How a Bayesian Hierarchical Model makes our roads safer

Eda Özdemir, Gururaj Desai, Tim Schleicher

Final presentation in the course 82005000 “Probabilistic Modelling”

Leuphana Universität Lüneburg, 04.07.2019

Agenda

I INTRODUCTION & RECAP

... of the overall approach used in the paper

II DESCRIPTIVE ANALYSIS

... to get a deeper understanding of the data and what the modelling aims at

III MODELLING & PREDICTING

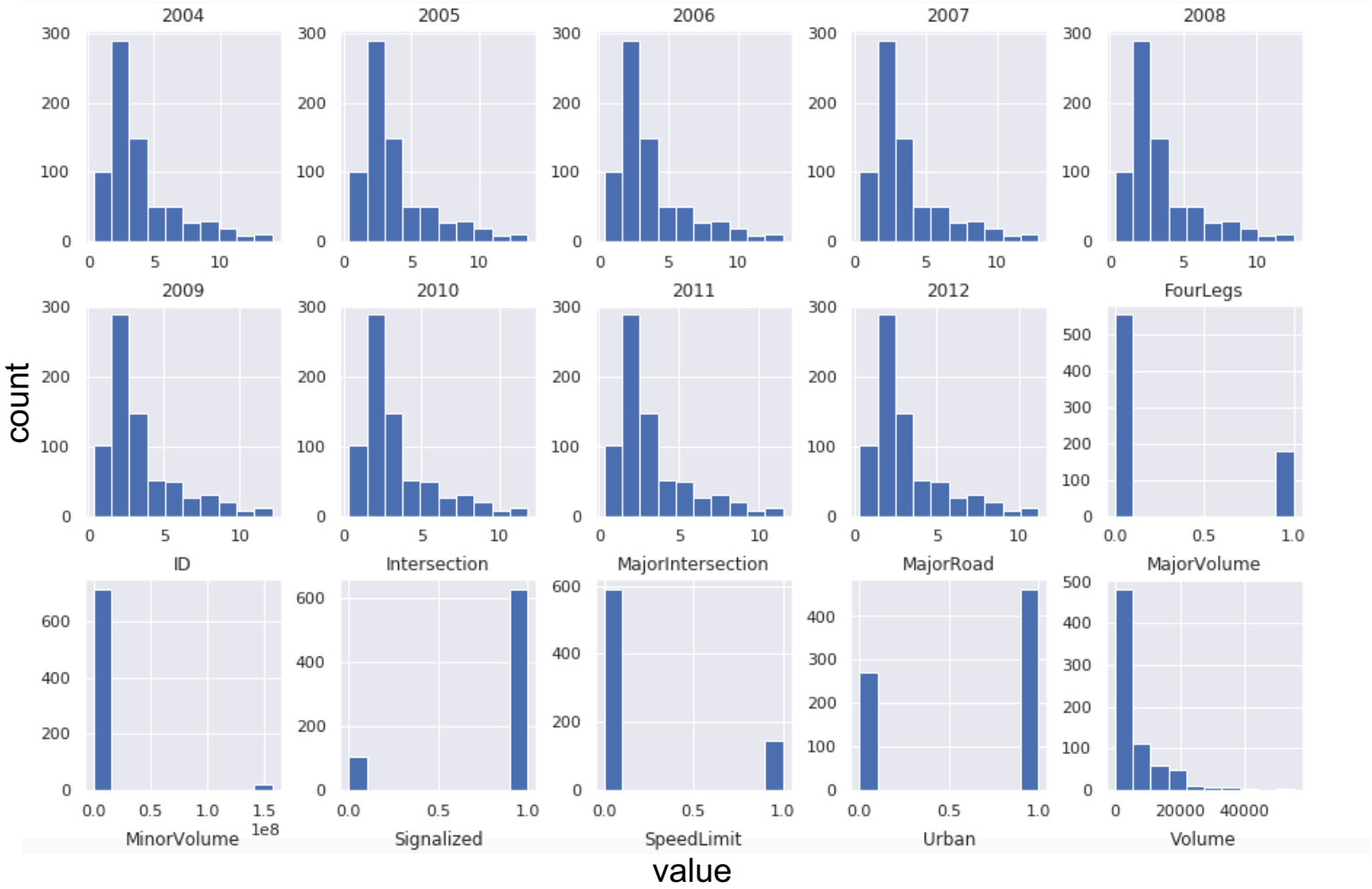
... future road safety hotspots in the city of Halle

IV NEXT STEPS

... based on our learnings and how we can improve the model

Our variables follow a distribution which we want to learn via our modeling

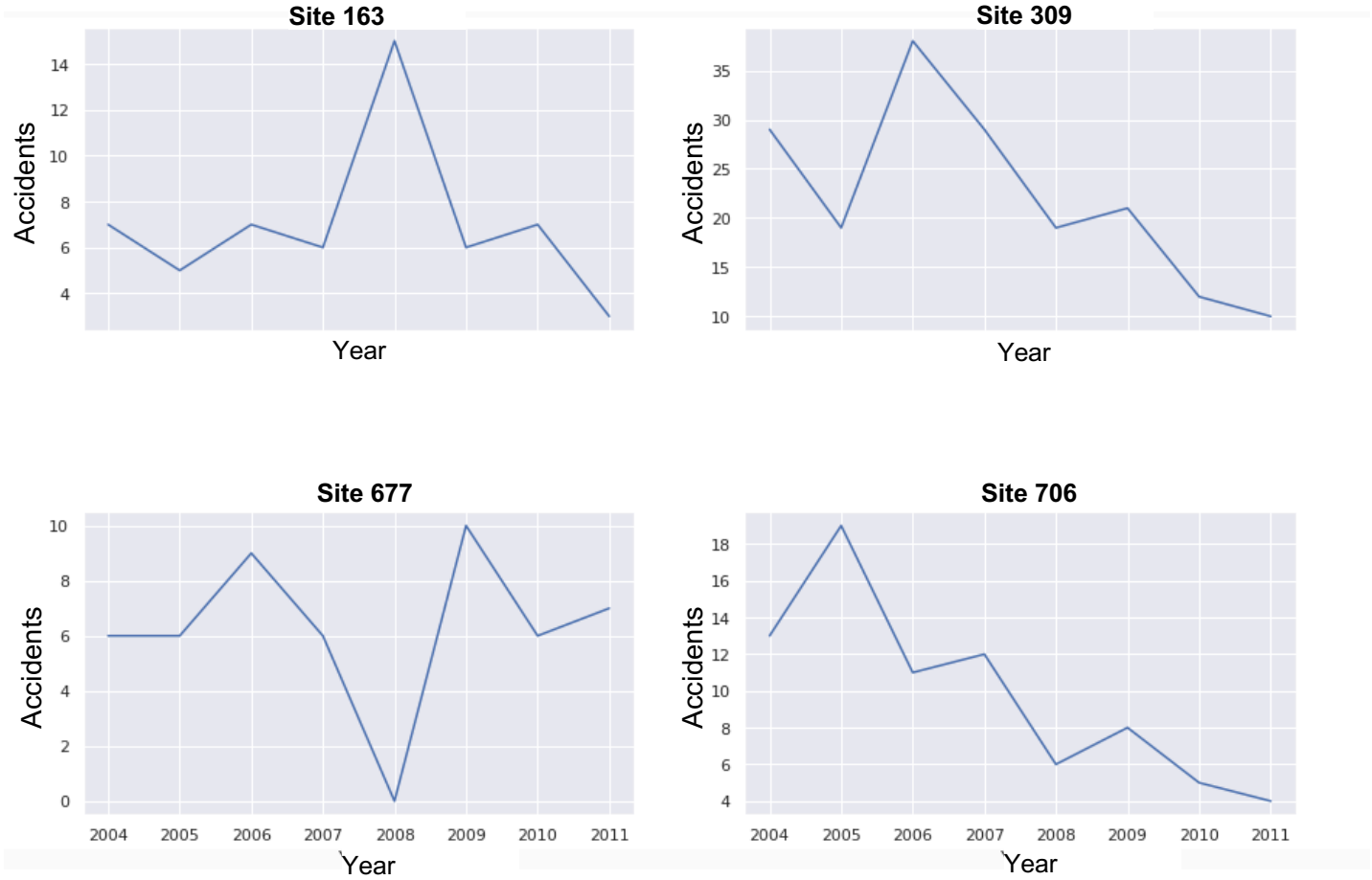
Distributions of all variables in the data



SOURCE: Own visualization based on data from Fawcett et al. 2017

Accidents at sites have random effects and trend – which we account for in our modelling

Time series plots of accidents at selected sites



- Sites 163 and 677 indicate potential **RTM effects**
- Sites 309 and 706 suggest evidence of **temporal trend**

SOURCE: Own approach based on Fawcett et al. 2017

We follow the same steps as Fawcett et al. in their paper

Bayesian Hierarchical Model

$$\lambda_j(t) = a_j \mu_j(t) \exp\{b_j t\}$$
$$a_j > 0; -\infty < b_j < \infty; t \leq 0$$

Accident Prediction Model

$$\mu_j(t) = \exp\left\{\beta_0 + \beta_t t + \sum_{p=1}^{n_p} \beta_p x_{p,j}\right\}$$

Posterior Sampling

Rank sites according to hotspot potential

Evaluation

SOURCE: Own depiction based on Fawcett et al. 2017

Agenda

I INTRODUCTION & RECAP

... of the overall approach used in the paper

II DESCRIPTIVE ANALYSIS

... to get a deeper understanding of the data and what the modelling aims at

III MODELLING & PREDICTING

... future road safety hotspots in the city of Halle

IV NEXT STEPS

... based on our learnings and how we can improve the model

To implement the Accident Prediction Model, we have to do data rearrangement and calculate the coefficients first (1/2)

Modelling the APM according to Fawcett et al. 2017

$$\mu_j(t) = \exp\left\{\beta_0 + \beta_t t + \sum_{p=1}^{n_p} \beta_p x_{p,j}\right\}$$

```
data_horizontal = pd.concat([data_2004, data_2005, data_2006, data_2007, data_2008,
                             data_2009, data_2010, data_2011, data_2012], axis=0)
```

	y	Volume	MajorVolume	MinorVolume	Urban	Intersection	Signalized	SpeedLimit	MajorRoad	MajorIntersection	FourLegs	time_coeff
0	20	55280	54919.43	360.98	1	1	1	60	1	0	0	-7
1	11	638	637.94	0.00	1	0	0	50	1	0	0	-7
2	6	1658	1103.58	554.75	1	1	1	45	1	1	0	-7
3	3	751	335.25	415.62	1	1	0	45	1	0	0	-7
4	0	2366	2364.20	1.33	1	1	0	30	0	0	0	-7
5	0	64778	42007.65	22770.83	1	0	0	60	1	1	1	-7
6	2	280	279.69	0.25	1	1	0	30	0	0	0	-7

SOURCE: Own approach based on Fawcett et al. 2017

To implement the Accident Prediction Model, we have to do data rearrangement and calculate the coefficients first (2/2)

Modelling the APM according to Fawcett et al. 2017

$$\mu_j(t) = \exp\left\{\beta_0 + \beta_t t + \sum_{p=1}^{n_p} \beta_p x_{p,j}\right\}$$

```
n_iterations = 100
formula = 'y ~ time_coeff + Volume + MajorVolume + MinorVolume + Urban + \
Intersection + Signalized + SpeedLimit + MajorRoad + MajorIntersection + FourLegs'

mu_model_all = pm.Model()
with mu_model_all:

    pm.glm.GLM.from_formula(formula=formula, data=data_horizontal,
                            family=pm.glm.families.NegativeBinomial())

    trace_a11 = pm.sample(n_iterations, tune=80, init='adapt_diag')
```

	mean	sd	mc_error	hpd_2.5	hpd_97.5	n_eff	Rhat
Intercept	0.000660	0.000927	0.000390	-0.000841	0.002206	1.589310	2.993953
time_coeff	-0.002817	0.000790	0.000330	-0.003876	-0.001617	1.784098	2.300166
Volume	0.000621	0.000366	0.000132	0.000024	0.001087	1.554851	3.481463
MajorVolume	-0.000610	0.000365	0.000132	-0.001074	-0.000014	1.555069	3.480915

SOURCE: Own approach based on Fawcett et al. 2017

Then, we calculate the mean accident rate for each site and each time ...

Modelling the APM according to Fawcett et al. 2017

$$\mu_j(t) = \exp\left\{\beta_0 + \beta_t t + \sum_{p=1}^{n_p} \beta_p x_{p,j}\right\}$$

```
def get_mu_for_site(time, site):

    model_learnt_coeff = mu_all_dataframe["mean"]

    accident_mean_predicted = model_learnt_coeff["Intercept"] + time * model_learnt_coeff["time_coeff"] + \
                              model_learnt_coeff["Volume"] * site[0] + model_learnt_coeff["MajorVolume"] * site[1] + \
                              model_learnt_coeff["MinorVolume"] * site[2] + model_learnt_coeff["Urban"] * site[3] + \
                              model_learnt_coeff["Intersection"] * site[4] + model_learnt_coeff["Signalized"] * site[5] + \
                              model_learnt_coeff["SpeedLimit"] * site[6] + model_learnt_coeff["MajorRoad"] * site[7] + \
                              model_learnt_coeff["MajorIntersection"] * site[8] + model_learnt_coeff["FourLegs"] * site[9]


    return accident_mean_predicted
```

SOURCE: Own approach based on Fawcett et al. 2017

… and encounter two problems we know well by now!

Modelling the APM according to Fawcett et al. 2017

$$\mu_j(t) = \exp\left\{\beta_0 + \beta_t t + \sum_{p=1}^{n_p} \beta_p x_{p,j}\right\}$$

	2004	2005	2006	2007	2008	2009	2010	2011	2012
0	6.808601	6.827808	6.847068	6.866383	6.885753	6.905177	6.924656	6.944189	6.963778
1	2.957803	2.966147	2.974514	2.982905	2.991320	2.999758	3.008220	3.016706	3.025216
2	2.722779	2.730460	2.738162	2.745886	2.753632	2.761400	2.769190	2.777001	2.784835
3	2.678493	2.686049	2.693626	2.701225	2.708845	2.716486	2.724149	2.731834	2.739540
4	1.945485	1.950973	1.956477	1.961996	1.967530	1.973081	1.978647	1.984228	1.989825
5	11.304993	11.336883	11.368863	11.400934	11.433095	11.465347	11.497689	11.530123	11.562649
6	1.900571	1.905932	1.911308	1.916700	1.922107	1.927529	1.932966	1.938419	1.943887

We do not account for RTM and trend yet!

SOURCE: Own approach based on Fawcett et al. 2017

Reminder: We account for RTM and trend by building a Bayesian Hierarchical Model on top of the APM

Model according for RTM and trend according to Fawcett et al. 2017

$$\mu_j(t) = \exp\left\{\beta_0 + \beta_t t + \sum_{p=1}^{n_p} \beta_p x_{p,j}\right\}$$

$$\lambda_j(t) = a_j \mu_j(t) \exp\{b_j t\}$$

$$\boldsymbol{\theta}_j^{(i)} = \left\{a_j, b_j, \tau_j, \lambda_j(t)\right\}^{(i)}$$

$$\text{NegBin}\left(r = \frac{\lambda_j(t)}{c(t)-1}, p = \frac{1}{c(t)}\right) \qquad t < 0$$

$$\text{Poisson}(\lambda_j(t)) \qquad t \geq 0$$

SOURCE: Own depiction based on Fawcett et al. 2017

For the modeling, we first have to initialize the priors according to what is specified in the paper

Prior initialization

```
%%time

n_samples = 1000

import theano.tensor as tt

modelMultAj = pm.Model()

with modelMultAj:

    tau = pm.Gamma("tau", 2,20)

    aj = pm.Gamma("aj_before", 4, 4)

    bn = pm.Normal("bn_before",0,0.1)

    bz = pm.Bernoulli("bz", 0.5)

    bn_array =  bn.random(size=734)

    bz_array = bn.random(size=734)

    bj_array = pm.Deterministic("bj", tt.mul(bn_array,bz_array))

    aj_array = pm.Deterministic("aj", tt.mul(1, aj.random(size=734)))
```

$$y_j(t)|\lambda_j(t)\sim \begin{cases} \text{Poisson}(\lambda_j(t)), & t \geq 0; \\ \text{NegBin}\left(r=\frac{\lambda_j(t)}{c(t)-1}, p=\frac{1}{c(t)}\right), & t<0; \end{cases}$$

$$\tau \sim \text{Gamma}(2, 20)$$

$$a_j \sim \text{Gamma}(\gamma, \gamma)$$

$$b_N \sim N(0, 0.1) \qquad b_Z \sim \text{Bernoulli}(0.5)$$

$$b_j = b_N b_Z$$

SOURCE: Own approach based on Fawcett et al. 2017

The pymc3 library allows us to smoothly apply a Negative Binomial Distribution to the “before” time period

```
models = []

t = -7

for time_range in range(2004, 2011):

    mu_j_t = mu_j[str(time_range)]

    aj_mj = pm.Deterministic("aj_mj_{}".format(time_range), tt.mul(aj_array,mu_j_t))

    lamda_aj_mj = pm.Deterministic("lamda_{}".format(time_range),
                                   tt.mul(aj_mj, tt.exp(tt.mul(t,bj_array))))

    lamda_final = pm.Deterministic("lamda_before_{}".format(time_range),
                                   (lamda_aj_mj) / ((np.exp(-1 * t * tau)) - 1))

    observed = data["y_{}".format(time_range)]

    current_model = pm.NegativeBinomial("before_time_{}".format(time_range), mu = lamda_final,
                                        alpha = 1/(np.exp(-1 * t * tau)),
                                        observed = observed)

    models.append(current_model)

    t = t + 1
```

$$y_j(t)|\lambda_j(t)\sim \begin{cases} \text{Poisson}(\lambda_j(t)), & t \geq 0; \\ \text{NegBin}\left(r=\frac{\lambda_j(t)}{c(t)-1}, p=\frac{1}{c(t)}\right), & t<0; \end{cases}$$

$$c(t)=\exp\{-t\tau\},\ t<0, \tau>0$$

$$\lambda_j(t)=a_j\mu_j(t)\exp\{b_j t\}, a_j>0; -\infty<b_j<\infty; t\leq 0$$

SOURCE: Own approach based on Fawcett et al. 2017

For the “after” time period data, we have applied a Poisson distribution

```
    t = 0

    for time_range in range(2011, 2013):

        mu_j_t = mu_j[str(time_range)]

        aj_mj = pm.Deterministic("aj_mj_{}".format(time_range), tt.mul(aj_array,mu_j_t))

        lamda_aj_mj = pm.Deterministic("lamda_after_{}".format(time_range),
                                       tt.mul(aj_mj, tt.exp(tt.mul(t,bj_array))))

        observed = data["y_{}".format(time_range)].values

        current_model = pm.Poisson("after_time_{}".format(time_range),
                                   mu = lamda_aj_mj, observed = observed)

        models.append(current_model)

        t = t + 1

    trace_final_model_before_and_after = pm.sample(n_samples, tune=10, init='adapt_diag')

print("DONE MODELING!!!!!!!!!!")
#pm.save_trace(trace_final_model_before_and_after, "./final_trace/", overwrite=True)
pm.model_to_graphviz(modelMultAj)
```

```
Multiprocess sampling (2 chains in 2 jobs)
CompoundStep
>NUTS: [bn_before, aj_before, tau]
>BinaryGibbsMetropolis: [bz]
Sampling 2 chains: 100%|██████████| 2020/2020 [03:16<00:00,  4.32draws/s]
```

$$y_j(t)|\lambda_j(t)\sim \begin{cases} \text{Poisson}(\lambda_j(t)), & t \geq 0; \\ \text{NegBin}\left(r=\frac{\lambda_j(t)}{c(t)-1}, p=\frac{1}{c(t)}\right), & t<0; \end{cases}$$

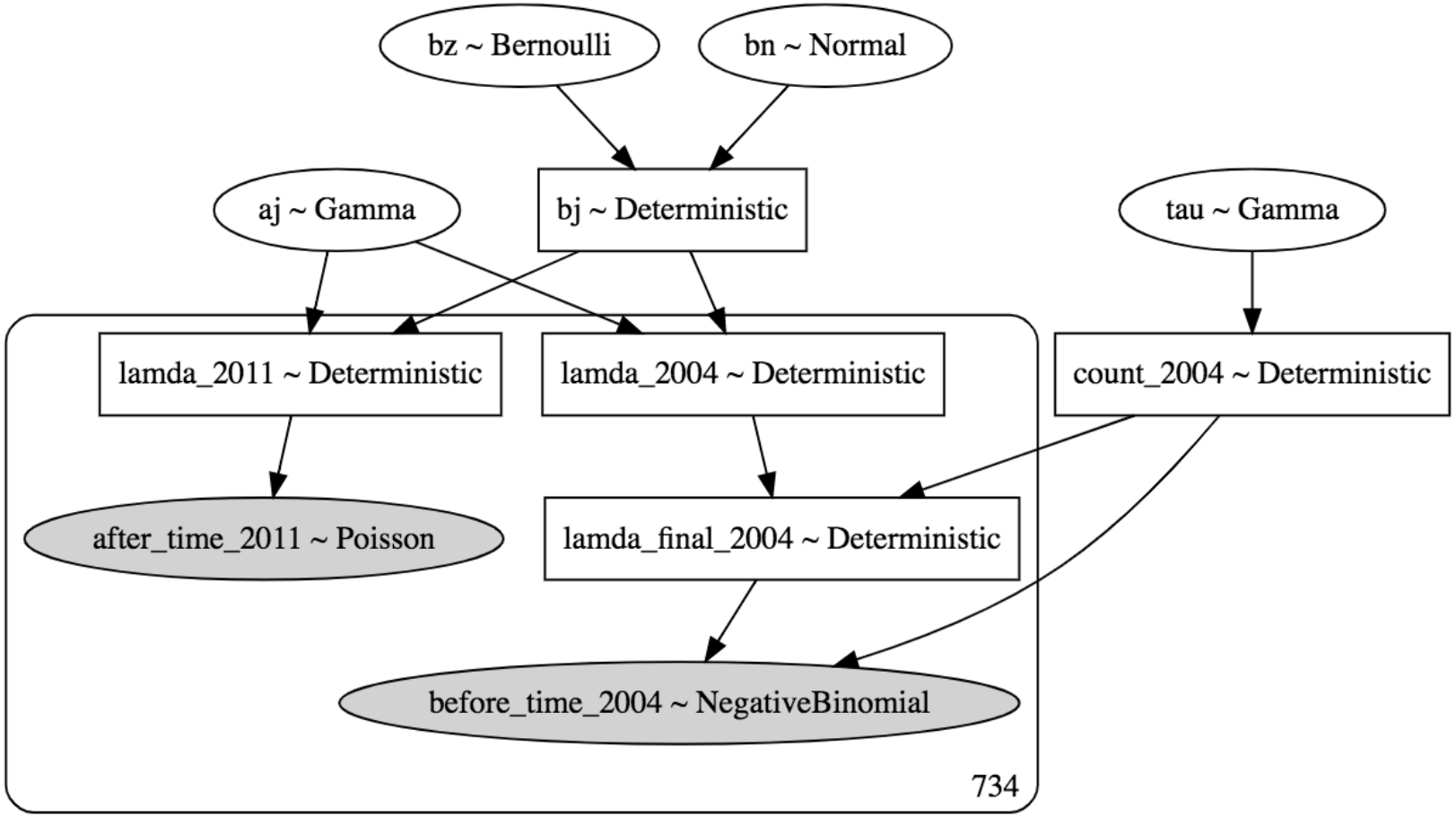
$$c(t)=\exp\{-t\tau\},\ t<0, \tau>0$$

$$\lambda_j(t)=a_j\mu_j(t)\exp\{b_j t\}, a_j>0; -\infty<b_j<\infty; t\leq 0$$

SOURCE: Own approach based on Fawcett et al. 2017

Plate notation of the model makes it easy to understand our model

Representation of our model based on Fawcett et al. 2017



SOURCE: Own approach based on Fawcett et al. 2017

As the posterior summary indicates, we have learnt RTM and local trend for each site

Posterior summary of our model based on Fawcett et al. 2017

	mean	sd	mc_error	hpd_2.5	hpd_97.5	n_eff	Rhat
bn_before	0.004656	1.027980e-01	2.594583e-03	-0.198652	0.195071	1366.438379	0.999535
bz	0.503000	4.999910e-01	6.237788e-03	0.000000	1.000000	6914.161542	0.999756
tau	0.127776	1.972335e-03	3.935674e-05	0.124210	0.131876	2149.869943	1.001179
aj_before	0.964517	4.505867e-01	2.251255e-02	0.206063	1.836473	334.584807	0.999634
bj__0	0.003606	1.474515e-16	4.336809e-20	0.003606	0.003606	1.001501	0.999500
bj__1	0.000947	2.894820e-17	3.252607e-20	0.000947	0.000947	1.001501	0.999500
bj__2	0.023206	4.649059e-16	6.938894e-19	0.023206	0.023206	1.001501	0.999500
bj__3	0.004288	1.387779e-16	0.000000e+00	0.004288	0.004288	1.001501	0.999500
bj__4	0.004436	2.688821e-17	8.673617e-20	0.004436	0.004436	1.001501	0.999500
bj__5	0.004538	2.688821e-17	8.673617e-20	0.004538	0.004538	1.001501	0.999500

SOURCE: Own approach based on Fawcett et al. 2017

For accident predictions we need to use learnt RTM, local trend and global trend along with mean accidents


```
def get_accidents(time, site, index):

    RTM = posterior_summary["mean"]["aj__{}".format(index - 1)]

    local_trend = posterior_summary["mean"]["bj__{}".format(index - 1)]

    mean_accident = get_mu_for_site(time, site)

    accident = RTM * mean_accident * np.exp(time * local_trend)

    return accident
```

SOURCE: Own approach based on Fawcett et al. 2017

According to posterior results, the model we built is close to Fawcett et al. 2017

Comparison of posterior results

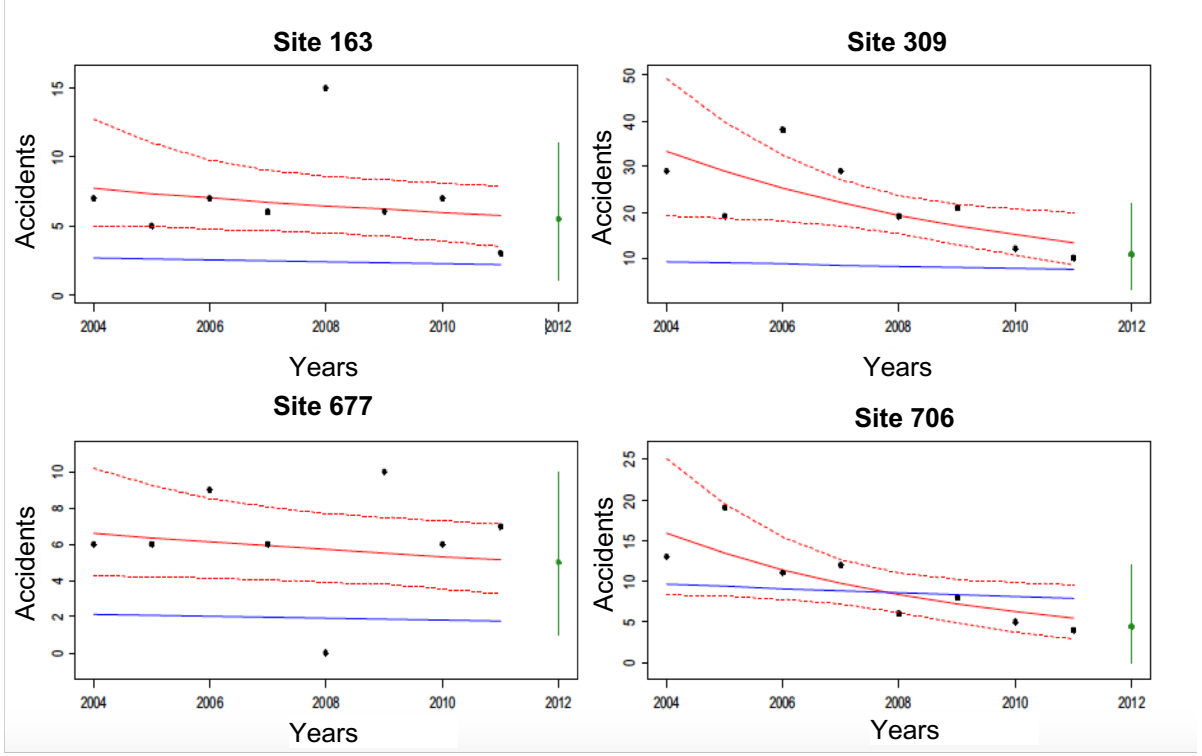
site	a_j	b_j	2008			2011			2012
			Observed	$\mu_j(t=-4)$ (APM)	$\lambda_j(t=-4)$	Observed	$\mu_j(t=0)$ (APM)	$\lambda_j(t-0)$	Prediction $(t-1)$
163	2.62 (1.43, 3.64)	−0.01 (−0.13, 0.02)	15	2.37	6.44 (4.56, 8.57)	3	2.17	5.72 (3.46, 7.87)	5.50 (1, 11)
309	1.61 (0.94, 2.61)	−0.10 (−0.20, 0.00)	19	8.23	19.40 (15.49, 23.60)	10	7.54	13.30 (8.55, 19.69)	10.93 (3, 22)
677	2.96 (1.80, 4.16)	−0.01 (−0.10, 0.05)	0	1.90	5.71 (3.95, 7.73)	7	1.75	5.18 (3.32, 7.17)	5.07 (1, 10)
706	0.63 (0.29, 1.20)	−0.12 (−0.25, 0.00)	6	8.61	8.39 (6.05, 11.03)	4	7.90	5.50 (2.89, 9.44)	4.42 (0, 11)

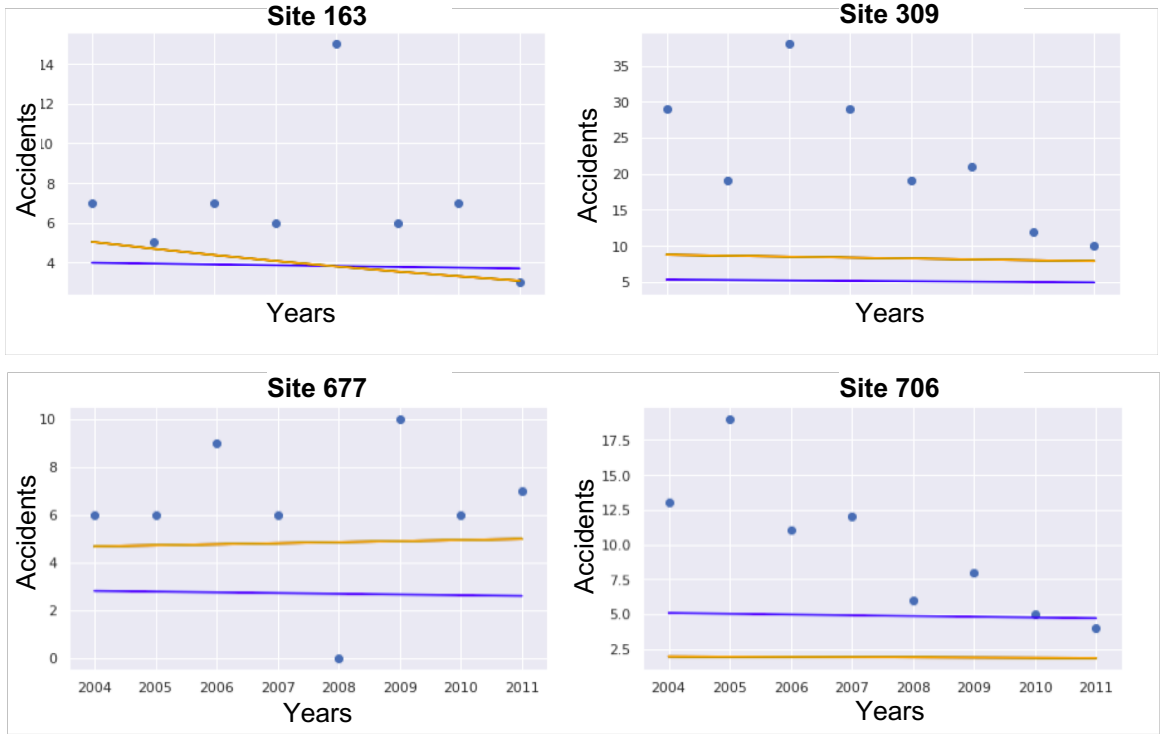
site_id	aj	bj	observed_2008	mj_2008	lamda_2008	mj_2011	observed_2011	lamda_2011	prediction_2012
162	0.832686	-0.059461	15.0	3.571223	3.795883	3.690532	3.0	3.073054	2.864107
308	1.601160	-0.004202	19.0	4.760884	8.244046	4.919937	10.0	7.877606	7.759115
676	1.922416	0.020133	0.0	2.504303	4.840023	2.587968	7.0	4.975151	5.021027
705	0.393558	0.002947	6.0	4.544583	1.893249	4.696410	4.0	1.848311	1.833571

SOURCE: Own approach based on Fawcett et al. 2017, Fawcett et al. 2017

Our model has learned the trend element correctly and is fitting quite well

Comparison of (predicted) accidents at sites over years

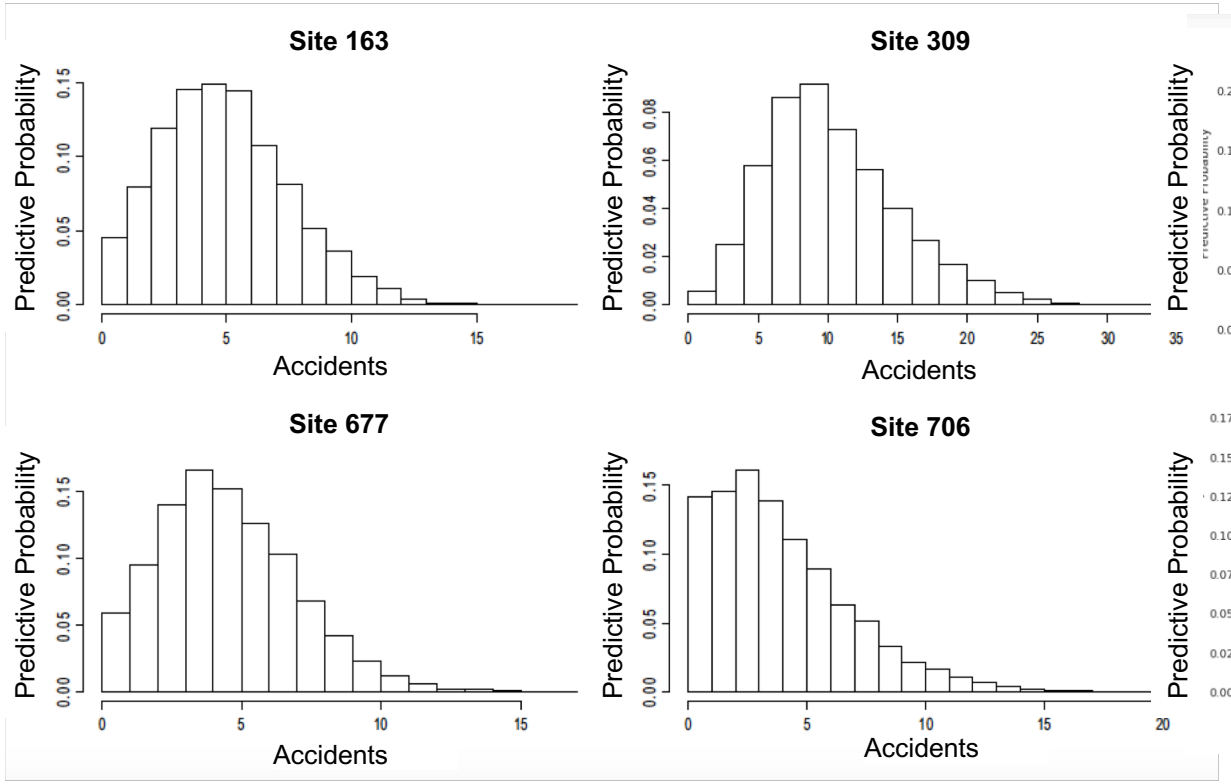


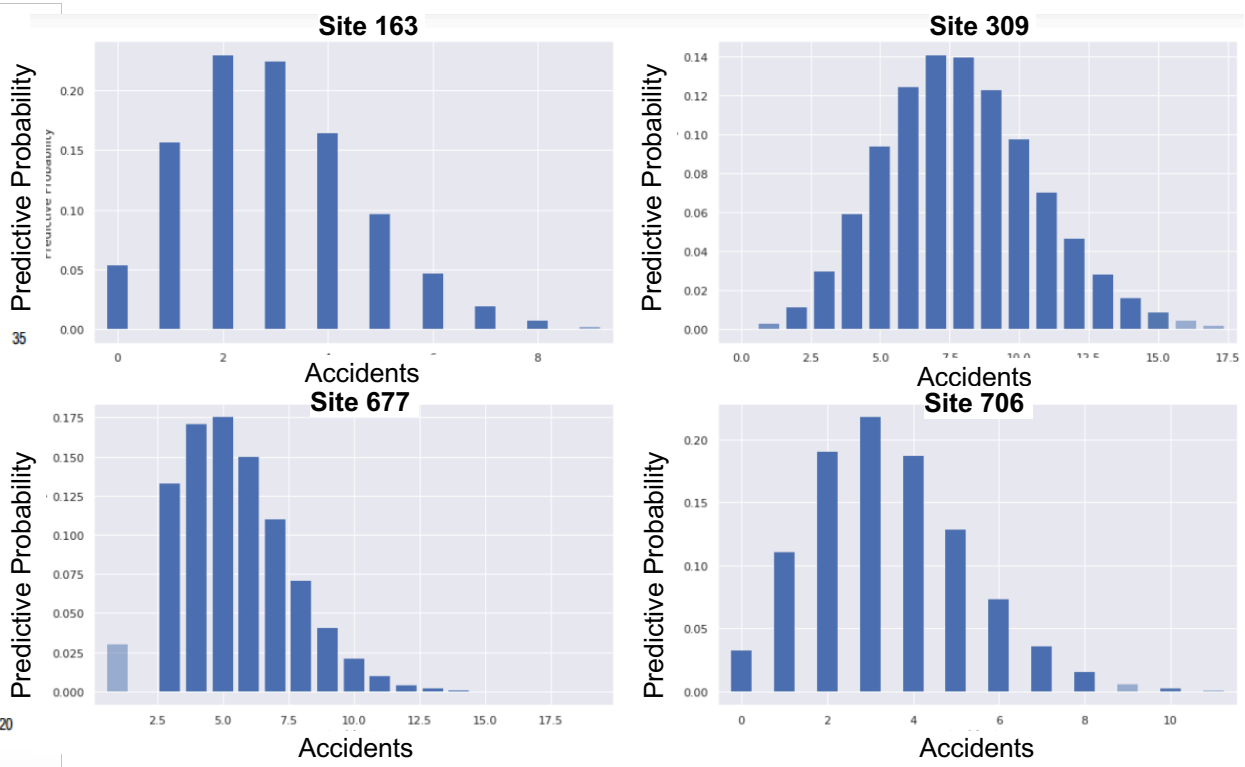


SOURCE: Own approach based on Fawcett et al. 2017, Fawcett et al. 2017

The model we built was able to generate similar posterior samples like in Fawcett et al. 2017

Comparison of posterior samples

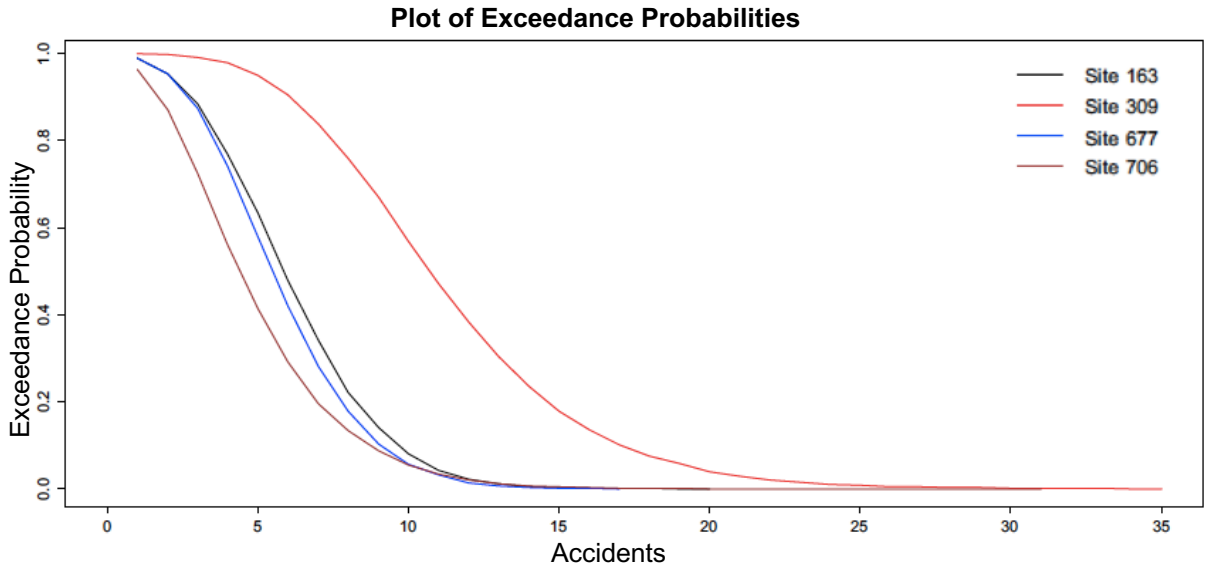


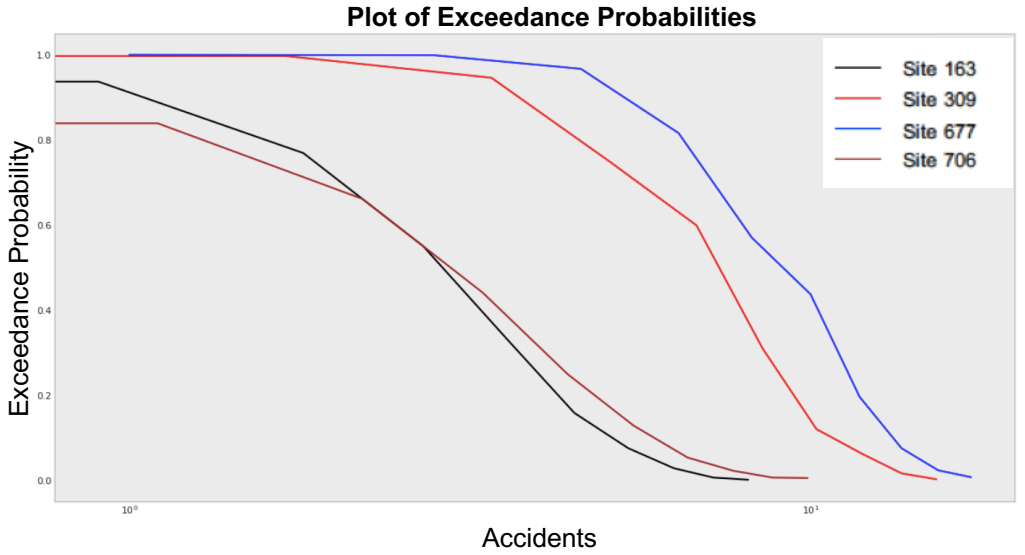


SOURCE: Own approach based on Fawcett et al. 2017, Fawcett et al. 2017

The exceedance probabilitiy graphs help us to define thresholds – again we have similar results

Comparison of exceedance probabilities





SOURCE: Own approach based on Fawcett et al. 2017, Fawcett et al. 2017

Agenda

I INTRODUCTION & RECAP

... of the overall approach used in the paper

II DESCRIPTIVE ANALYSIS

... to get a deeper understanding of the data and what the modelling aims at

III MODELLING & PREDICTING

... future road safety hotspots in the city of Halle

IV NEXT STEPS

... based on our learnings and how we can improve the model

To make our model even better, we have clearly defined next steps to be implemented

- We will train the model with **10,000 iterations**
- We will model using the recommended library by Fawcett et al.: **R-JAGS**
- We will further **evaluate the results** and adjust the model

We were able to successfully rebuild the model by Fawcett et al. – but there is still room for improvement

Q&A

We also implemented a Neural Net to predict accidents – with an “ok” result

```
labels = data_Halle["Accidents"]
```

```
type(data_Halle)
```

```
pandas.core.frame.DataFrame
```

```
data_Halle.columns
```

```
Index(['Unnamed: 0', 'Year', 'ID', 'Accidents', 'Volume', 'MajorVolume',
       'MinorVolume', 'Urban', 'Intersection', 'Signalized', 'SpeedLimit',
       'MajorRoad', 'MajorIntersection', 'FourLegs', 'Time', 'Miu'],
      dtype='object')
```

```
features = data_Halle[['Volume', 'MajorVolume',
       'MinorVolume', 'Urban', 'Intersection', 'Signalized', 'SpeedLimit',
       'MajorRoad', 'MajorIntersection', 'FourLegs', 'Time']]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
```

```
from sklearn.neural_network import MLPRegressor
clf = MLPRegressor(activation='identity', solver='adam', alpha=1e-10, random_state=1, max_iter=1000)
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(np.array(y_test), predictions)
```

```
23.058867070489782
```

SOURCE: Own model

Thank you!