Homework2

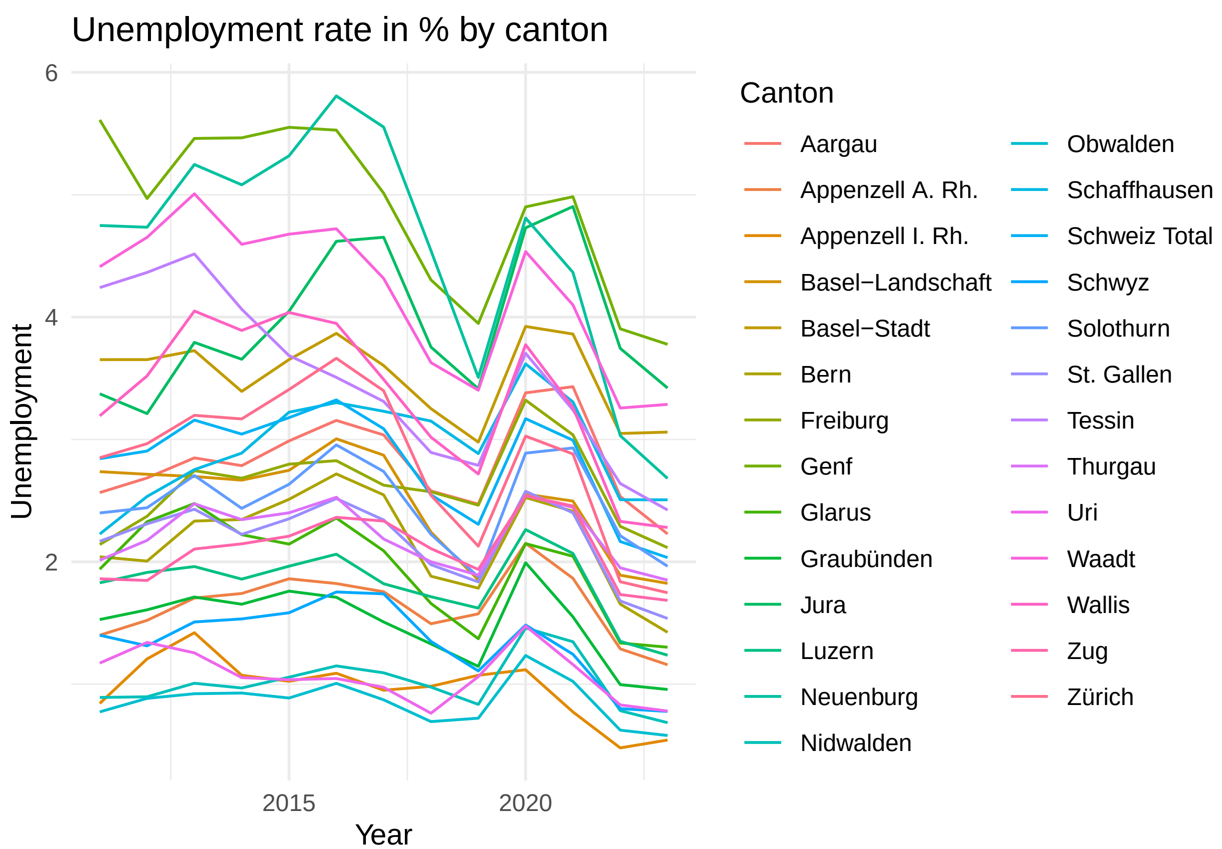
Erdan Beka, Cyril Scheuermann, Roman Krass, Keijo Nierula

2024-05-21

Data exploration

In this part we visualize the raw data to have an overview of the unemployment rate. We will do this for each canton and then summarize the data to get an overview of the unemployment rate in Switzerland.

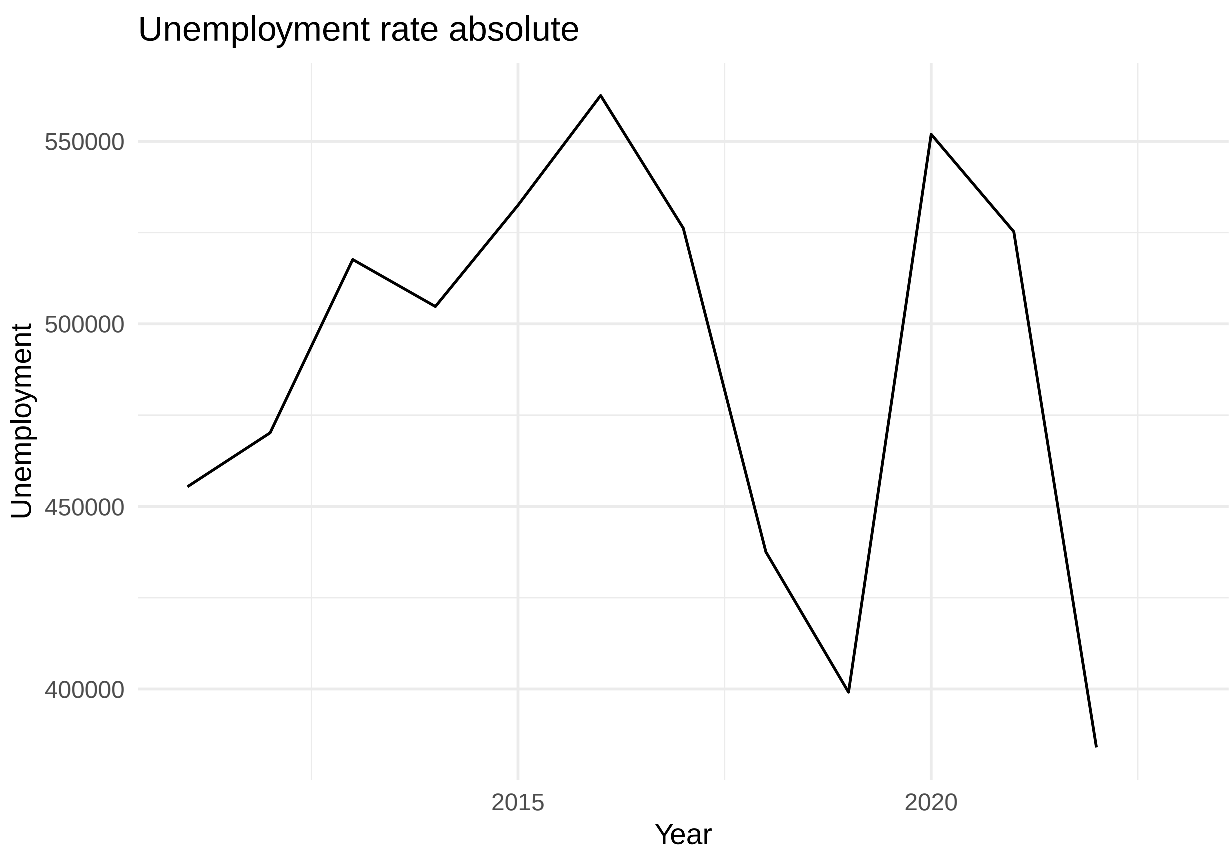
```
# visualize the data for each canton
data |>
    ggplot(aes(x = Year, y = Unemployment, color = Kanton)) +
    geom_line() +
    theme_minimal() +
    labs(title = "Unemployment rate in % by canton", color = "Canton")
```



In this plot we can see the unemployment rate for each canton in % over the years. We can see that the percentage of unemployment is different for each canton but they all follow the same trend. This is for example visible in the year 2020 where the unemployment rate increased for all cantons.

```
# visualize the data summarized
data |>
    index_by(Year) |>
    summarise(Unemployment = sum(Unemployment_number, na.rm = FALSE)) |>
    ggplot(aes(x = Year, y = Unemployment)) +
    geom_line() +
    theme_minimal() +
    labs(title = "Unemployment rate absolute")
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

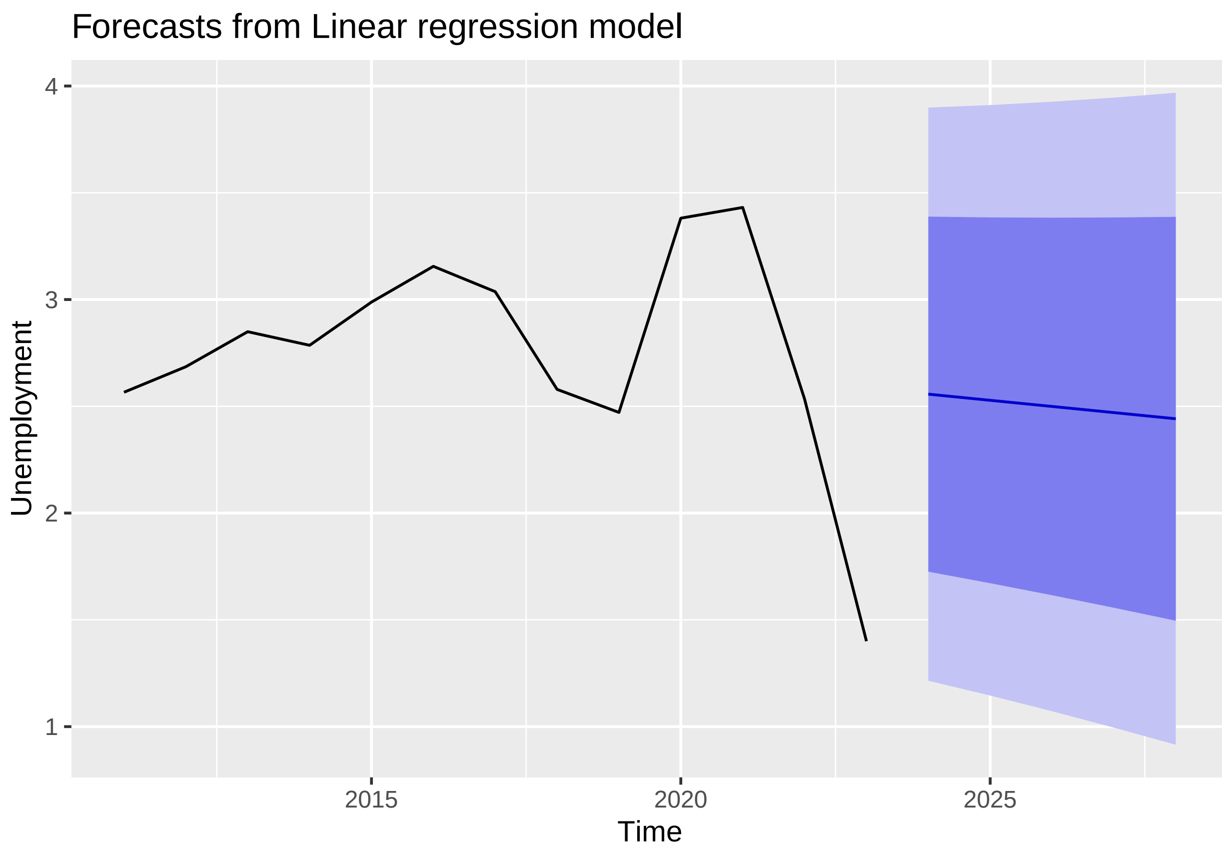


In this plot we can see the absolute number of unemployed people in Switzerland over the years. We can see that the number of unemployed people increased in 2020 and decreased a lot in 2022.

Forecasting

Linear Trend Model

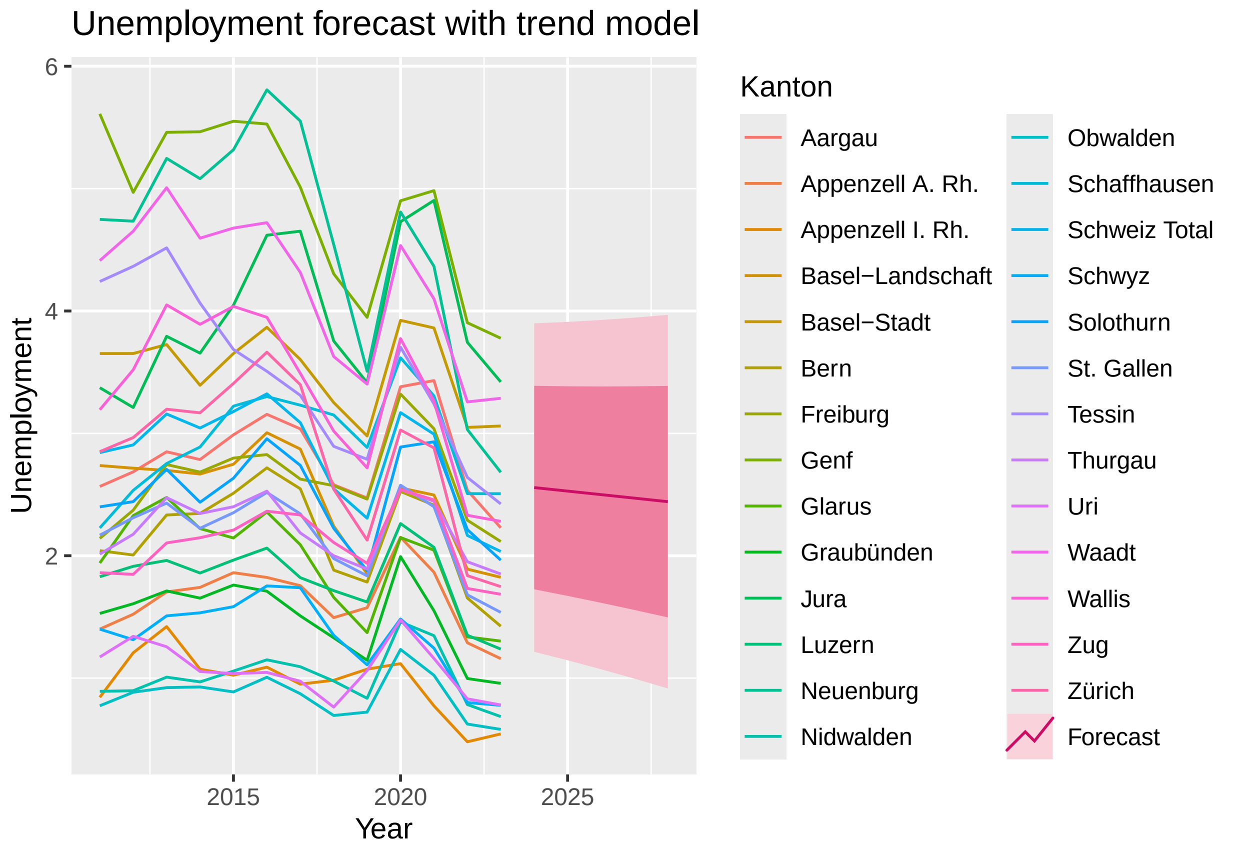
```r
# Define trend model for the Unemployment
trend_model <- tslm(Unemployment ~ trend, data = data_ts)

# Forecast the unemployment rate for the next 5 years
trend_fc <- trend_model |>
    forecast(h = 5)

# Plot the forecast
trend_fc |>
    autoplot()
```



When we look at the linear trend model we can see that the unemployment rate is forecasted to decrease over the next 5 years. The range of the forecast is quite large. But this is also because of the big decrease in overall unemployment in 2021 and 2022.

visualize Results

```r
# Plot forecast for all cantons
autoplot(data_tsibble) +
    autolayer(trend_fc, series = "Forecast") +
    xlab("Year") +
    ylab("Unemployment") +
    ggtitle("Unemployment forecast with trend model")
```



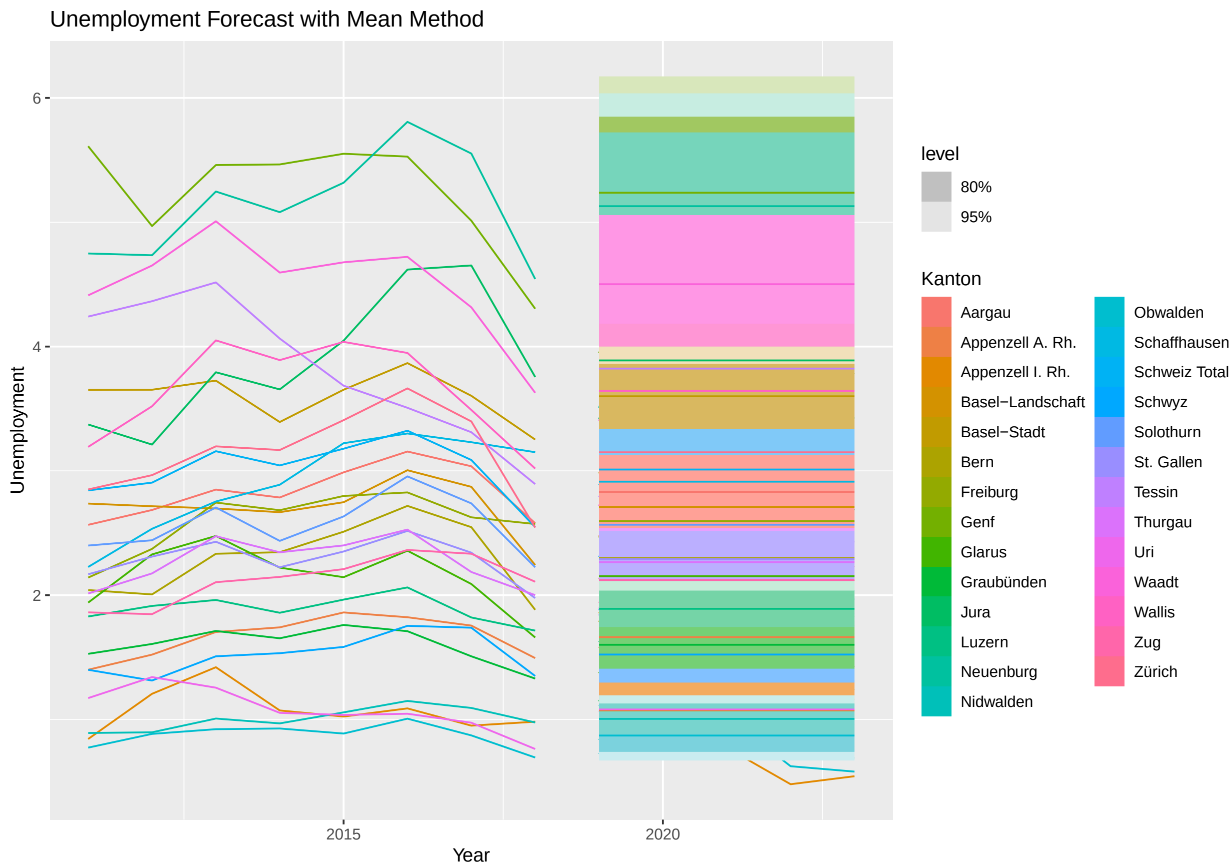
When we look at the forecast for all cantons we can see that the unemployment rate is forecasted to decrease. But because the forecast is combined for all cantons we can't see the differences between the cantons. Because the unemployment rate is combined from all cantons the range isn't suitable for all cantons. For example Appenzell Innerrhoden has a very low unemployment rate compared to other cantons. The forecast is why higher than the current unemployment rate. Which is not what the general forecast shows.

Next we try the mean method to get another forecast:

```
# First we split the data into training and test sets
# This is done so that we can evaluate the quality of the forecasts later
train_data <- data_tsibble |>
    filter(Year <= 2018)

test_data <- data_tsibble |>
    filter(Year > 2018)


# Mean Forecasting Method
mean_model <- train_data |>
    model(mean_fc = MEAN(Unemployment))

mean_fc <- mean_model |>
    forecast(new_data = test_data)

# Plot the forecast
autoplot(train_data, series = "Training Data") +
```

```r
    autolayer(test_data, series = "Test Data") +
    autolayer(mean_fc, series = "Mean Forecast") +
    xlab("Year") +
    ylab("Unemployment") +
    ggtitle("Unemployment Forecast with Mean Method")
```

```
## Warning in geom_line(...): Ignoring unknown parameters: `series`

## Warning in geom_line(eval_tidy(expr(aes(!!!aes_spec))), data = object, ..., :
## Ignoring unknown parameters: `series`

## Warning in ggdist::geom_lineribbon(without(intvl_mapping, "colour_ramp"), :
## Ignoring unknown parameters: `series`

## Warning in geom_line(mapping = without(mapping, "shape"), data =
## unpack_data(object[single_row[["FALSE"]], : Ignoring unknown parameters:
## `series`
```

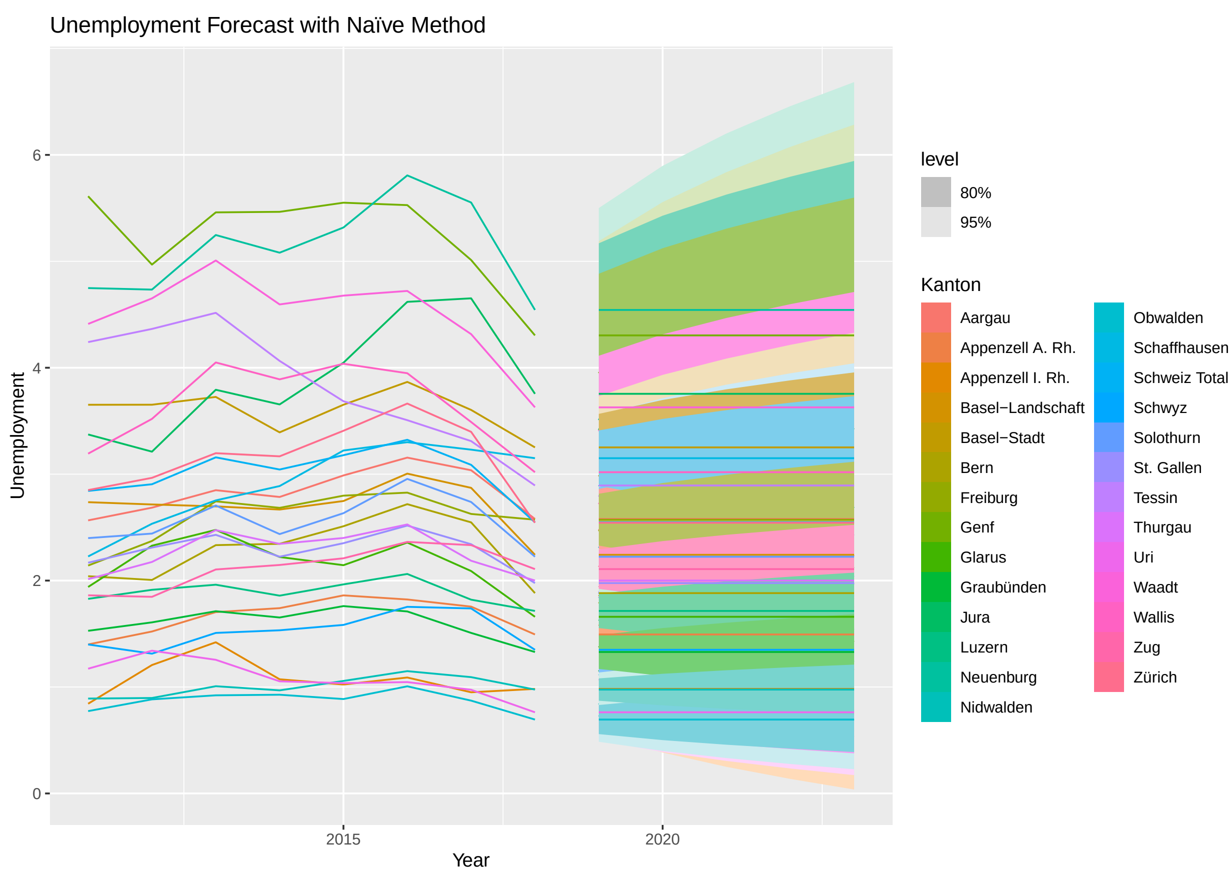


Naive method:

```r
# Naïve Forecasting Method
naive_model <- train_data |>
    model(naive_fc = NAIVE(Unemployment))
```

```r
naive_fc <- naive_model |>
    forecast(new_data = test_data)

# Plot the forecast
autoplot(train_data, series = "Training Data") +
    autolayer(test_data, series = "Test Data") +
    autolayer(naive_fc, series = "Naïve Forecast") +
    xlab("Year") +
    ylab("Unemployment") +
    ggtitle("Unemployment Forecast with Naïve Method")
```

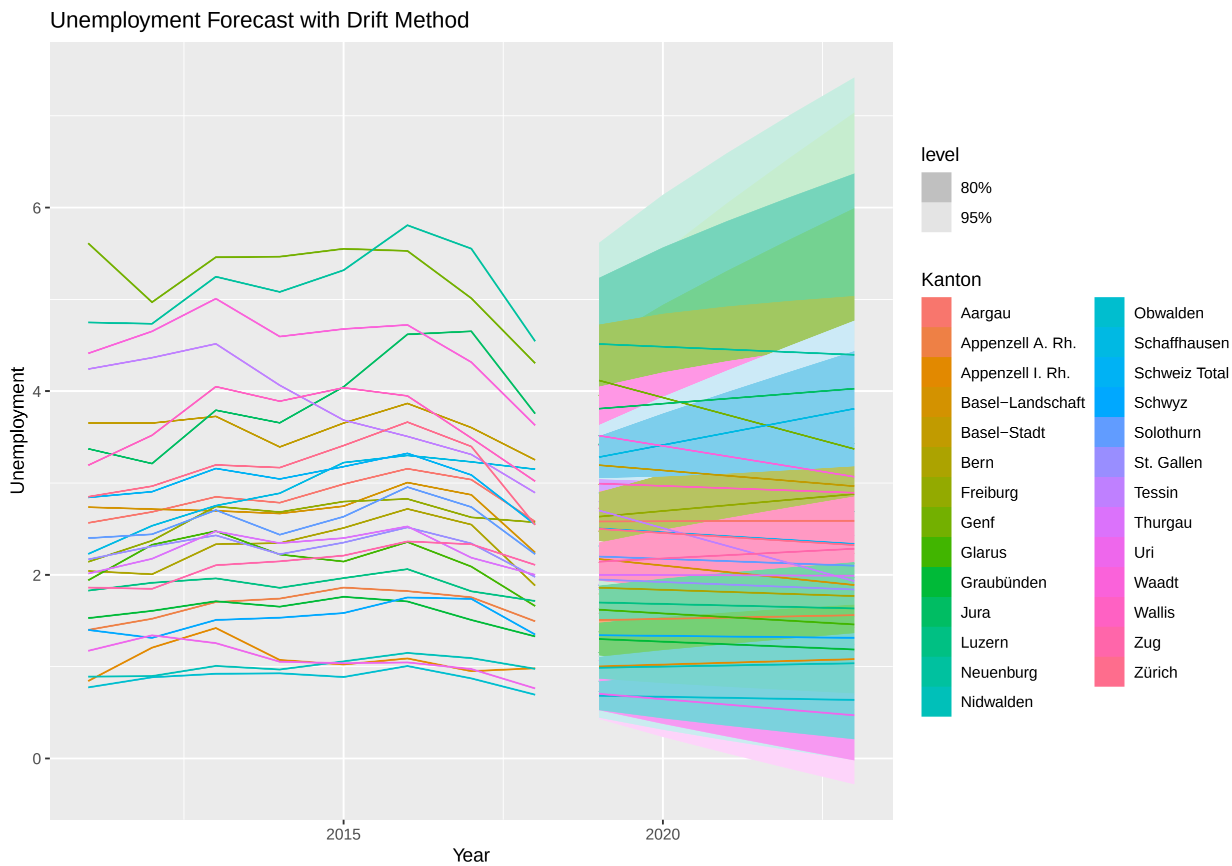
```
## Warning in geom_line(...): Ignoring unknown parameters: ‘series‘

## Warning in geom_line(eval_tidy(expr(aes(!!!aes_spec))), data = object, ..., :
## Ignoring unknown parameters: ‘series‘

## Warning in ggdist::geom_lineribbon(without(intvl_mapping, "colour_ramp"), :
## Ignoring unknown parameters: ‘series‘

## Warning in geom_line(mapping = without(mapping, "shape"), data =
## unpack_data(object[single_row[["FALSE"]], : Ignoring unknown parameters:
## ‘series‘
```



Drift Method:

```r
# Drift Forecasting Method
drift_model <- train_data |>
    model(drift_fc = RW(Unemployment ~ drift()))

drift_fc <- drift_model |>
    forecast(new_data = test_data)

# Plot the forecast
autoplot(train_data, series = "Training Data") +
    autolayer(test_data, series = "Test Data") +
    autolayer(drift_fc, series = "Drift Forecast") +
    xlab("Year") +
    ylab("Unemployment") +
    ggtitle("Unemployment Forecast with Drift Method")
```

```
## Warning in geom_line(...): Ignoring unknown parameters: `series`

## Warning in geom_line(eval_tidy(expr(aes(!!!aes_spec))), data = object, ..., :
## Ignoring unknown parameters: `series`

## Warning in ggdist::geom_lineribbon(without(intvl_mapping, "colour_ramp"), :
## Ignoring unknown parameters: `series`

## Warning in geom_line(mapping = without(mapping, "shape"), data =
## unpack_data(object[single_row[["FALSE"]], : Ignoring unknown parameters:
## `series`
```



Just from looking at the plots, it is hard to tell which method is the best. We will now calculate the accuracy of the forecasts to get a better idea of which method is the best.

```
# Calculate accuracy metrics
mean_accuracy <- mean_fc |>
    accuracy(test_data)

naive_accuracy <- naive_fc |>
    accuracy(test_data)

drift_accuracy <- drift_fc |>
    accuracy(test_data)

# For the Linear Trend Model we calculate the accuracy without the test dataset because we were not abl
trend_accuracy <- trend_fc |>
  accuracy()


mean_RMSE <- mean(mean_accuracy$RMSE)
naive_RMSE <- mean(naive_accuracy$RMSE)
drift_RMSE <- mean(drift_accuracy$RMSE)
trend_RMSE <- mean(trend_accuracy[,"RMSE"])
cat("Mean RMSE: ", mean_RMSE, "\n")
```

```
## Mean RMSE:  0.6047311
```

```
cat("Naïve RMSE: ", naive_RMSE, "\n")
```

```
## Naïve RMSE:  0.4651364
```

```
cat("Drift RMSE: ", drift_RMSE, "\n")
```

```
## Drift RMSE:  0.5048573
```

```
cat("Linear Trend RMSE: ", trend_RMSE, "\n")
```

```
## Linear Trend RMSE:  0.4834281
```

```
cat("We can see that the Naïve method has the lowest RMSE, which means it is the most accurate method. 
```

```
## We can see that the Naïve method has the lowest RMSE, which means it is the most accurate method.
```

```
mean(naive_accuracy$MAPE)
```

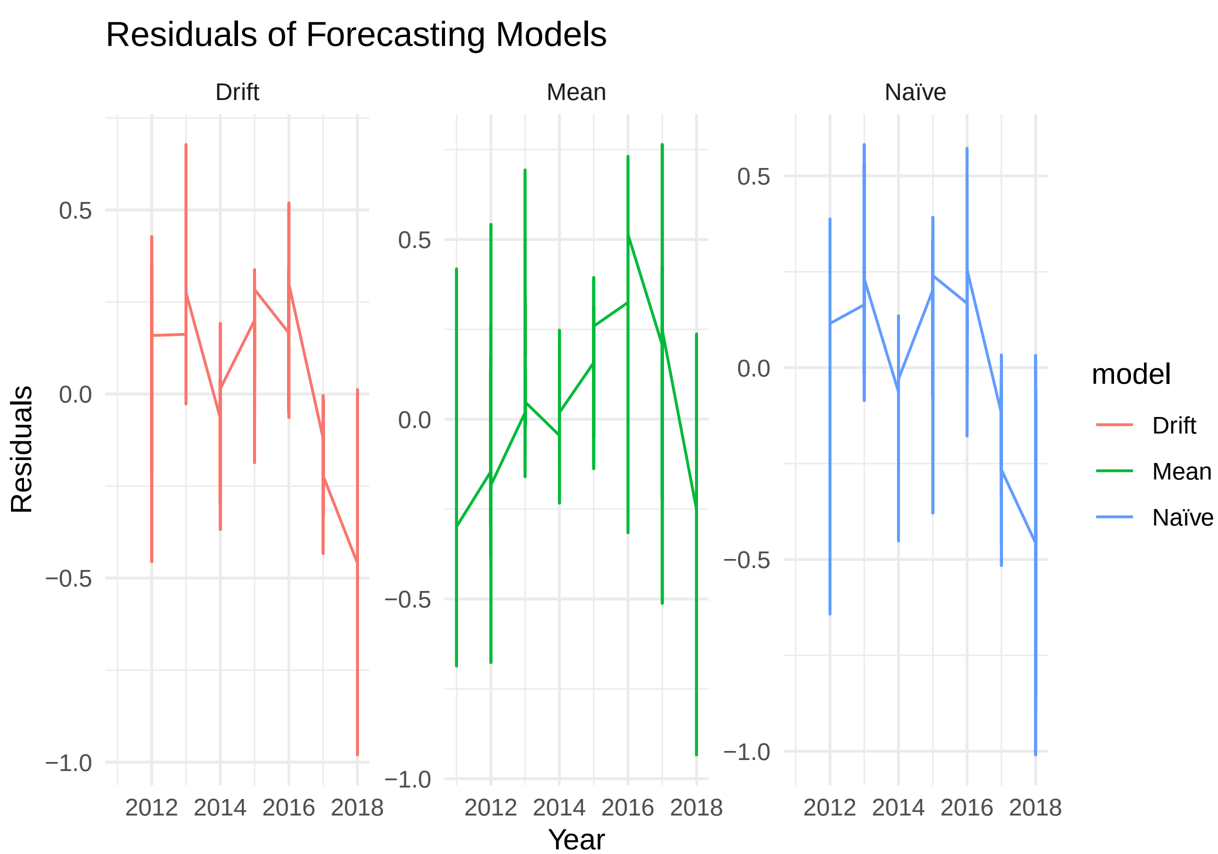
```
## [1] 20.45493
```

We can see that the Naïve method has the lowest RMSE, which means it is the most accurate method. It also has the lowest MAPE (20.45), which means it is the most accurate method so far. The Mean method has the highest RMSE, which means it is the least accurate method.

To get a better idea of the accuracy we visualize the performance of the models by combining the residuals:

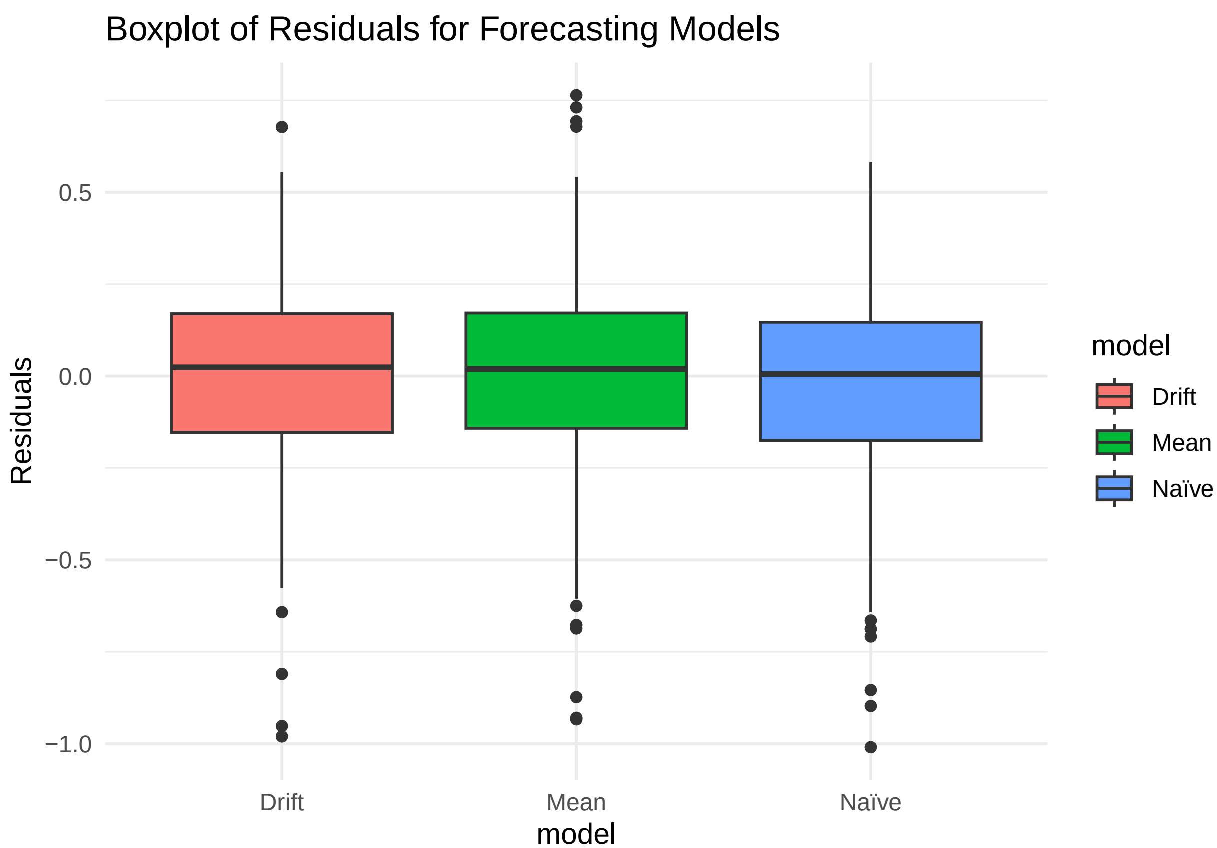
```r
# Residuals for each model
mean_resid <- augment(mean_model) |>
    mutate(model = "Mean")

naive_resid <- augment(naive_model) |>
    mutate(model = "Naïve")

drift_resid <- augment(drift_model) |>
    mutate(model = "Drift")

# Combine residuals
residuals <- bind_rows(mean_resid, naive_resid, drift_resid)

# Plot residuals
residuals |>
    ggplot(aes(x = Year, y = .resid, color = model)) +
    geom_line() +
    facet_wrap(~model, scales = "free_y") +
    theme_minimal() +
    labs(title = "Residuals of Forecasting Models", y = "Residuals")
```

```
## Warning: Removed 54 rows containing missing values or values outside the scale range
## (‘geom_line()‘).
```



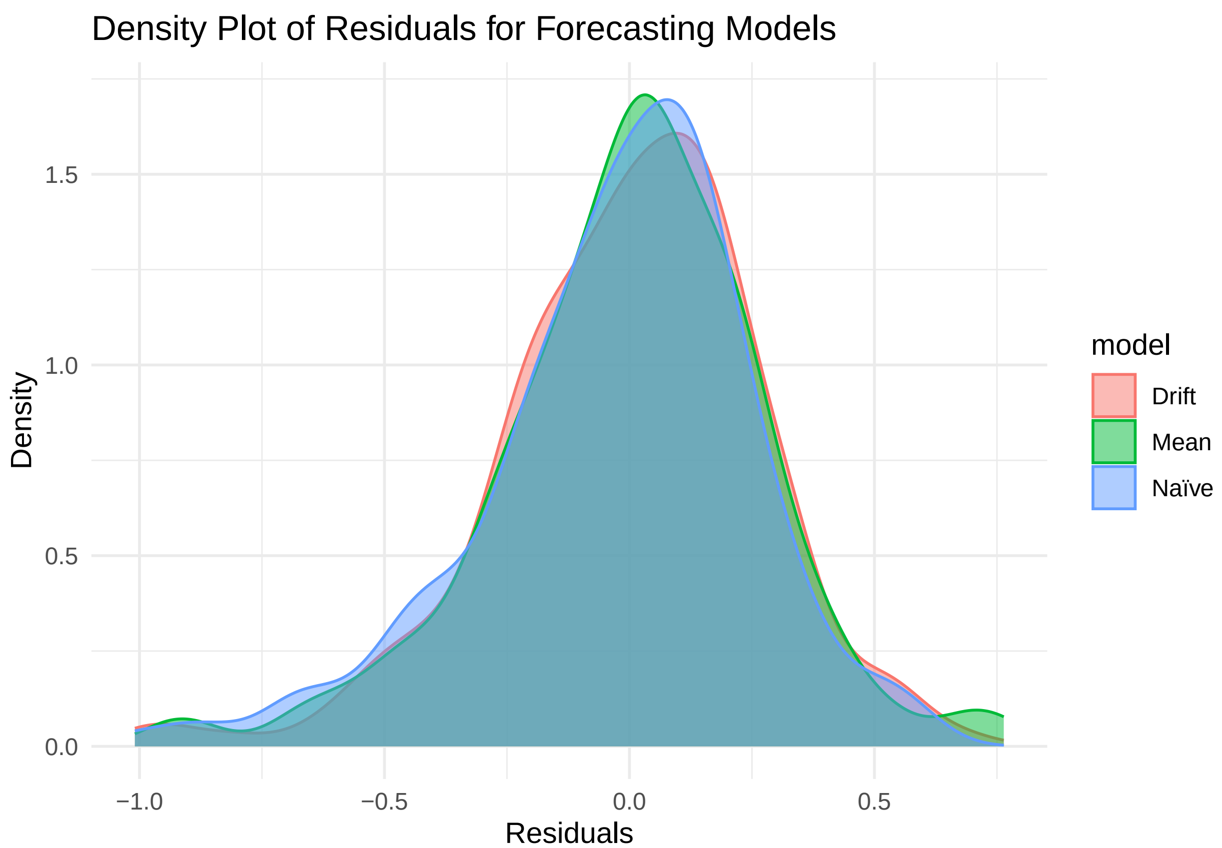
```
# Boxplot of residuals
residuals |>
    ggplot(aes(x = model, y = .resid, fill = model)) +
    geom_boxplot() +
    theme_minimal() +
    labs(title = "Boxplot of Residuals for Forecasting Models", y = "Residuals")
```

```
## Warning: Removed 54 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```




```
# Density plot of residuals
residuals |>
    ggplot(aes(x = .resid, fill = model, color = model)) +
    geom_density(alpha = 0.5) +
    theme_minimal() +
    labs(title = "Density Plot of Residuals for Forecasting Models", x = "Residuals", y = "Density")
```

```
## Warning: Removed 54 rows containing non-finite outside the scale range
## (`stat_density()`).
```



The density plot shows that the residuals of the Naïve method are the closest to a normal distribution, which is a good sign for the accuracy of the model. The boxplot shows that the Naïve method has the smallest range of residuals, which means it is the most consistent method. The residuals of the Mean method have the largest range, which means it is the least consistent method.

Actual Forecast with Naive Method

```
# Extend the forecast horizon to December 2024
last_year <- max(data$Year)
forecast_horizon <- 2024 - last_year

# Generate extended Naïve forecasts for each canton
extended_naive_fc <- data |>
    group_by(Kanton) |>
    model(naive_fc = NAIVE(Unemployment)) |>
    forecast(h = forecast_horizon)

# Extract the forecasted values
forecast_table <- extended_naive_fc |>
    as_tibble() |>
    select(Year, Kanton, .mean)

# Display the forecasted values
kable(forecast_table)
```

Year	Kanton	.mean
2024	Aargau	2.2280062
2024	Appenzell A. Rh.	1.1585694
2024	Appenzell I. Rh.	0.5435785
2024	Basel-Landschaft	1.8237969
2024	Basel-Stadt	3.0601221
2024	Bern	1.4238516
2024	Freiburg	2.1158366
2024	Genf	3.7773164
2024	Glarus	1.3022337
2024	Graubünden	0.9567779
2024	Jura	3.4204309
2024	Luzern	1.2368501
2024	Neuenburg	2.6816396
2024	Nidwalden	0.6853918
2024	Obwalden	0.5806569
2024	Schaffhausen	2.5067347
2024	Schweiz Total	2.0346054
2024	Schwyz	0.7773620
2024	Solothurn	1.9636876
2024	St. Gallen	1.5370276
2024	Tessin	2.4233714
2024	Thurgau	1.8503221
2024	Uri	0.7802802
2024	Waadt	3.2865992
2024	Wallis	2.2806628
2024	Zug	1.6850007
2024	Zürich	1.7471755

TODO: maybe remove this? We already calculated accuracy above.

```r
# Display the accuracy metrics for each model
accuracy_metrics <- tibble(
    Model = c("Mean", "Naïve", "Drift"),
    RMSE = c(mean_RMSE, naive_RMSE, drift_RMSE),
    MAPE = c(mean(mean_accuracy$MAPE), mean(naive_accuracy$MAPE), mean(drift_accuracy$MAPE))
)

accuracy_metrics %>%
    knitr::kable(caption = "Accuracy Metrics for Different Forecasting Models")
```

Table 2: Accuracy Metrics for Different Forecasting Models

Model	RMSE	MAPE
Mean	0.6047311	27.71496
Naïve	0.4651364	20.45493
Drift	0.5048573	21.31522

The table above shows the RMSE and MAPE values for the Mean, Naïve, and Drift forecasting models. The Naïve method has the lowest RMSE and MAPE, indicating it is the most accurate model among the three. We decided not to make something with the seasonal model, because we only have yearly data and no monthly data.