(species[si,2])",
        xlabel="Wells", ylabel="Max. conc.", bar_position = :dodge, bar_width=0.5,
        xticks=(1:5, wellarr))
    push!(plotarr,p)
end
plot(plotarr..., layout=(5,1))
plot!(size=(700,1000))
```

[21]:

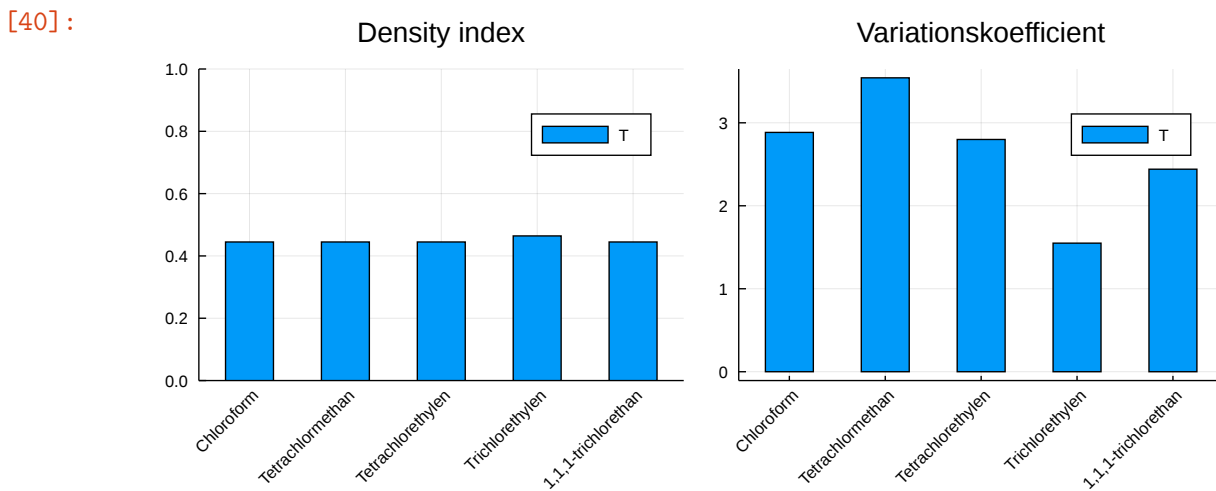


```
[22]: png("export/Species_wells_bar.png")
```

```
[37]: di_T = []
      sd_T = []
      for i in 1:size(T,2)
          data = collect(Iterators.flatten(T[:,i,:]))
          values = [data[i] for i=1:length(data) if !isnan(data[i])]
          di = length(values)/length(data)
          sd = std(values)/mean(values)
          append!(di_T, di)
          append!(sd_T, sd)
      end
```

```
[40]: using Plots

      l = @layout [a b]
      p1 = bar(species[:,2], di_T, ylim=(0, 1), bar_width=0.5, title = "Density_
      ↪index", label = "T", xrotation=45, size=(800, 400))
      p2 = bar(species[:,2], sd_T, bar_width=0.5, title = "Variationskoefficient",
      ↪label = "T", xrotation=45, size=(800, 400))
      plot(p1, p2, layout = l)
```



```
[41]: png("export/Stats.png")
```


I. Projektlog

16. december 2019

Rapport Gennemlæsning og sidste tilretning

8-15. december 2019

Mailkorrespondence Sender et af mine koncentrations-plots til Monty, hvor det ser ud som om at stofferne har skiftet plads i den estimerede tensor. Spørger om der kan være en "bug" i `mixmatchcompute()` funktionen. Monty bekræfter, at det godt kunne se sådan ud og opfordrer mig til at justerer dette når jeg plotter. Kan ikke nå at få dette med i rapporten, men vil præsentere dette til eksamen.

Aflevering Download og pakning af kode

Rapport Rapporten skrevet færdig og sidste bilag og figure er indsat.

Visualisering Tilretning og endelige kørsler af `nbInspect.ipynb` for 5 tests, så visualiseringerne bliver ensartede. Export af udvalgte figurer.

NMFk Endelige testkørsler test 1-6

7. december 2019

Visualisering Oprettelse af `nbInspect.ipynb` på baggrund af visualiseringstests

Visualiseringer Plot af kilders maksimale påvirkning af boringer i MapInfo (... den nemme løsning for tøse-drenge).

1. december 2019

NMFk Forskellige småjustering af tests og nye kørsler

NMFk Implementering af logning i test-script - nye kørsler

Rapport Udskrivning af kode og samling af bilag

30. november 2019

NMFk Oprettelse af 5 formaliserede tests

NMFk

24. november 2019

Visualisering Test af diverse visualiseringer - arbejde og neddykning i Julia pakken [Plots]

Visualiseringer Undersøge mulighederne for spatiale analyser og visualiseringer i Julia. Studie af GMT.jl (Generic Mapping Tools), som er et meget brugt bibliotek i mange moduler og i mange sprog. Opgiver dette, fordi funktionaliteten er for "generisk" - kræver nog at man laver nogle wrappere og conviniance-funktioner for at det bliver operationelt. Har ikke tid til dette nu.

19. - 21. november 2019

Mailkorrespondence Jeg sender kort med maks konc. i boringerne for hver stoftype til Monty og et kort der viser de potentielle kilder, der kan påvirke boringerne i området, med henblik på at det kan give et indtryk af et forventet resultat.

Mailkorrespondence Monty melder tilbage at resultatet ser mærkeligt ud, og han kan ikke forstå hvad der er årsagen. Foreslår at vi prøver at udvælge en begrænset tidsramme, hvor der er nok data til at kilderne kan identificeres.

Mailkorrespondence Fremsender nyt datasæt til Monty, der kun omfatter klorerede opløsningsmidler. Har ændret strukturen i datasættet, og sender derfor også mit script til at køre analysen, samt resultatet og mine plot af hhv. species, boringer og kildernes estimerede påvirkning af boringerne. Jeg gør opmærksom på at resultatet ikke stemmer med virkeligheden og foreslår nogle mulige forklaringer.

NTFk Inspektion af Monty's kildekode på github. Prøver at forstå, hvad funktioner `execute()` og `mixmatchcompute()` gør. Dette er svært - koden har kun meget sparsomme bemærkninger, der kan hjælpe. Begynder dog at få lidt bedre fornemmelse for Julia.

17. november 2019

NTFk Udarbejdelse af kort med maks koncentrationer for alle boringer.

16. november 2019

NTFk Test af kørsel af julia-script via tmux. Her kobler jeg mig på den kørende container via en tmux-session, og lader scriptet køre i lab-containeren mens jeg spiser aftensmad. Efterfølgende kan jeg koble mig på igen og konstaterer at scriptet er færdig med at køre. Dette fungerer perfekt!

NTFk Udarbejdelse af Julia-script, der indlæser datasættet, genererer T-vektoren og kører NTFk.

10. november 2019

NTFk Udarbejdelse af notesbog til systematisk dataudtræk fra SQL serveren. Dette er en anden struktur end det første datasæt. Datasættet gemmes på drevet som csv-fil.

NTFk Test af tmux, herunder at kunne logge på lvm01 via ssh, oprette en tmux-session, starte en process, koble mig ud af tmux-sessionen, logge ud af ssh og efterfølgende kunne logge ind igen og 'koble mig tilbage på tmux-sessionen med den kørende process. Det virker.

9. november 2019

visualisering Udfordring med visualiseringer. Mangler færdigheder i Julia for at få det til at se ud som jeg gerne vil have. Læsning af manualer og google-søgninger.

3. november 2019

NTFk Første estimerede koncentrationer med funktionen `mixmatchcompute()`

Visualisering Koncentrationer visualiseres med Montys hjælpefunktioner fra Mads. Det er svært at skelne mellem faktuelle og modellerede resultater - for mange prikker og farver oven i hinanden. Skal finde en bedre måde at gøre det på.

2. november 2019

Mailkorrespondance Monty oplyser at man ikke kan gange de to matricer H og W med julias normale matrice-funktion. Man skal bruge funktionen `mixmatchcompute()` fra NMFk pakken.

NTFk Forsøger at gange H og W med standard Julia syntaks. Dette virker ikke. Bruger lang tid på google-fejlsøgning uden held - det burde virke, siger Google.

27. oktober 2019

Lab Ombygning af JupyterLab container, så GPU-beregninger understøttes. Dette er meget omfattende. Mange fejl, men det ender med at containeren kan bygges.

NMFk efter granskning af H og W må jeg konkludere, at alle kilder tilsyneladende påvirker alle borer i modellen. Dette virker mærkeligt.

26. oktober 2019

Mailkorrespondance Monty indikere mellem linjerne at datasættet er meget "sparse" og at det giver usikker heder i forhold til at finde antallet af kilderne.

NTFk Prøve-kørsler med forskellige kombinationer af borer og species.

Lab Test af NVIDIA/CUDA docker-container på min egen linux-pc. Dette virker.

Lab Test installation af NVIDIA/CUDA-containeren på lvm01. Fejl - kræver at der bliver installeret CUDA-tools på host-maskinen først. Efter dette er gjort, fungerer NVIDIAI-containeren.

23. oktober 2019

NTFk Overvejelser og læsning omkring muligheder for lange kørsler i Jupyter og ssh, uden at skulle holde klienten åben. Muligheder for at kunne "detache". Lander på tmux som en mulig løsning - jeg har ofte læst om tmux og har haft lyst til at lære det, men aldrig fået det prøvet.

22. oktober 2019

Mailkorrespondance Monty sender resultatet for den første kørsel og beder mig kikke på det og plotte resultaterne.

NTFk gentager Montys demo-kørsel (lille datasæt).

21. oktober 2019

Mailkorrespondance Monty melder at han har kørt algoritmen og har fundet en god løsning med 4 kilder. Viser hvordan man kan bruge funktionen `checkarray-entries()` til at frasortere de boringer med for få data.

20. oktober 2019

Visualisering Første forsøg på at inspicere og visualisere resultaterne. Dette er en udfordring, da jeg ikke kender Julia særlig godt. Bruger en del tid på at læse og søge på nettet.

SQL Server Implementering af cron-job til opdatering af databaser. Genbygning af containeren.

19. oktober 2019

NMFk kørslen bliver først færdig søndag aften. Den lange kørselstid bliver en udfordring - skal undersøge muligheden for at køre koden via GPU eller udskalerer beregningerne.

18. oktober 2019

SQL Server Undersøger muligheden for at flytte script til download af databaser ind i Docker kontaineren, herunder muligheden for at køre et script som det første, når kontaineren starter. Laver første forsøg med dette der fejler.

Mailkorrespondance Monty melder at der er et issue i forhold til at datasættet er alt for "sparse" dvs. for få data på nogen af boringerne og analyserne.

NMFk ... og den kører stadigvæk lørdag aften, da jeg går i seng!

17. oktober 2019

Mailkorrespondance Montys sender resultatet af kørsel på det første datasæt samt instruks for preprocessing af datafil og NMFk-kørsel. Oplyser at NMFk-pakken er blevet opdateret og beder mig hente den nyeste.

NMFk Testkørsel ud fra Montys instrukser i min Jupyter notebook. Koden fungerer og kan køres - 1. kørsel sættes i gang ... den kører stadig da jeg går i seng

13. oktober 2019

Teori Fortsatte studier af NTF herunder Tucker-decomposition, samt nærlæser Monty's NTFk-artikler for at forstå metoden.

Teori Første bud på et teoriafsnit

12. oktober 2019

Teori Læser om dekomposition, HOSVD og NMF / NTF.

6. oktober 2019

Teori Læsning af [27]

Jupyter notebook Test af at overføre Montys konfiguration af hans docker image til Jupyter datascience-notebook billedet. Dette lykkedes efter mange fejl og fejlrettelser samt google-søgninger - kan nu køre den justerede datascience-notebook, og importerer NTFk og Mads uden fejl. Prøver at køre tests beskrevet på Montys github side, og dette fungerer.

5. oktober 2019

Rapport Indledning og problemformulering er skrevet

Teori Læsning af [25] og [26]

Jupyter notebook Testkørsel af Montys Docker image. Det virker og NTFk kan importeres uden fejl. Montys docker image giver dog ikke adgang til Jupyter - kun en terminal. Hvis jeg skal lave projektet via en Jupyter notebook, er jeg derfor nød til enten at tilføje Jupyter til Montys image eller at tilføje Montys konfiguration til Jupyter's Datascience-notebook. Da jeg også har brug for at kommunikere med SQL Serveren, og derfor alligevel skal konfigurere og installere, bliver det nok det sidste jeg forsøger.

30. september 2019

Mailkorrespondance Monty sender link til Docker image til NTFk.

SQL server Udbygning af SQL scripts, så de bliver nemmere at bruge.

29. september 2019

Testdataset testdataset sendt til Monty.

Teori Introduktion til programmeringssproget *Julia* læst / skimmet og de første forsøg med sproget er gennemført.

28. september 2019

Testdatasæt Udvalgelse af de vigtigste analyseparametre til datasættet baseret på mere end 100 forekomster af en parameter i datasættet. Herefter pivotering af datasættet baseret på analyseparametre (species), med max-værdi for hvert boring og år.

24. - 25. september 2019

Mailkorrespondance Monty oplyser at han forventer at lave et Docker-billede til hans NTFk-pakke.

22. september 2019

Jupyter Notebook Forsøg med at oprette nyt docker-billede ud fra Datascience-Notebook fra Jupyter, med konfiguration og installation af afhængigheder som fremgår af Montys Docker-billede.

Rapport Gentage udvalgte dele af processen omkring oprettelse af VM og lave screendumps af udvalgte dele til dokumentation og rapport.

21. september 2019

Jupyter Notebook Fandt ud af at Monty udstiller et docker-billede til en af sine andre pakker (MADS), som er en af de pakker der fejler. Har oprettet en docker-container fra dette billede og testet at fælgende:

- MADS-pakken kan tilføjes og kompileres uden fejl - OK
- NTFk-pakken kan tilføjes, men den fejler stadig pga. manglende afhængigheder

17. september 2019

Vejledermøde Drøftelse af fundne fejl og mulig strategi herfra. Beslutning om at kontakte Monty og spørge han om råd i forhold til fejl. Endvidere drøftelse af plan B, hvis disse ikke kan løses og NTFk pakken dermed ikke kan køres lokalt.

16. september 2019

Test datasæt SQL til sammensmeltning af de to datasæt, som bedst muligt tager højde for redundante analyser.

GitHub Oprette [/koetter/NTFk-POC] på min Github konto, så denne kan bruges til versions-styring og backup af projektet.

15. september 2019

SQL server Logget på SQL serveren og restorede de to downloadede backup-filer til databaser på serveren manuelt. Test af at kunne trække data ud fra begge databaser.

Jupyter Notebook Oprettelse af docker container ud fra billedet "Datascience-notetook" fra Jupyter, hvor Julia er installeret. Test af følgende:

- Julia OK
- Julia ODBC-pakke - FEJL, der skal installeres ODBC-biblioteker som en del af dockerbilledet
- Julia NTFk-pakke - FEJL ... mange fejl! Der mangler Python-pakker, pakker skal recompileres m.m.

Jupyter Notebook Opbygning af nyt Docker image med udgangspunkt i Datascience-Notebook, hvor nødvendige fejl forsøges rettet, ved installation af manglende pakker. Iterativ process trial-and-error. Kan løse de fleste fejl, men ender med at der mangler nogle biblioteker i styresystemet, og kan ikke komme længere.

14. september 2019

Azure VM Geninstallation af VM i Azure, med justerede parametre og adgangs-konfiguration. Adgang fra forskellige enheder virker nu, og ssh-adgang kan stadig baseres på public-private key, ved at public key uploades til brugerens ssh-mappe.

Azure VM Ved forsøg på at anvende dataDisken til persistent data-lagring for SQL-serveren, blev jeg opmærksom på at Datadysken ikke er blevet mountet. Har undersøgt hvordan disken kan findes / identificeres og hvordan denne kan automountes vha. /etc/fstab og har fået gjort dette.

Azure VM Bash-script til download af databaser sat i gang igen

10. september 2019

Test datasæt SQL til dataudtræk fra GeoGIS databasen, som kan danne grundlag for test-datasæt til Monty. Dataudtræk lavet, så det har samme struktur som udtrækket fra Jupiter.

8. september 2019

Rapport L^AT_EXrapport skabelon er blevet oprettet og foreløbig overordnet indholdsfortegnelse er lavet, så jeg kan begynde skriveprocessen. Bilag H er oprettet som projektlog, så udførte opgaver, problemer, løsninger og andet kan dokumenteres.

Azure VM Begge databaser er blevet downloadet vha. bash-script, oprettet dagen før.

SQL Server Restorede Jupiter-databasen ind på serveren. Ved forsøg på restore af GeoGIS databasen opstod fejl. Efter længere fejlsøgning kunne jeg konstatere, at problemet skyldes pladsmangel på disk. Både download og docker-containerens persistente datalagring sker på maskinens OSDisk, som ikke er særlig stor. Skal have flyttet data og notebooks til maskinens datadisk i stedet.

7. september 2019

Docker Oprettelse af MS SQL container ud fra <https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-2017&>

`pivots=cs1-bash`. Kunne efterfølgende ikke logge på sql serveren. Prøvede at oprette containeren ud fra linux og Ubuntu specifikke images (https://hub.docker.com/_/microsoft-mssql-server), men dette hjalp ikke. Fandt at problemet var parameteren `SA_PASSWORD`, der skal sættes med `-e *****` hvor hele parameteren er i anførselstegn og ikke bare passwordet.

VM Oprettelse af bash-script, der kan downloade hhv. regionens GeoGIS-database og Jupiter-databasen som backup-filer, så disse kan restores ind på SQL serveren.

4. september 2019

Test datasæt SQL til dataudtræk fra Jupiter databasen, som kan danne grundlag for test-datasæt til Monty

2. september 2019

Vejledermøde Drøftelse af projektide og projektdefinition.

1. september 2019

Azure VM Installation af programmer via apt - dette bør scriptes, så det kan køres automatisk ved redeployment

Docker installation Docker blev installeret efter vejledning på <https://docs.docker.com/install/linux/docker-ce/ubuntu/> og testet.

Azure VM Erkendelse af at brugeradgang med ssh-key giver en udfordring med adgang fra andre enheder. Forsøgte at logge på med min ubuntu workstation i stedet for min mac, men ssh-nøglen er enhedsspecifik. Skal derfor omkonfigurere VM'en med brugeradgang via password i stedet, så jeg kan tilgå den fra andre enheder.

31. august 2019

Azure VM Oprettelse af VM i enhedens test-ressourcegruppe i Azure. VM oprettet med følgende specifikationer:

- VM navn: *****
- Styresystem: Ubuntu 18.04
- Størrelse: Standard D4s v3 (4 vcpus, 16 GiB memory)
- OSdisk: ***** _OsDisk_1_b23a... , 30 GiB (Premium SSD)
- Data disk: ***** -DataDisk0, 1023 GiB (Premium SSD)
- Networkinterface og Public IP
- Brugeradgang opsat med brugernavn ak og public-private ssh-key (sha256)

Azure VM Test af adgang til VM via ssh - ok

28. - 30. august 2019

Mailkorrespondance Reach out og første korrespondance med Monty. Oplyser at hans kode vil være udfordrende at forstå og at arbejde med, da den ikke er dokumenteret. Han tilbyder at køre analysen for mig og sende mig resultaterne. Vi aftaler i stedet at jeg kan bruge hans resultater til at validerer mine egne kørsler.