Macroanomaly: Detecting outliers in time series data

Luciano Perfetti Villa David Newhouse Olivier Dupriez

```r
library(macroanomaly)
library(imf.data)
library(collapse)
```

Introduction

This package provides functions to detect outliers in time series data, particularly focusing on macroeconomic indicators. It includes methods for detecting anomalies using various statistical techniques from a range of packages. The package allows the user to detrend, deseasonalize and normalize the data, making it suitable for anomaly detection.

At the moment the package supports the following methods:

- Z-score based detection ("zscore")
- Isolation Forests using the `isotree` package ("isotree")
- Outlier Trees using the `outliertree` package ("outliertree") (Warning: this method might not work properly for datasets with more than 1000 rows)
- Tsoutliers based on the IQR, as implemented in the `forecast` package ("tsoutlier")
- Point and collective anomalies using the `capa` method in the `anomaly` package ("capa")

Example Data

To explain each method, we will use two example datasets: an IMF dataset and a World Bank dataset. The IMF dataset contains monthly Consumer Price Index (CPI) data, while the World Bank dataset contains various yearly macroeconomic indicators. For the latter dataset, the package contains a function to load the data and prepare it for analysis.

```r
# Load IMF data
imf.data::load_datasets("CPI")$get_series(freq = "M",
                                          indicator = "PCPI_IX",
                                          start_period = "2015-01-01",
                                          end_period = "2023-12-31") |>
  fmutate(TIME_PERIOD = lubridate::ym(TIME_PERIOD)) |>
  pivot(
    ids = "TIME_PERIOD",
    how = "longer") |>
  fmutate(
    value = as.numeric(value),
    Country = stringr::str_split_i(variable, "\\.", 2),
    Frequency = stringr::str_split_i(variable, "\\.", 1),
```

```
    variable = stringr::str_split_i(variable, "\\.", 3)) -> imf_data_long


# Load World Bank data
wdi_download(.path = tempdir()) |>
  pivot(
    ids = c("Year", "Country.Name", "Country.Code"),
    how = "longer",
    names = list("Indicator.Code", "Indicator.Value")
    ) |>
  fsubset(Indicator.Code %in% c("EG.CFT.ACCS.ZS",
                                "NY.GDP.MKTP.CN.AD") &
            Year > 2005) -> wdi_data_long_subset
```

Many of these time series contain missing values or missing periods, which can usually affect the detection of anomalies. The package provides a functionality that detects missing periods and automatically imputes missing data based on a selected method from the `imputeTS` package. If the user prefers to impute the data using a different method, they can do so before applying the anomaly detection methods. However, as with most cases of missing data, if there is no data points for a country and indicator, the current implementation cannot impute data. Therefore, if the user will not impute these data, it is better to exclude them from the analysis. The `normalize` function will let the user know that there are countries with missing values, and that these should be excluded.

```
# Filter the data to remove countries with no data points
nobs <- fnobs(wdi_data_long_subset$Indicator.Value,
              g=wdi_data_long_subset[c("Country.Code","Indicator.Code")],TRA=2)
wdi_data_long_subset |> fsubset(!is.na(nobs)) -> wdi_data_long_subset_filtered

imf_data_long |>
  fsubset(!Country %in% c("YE")) -> imf_data_long_filtered
```

Normalization and imputation

The package provides a `normalize` function that allows the user to normalize the data, detrend it, and deseasonalize it. The user can choose to impute missing values using a method from the `imputeTS` package. The normalization is done by country and indicator, and the user can choose to keep the decomposed data (i.e., trend and seasonal components) for further analysis. Note that the normalization is done on the long format data, which is suitable for anomaly detection methods. Therefore, it is required that the data is in long format, with columns for the country, year, and value. The `normalize` function will return a long format data frame with the normalized values, as well as the decomposed components if requested.

```
# Normalize the IMF data
imf_data_long_filtered |>
  normalize(.indicator_col = "variable",
            .frequency = "monthly",
            .value_col = "value",
            .country_col = "Country",
            .detrend = TRUE,
            .impute = TRUE,
            .time_col = "TIME_PERIOD")  -> imf_data_long_normalized

# Normalize the World Bank data
```

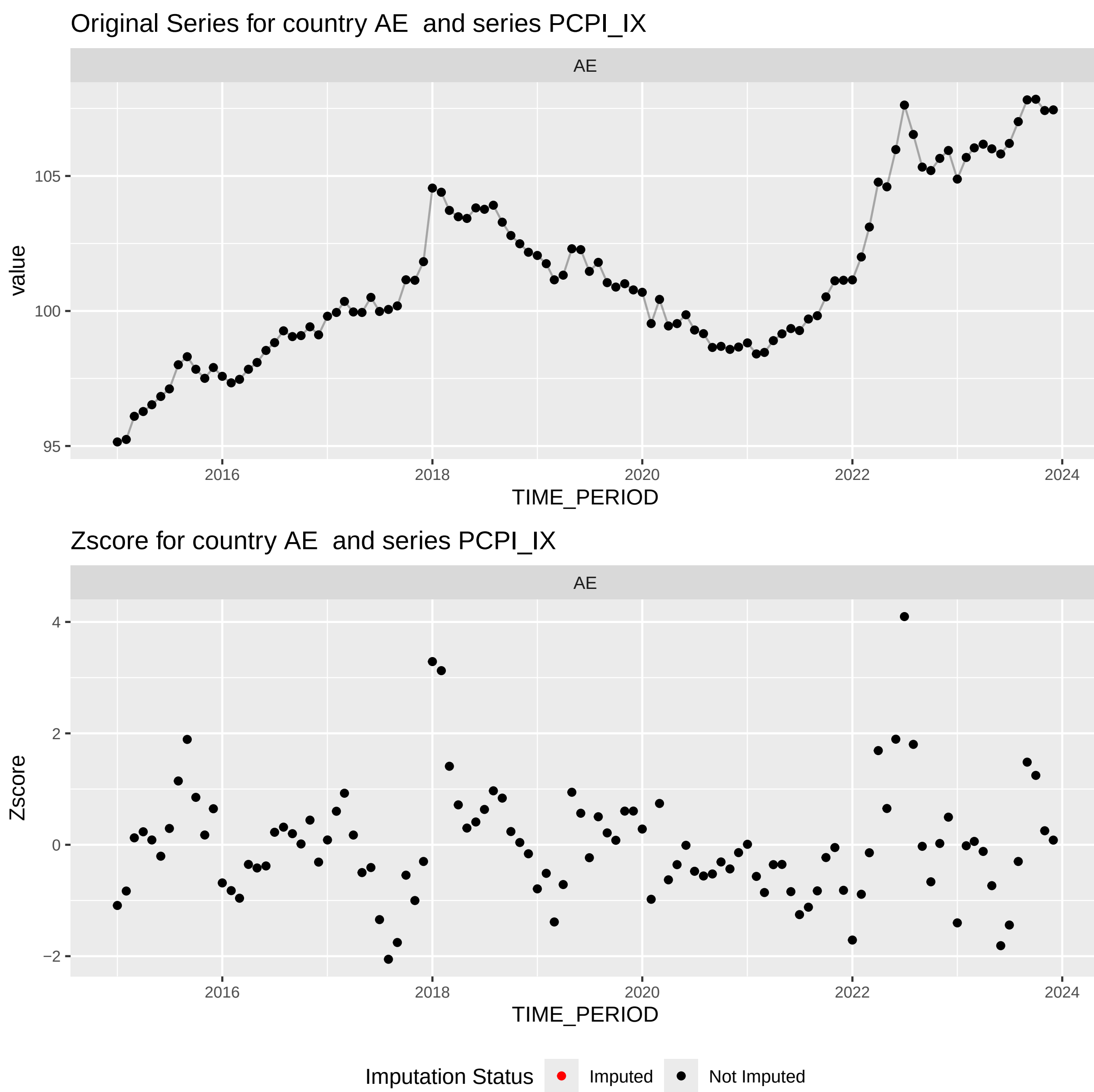
```r
wdi_data_long_subset_filtered |>
  normalize(.value_col= "Indicator.Value",
            .country_col = c("Country.Code", "Country.Name"),
            .indicator_col = "Indicator.Code",
            .time_col = "Year",
            .detrend = TRUE,
            .impute = TRUE
            ) -> wdi_data_long_subset_normalized
```

We can get a summary of the normalized data, which will show the number of countries and indicators, as well as the number of gaps and missing values. This is useful to understand the data before applying the anomaly detection methods.

```r
# Summary of the normalized IMF data
summary(imf_data_long_normalized)
#> Macroanomaly Normalized Object
#> Country Columns:  Country
#> Total number of countries:  187
#> Time Columns:  TIME_PERIOD
#> Time column from:  2015 Jan  to  2023 Dec
#> Total number of gaps:  0
#> Indicator Columns:  variable
#> Number of indicators:  1
#> Value Column:  value
#> Total number of missing values:  904
#> Frequency:  monthly
#>
#> Specified options:
#> Detrend:  TRUE
#> Impute:  TRUE
#> Impute Method:  na_interpolation
#> Season:  NULL
#> Keep Decomposition Variables:  FALSE
#> Long Format:  TRUE
#>
#> Data Summary:
#> Summary of:  value
#>     Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
#>     0.48    97.40   104.60   332.93   127.00 68887.67
#>
#> Summary of Zscore:
#>      Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
#> -4.519296 -0.546970 -0.001506  0.000000  0.503838  7.849954
```

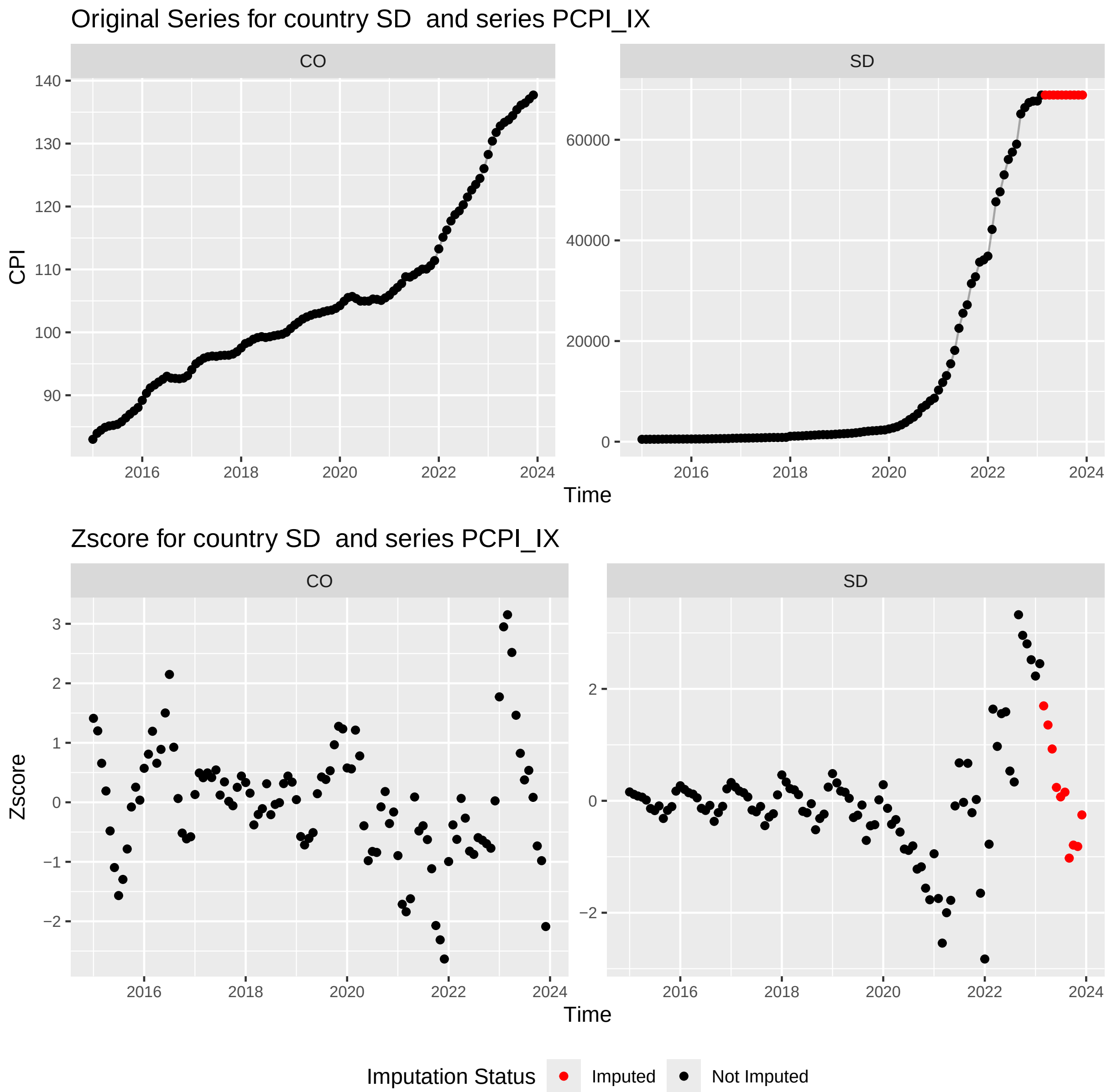
We can also plot the original and normalized data to see the effect of normalization. The `plot_normalized` function will plot the original and normalized data for each country and indicator. The highlighted points in red are the imputed values. It is important to account for this, since some future outliers may be imputed values, and therefore not real anomalies. By default, the `plot` method will plot the first country and indicator in the data. The user can specify the country and indicator to plot by using the `country` and `indicator` arguments. Also, the use can change the x-axis and y-axis labels.

```r
# Plot the normalized IMF data
plot(imf_data_long_normalized)
```



Let's look at Sudan. We can see that the CPI for the months after 2023 are not recorded, and therefore the values are imputed.

```r
# Plot the normalized IMF data for Sudan
plot(imf_data_long_normalized, country = c("SD", "CO"),
     x.lab = "Time", y.lab = "CPI")
```



Anomaly Detection

The package provides several methods for detecting anomalies in the normalized data. The user can choose the method to use, and the package will apply the method to the normalized data. The `detect` function will use the object from `normalize` and apply the selected method to the data. The user can choose to apply multiple methods at once, and the package will return a data frame with the results for each method. The results will include the outlier indicator, the z-score, and the imputed values if applicable. If multiple methods are applied, the results will be combined into a single data frame with the results for each method, and a final variable with the total number of outlier indicator. The user can then use the results to analyze the anomalies in the data.

Each method can have additional information that the user can access. For example, the `isotree` method will return the outlier scores, which can be used to rank the anomalies. The `capa` method will return the point and collective anomalies, which can be used to analyze the anomalies in more detail. To allow for these additional columns to be added to the results, the user can use the `additional_cols` argument in the `detect` function. This will return a data frame with the additional columns for each method.

It is important to understand that the tree methods, `isotree` and `outliertree`, allow the user to include additional columns that are used as covariates. Also, since the method relies on the additional countries and indicators, the results are relative to the other countries and indicators in the data. Therefore, it is important to include all relevant countries and indicators in the data before applying the methods. The results from using one single country may vary substantially from the results using all countries and indicators. The `capa` method is not affected by this, since it applies the method to each country and indicator separately.

Let's apply the `detect` function to the normalized IMF data and the World Bank data. We will first apply the `tsoutlier` and then apply multiple methods, i.e., the `zscore`, `isotree`, and `capa`. Each method can have additional arguments, see the help documentation for more details (`?detect`). The `detect` function will return a data frame with the results for each method, and the user can then use the results to analyze the anomalies in the data.

```
# Detect anomalies in the IMF data using tsoutlier
imf_data_long_normalized |>
  detect(.method = "tsoutlier") -> imf_data_long_tsoutlier

# Detect anomalies in the IMF data using multiple methods
imf_data_long_normalized |>
  detect(.method = c("zscore", "isotree", "capa"),
         .args = list(zscore = c(.min_seg_len = 3))
         ) -> imf_data_long_multiple_methods
#> Warning: Error in detecting point anomalies: object 'p_anom_daf' not found
```

Similar to the `summary` method with the normalized data, we can get a summary of the detected anomalies. This will show some information about the dataset. Let see some information from the IMF data.

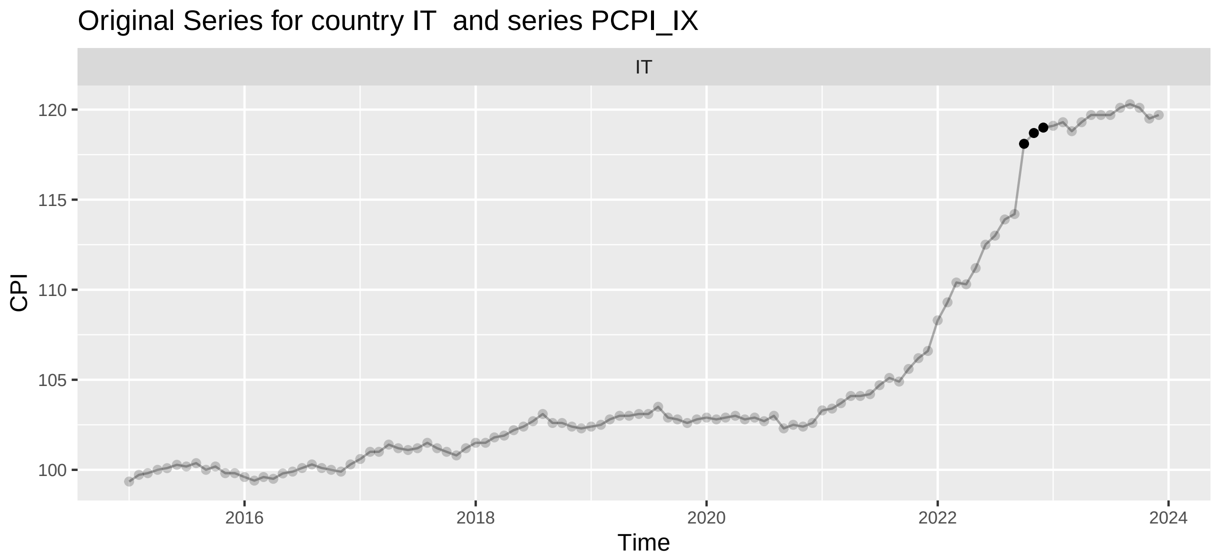
```
# Summary of the detected anomalies in the IMF data
summary(imf_data_long_multiple_methods)
#> Summary of Macroanomaly detect:
#>   Method(s) used for detection: zscore, isotree, capa
#>   Number of rows: 20196
#>   Number of columns: 10
#>
#>  Information of outliers:
#>   Number of countries with outliers: 179 of a total of 187 countries
#>   Number of indicators with outliers: 1 of a total of 1 indicators
#>   Number of time periods with outliers: 108 of a total of 108 time periods
#>   Number of outliers detected: 3193
#>   Country with most outliers: VE
#>     - Number of outliers in this country: 93
```

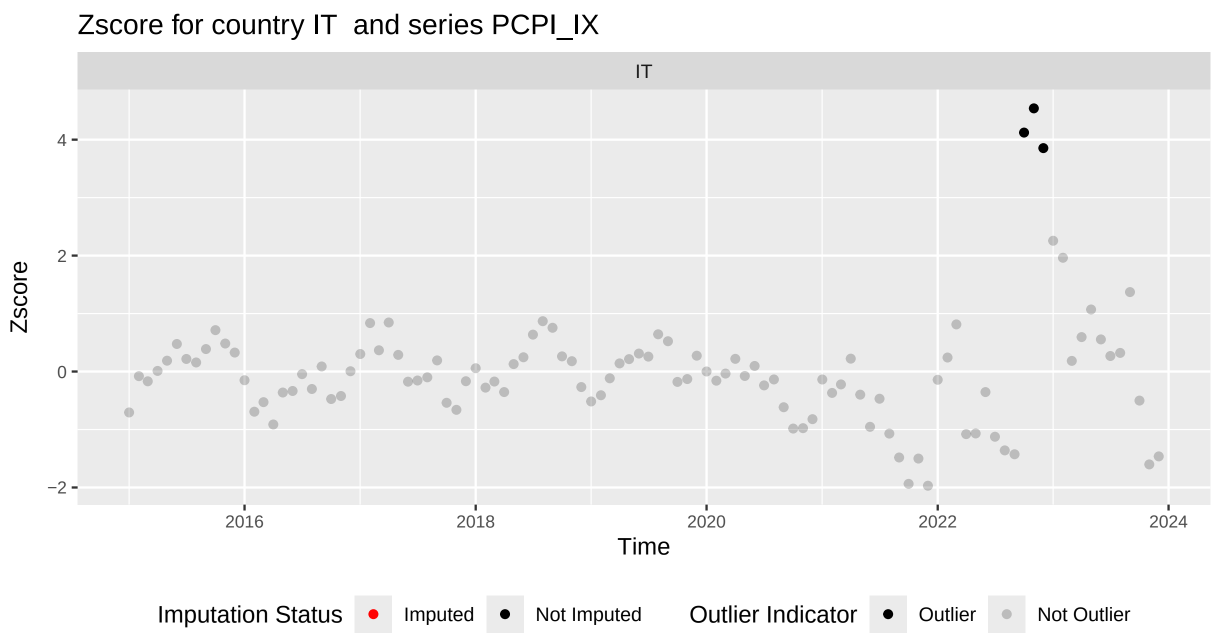
Before continuing, we mentioned above that `outliertree` method might not work properly for datasets with more than 1000 rows. Therefore, we will use the `outliertree` method only for a subset of the IMF data, i.e., for Argentina and Zimbabwe. This subset contains 216 observations, since the GitHub issue mentions the problems with 2,000 observations. The `outliertree` method will return a data frame with the results for each country and indicator, and the user can then use the results to analyze the anomalies in the data. At the moment, the `detect` function using the `outliertree` method will ignore the column that contains the original value (i.e., `value` column), and will only return detect using the `Zscore`. The user can then use the `plot` method to visualize the results.

```r
# Detect anomalies in the IMF data using outliertree
imf_data_long |>
  collapse::fsubset(Country %in% c("AR", "ZW")) |>
  normalize(.indicator_col = "variable",
            .frequency = "monthly",
            .value_col = "value",
            .country_col = "Country",
            .detrend = TRUE,
            .impute = TRUE,
            .time_col = "TIME_PERIOD") |>
    detect(.method = "outliertree",
           .args = list(outliertree = list(.cols = c("Zscore", "Country", "TIME_PERIOD")))
           ) -> imf_data_long_outliertree
#> Message from outliertree:
#> Reporting top 2 outliers [out of 2 found]
#>
#> row [108] - suspicious column: [Zscore] - suspicious value: [7.85]
#>  distribution: 99.074% <= 2.92 - [mean: -0.07] - [sd: 0.66] - [norm. obs: 214]
#>
#>
#> row [210] - suspicious column: [Zscore] - suspicious value: [7.65]
#>  distribution: 99.074% <= 2.92 - [mean: -0.07] - [sd: 0.66] - [norm. obs: 214]
```

We can also plot the detected anomalies using the `plot` method. The `plot` method will plot the original and normalized data, and highlight the detected anomalies. As before, the imputed values will be red, but the outliers will have a bright color (i.e, red or black). The user can specify the country and indicator to plot by using the `country` and `indicator` arguments, as well as the x-axis and y-axis labels.

```r
# Plot the detected anomalies in the IMF data
plot(imf_data_long_multiple_methods, country = "IT",
     x.lab = "Time", y.lab = "CPI")
```





Let's replicate the same for the World Bank data. We will apply the `zscore` method first, and then apply multiple methods, i.e., the `tsoutlier`, `isotree`, and `capa`. The results will be similar to the IMF data, but with different anomalies detected.

```r
# Detect anomalies in the World Bank data using zscore
wdi_data_long_subset_normalized |>
  detect(.method = "zscore") -> wdi_data_long_zscore

# Detect anomalies in the World Bank data using multiple methods
wdi_data_long_subset_normalized |>
  detect(.method = c("tsoutlier", "isotree", "capa"),
         .args = list(capa = c(.min_seg_len = 3),
                      isotree = c(.threshold = 0.7))
         ) -> wdi_data_long_multiple_methods
```

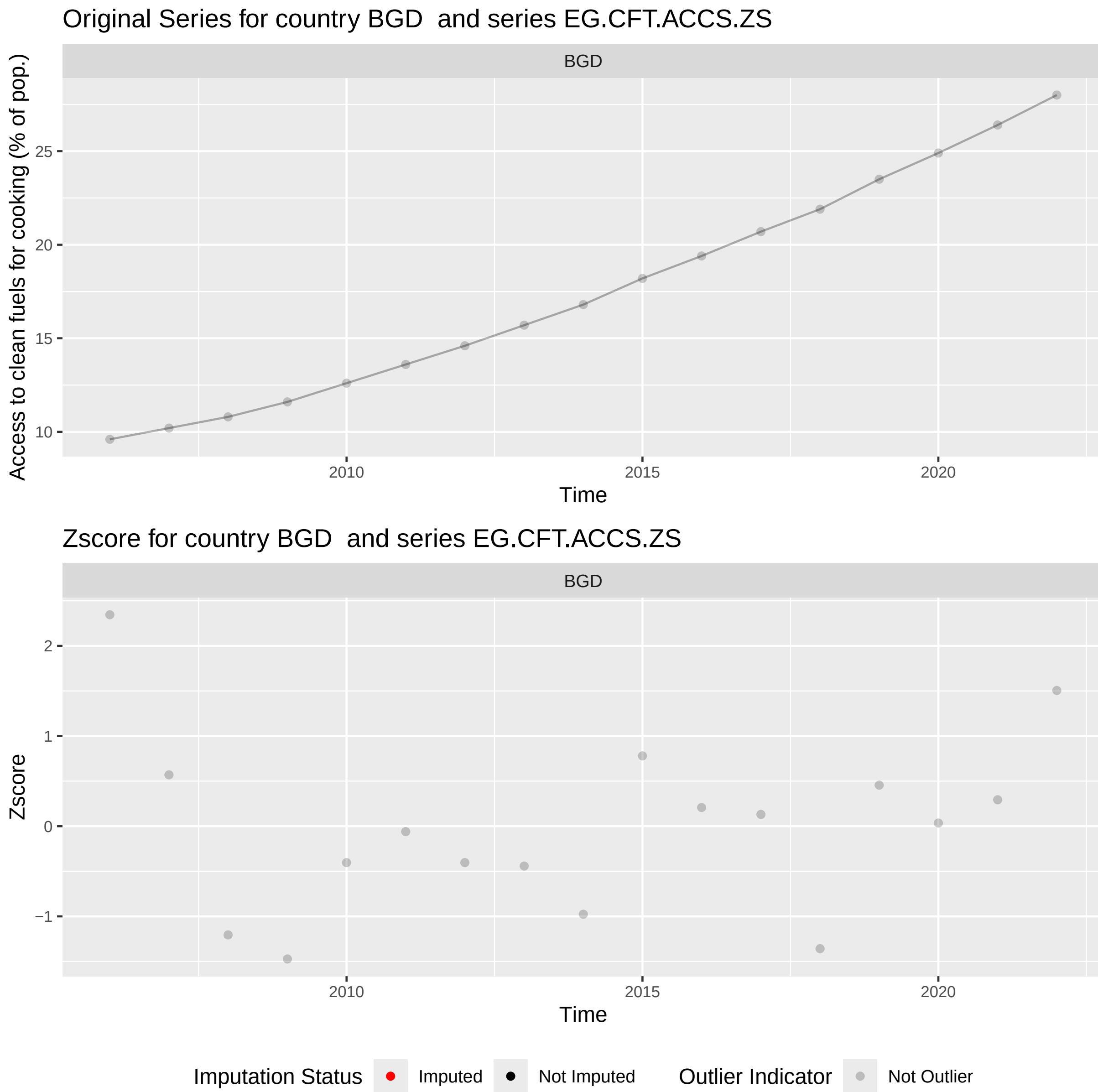
```
#> Warning: Missing values found in Zscore column. These will be removed from the
#> analysis.
```

We can get a summary of the detected anomalies in the World Bank data, which will show some information about the dataset.

```
# Summary of the detected anomalies in the World Bank data
summary(wdi_data_long_multiple_methods)
#> Summary of Macroanomaly detect:
#>   Method(s) used for detection: tsoutlier, isotree, capa
#>   Number of rows: 7764
#>   Number of columns: 11
#>
#>  Information of outliers:
#>   Number of countries with outliers: 58 of a total of 262 countries
#>   Number of indicators with outliers: 2 of a total of 2 indicators
#>   Number of time periods with outliers: 10 of a total of 18 time periods
#>   Number of outliers detected: 87
#>   Country with most outliers: FCS
#>     - Number of outliers in this country: 5
```

We can also plot the detected anomalies in the World Bank data, similar to the IMF data. The user can specify the country and indicator to plot by using the `country` and `indicator` arguments, as well as the x-axis and y-axis labels.

```
# Plot the detected anomalies in the World Bank data
plot(wdi_data_long_multiple_methods,
     country = "BGD",
     indicator = "EG.CFT.ACCS.ZS",
     x.lab = "Time",
     y.lab = "Access to clean fuels for cooking (% of pop.)")
```



Finally, we can save the results to a CSV file using `write_to_csv` function. The user has the option to save the results by rank and limiting the number of outliers to preserve; or all the results if needed. The user can specify the file name and path, and have the option to save additional columns from the different methods or only the main columns.

```
# Save the results to a CSV file only the first 30 outliers
write_to_csv(imf_data_long_multiple_methods,
             file = "imf_data_anomalies.csv",
             top_outliers = 30,
             additional_cols = FALSE)

# Save the results to a CSV file with all outliers
write_to_csv(wdi_data_long_multiple_methods, file = "wdi_data_anomalies.csv")
```

Conclusion

The `macroanomaly` package provides a set of functions to detect anomalies in time series data, particularly focusing on macroeconomic indicators. The package allows the user to normalize, detrend, and deseasonalize the data, making it suitable for anomaly detection. The user can choose from several methods to detect anomalies, including Z-score based detection, Isolation Forests, Outlier Trees, IQR based outliers, and Point and Collective Anomalies using the `anomaly` package. The package also provides functions to plot the results and summarize the detected anomalies.

Session Info

```
sessionInfo()
#> R version 4.4.1 (2024-06-14 ucrt)
#> Platform: x86_64-w64-mingw32/x64
#> Running under: Windows 11 x64 (build 26100)
#>
#> Matrix products: default
#>
#>
#> locale:
#> [1] LC_COLLATE=English_United States.utf8
#> [2] LC_CTYPE=English_United States.utf8
#> [3] LC_MONETARY=English_United States.utf8
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.utf8
#>
#> time zone: America/New_York
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] collapse_2.1.2          imf.data_0.1.7          macroanomaly_0.0.0.9000
#>
#> loaded via a namespace (and not attached):
#>  [1] gtable_0.3.6         anytime_0.3.12       xfun_0.52
#>  [4] ggplot2_3.5.2        tsibble_1.1.6        lattice_0.22-6
#>  [7] quadprog_1.5-8       vctrs_0.6.5          tools_4.4.1
#> [10] Rdpack_2.6.4         generics_0.1.4       curl_6.4.0
#> [13] parallel_4.4.1       tibble_3.3.0         xts_0.14.1
#> [16] pkgconfig_2.0.3      R.oo_1.27.1          RColorBrewer_1.1-3
#> [19] imputeTS_3.3         distributional_0.5.0 lifecycle_1.0.4
#> [22] stringr_1.5.1        compiler_4.4.1       farver_2.1.2
#> [25] stinepack_1.5        RhpcBLASctl_0.23-42  outliertree_1.10.0
#> [28] htmltools_0.5.8.1    feasts_0.4.1         yaml_2.3.10
#> [31] pillar_1.11.0        tidyr_1.3.1          ellipsis_0.3.2
#> [34] R.utils_2.13.0       nlme_3.1-164         fracdiff_1.5-3
#> [37] tidyselect_1.2.1     digest_0.6.37        stringi_1.8.7
#> [40] dplyr_1.1.4          purrr_1.0.4          labeling_0.4.3
#> [43] tseries_0.10-58      cowplot_1.2.0        fastmap_1.2.0
#> [46] grid_4.4.1           colorspace_2.1-1     cli_3.6.4
```

```
#> [49] magrittr_2.0.3       patchwork_1.3.1      fabletools_0.5.0
#> [52] withr_3.0.2          scales_1.4.0         forecast_8.24.0
#> [55] anomaly_4.3.3        lubridate_1.9.4      timechange_0.3.0
#> [58] TTR_0.24.4           rmarkdown_2.29       quantmod_0.4.28
#> [61] ggtext_0.1.2         nnet_7.3-19          timeDate_4041.110
#> [64] progressr_0.15.1     zoo_1.8-14           R.methodsS3_1.8.2
#> [67] isotree_0.6.1-4      urca_1.3-4           evaluate_1.0.4
#> [70] knitr_1.50           rbibutils_2.3        lmtest_0.9-40
#> [73] rlang_1.1.6          gridtext_0.1.5       Rcpp_1.0.14
#> [76] glue_1.8.0           xml2_1.3.8           jsonlite_2.0.0
#> [79] rstudioapi_0.17.1    R6_2.6.1
```