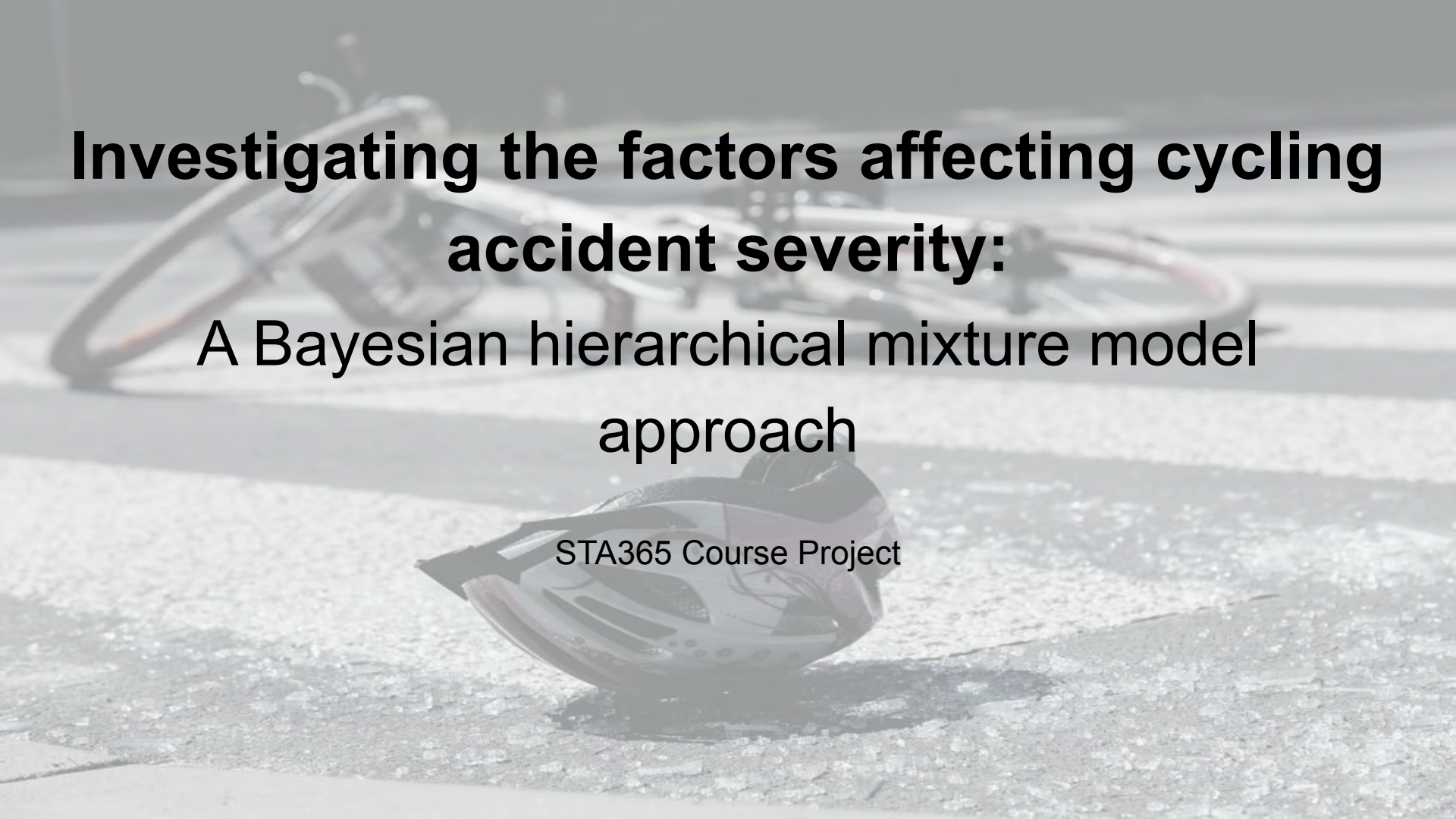


Investigating the factors affecting cycling accident severity:

A Bayesian hierarchical mixture model approach

STA365 Course Project

Introduction

Identify problem → EDA → Build model → Assess results

Cycling is increasingly popular → More people at risk of injuries

Aim to see how their severity can be reduced

Open-source data of Bicycle Accidents in Great Britain (1979 to 2018):

Analyse data patterns

Combine variables to create a Bayesian models

Compare different models using WAIC

Look at inference diagnostics

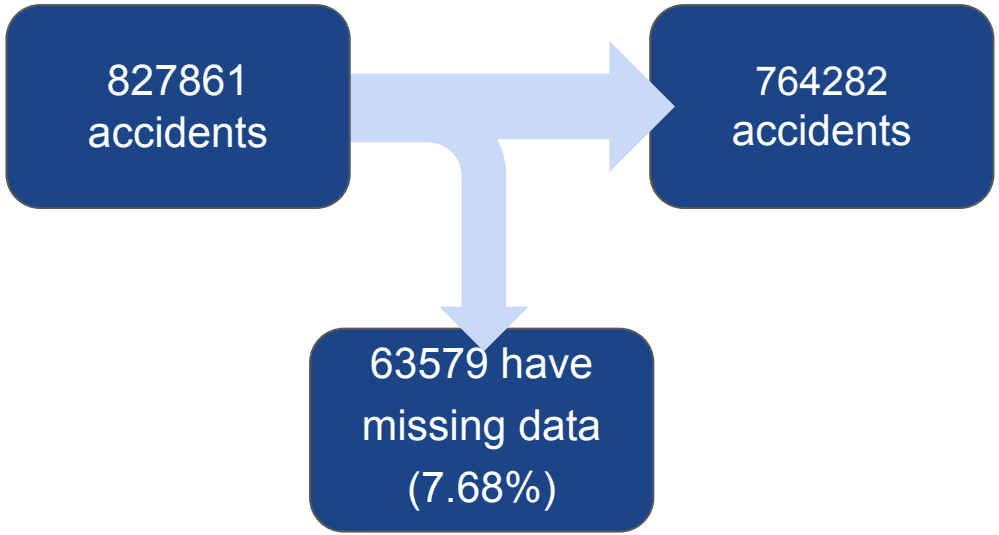
Create MCMC chains, look at convergence and ess

EDA

Data from two datasets

Accidents Dataset
Accidents Index
Date
Day
Time
Speed limit
Road type
Road conditions
Weather conditions
Light conditions

Cyclists Dataset
Accidents Index
Age Group
Gender
Severity



Merge the two datasets

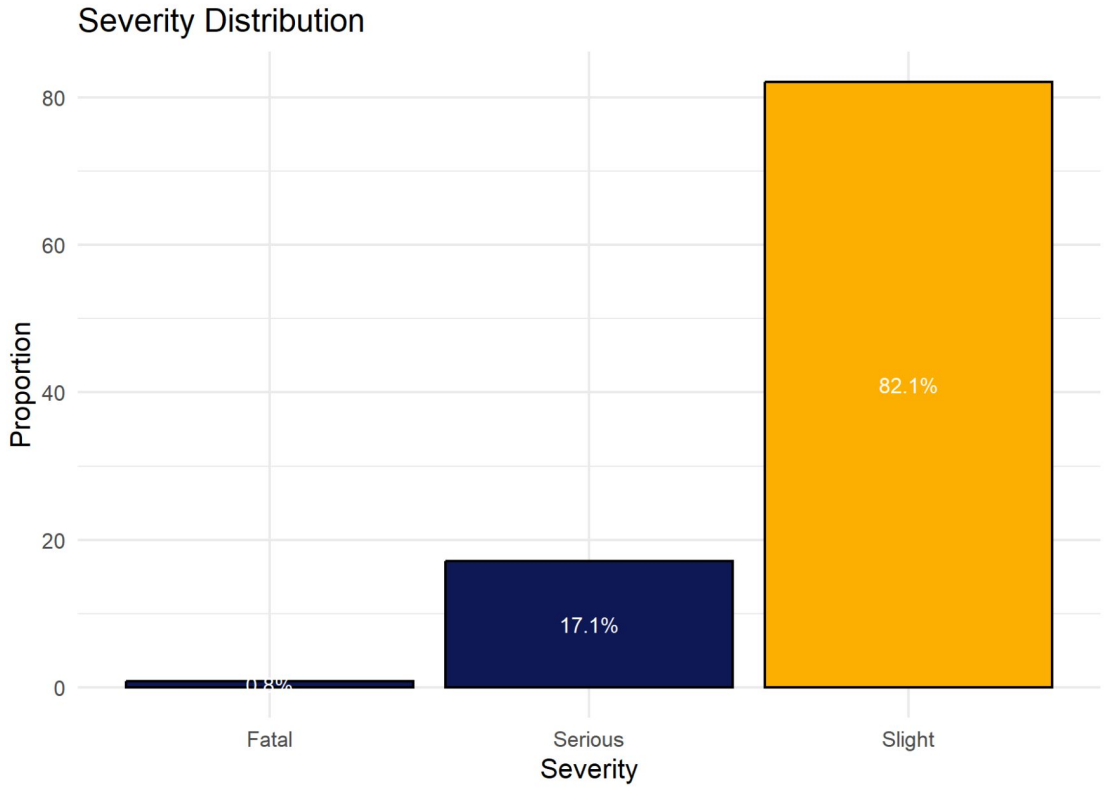
When
Date
Day
Time

Conditions
Speed limit
Road type
Road conditions
Weather conditions
Light conditions

Cyclist’s information
Age Group
Gender

Response variable
Severity

Response Variable: Severity



Binary indicator:
Fatal / Serious = 1
Slight = 0

Cyclist's characteristics in terms of severe rate

Gender	Proportion	Severe rate
Male	0.7979	18.2%
Female	0.2021	17.0%

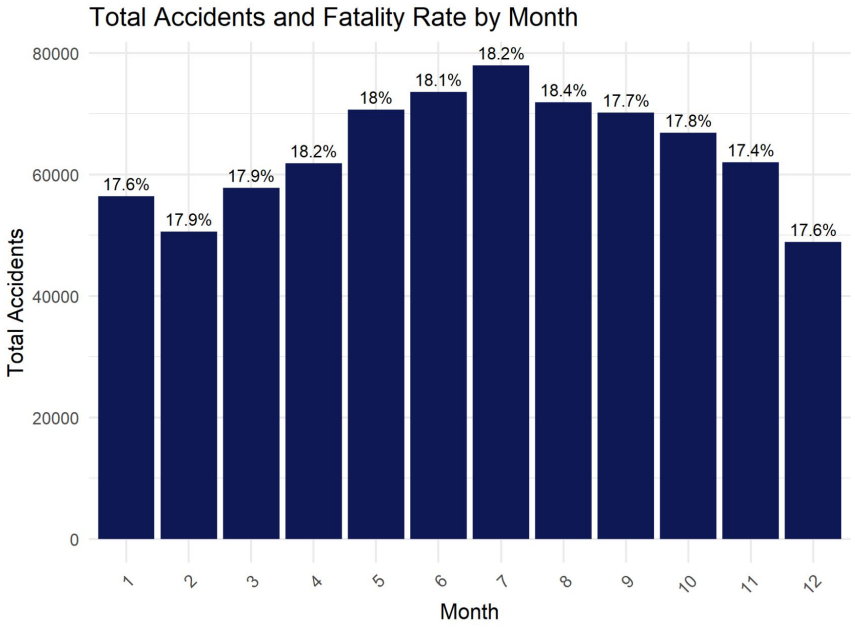
Age Group	Proportion	Severe rate
55 or below	0.9304	17.2%
Over 55	0.0695	27.1%

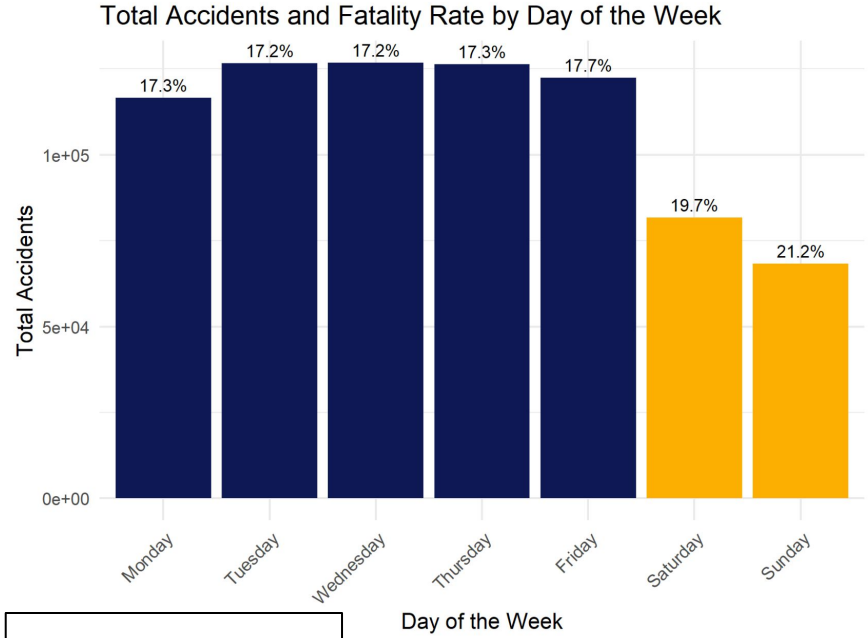
Severe rate	55 or below	Over 55
Male	17.6%	27.0%
Female	16.0%	27.4%

Subpopulations behave differently

Severe rate: proportion of accidents being serious or fatal

Month and Weekday

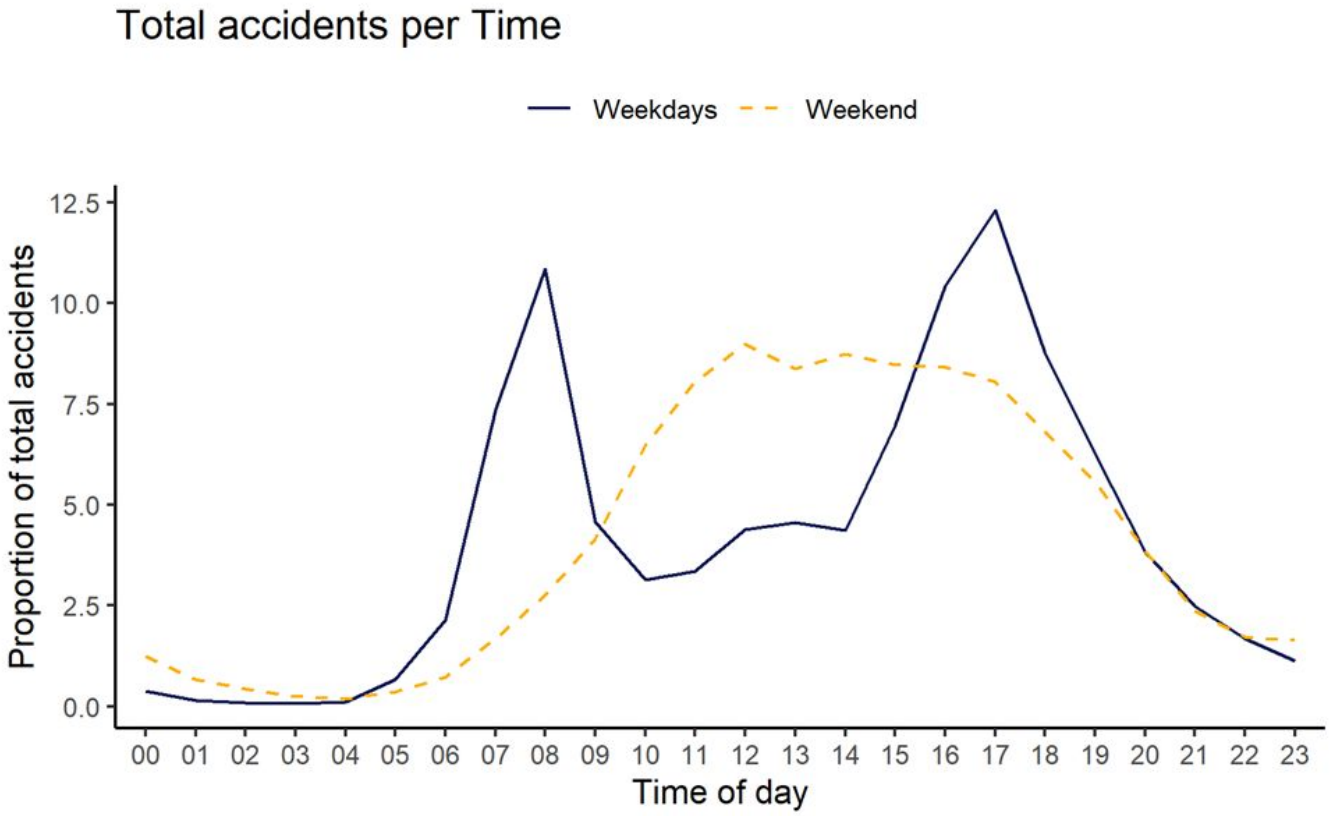




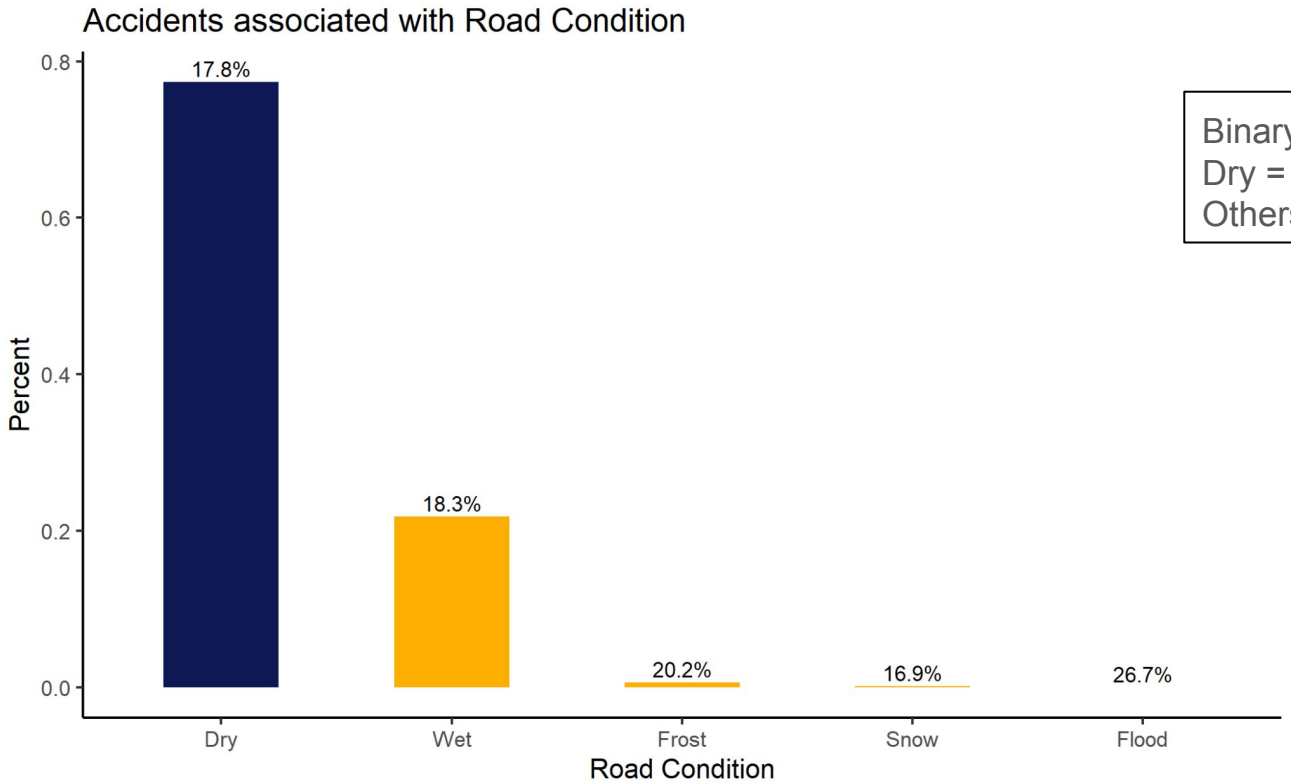
*%: proportion of accidents being serious or fatal

Binary indicator:
Weekdays = 1
Weekend = 0

Time


Total accidents per Time
Weekdays
Weekend
Proportion of total accidents
12.5
10.0
7.5
5.0
2.5
0.0
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of day

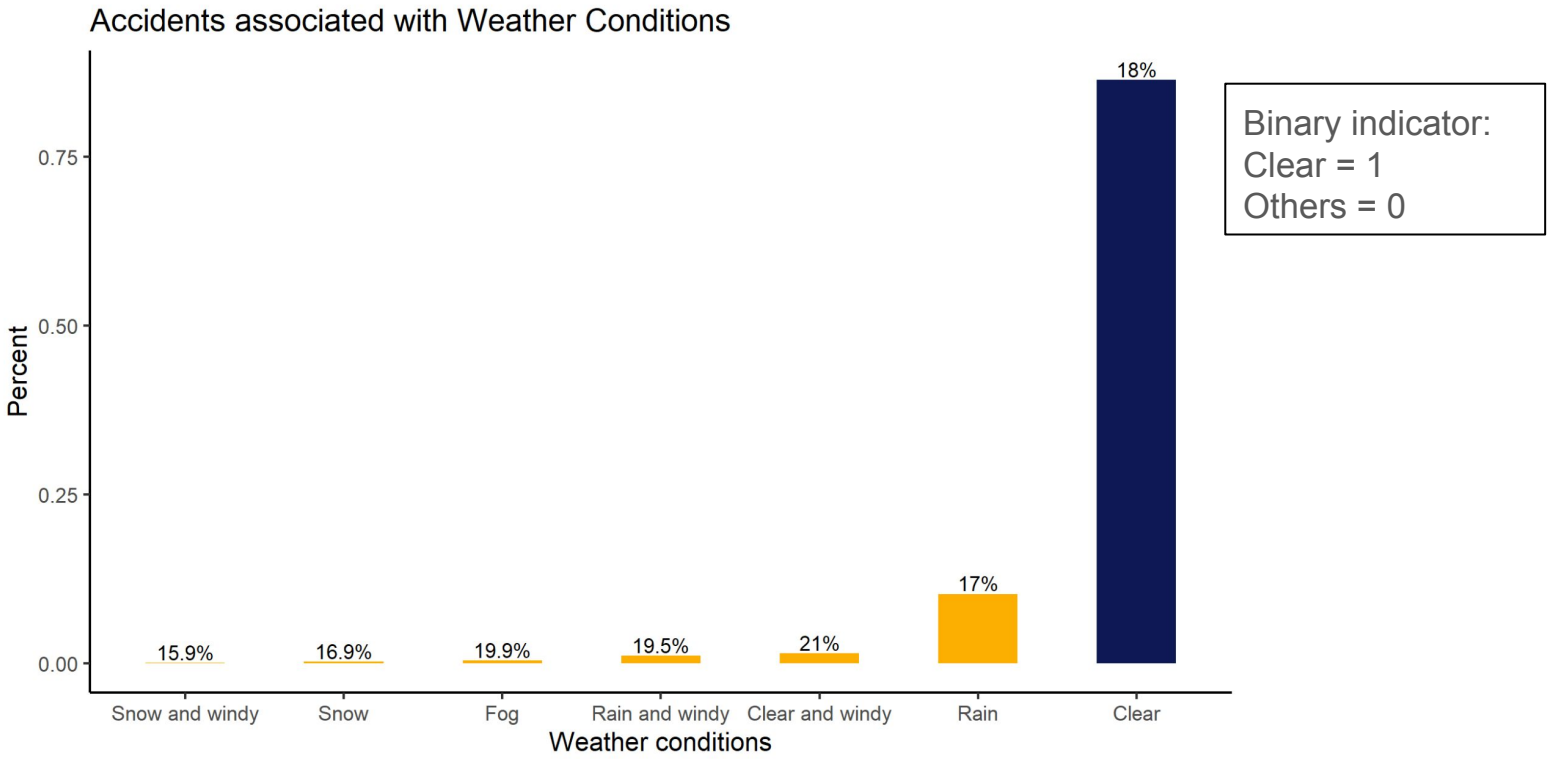
Road condition



Binary indicator:
Dry = 1
Others = 0

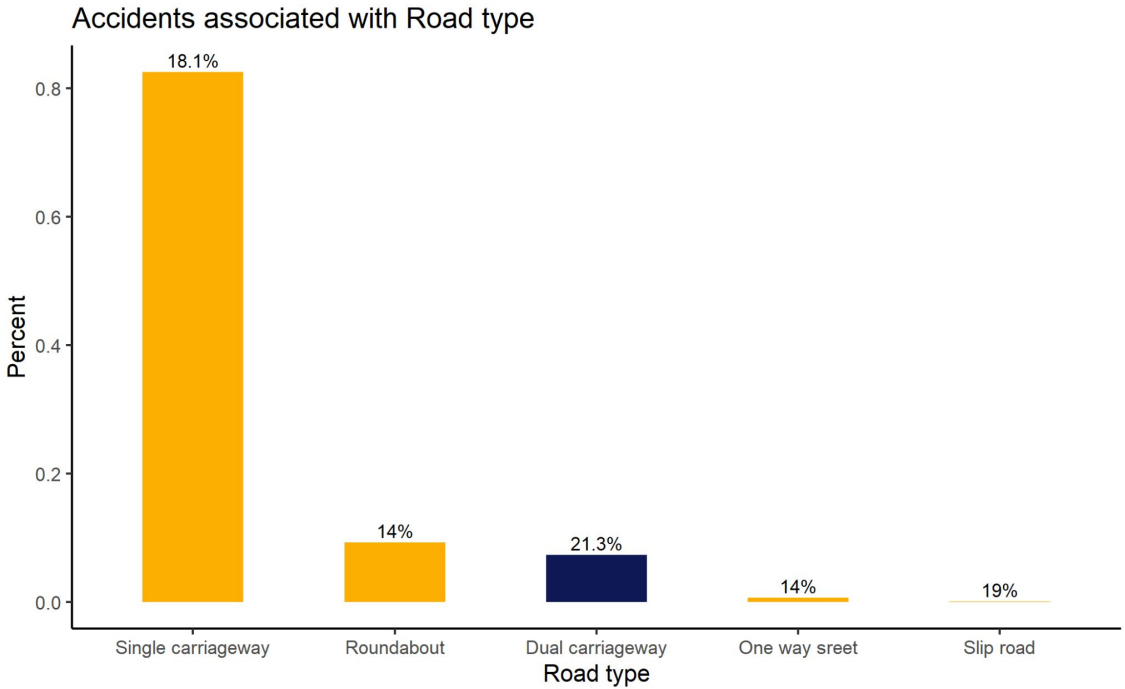
*%: proportion of accidents being serious or fatal

Weather condition



*%: proportion of accidents being serious or fatal

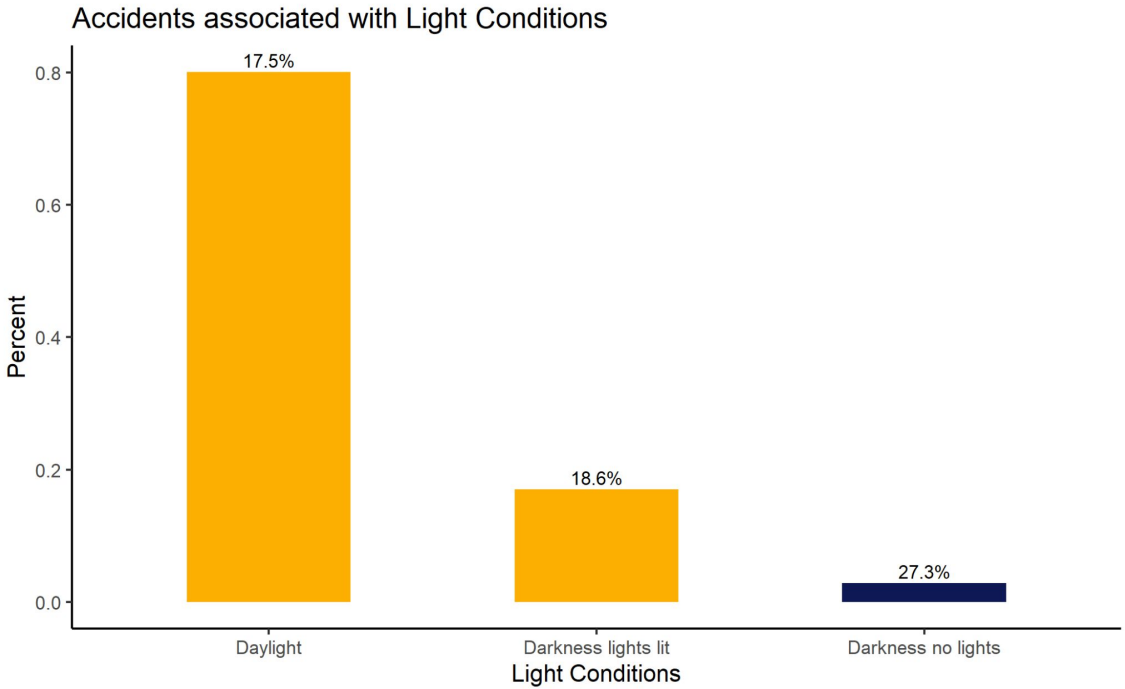
Road Type



Binary indicator:
Dual carriageway = 1
Others = 0

*%: proportion of accidents being serious or fatal

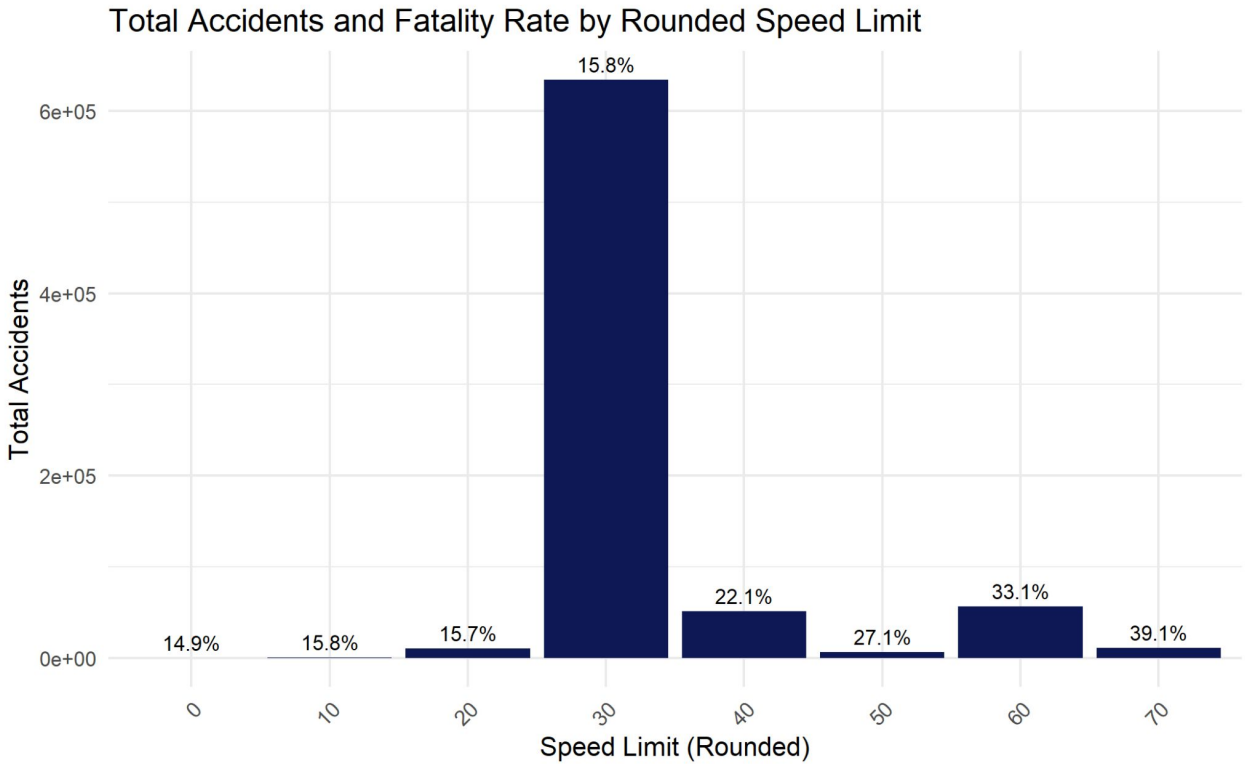
Light condition



Binary indicator:
Darkness no lights = 1
Others = 0

*%: proportion of accidents being serious or fatal

Speed limit



*%: proportion of accidents being serious or fatal

Missing Value Imputation

Model Setup

- Suppose we only have 4 variables, their joint distribution can be decomposed as follows
- $p(Speed\_limit, DryRoad, Clear, Night) = p(Speed\_limit)p(DryRoad|Speed\_limit)p(Clear|Speed\_limit, DryRoad)$ $p(Night|Speed\_limit, DryRoad, Clear)$
- Each conditional distribution can be viewed as a posterior and hence can be modelled as Bayesian.
- The Bayesian linear model!
- E.g. $p(DryRoad|Speed\_limit) \sim N(\alpha_1 + Speed\_limit\beta_1, \sigma_1^2)$ where $\alpha_1, \beta_1, \sigma_1^2$ follows some prior distributions. If x_2 is binary, then $p(DryRoad|\alpha_1 + Speed\_limit\beta_1) \sim Bern(\sigma(\alpha_1 + Speed\_limit\beta_1))$ where $\sigma(x)$ is the sigmoid function.

Bottleneck

- This specification requires the computation of too many parameters.
- In $p(Speed\_limit, DryRoad, Clear, Night) = p(Speed\_limit)p(DryRoad|Speed\_limit)p(Clear|Speed\_limit, DryRoad$ $p(Night|Speed\_limit, DryRoad, Clear)$
- , we need to compute the posterior $p(x_1)$ and 3 Bayesian linear models.
- This results in $2 + \sum_{i=2}^{n} i$ parameters if we have n features and assume normality.
- We have 10 variables in the dataset, so 56 parameters!!!

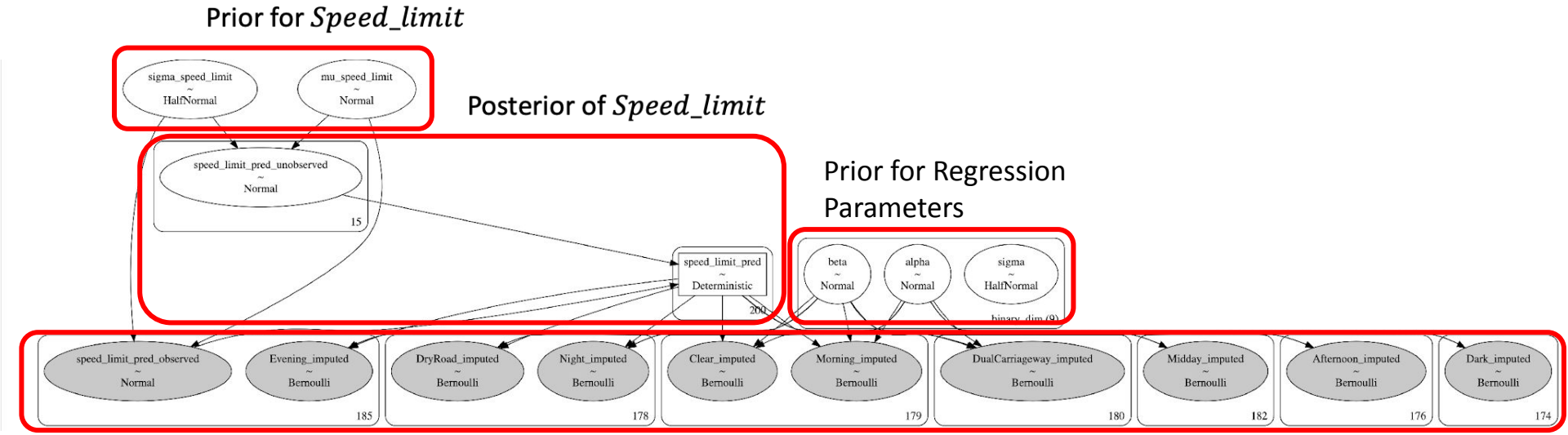
A Simplification

- If all the other features are independent given $Speed\_limit$, then
- $p(Speed\_limit, DryRoad, Clear, Night) =$
  $p(Speed\_limit)p(DryRoad|Speed\_limit)p(Clear|Speed\_limit, DryRoad)$
  $p(Night|Speed\_limit, DryRoad, Clear)$

can be simplified to

- $p(Speed\_limit, DryRoad, Clear, Night) =$
  $p(Speed\_limit)p(DryRoad|Speed\_limit)p(Clear|Speed\_limit)p(Night|Speed\_limit)$
- So we are left with much less parameters to work with!

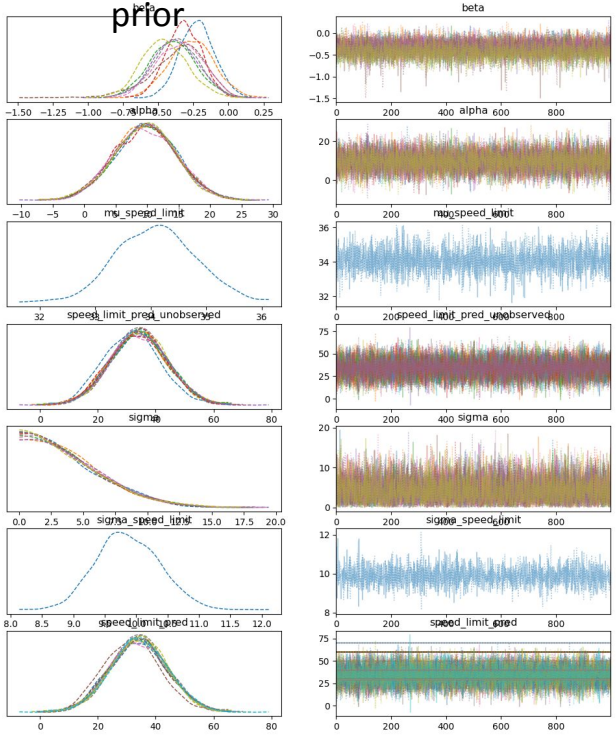
Model Implementation



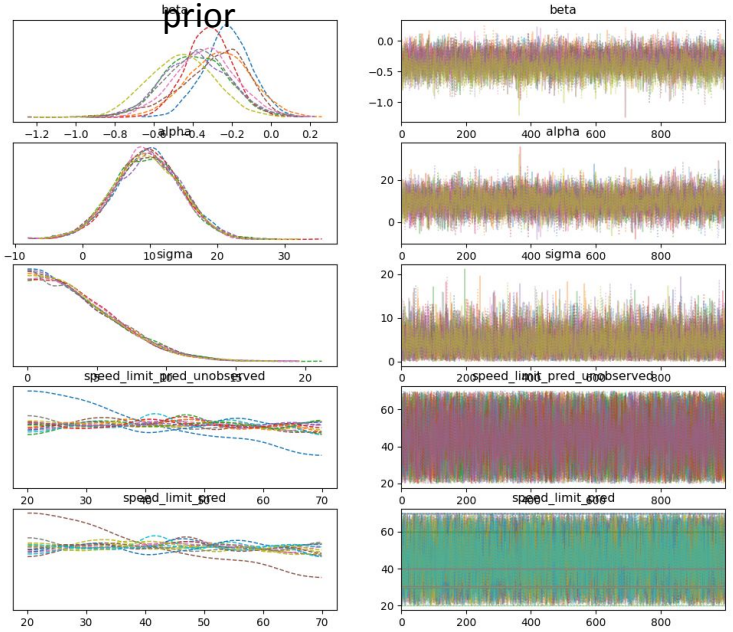
Samples from the posterior predictive of logistic regression (the imputed values)

Diagnostics

Trace plot for normal prior



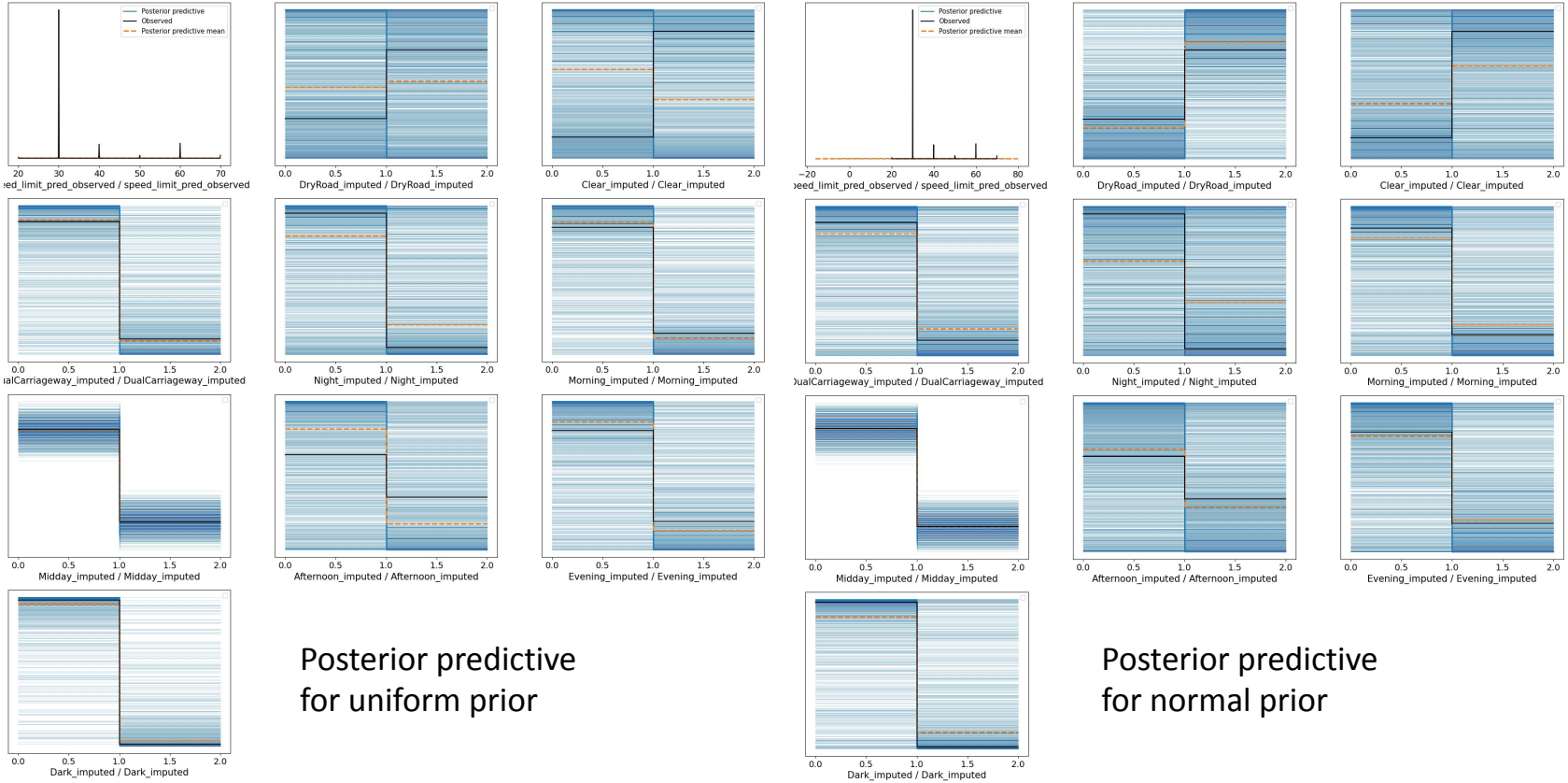
Trace plot for uniform prior



Diagnostics

Number of MCMC samples	2000	
ess_bulk	Lowest: 1484 ; Highest: 2255	All have good effective sample size
Split r_hat	All < 1.05 & extremely close to 1.0	Passed similarity test

Posterior Predictive Distributions



Posterior predictive for uniform prior

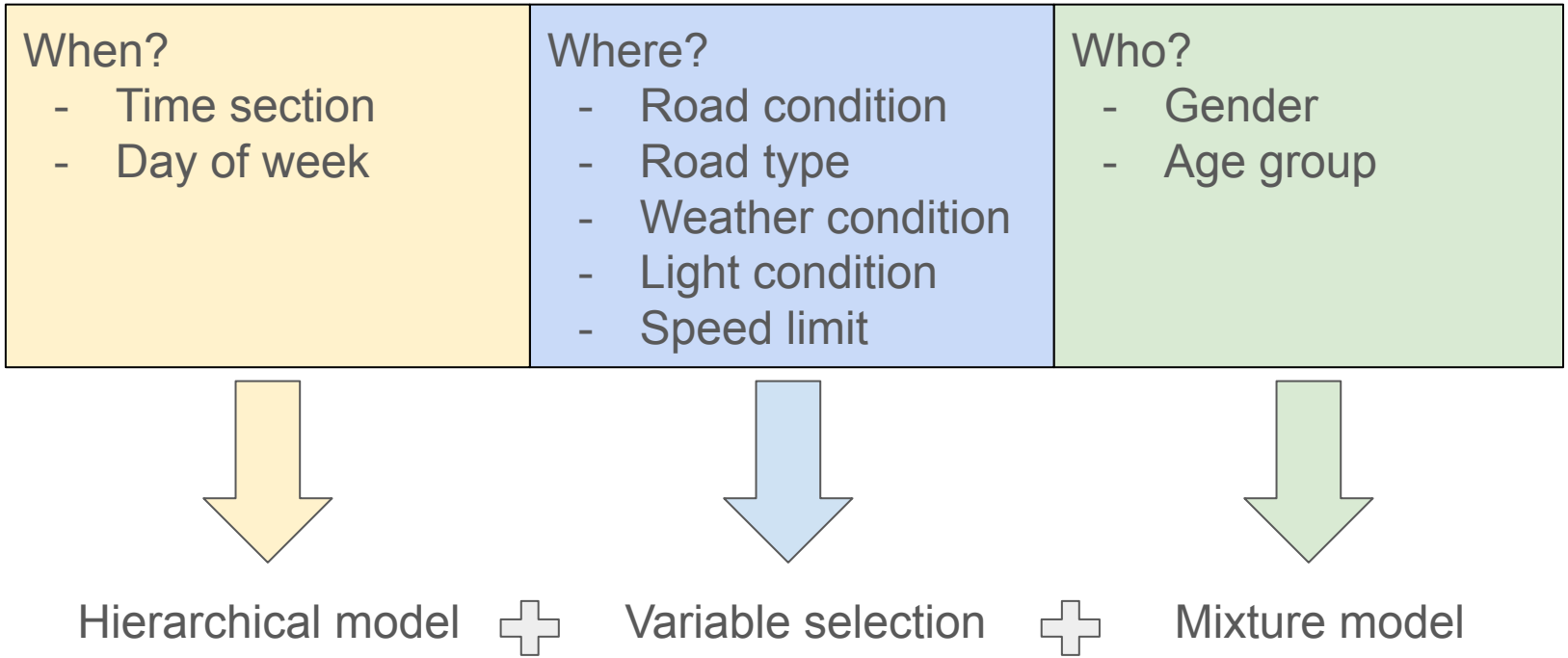
Posterior predictive for normal prior

Code to Obtain imputed values

```python
def get_imputed(idata, data):
    imputed_data = data.copy()

    imputed_speed_limit = az.extract(idata, group="posterior_predictive",
                                     num_samples=200)["speed_limit_pred"].mean(axis=1)
    mask = imputed_data["Speed_limit"].isnull()
    imputed_data.loc[mask, "Speed_limit"] = imputed_speed_limit.values[imputed_data[mask].index]

    for var in ["DryRoad", "Clear", "DualCarriageway", "Night", "Morning",
                "Midday", "Afternoon", "Evening", "Dark"]:
        imputed_var = az.extract(idata, group="posterior_predictive", num_samples=200)[f"{var}_imputed"].mean(axis=1)
        imputed_var = np.round(imputed_var, 0)
        mask = imputed_data[var].isnull()
        #imputed_data.loc[mask, var] = imputed_var.values[imputed_data[mask].index]
        imputed_data.loc[mask, var] = imputed_var[:mask.sum()]

    assert imputed_data.isnull().sum().sum() == 0, "Some variables still contain missing values after imputation."
    imputed_data.columns = ["imputed_" + col for col in imputed_data.columns]

    return imputed_data


imputed_data_normal = get_imputed(idata_normal, df4)
imputed_data_uniform = get_imputed(idata_uniform, df4)
imputed_data_normal.head(5)
```

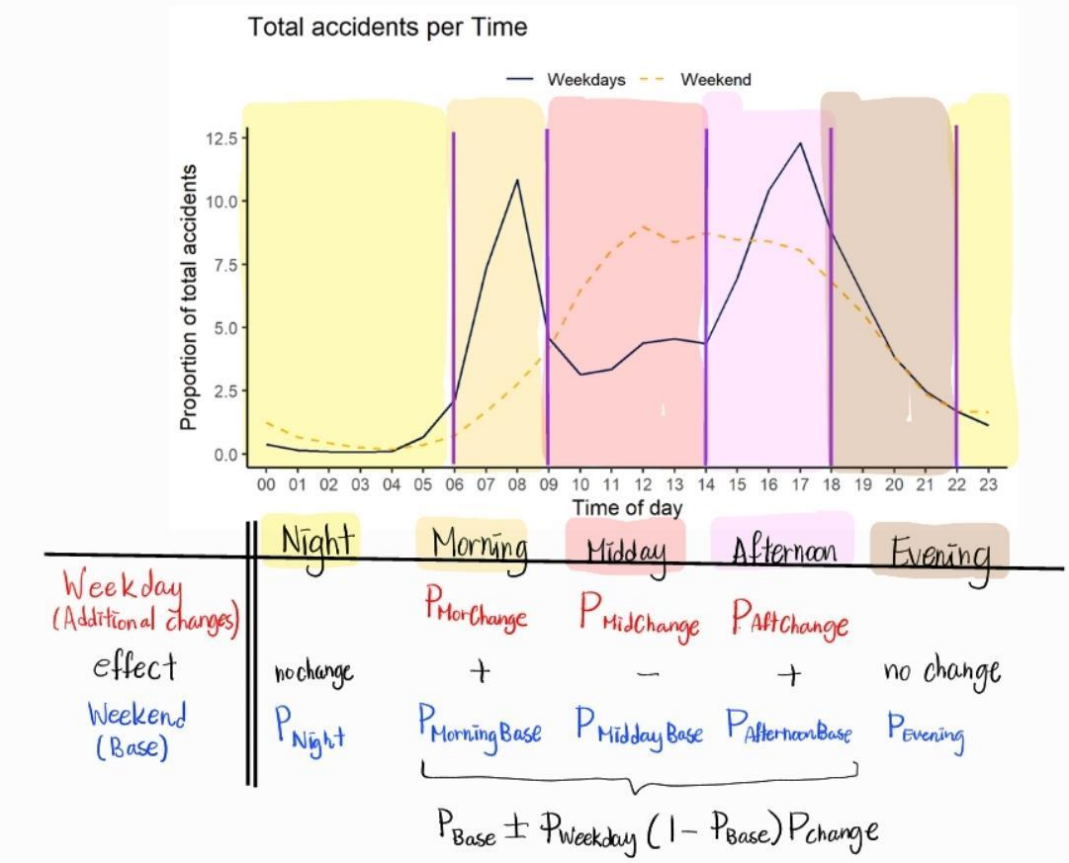
Model building

Outline


When?
- Time section
- Day of week
Where?
- Road condition
- Road type
- Weather condition
- Light condition
- Speed limit
Who?
- Gender
- Age group
Hierarchical model
Variable selection
Mixture model

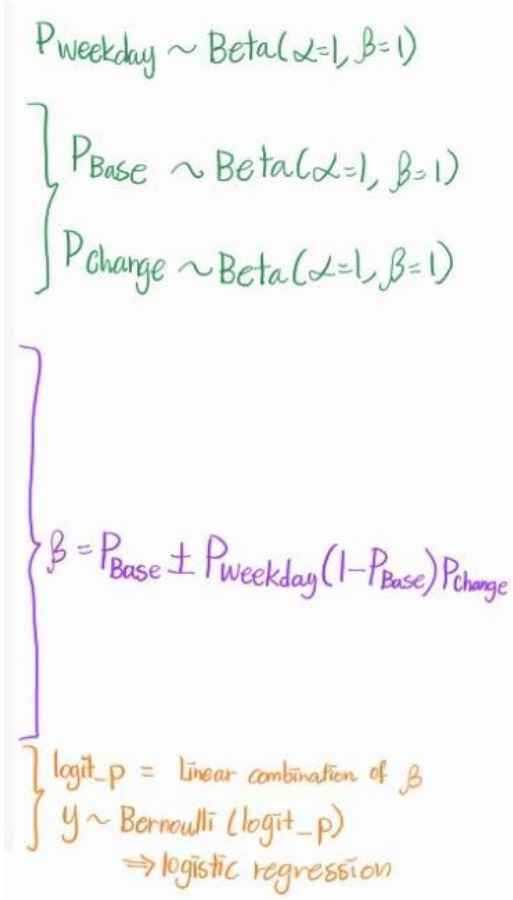
Hierarchical structure

Hierarchical structure justification

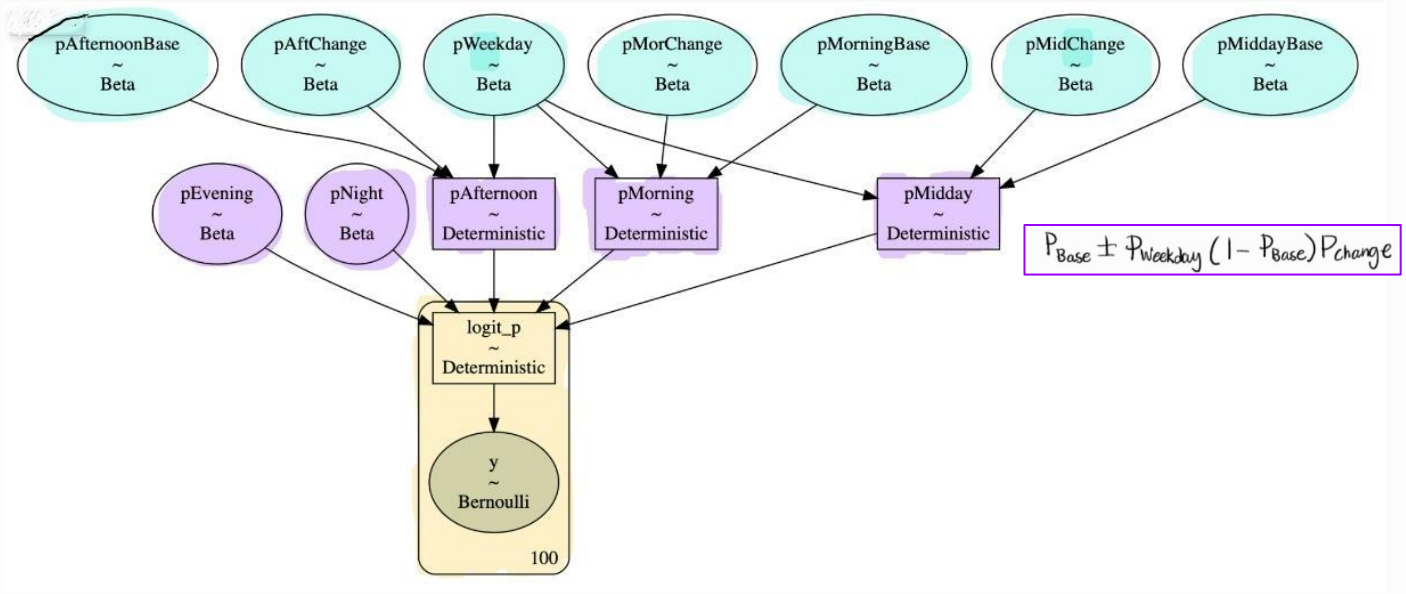


Hierarchical structure set up

```
with pm.Model() as timeday_model:
    # pWeekdayday Probability of being a weekday
    pWeekday = pm.Beta('pWeekday', alpha=1, beta=1)

    # Time weekday probabilities
    pNight = pm.Beta('pNight', alpha=1, beta=1)
    pMorningBase = pm.Beta('pMorningBase', alpha=1, beta=1)
    pMiddayBase = pm.Beta('pMiddayBase', alpha=1, beta=1)
    pAfternoonBase = pm.Beta('pAfternoonBase', alpha=1, beta=1)
    pEvening = pm.Beta('pEvening', alpha=1, beta=1)

    # Time weekend probabilities
    pMorChange = pm.Beta('pMorChange', alpha=1, beta=1)
    pMidChange = pm.Beta('pMidChange', alpha=1, beta=1)
    pAftChange = pm.Beta('pAftChange', alpha=1, beta=1)

    # pMorningBase + pWeekday*(1-pMorningBase)*pMorChange
    no_negatives_2 = pm.math.switch(pm.math.lt(pMorningBase + pWeekday * (1 - pMorningBase) * pMorChange, 0), 0,
                                    pMorningBase + pWeekday * (1 - pMorningBase) * pMorChange)
    in_unit_interval_2 = pm.math.switch(pm.math.gt(no_negatives_2, 1), 1,
                                        pMorningBase + pWeekday * (1 - pMorningBase) * pMorChange)
    pMorning = pm.Deterministic('pMorning', in_unit_interval_2)

    # pMiddayBase - pWeekday*(1-pMiddayBase)*pMidChange
    no_negatives_3 = pm.math.switch(pm.math.lt(pMiddayBase - pWeekday * (1 - pMiddayBase) * pMidChange, 0), 0,
                                    pMiddayBase - pWeekday * (1 - pMiddayBase) * pMidChange)
    in_unit_interval_3 = pm.math.switch(pm.math.gt(no_negatives_3, 1), 1,
                                        pMiddayBase - pWeekday * (1 - pMiddayBase) * pMidChange)
    pMidday = pm.Deterministic('pMidday', in_unit_interval_3)

    # pAfternoonBase + pWeekday*(1-pAfternoonBase)*pAftChange
    no_negatives_4 = pm.math.switch(pm.math.lt(pAfternoonBase + pWeekday * (1 - pAfternoonBase) * pAftChange, 0), 0,
                                    pAfternoonBase + pWeekday * (1 - pAfternoonBase) * pAftChange)
    in_unit_interval_4 = pm.math.switch(pm.math.gt(no_negatives_4, 1), 1,
                                        pAfternoonBase + pWeekday * (1 - pAfternoonBase) * pAftChange)
    pAfternoon = pm.Deterministic('pAfternoon', in_unit_interval_4)

    # Linear combination of predictors
    logit_p = pNight * x6 + pMorning * x7 + pMidday * x8 + pAfternoon * x9 + pEvening * x10
    logit_p = pm.Deterministic('logit_p', logit_p)

    y = pm.Bernoulli('y', logit_p=logit_p, observed=y)
```



Visualization of structure



$P_{Base} \pm P_{Weekday}(1 - P_{Base})P_{change}$

Base & Change priors

Combined priors of β

Linear combination of β

Outcome logistic regression model

Variable selection

Variable selection set up

```python
predictors = ['DryRoad', 'Dark', 'DualCarriageway', 'Clear','Speed_limit']

# Get predictors and target variable as numpy arrays
X = cleandata[predictors].values[-100:]
y = cleandata['Severity_binary'].values[-100:]

# Number of predictors
n_predictors = X.shape[1]

# Define Bayesian binary regression model with spike-and-slab priors for variable selection
with pm.Model() as model:
    # Coefficients for predictors
    slab = pm.Normal('slab', mu=0, sigma=1, shape=n_predictors)

    # Indicator variables for spike-and-slab
    spike = pm.Bernoulli('spike', p=0.5, shape=n_predictors)

    # Create deterministic variable for selected coefficients
    beta = pm.Deterministic('beta', slab * spike)

    # Logistic regression model
    p = pm.math.invlogit(pm.math.dot(X, beta))
    y_obs = pm.Bernoulli('y_obs', p=p, observed=y)
    idata = pm.sample()

    # Sample from the posterior
    trace = pm.sample(2000, tune=1000, return_inferencedata=False)

# Extract selected predictors based on posterior samples of gamma
selected_predictors = np.array(predictors)[np.mean(trace['spike'], axis=0) > 0.5]
print("Selected predictors:", selected_predictors)
```

Condition predictors:
- Road Condition
- Light Condition
- Road Type
- Weather Condition
- Speed limit

} binary

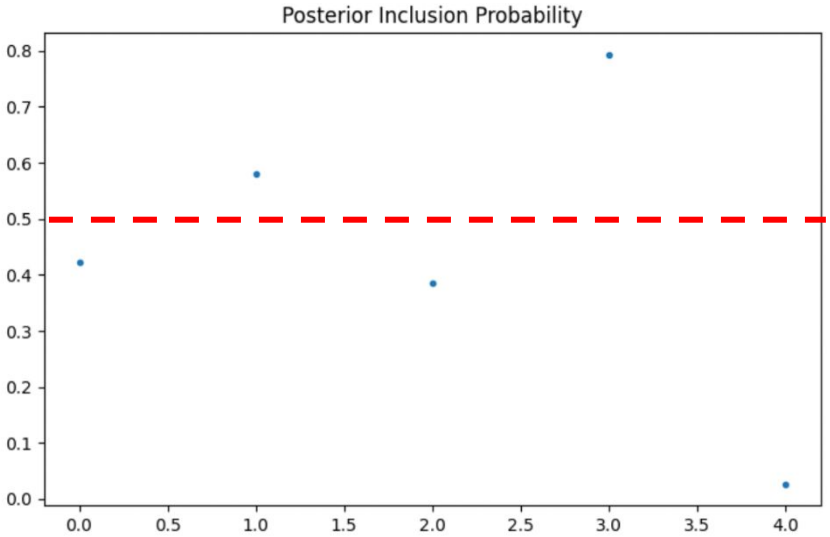
Slab ~ Normal (0,1)
Spike ~ Bernoulli (0.5)

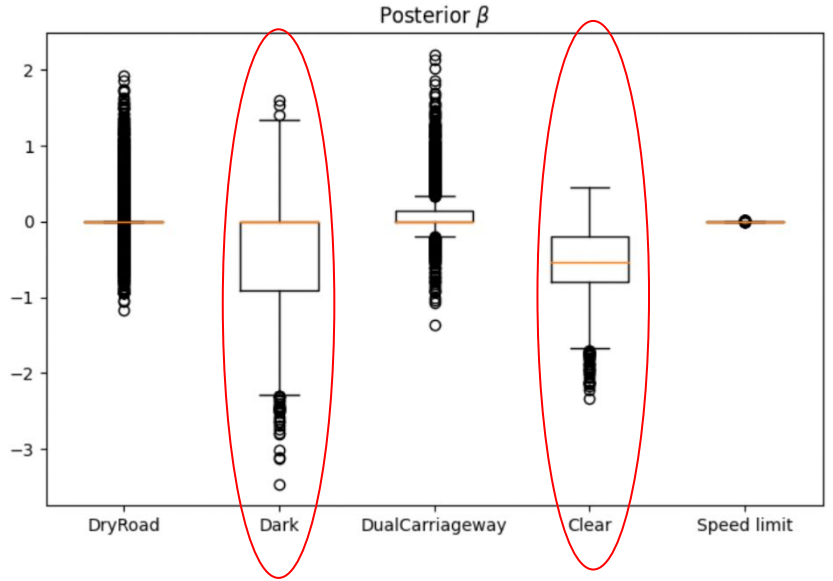
} for each predictor

β = Slab • Spike

logit_p = Linear combination of β
y ~ Bernoulli (logit_p)
⇒ logistic regression

Spike and slab posterior inferences results





Selected condition variables: Light condition, Weather condition

Complete model

Hierarchical Time/Day structure + selected variables

```
with pm.Model() as complete_model:

    # pWeekday Probablity of being a weekday
    pWeekday = pm.Beta('pWeekday', alpha=1, beta=1)

    # Time weekday probabilities
    pNight= pm.Beta('pNight', alpha=1, beta=1) ## = p1
    pMorningBase= pm.Beta('pMorningBase', alpha=1, beta=1)
    pMiddayBase= pm.Beta('pMiddayBase', alpha=1, beta=1)
    pAfternoonBase= pm.Beta('pAfternoonBase', alpha=1, beta=1)
    pEvening= pm.Beta('pEvening', alpha=1, beta=1) ## = p5
    # Time weekend probabilities
    pMorChange= pm.Beta('pMorChange', alpha=1, beta=1)
    pMidChange= pm.Beta('pMidChange', alpha=1, beta=1)
    pAftChange= pm.Beta('pAftChange', alpha=1, beta=1)

    # pMorningBase + pWeekday*(1-pMorningBase)*pMorChange
    no_negatives_2 = pm.math.switch(pm.math.lt(pMorningBase + pWeekday*(1-pMorningBase)*pMorChange,0), 0, pMorningBase + pWeekday*(1-pMorningBase)*pMorChange)
    in_unit_interval_2 = pm.math.switch(pm.math.gt(no_negatives_2,1), 1, pMorningBase + pWeekday*(1-pMorningBase)*pMorChange)
    pMorning = pm.Deterministic('pMorning', in_unit_interval_2)

    # pMiddayBase - pWeekday*(1-pMiddayBase)*pMidChange
    no_negatives_3 = pm.math.switch(pm.math.lt(pMiddayBase - pWeekday*(1-pMiddayBase)*pMidChange,0), 0, pMiddayBase - pWeekday*(1-pMiddayBase)*pMidChange)
    in_unit_interval_3 = pm.math.switch(pm.math.gt(no_negatives_3,1), 1, pMiddayBase - pWeekday*(1-pMiddayBase)*pMidChange)
    pMidday = pm.Deterministic('pMidday', in_unit_interval_3)

    # pAfternoonBase + pWeekday*(1-pAfternoonBase)*pAftChange
    no_negatives_4 = pm.math.switch(pm.math.lt(pAfternoonBase + pWeekday*(1-pAfternoonBase)*pAftChange,0), 0, pAfternoonBase + pWeekday*(1-pAfternoonBase)*pAftChange)
    in_unit_interval_4 = pm.math.switch(pm.math.gt(no_negatives_4,1), 1, pAfternoonBase + pWeekday*(1-pAfternoonBase)*pAftChange)
    pAfternoon = pm.Deterministic('pAfternoon', in_unit_interval_4)

    # Intercept
    intercept = pm.Normal('intercept', mu=0, sigma=1)

    # Priors of selected condition predictors
    beta_dark = pm.Normal('beta_dark', mu=0, sigma=1)  # Prior for 'Dark'
    beta_clear = pm.Normal('beta_clear', mu=0, sigma=1)  # Prior for 'Clear'

    # Linear combination of predictors
    logit_p = intercept + pNight*x6 +pMorning*x7 +pMidday*x8 +pAfternoon*x9 +pEvening*x10 + beta_clear*x2 + beta_dark*x4
    logit_p = pm.Deterministic('logit_p',logit_p)

    Y = pm.Bernoulli('Y', logit_p = logit_p, observed=y)

    idata_2 = pm.sample()
```

Hierarchical structure

Standard normal prior

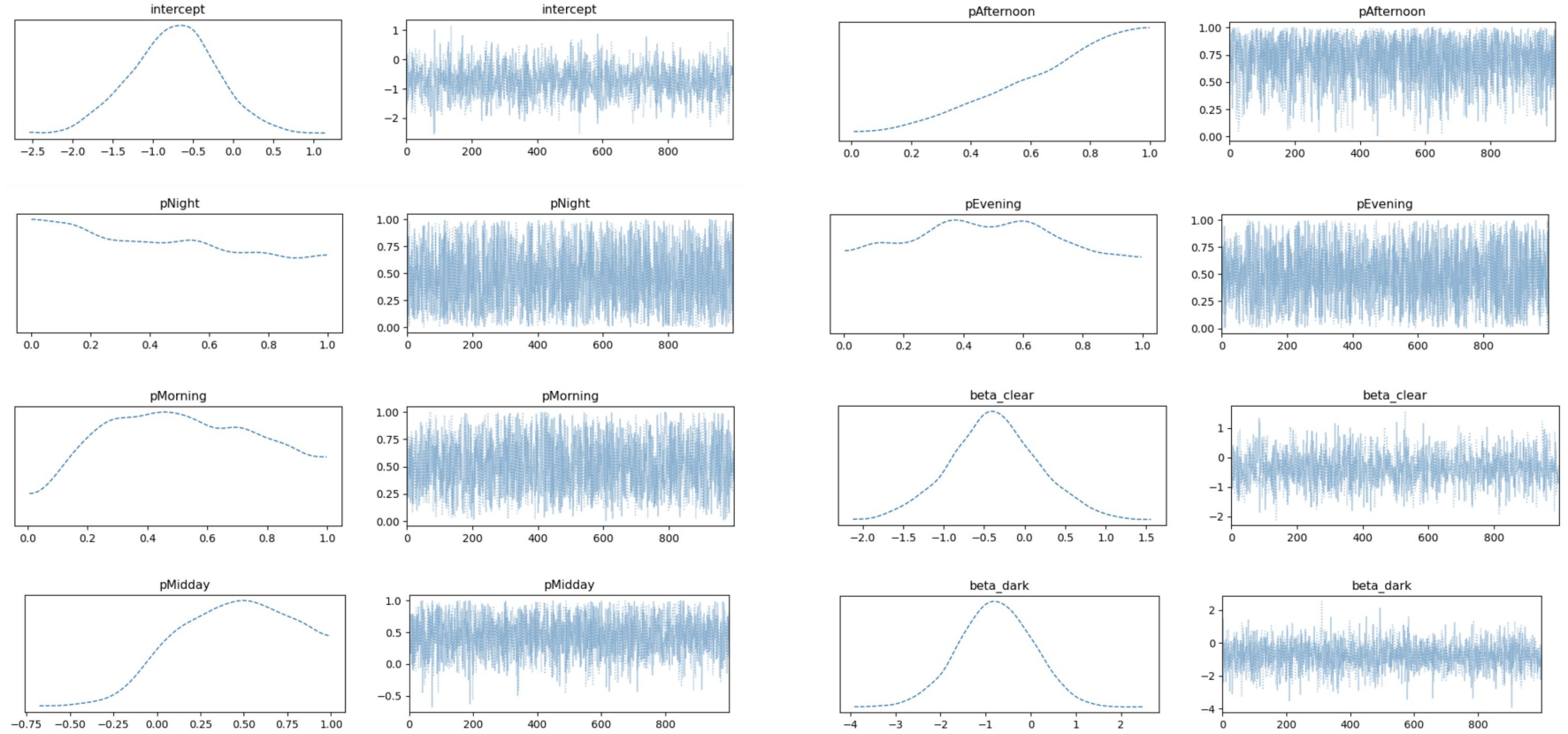
Standard normal prior

Selected condition predictors

Logistic regression model

Inference Diagnostic PART 1

Posterior distributions of betas

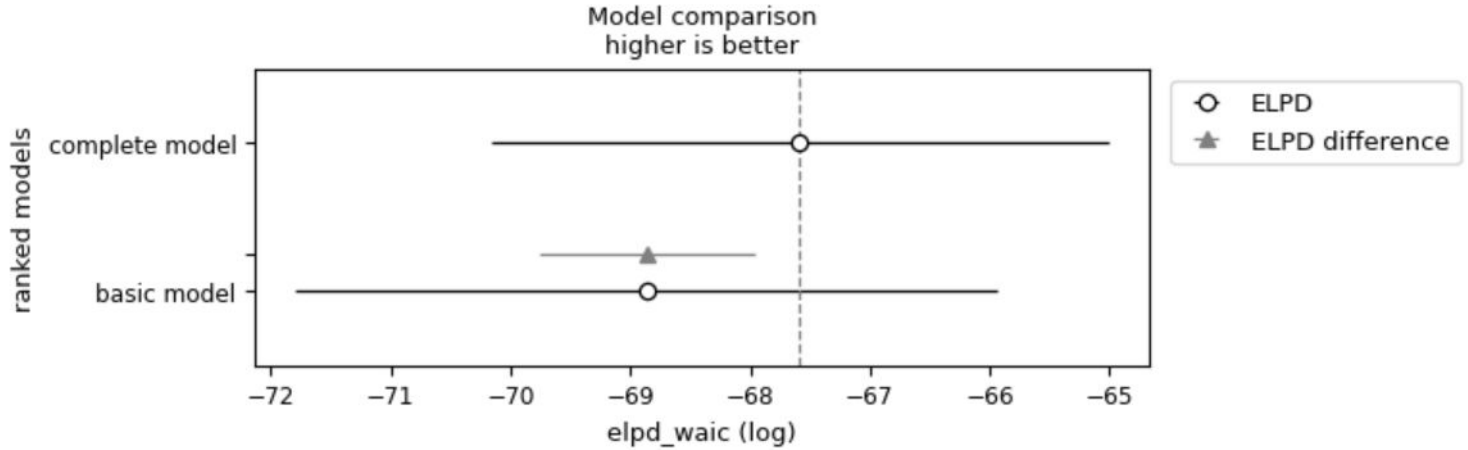


Convergence and effective sample size

Number of MCMC samples	2000	
Number of variables	115	
ess_bulk	Lowest: 1389 (~70%); Highest: 2662 (~133%)	High effective sample size
Split h_hat	All below 1.05	Passed similarity test

Modeling Diagnostics PART 1

WAIC comparison



Model	Simple logistic regression model	Our model
Description	Normal priors on all betas	• Hierarchical structure on Time/Day • Normal priors on selected condition predictors
elpd_waic	-68.830915	-67.620926

Mixture model

Mixture model

```
with pm.Model() as mixture_model:
    # Define mixture component for Gender
    alpha_gender_0 = pm.Normal('alpha_gender_0', mu=0, sigma=1)
    beta_gender_0 = pm.Normal('beta_gender_0', mu=0, sigma=1)

    alpha_gender_1 = pm.Normal('alpha_gender_1', mu=0, sigma=1)
    beta_gender_1 = pm.Normal('beta_gender_1', mu=0, sigma=1)

    # Mixture component for Gender
    mixture_gender = pm.Bernoulli('mixture_gender', p=0.5)  # Prior probability of belonging to each component

    # Define mixture component for over55
    alpha_over55_0 = pm.Normal('alpha_over55_0', mu=0, sigma=1)
    beta_over55_0 = pm.Normal('beta_over55_0', mu=0, sigma=1)

    alpha_over55_1 = pm.Normal('alpha_over55_1', mu=0, sigma=1)
    beta_over55_1 = pm.Normal('beta_over55_1', mu=0, sigma=1)

    # Mixture component for over55
    mixture_over55 = pm.Bernoulli('mixture_over55', p=0.5)  # Prior probability of belonging to each component

    # Define mixture probabilities
    mixture_prob_gender = pm.math.switch(mixture_gender, 1, 0)
    mixture_prob_over55 = pm.math.switch(mixture_over55, 1, 0)
```

Gender

over55

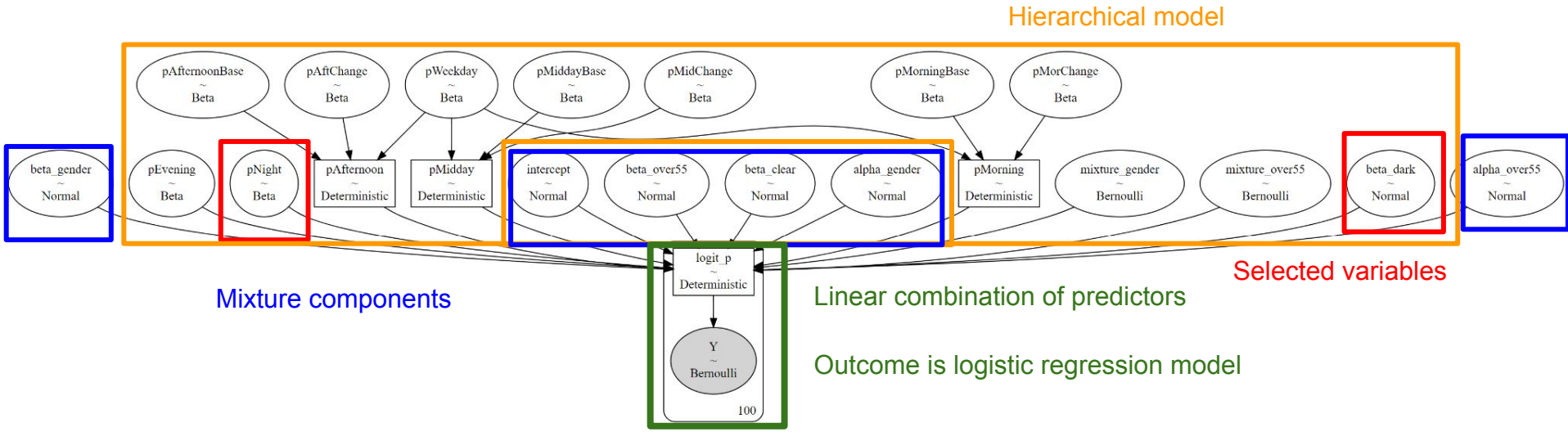
Mixture model structure

Hierarchical structure and variable selection

```
# Combine mixture components with the existing linear combination
logit_p = intercept + pNight * x6 + pMorning * x7 + pMidday * x8 + pAfternoon * x9 + pEvening * x10 + beta_clear * x2 + beta_dark * x4 \
          + mixture_prob_gender * (pm.math.switch(mixture_gender, alpha_gender_1 + beta_gender_1 * xGender, alpha_gender_0 + beta_gender_0 * xGender)) \
          + mixture_prob_over55 * (pm.math.switch(mixture_over55,alpha_over55_1 + beta_over55_1 * xover55,alpha_over55_0 + beta_over55_0 * xover55))
logit_p = pm.Deterministic('logit_p',logit_p)

# Define likelihood
Y = pm.Bernoulli('Y', logit_p=logit_p, observed=y)

# Sample from the model
idata_4 = pm.sample()
```

Logistic regression model with mixture

Visualisation of structure



Inference Diagnostic PART 2

Posterior inference of betas

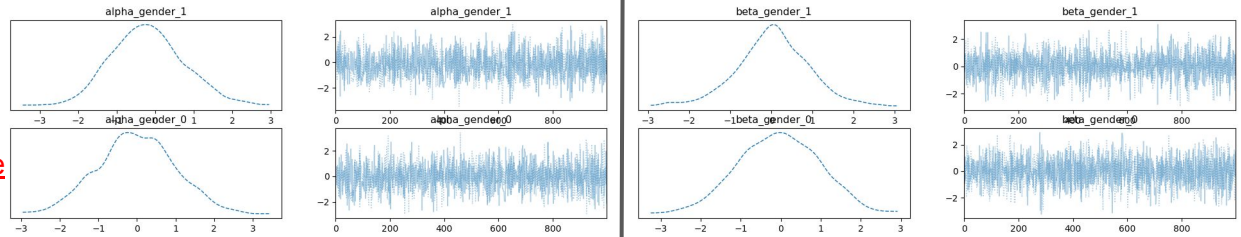
Alpha

Beta

Gender

Male

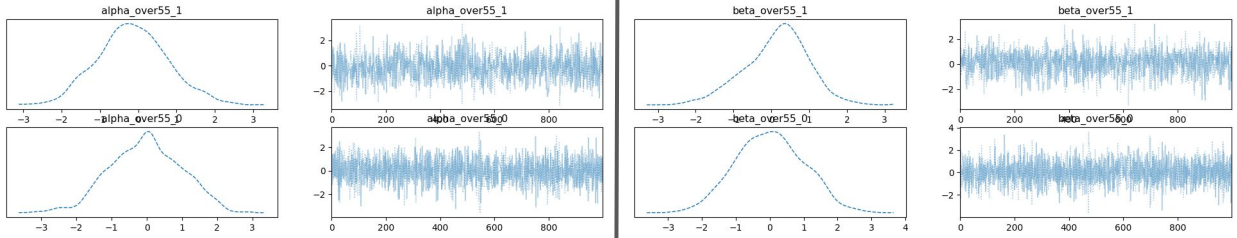
Female



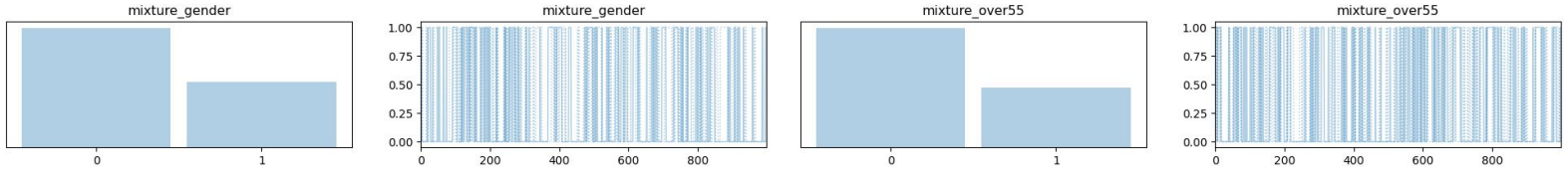
Age Group

Over 55

Under 55



Mixtures



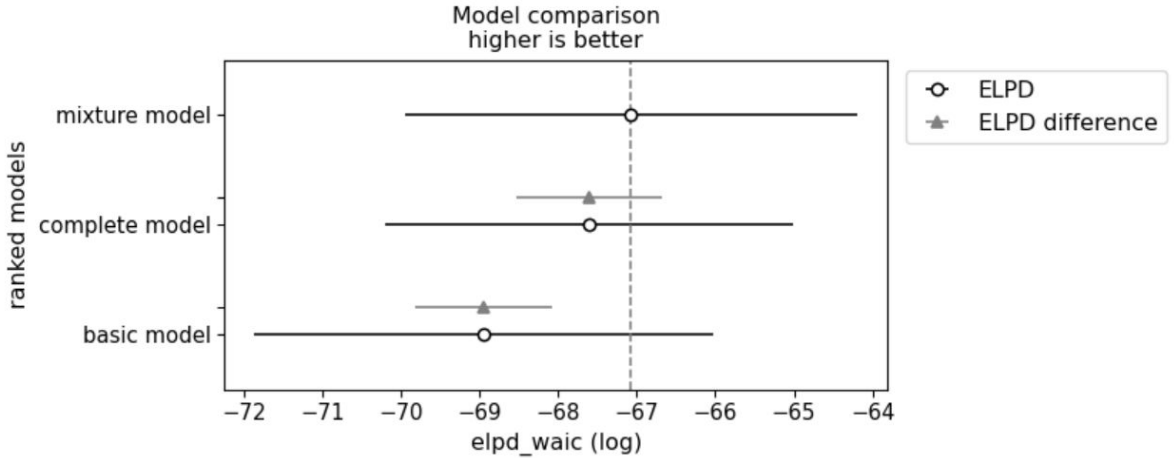
Convergence and effective sample size

Number of MCMC samples	2000	
Number of variables	121	
ess_bulk	Lowest: 261 (~13%); Highest: 2668(~133%)	Most have good effective sample size
Split r_hat	All < 1.05	Passed similarity test

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha_gender	-0.10	0.97	-2.02	1.64	0.03	0.02	1419.56	1315.52	1.00
beta_gender	0.02	0.97	-1.81	1.82	0.02	0.02	2039.80	1303.39	1.00
mixture_gender	0.27	0.45	0.00	1.00	0.02	0.02	439.77	439.77	1.01
alpha_over55	-0.18	0.95	-1.86	1.62	0.03	0.02	1445.17	1232.66	1.00
beta_over55	0.14	0.86	-1.71	1.65	0.02	0.03	2027.23	1048.52	1.00
mixture_over55	0.36	0.48	0.00	1.00	0.03	0.02	264.84	264.84	1.00

Modeling Diagnostics PART 2

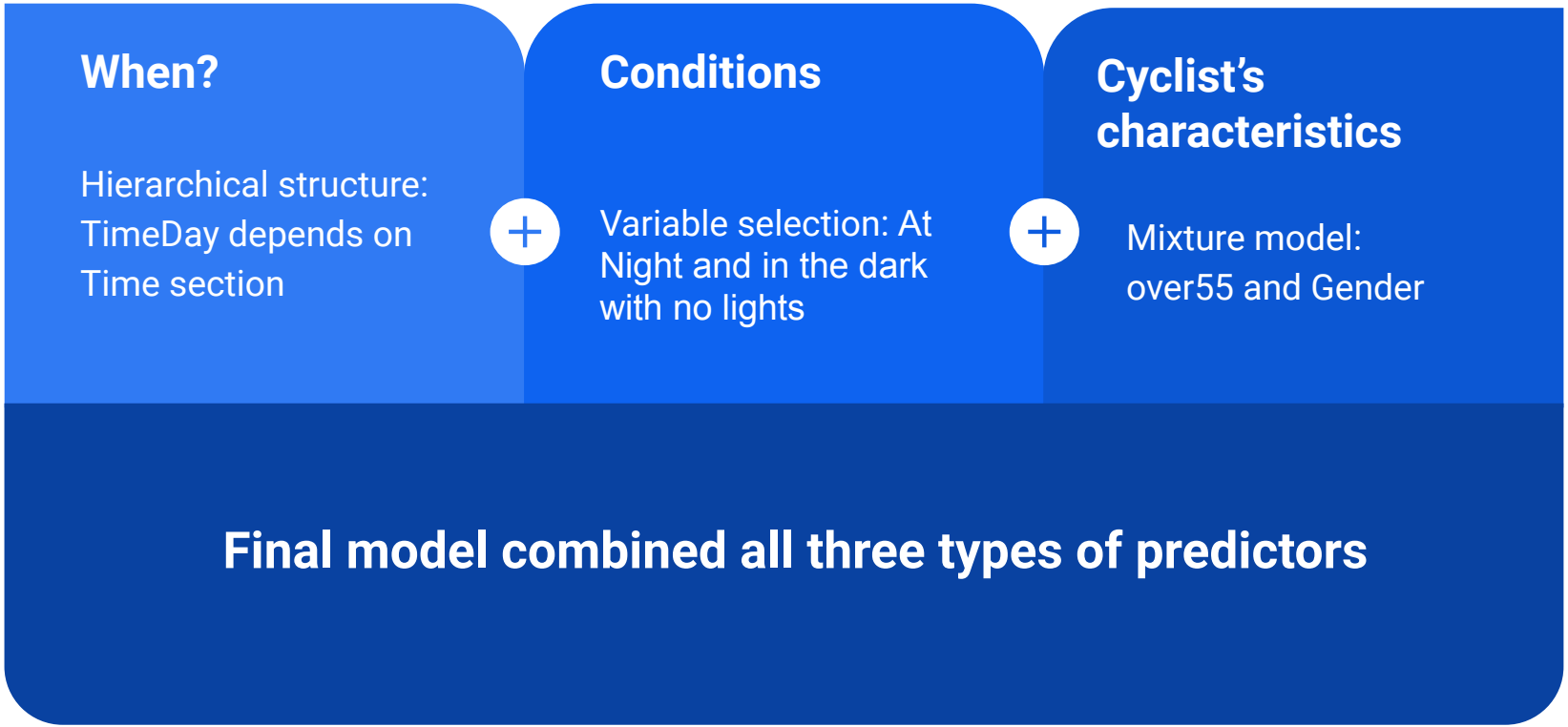
WAIC comparison



Model	Simple logistic regression model	Our model without mixture	Our model with mixture
Description	Normal priors on all betas	• Hierarchical structure on Time/Day • Normal priors on selected condition predictors	• Hierarchical structure on Time/Day • Normal priors on selected condition predictors • Mixture component on Gender and Age
elpd_waic	-68.830915	-67.620926	-66.733539

Conclusion

Conclusion


When?
Hierarchical structure: TimeDay depends on Time section
Conditions
Variable selection: At Night and in the dark with no lights
Cyclist's characteristics
Mixture model: over55 and Gender
Final model combined all three types of predictors