Outlier detection

We first set up some functions used to run the MATLAB scripts and plot the results. Click here for the analysis. Alternatively click one of these four links for the analysis of any specific dataset:

- op1_dl77_mode3_NHW.mat
- op2_dl_74_mode3_NHW.mat
- op1_dl_78_mode3_NHM.mat
- op2_dl_76_mode3_NHM.mat

The code for the plot in the paper is here.

In [1]:
```
import warnings
from pathlib import Path
from shutil import copy
from subprocess import run
from tempfile import TemporaryDirectory
from time import time

import jinja2
import matplotlib.pyplot as plt
import numpy as np
import tensorly as tl
import tlviz
import xarray as xr
from pygments import highlight
from pygments.lexers import MatlabLexer
from pygments.formatters import HtmlFormatter

from IPython.display import display, Markdown, HTML
from scipy import io
```

In [2]:
```
def matlab_to_tl(tensor_toolbox_obj):
    factors = [
        fm[0] for fm in tensor_toolbox_obj['u'][0][0].copy()
    ]
    weights = tensor_toolbox_obj['lambda'][0][0].squeeze().copy()
    return tl.cp_tensor.CPTensor((weights, factors))

def parse_matlab(matfile):
    # Load relevant data from MATLAB file
    cp_tensor = matlab_to_tl(matfile['bestF'])
    X = matfile['data'].copy()
    leverage = matfile['node_leverage'].squeeze().copy()
    residual = matfile['node_residual'].squeeze().copy()
    residual /= residual.sum()
    outlier_nodes = matfile['outlier_nodes'].squeeze().tolist()

    # Convert dataset to xarray
    if X.shape[-1] == 11:
        t = np.arange(1, 12)
        t_name = "month"
    elif X.shape[-1] == 7:
        t = np.arange(1, 8)
        t_name = "day of week"
    else:
        raise ValueError
```

```
    dataset = xr.DataArray(
        X,
        coords={
            "node": np.arange(1, len(X)+1),
            "hour": [2, 14, 19],
            t_name: t,
        },
        dims=["node", "hour", t_name]
    )

    # Remove outlier nodes
    dataset = dataset.drop(outlier_nodes, "node")

    # Impute missing data
    Xhat = cp_tensor.to_tensor()
    dataset.values[dataset.isnull().values] = Xhat[dataset.isnull().values]

    cp_tensor = tlviz.postprocessing.label_cp_tensor(cp_tensor, dataset)

    return cp_tensor, dataset, leverage, residual,


def run_experiment(datafile, params, outlier_nodes, print_matlab_code=False):
    with open("ApplyCPWOPT.m.j2") as f:
        template = f.read()

    with TemporaryDirectory() as tmpdir:
        tmpdir = Path(tmpdir)
        script_path = tmpdir / "ApplyCPWOPT.m"
        copy("ResidualLeverage_3wayNAN.m", tmpdir / "ResidualLeverage_3wayNAN.m")
        matlab_code = jinja2.Template(template).render(**params, savedir=str(tmpdir), ou
        with open(script_path, "w") as f:
            f.write(matlab_code)

        if print_matlab_code:
            display(Markdown("#### Running MATLAB code:"))
            display(HTML(f'<style>{ HtmlFormatter().get_style_defs(".highlight") }</styl
            display(HTML(highlight(matlab_code, MatlabLexer(), HtmlFormatter())))

        t0 = time()
        run(
            f"matlab -nosplash -nodesktop -r run('{script_path}')".split(),
        )
        t1 = time()
        print(f"Decomposed dataset in {t1 - t0:.0f}s.")
        return io.loadmat(tmpdir / datafile)


def plot_outliers(cp_tensor, dataset, leverage, residual):
    fig, ax = plt.subplots(dpi=200)
    with warnings.catch_warnings():  # Ignore warnings due to Pandas plotting
        warnings.simplefilter("ignore")
        ax = tlviz.visualisation.outlier_plot(
            cp_tensor,
            dataset,
            residual_rules_of_thumb='bonferroni p-value',
            leverage_rules_of_thumb='bonferroni p-value',
            p_value=[0.01],
            ax=ax
        )
        # Check that leverage and residuals coincide with those computed with MATLAB
        ax.scatter(leverage, residual, marker='x', color='tomato')
        tlviz.visualisation.components_plot(cp_tensor)
    plt.show()

def run_analysis(datafile, outlier_nodes, plot, print_matlab_code=False):
```

```
    params = {
        "operator": datafile.split("_dl_")[0][-1],
        "num_nodes": int(datafile.split("_dl_")[1].split("_")[0]),
        "granularity": datafile.split("_NH")[1][0]
    }

    matfile = run_experiment(datafile, params, outlier_nodes, print_matlab_code=print_ma
    cp_tensor, dataset, leverage, residual, = parse_matlab(matfile)
    if plot:
        plot_outliers(cp_tensor, dataset, leverage, residual)
    return cp_tensor, dataset, leverage, residual
```

Finding outliers

The following cells are running Mah-Rukh's MATLAB scripts. The first cell prints out the code that's being run.

op1_dl77_mode3_NHW.mat

In [3]:
```
datafile = "op1_dl_77_mode3_NHW.mat"
```

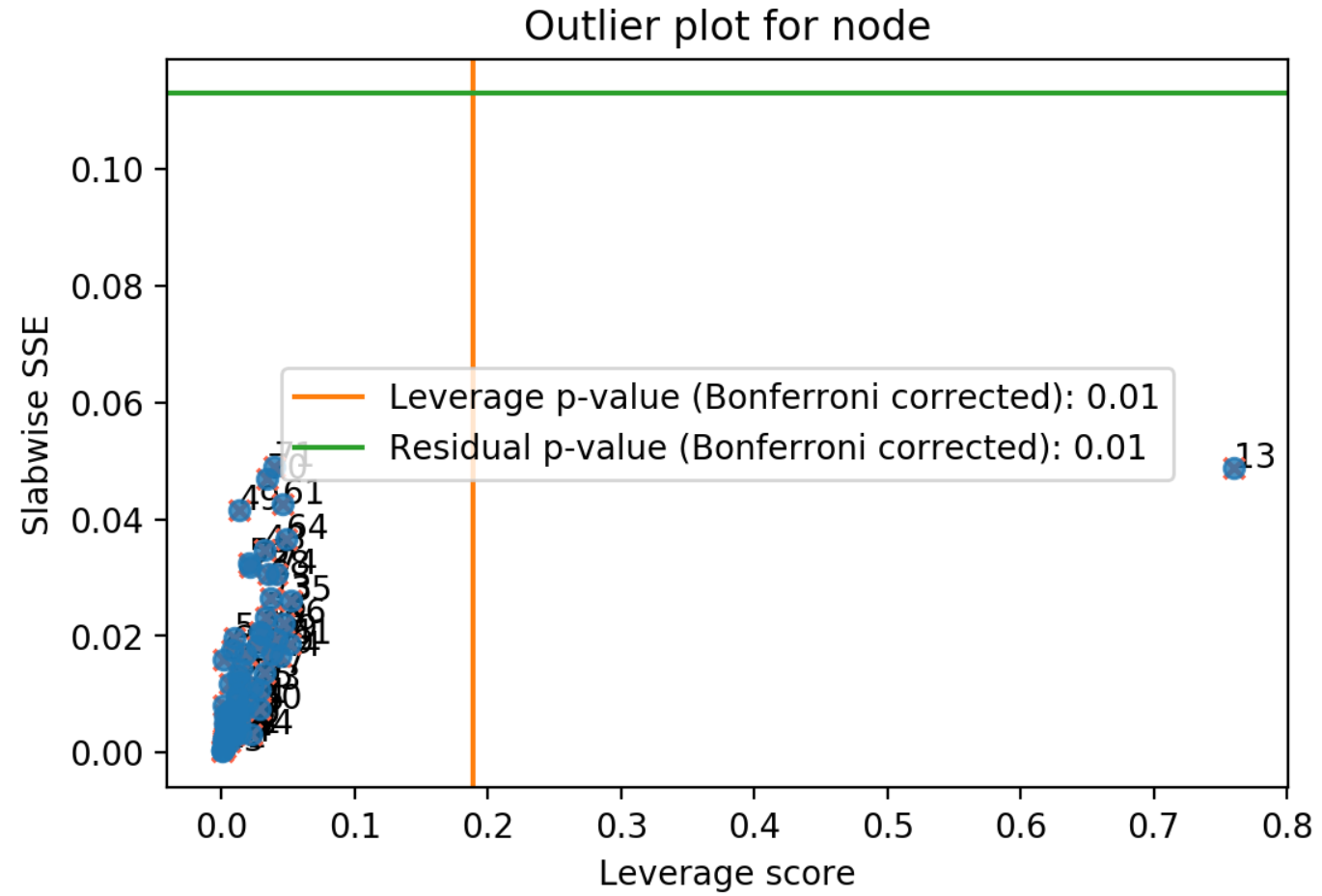
In [4]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[], plot=
```

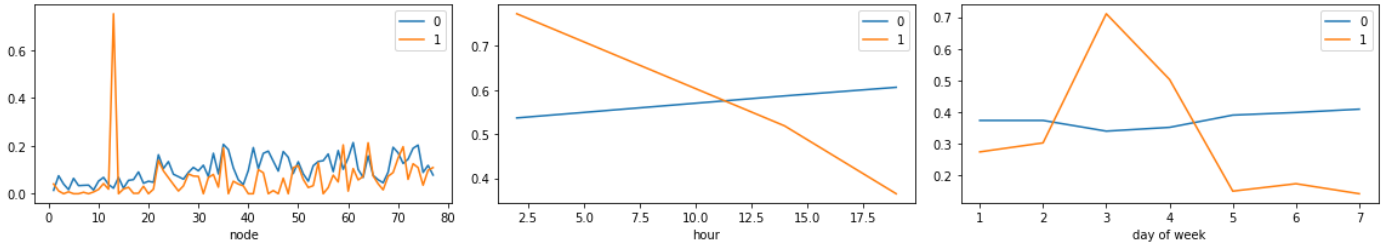
Running MATLAB code:

```
clear
addpath(genpath("/home/mariero/matlab/toolboxes/tensor_toolbox"))
data_file = 'op1_dl_77_mode3_NHW.mat'
load(['/home/mariero/matlab/network_speed_analysis/data/' data_file])


num_timeslices = 7

data = reshape((A),[77 3 num_timeslices]);  % For NHW

%% OUTSIDE LOOP-----------------------------------------------------------
X = data;

% STEP 2: Use this step when removing outlier nodes
outlier_nodes = []
X(outlier_nodes,:,:)=[];
%----------------------
C_wopt = X;
C_parafac= X;
C_wopt(isnan(C_wopt))=0; % C_wopt has zero values only at the NaN locations
% convert double matrices to tensors
C_wopt = tensor(C_wopt);
W_nan = ~isnan(X);
W_nan = tensor(W_nan);

% Set up optimization parameters
% ------------------------------
% number of limited memory vectors
lbfgsb_options.m = 5 % default 5
% a tolerance setting
lbfgsb_options.factr = 1e7 % default 1e7
```

```
lbfgsb_options.maxIts = 10000; % default 100
lbfgsb_options.maxTotalIts = 50000; % default 500
lbfgsb_options.pgtol = 1e-7; % tolerance related to gradient, default 1e-5


% WITH NAN-- RUN IT FOR FIND BEST INITIALIZATION ---FOR R==1 AND R==2
% WITH 1 2 and 3 COMPONENT
% Calculate two best factors and perform congruence check on them
R=2
min_ff = 100000
min_index = 0
BestU = 0
parfor(j = 1:50)
    [F{j}, U{j}, out{j}]  = cp_wopt(C_wopt, W_nan, R, 'skip_zeroing', true, 'opt', 'lb
fgsb', 'opt_options', lbfgsb_options, 'lower', 0);
    ff(j) = out{j}.f;
%    if(min_ff > ff(j))
%       min_ff = ff(j);
%       BestU = U{j};
%       min_index = j;
%       bestF = F{j};
%    end
end

idx = find(ff == min(ff))
bestF = F{idx};

%% RESUDUAL AND LEVERAGE GRAPH
[node_residual, node_leverage] = ResidualLeverage_3wayNAN(C_wopt, W_nan, bestF, 1);

%% Save data
savepath = ['/tmp/tmp5rkhqf4y/' data_file]
save(savepath)
```

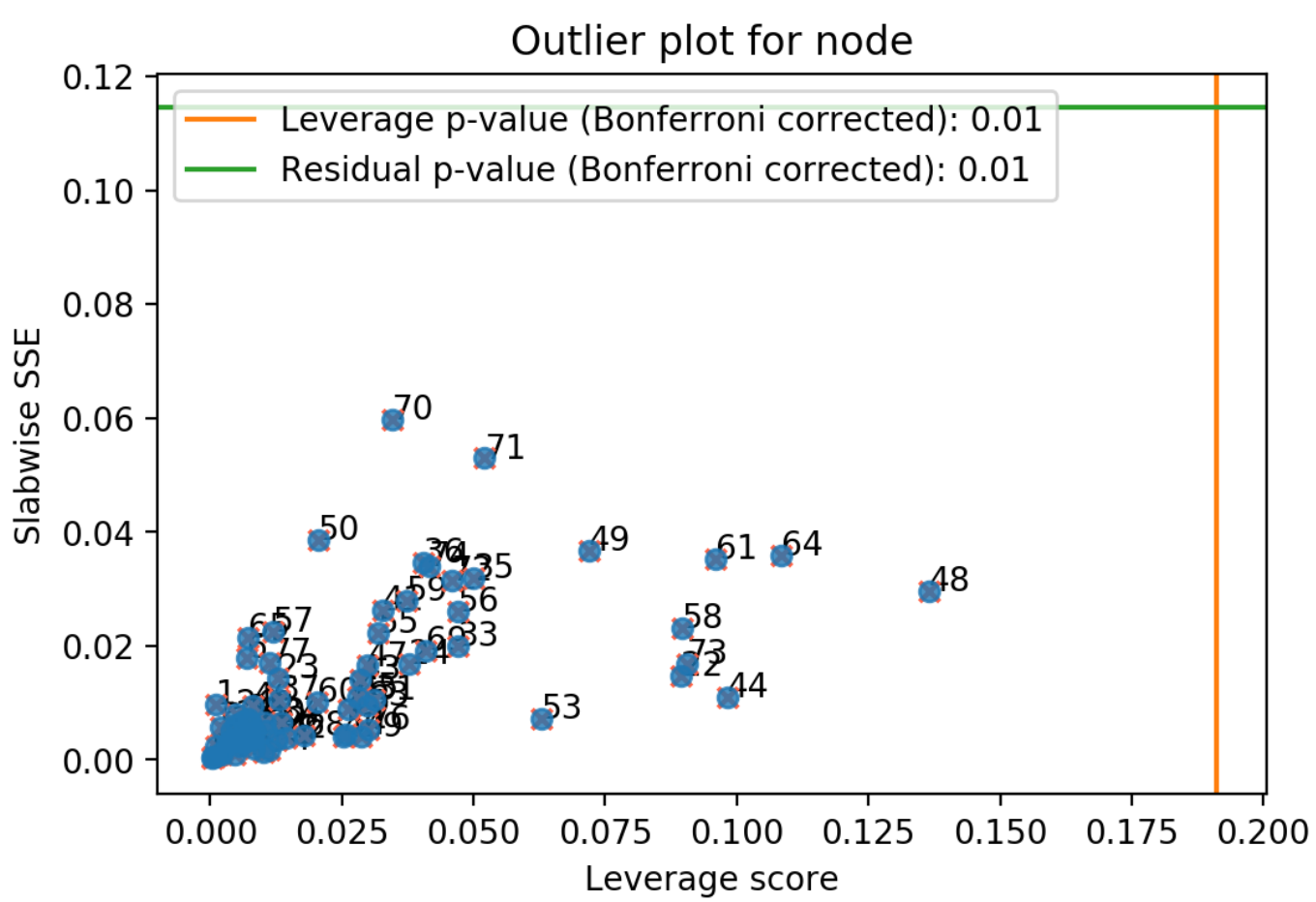
```
Decomposed dataset in 45s.
```

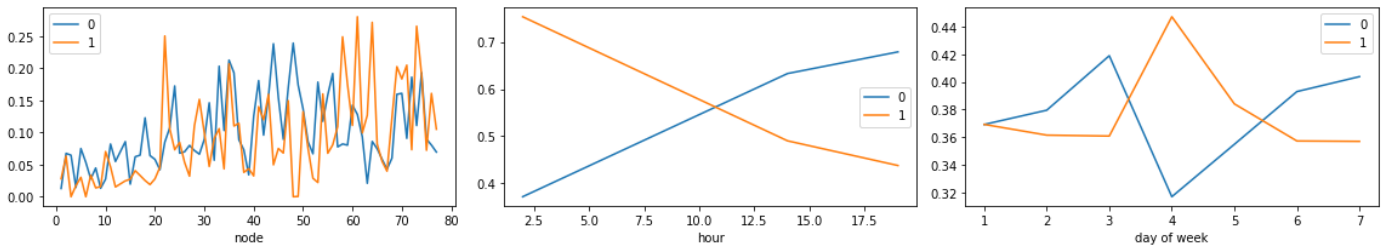




In [5]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[13], plo
```

```
Decomposed dataset in 50s.
```





In [6]:
```
print("Dataset shape:", dataset.shape)
```

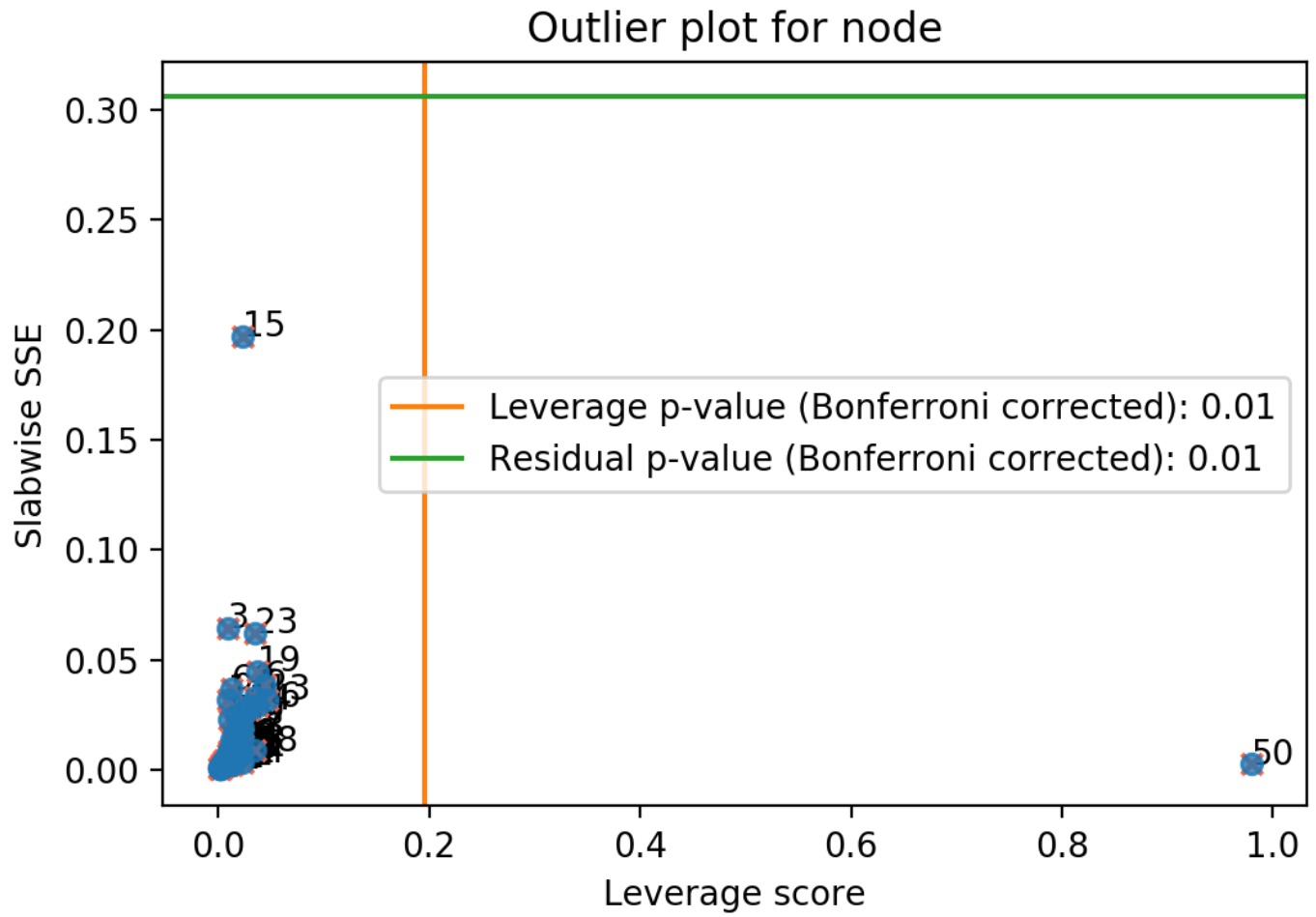
```
Dataset shape: (76, 3, 7)
```

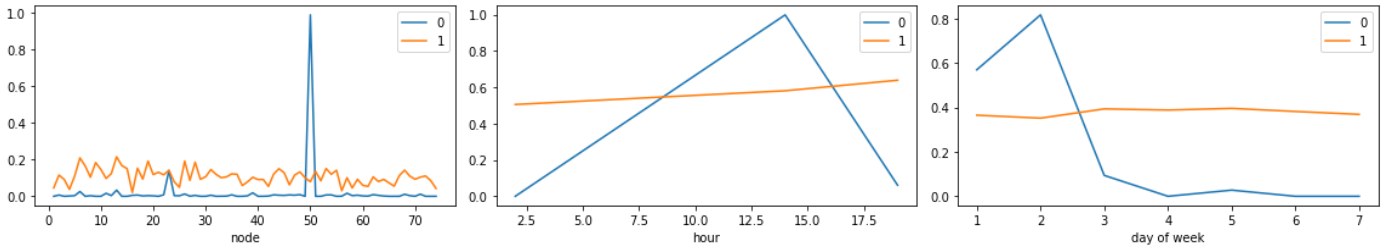
op2_dl_74_mode3_NHW.mat

In [7]:
```
datafile = "op2_dl_74_mode3_NHW.mat"
```

In [8]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[], plot=
```

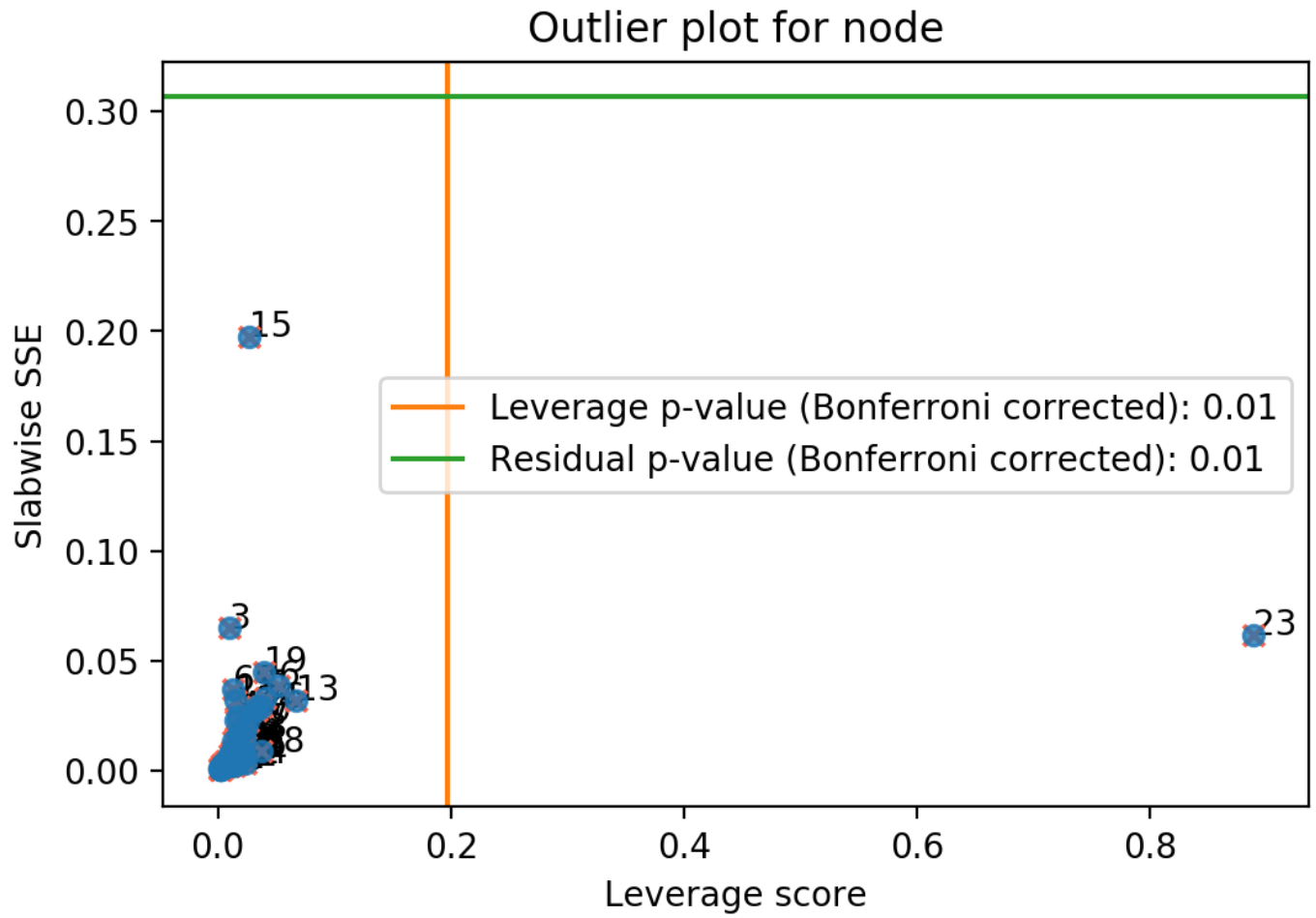
```
Decomposed dataset in 48s.
```

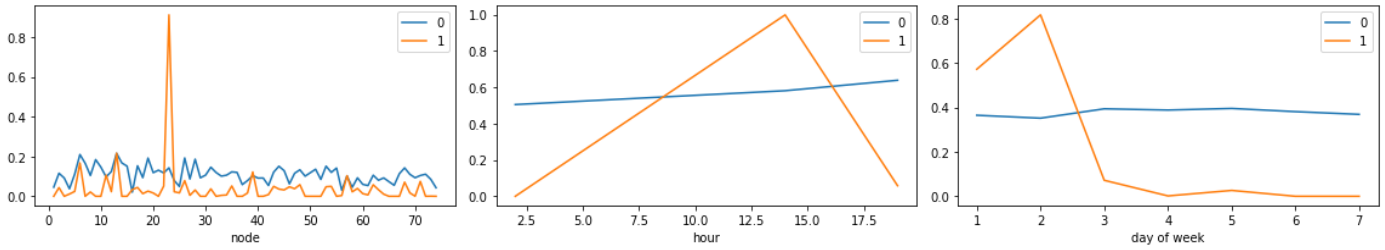




In [9]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[50], plo
```

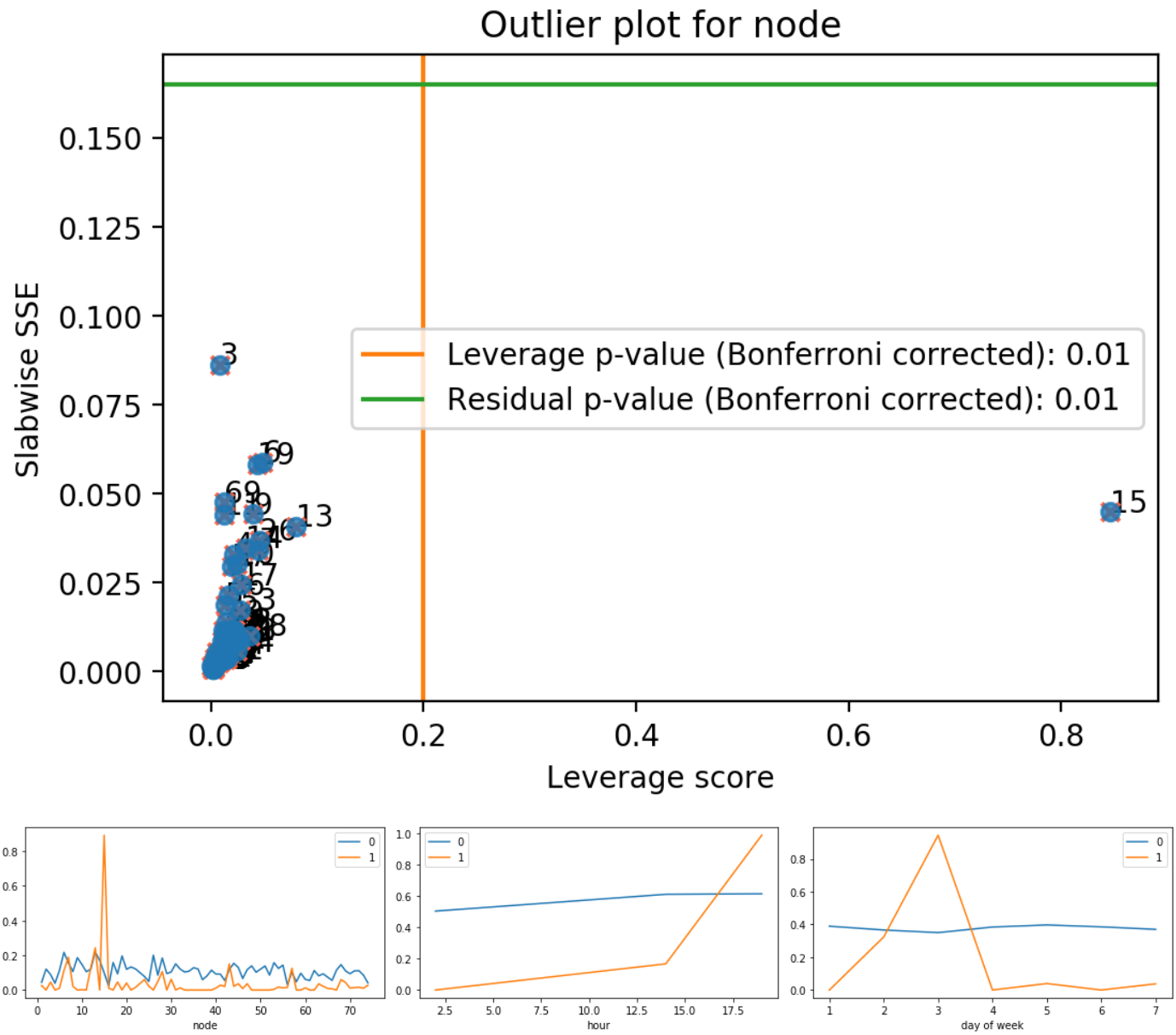
```
Decomposed dataset in 48s.
```





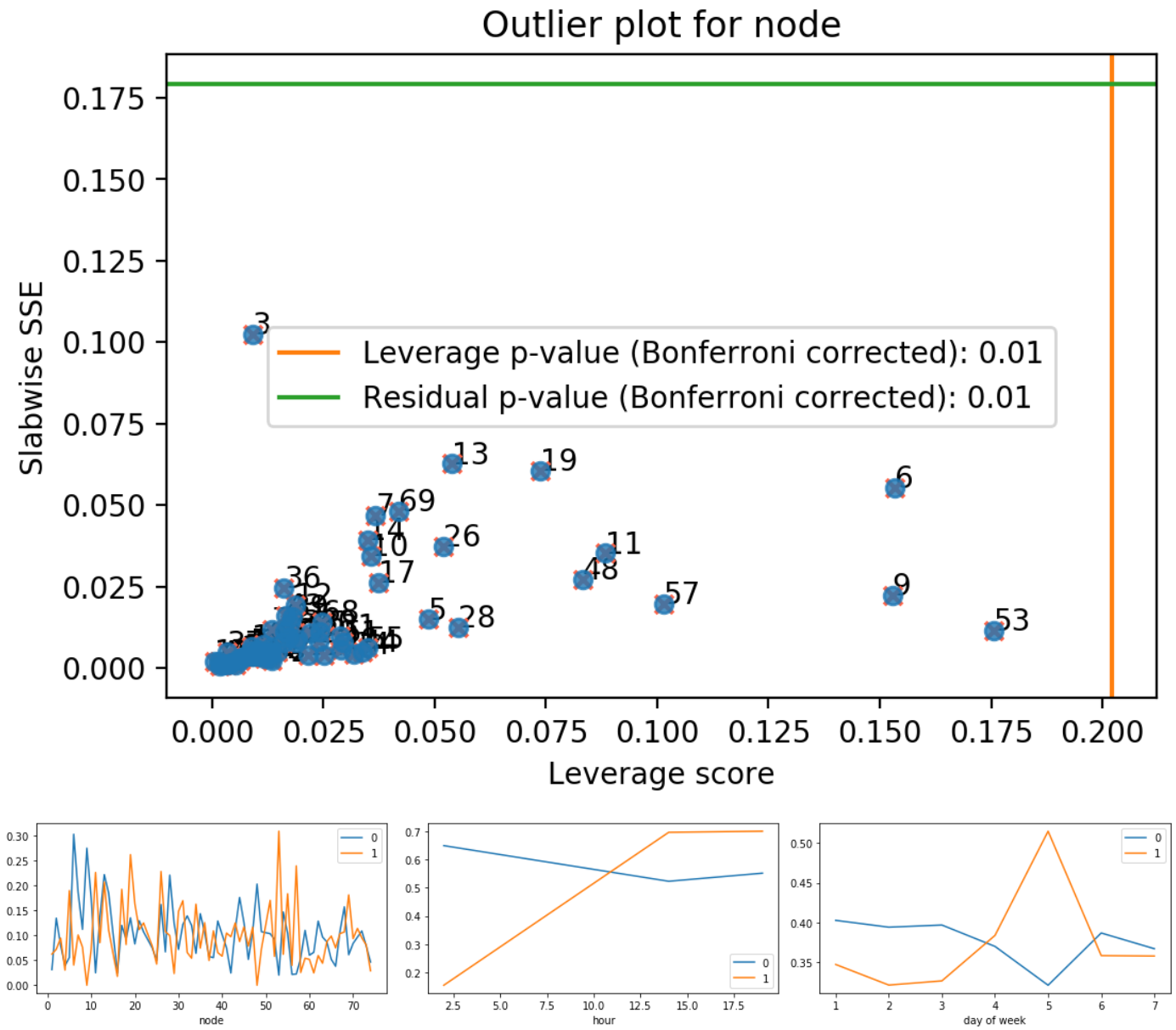
In [10]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[50, 23],
```

```
Decomposed dataset in 44s.
```



In [11]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[50, 15,
```

```
Decomposed dataset in 45s.
```



In [12]:
```
print("Dataset shape:", dataset.shape)
```

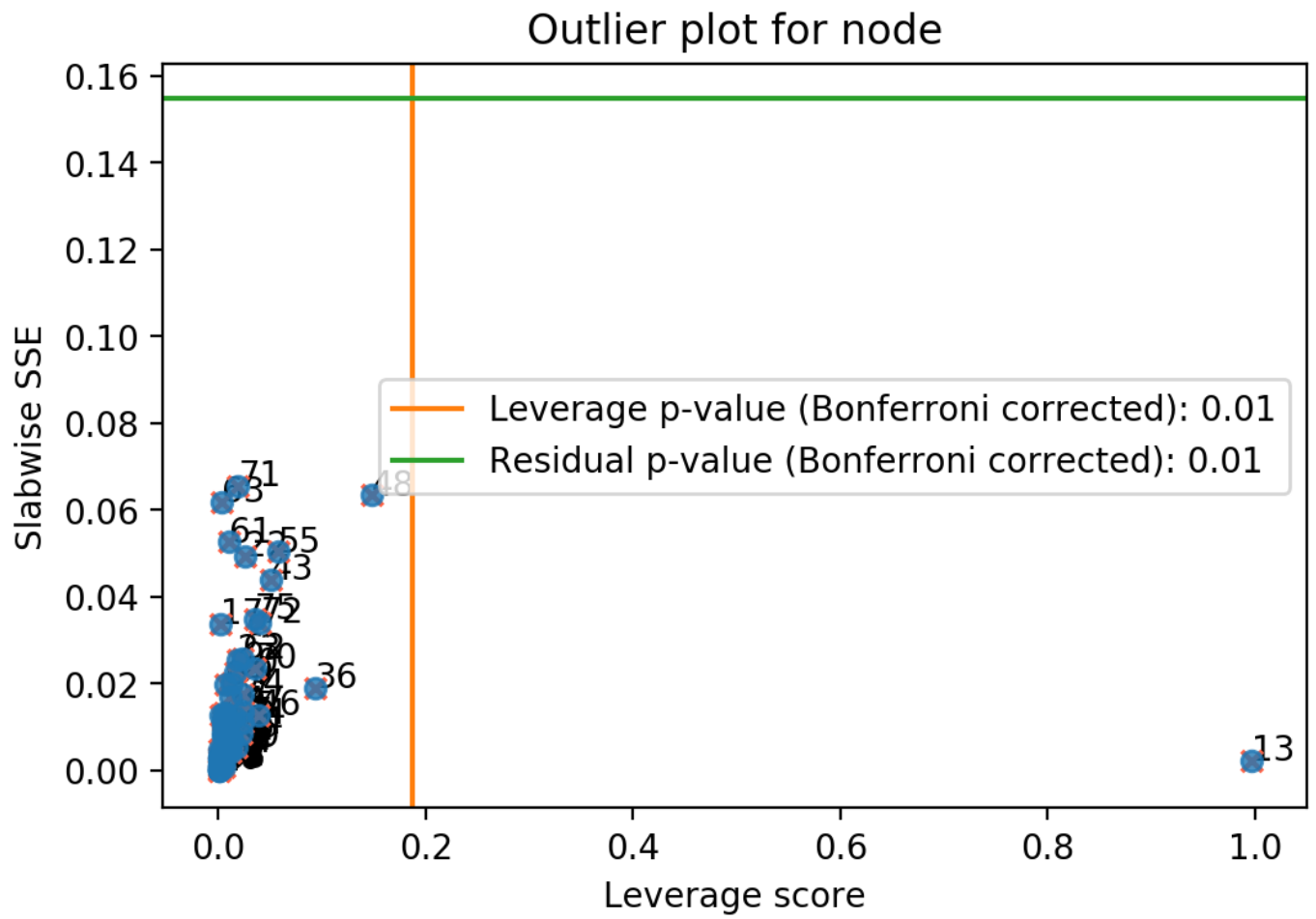
```
Dataset shape: (71, 3, 7)
```

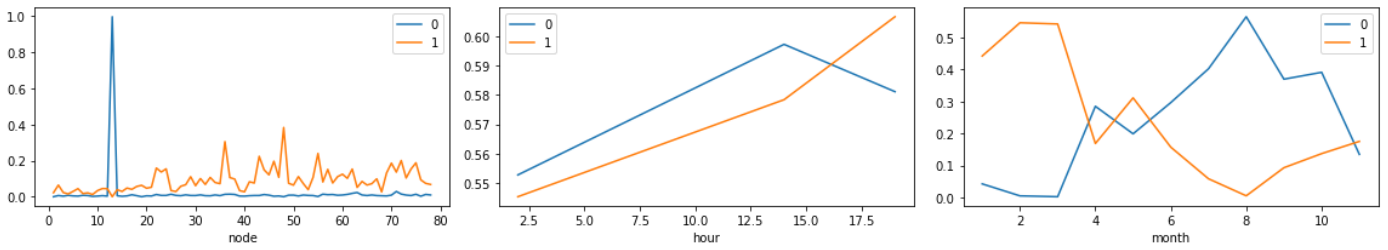
op1_dl_78_mode3_NHM.mat

In [13]:
```
datafile = "op1_dl_78_mode3_NHM.mat"
```

In [14]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[], plot=
```

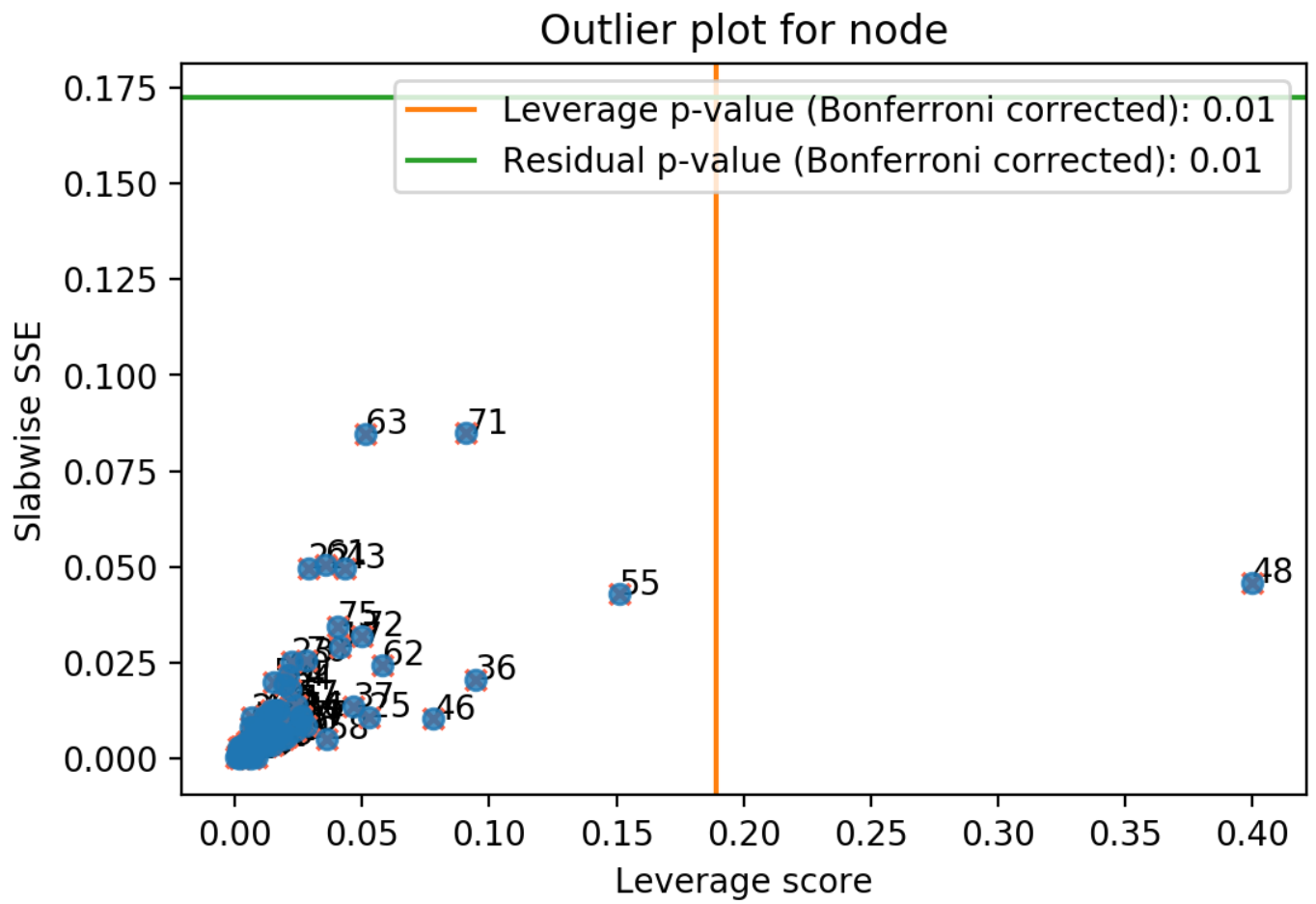
```
Decomposed dataset in 47s.
```

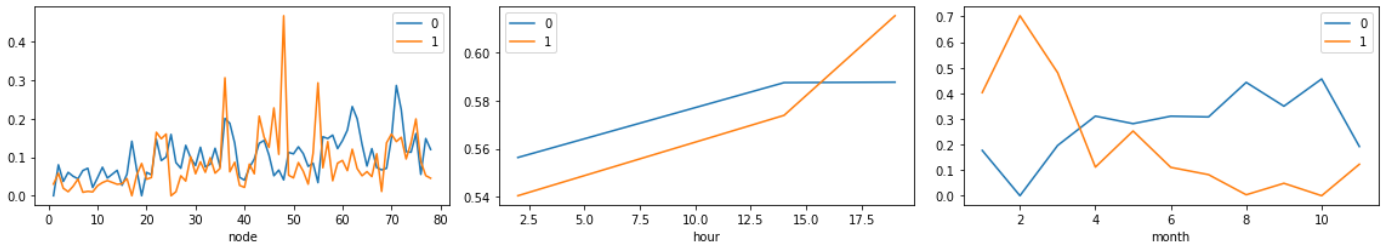




In [15]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[13], plo
```

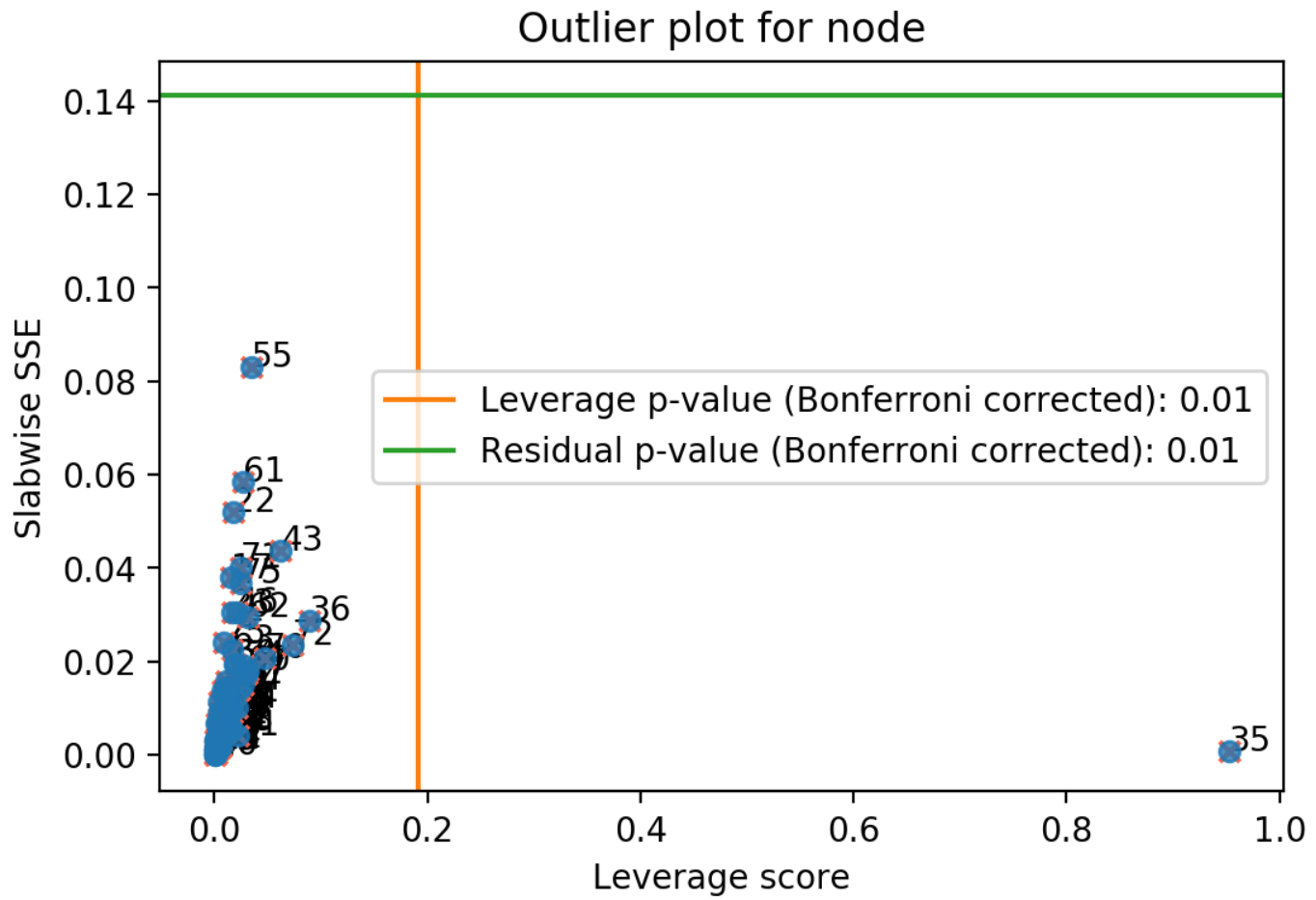
```
Decomposed dataset in 45s.
```

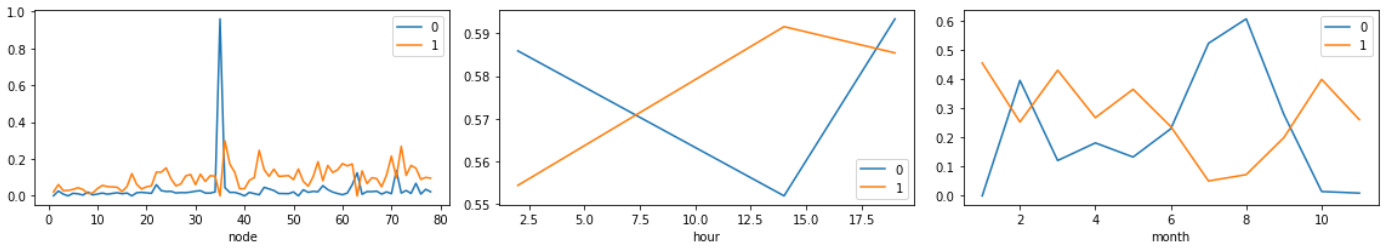




In [16]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[13, 48],
```

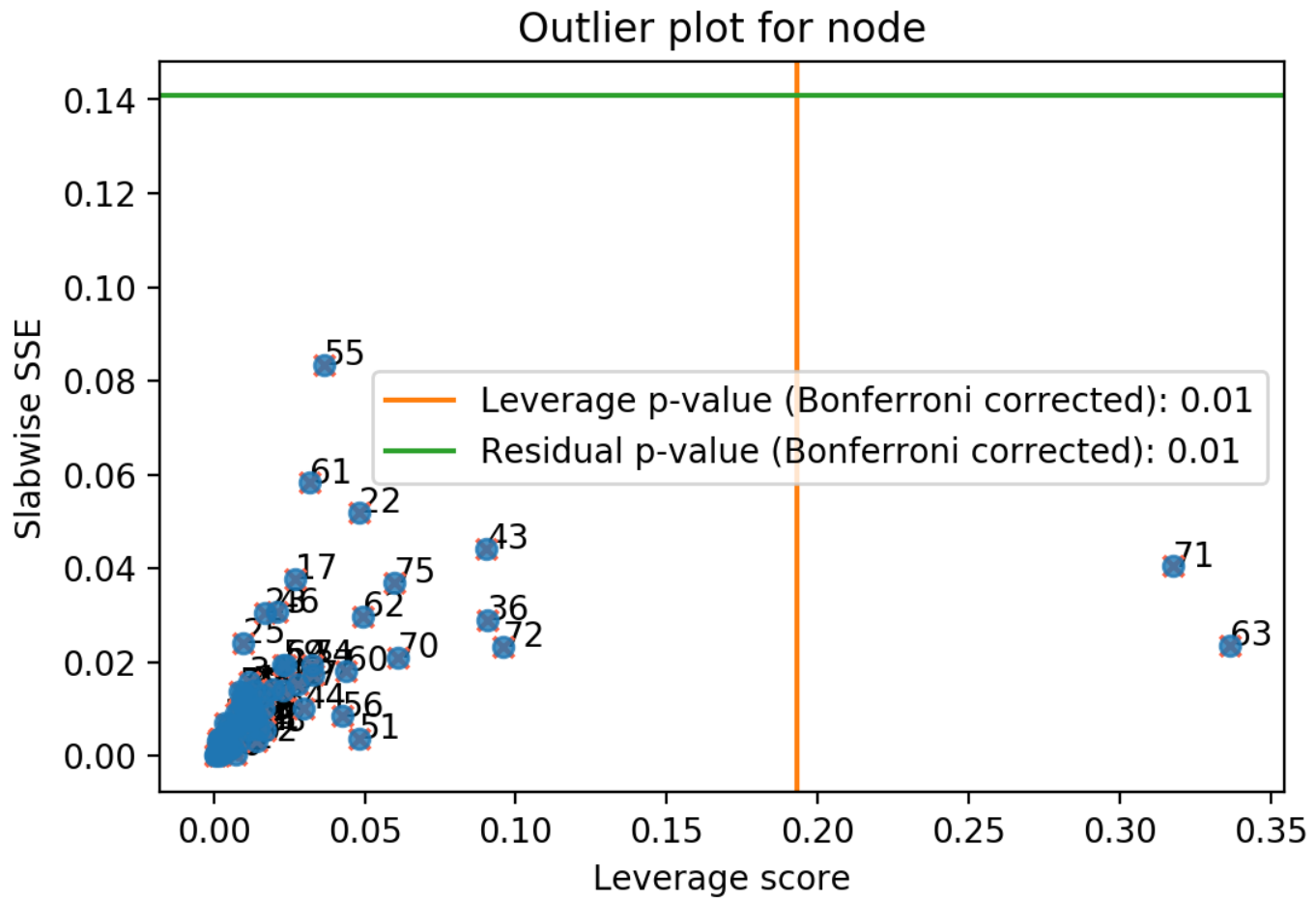
```
Decomposed dataset in 45s.
```

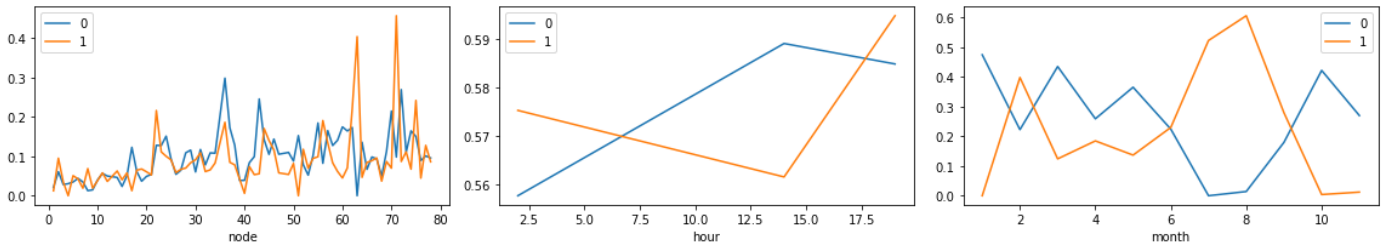




In [17]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[13, 35,
```

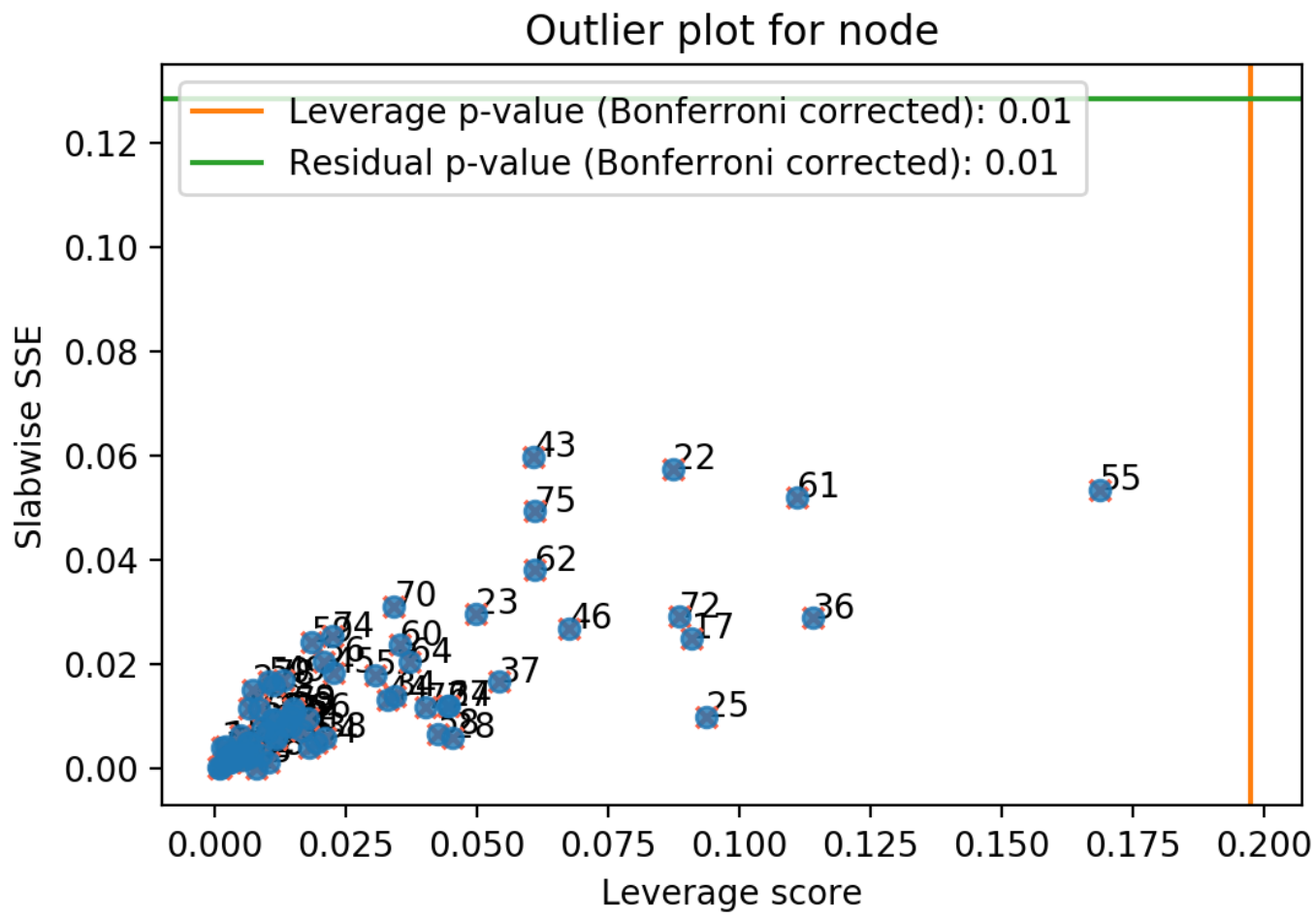
```
Decomposed dataset in 44s.
```

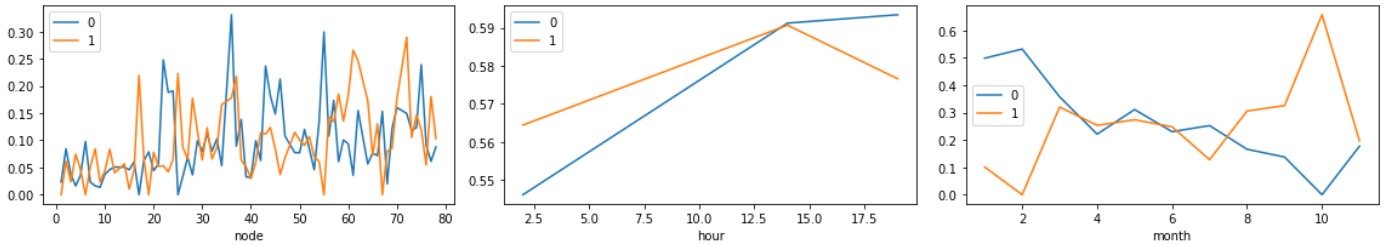




In [18]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[13, 35,
```

```
Decomposed dataset in 44s.
```





In [19]:
```
print("Dataset shape:", dataset.shape)
```

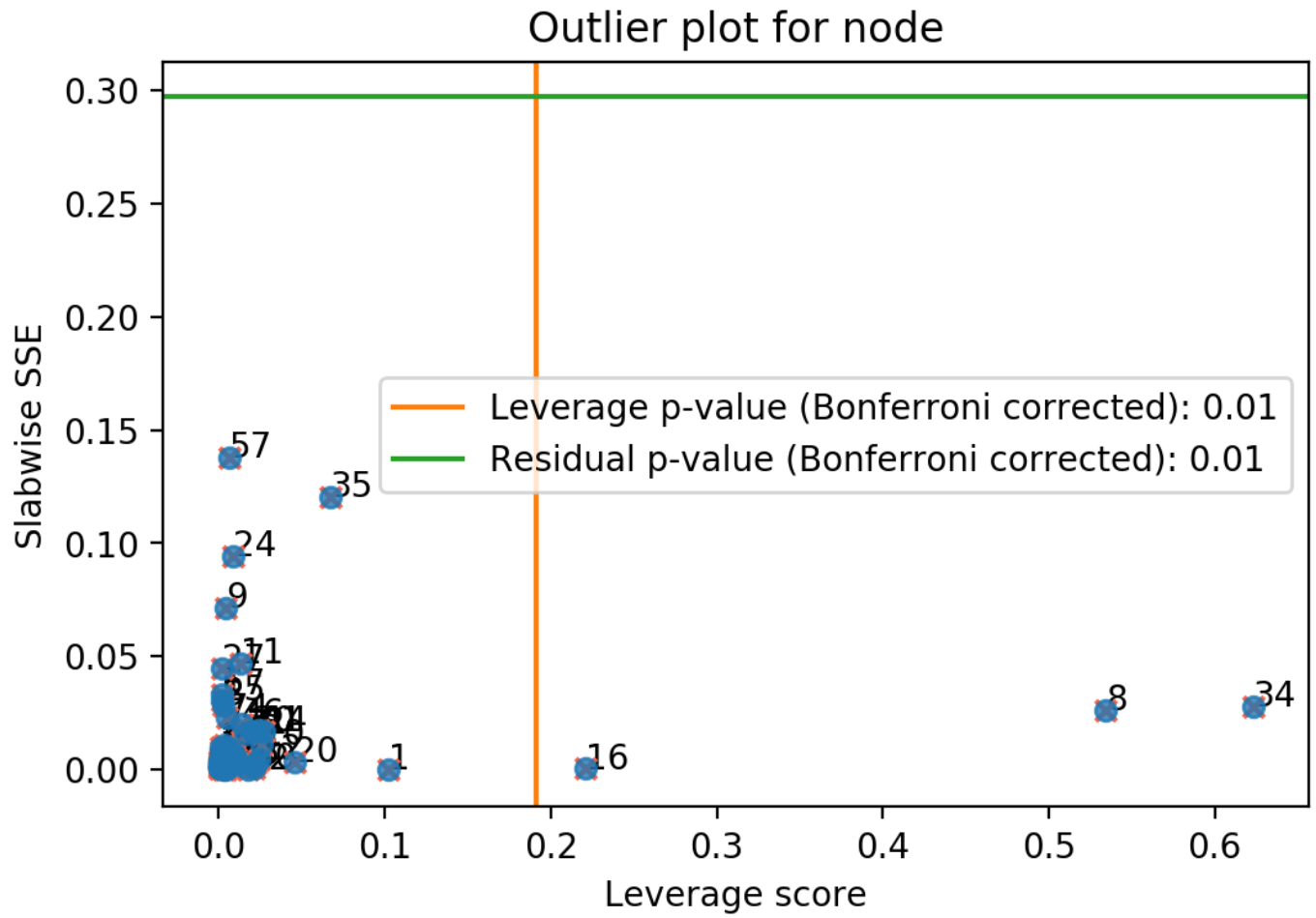
```
Dataset shape: (73, 3, 11)
```

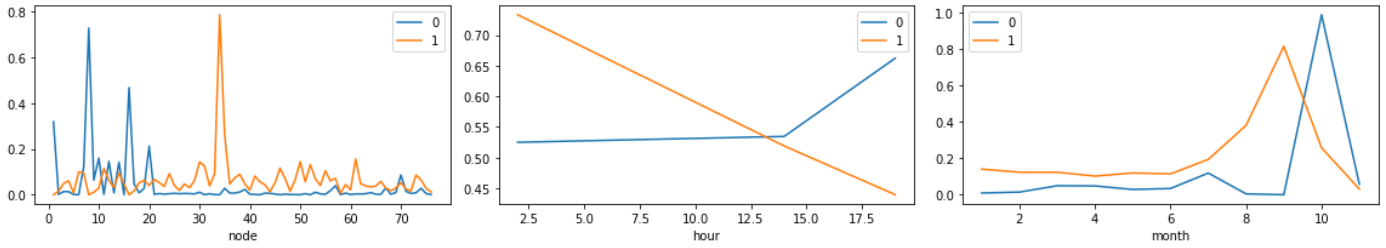
op2_dl_76_mode3_NHM.mat

In [20]:
```
datafile = "op2_dl_76_mode3_NHM.mat"
```

In [21]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[], plot=
```

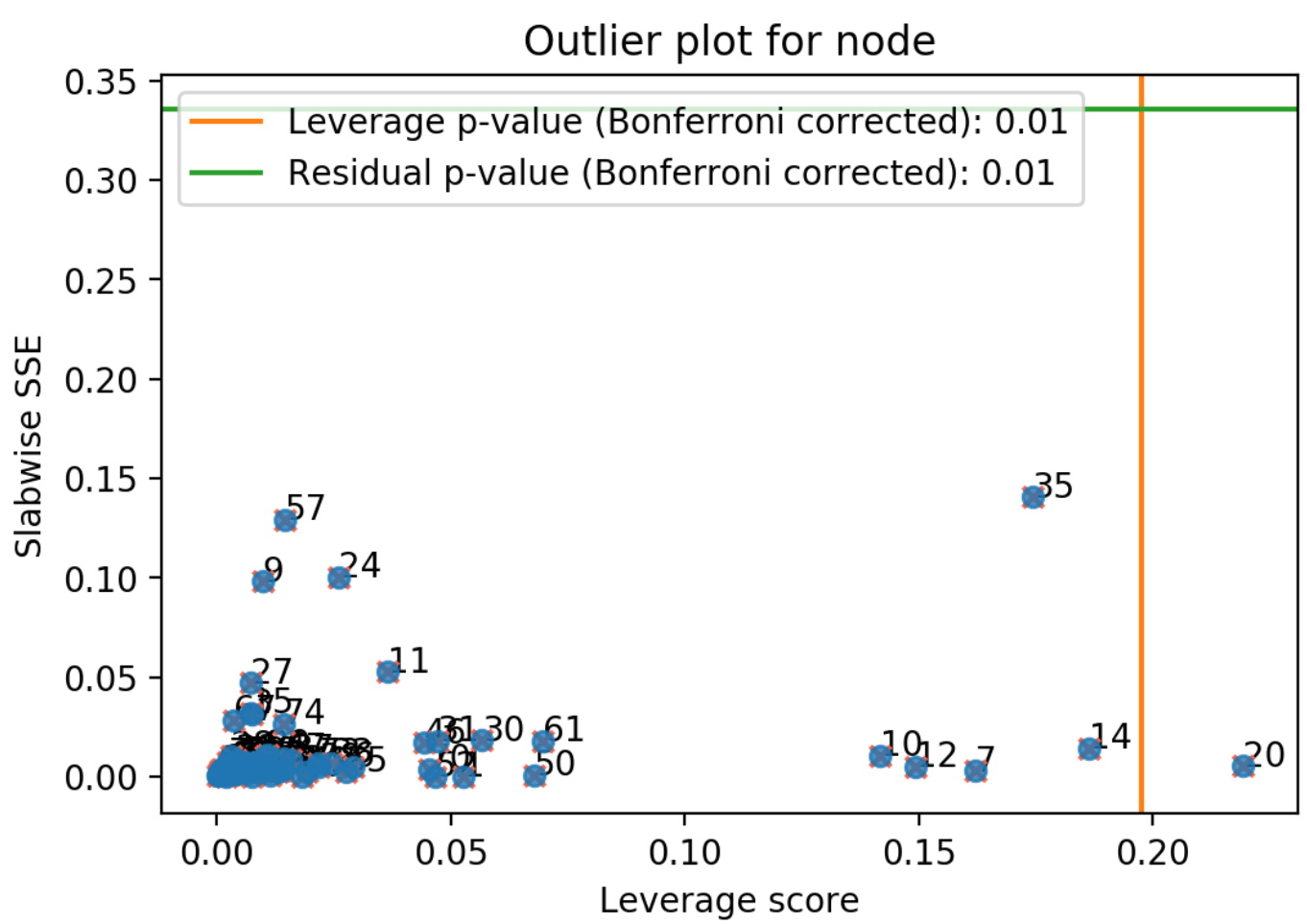
```
Decomposed dataset in 60s.
```

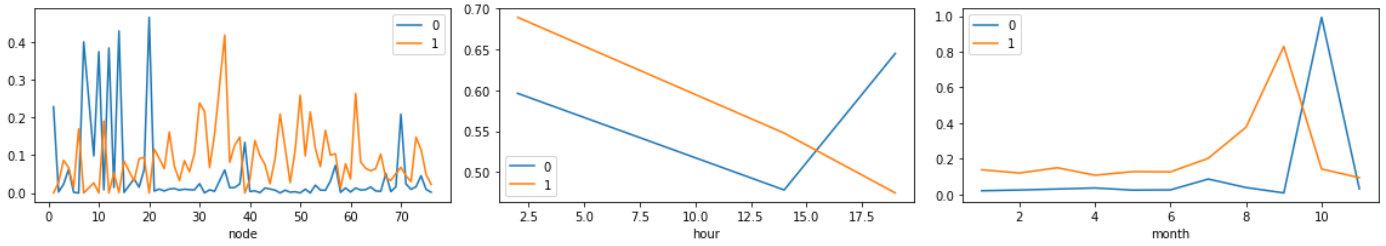




In [22]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[34, 8, 1
```

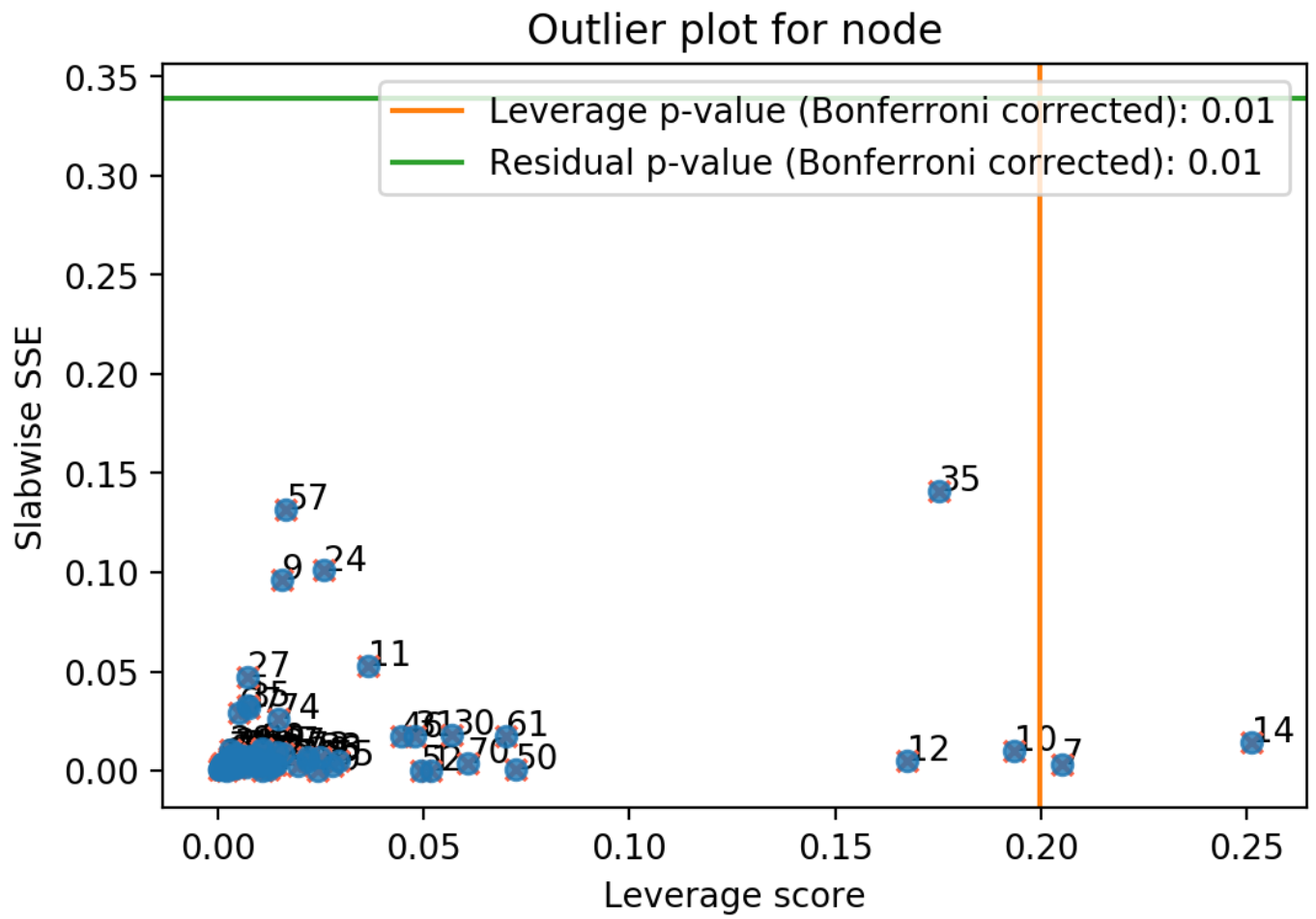
```
Decomposed dataset in 53s.
```

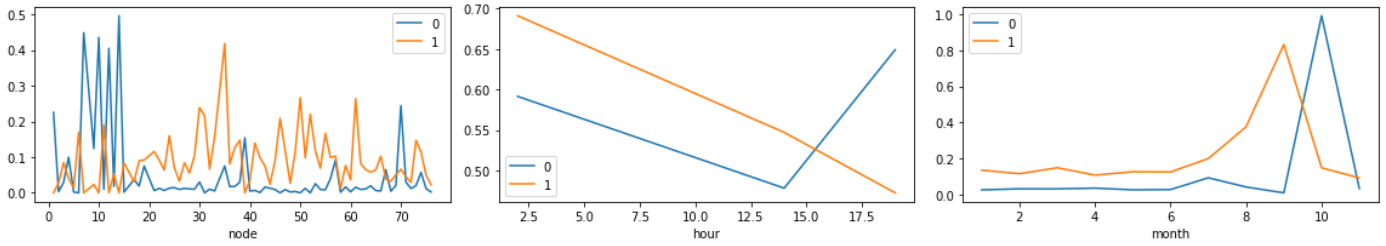




In [23]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[34, 8, 1
```

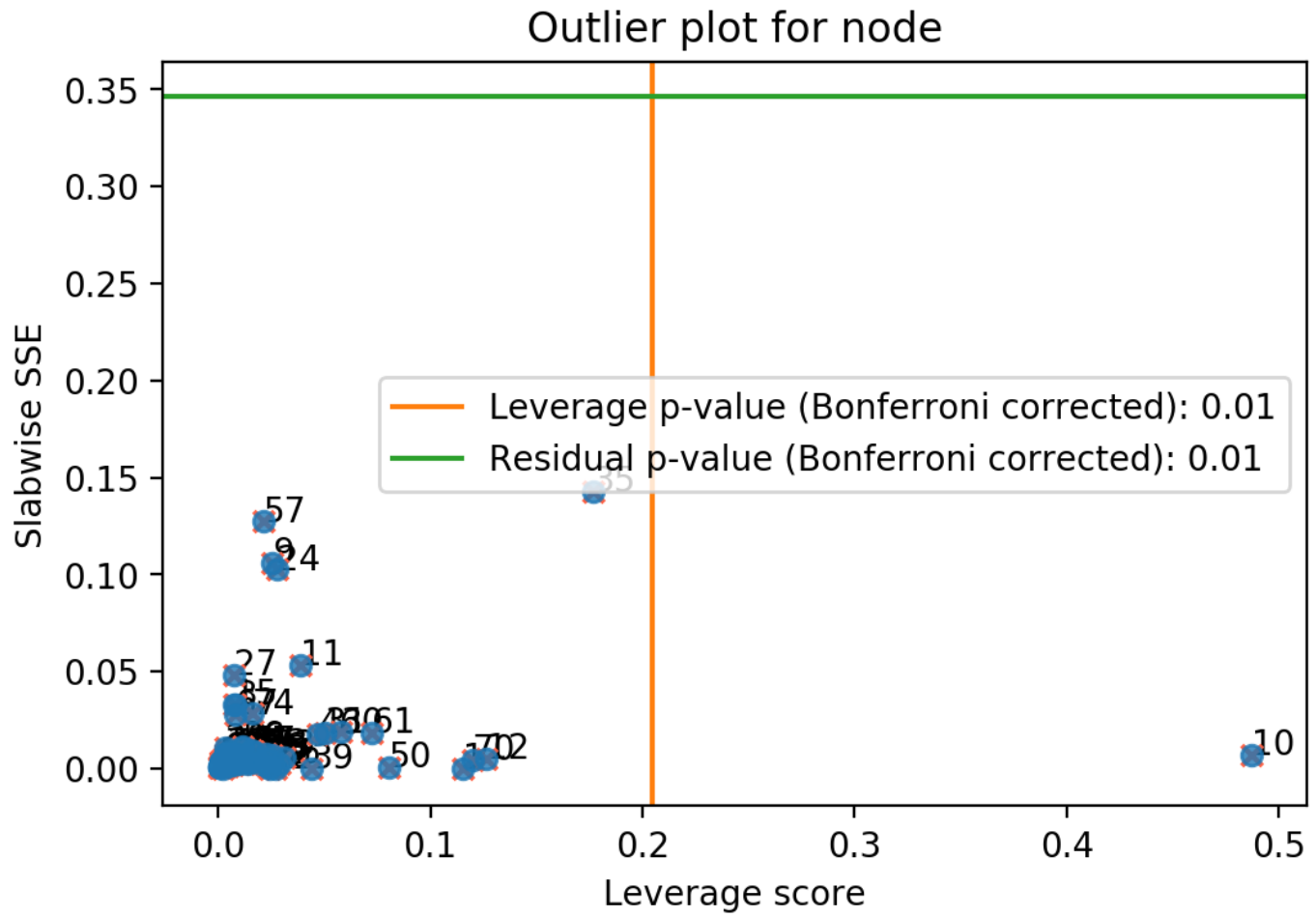
```
Decomposed dataset in 59s.
```

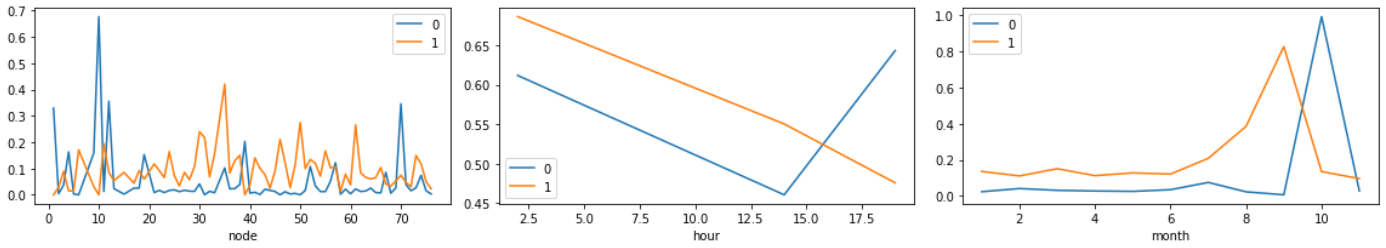




In [24]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[34, 8, 1
```

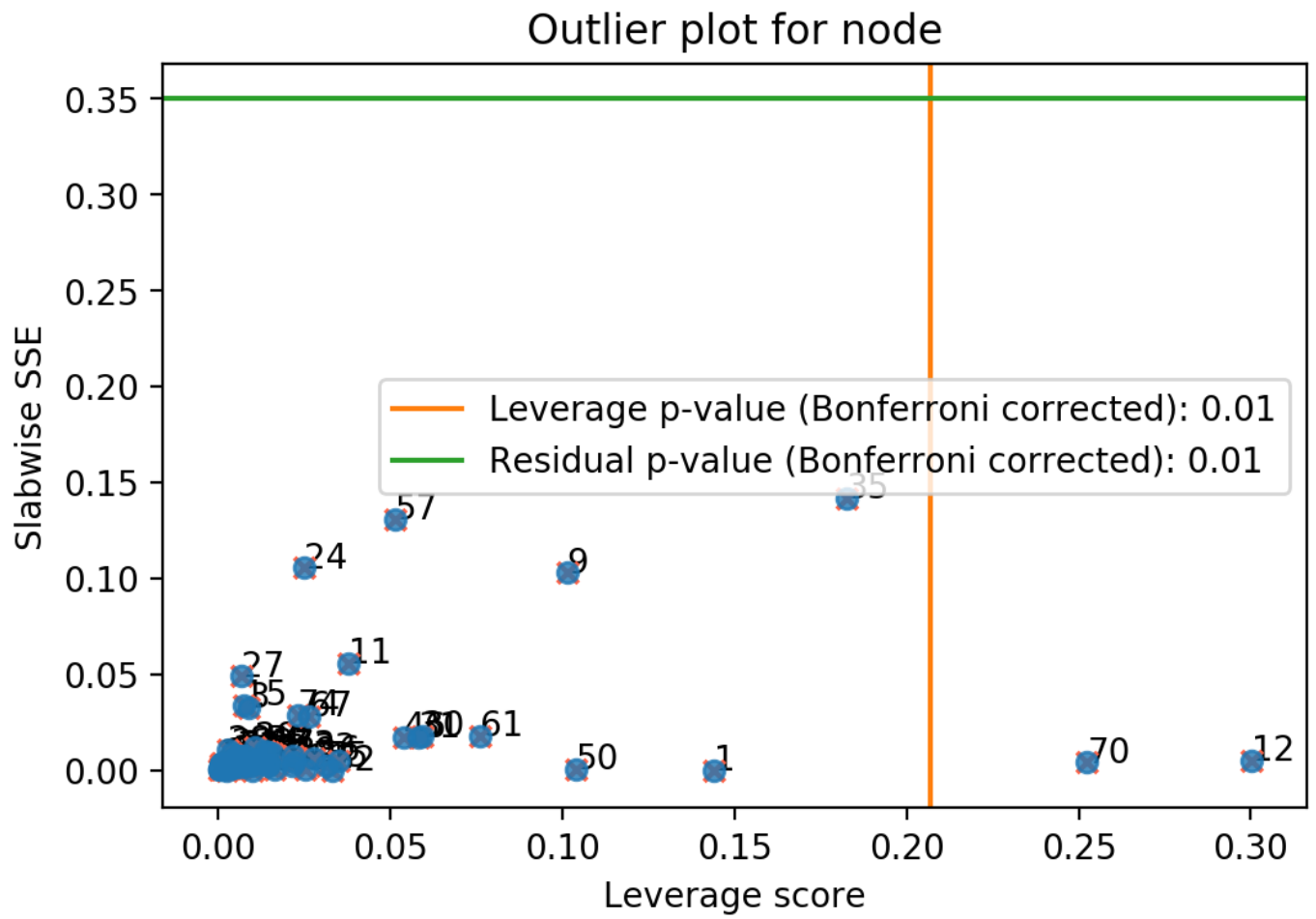
```
Decomposed dataset in 59s.
```

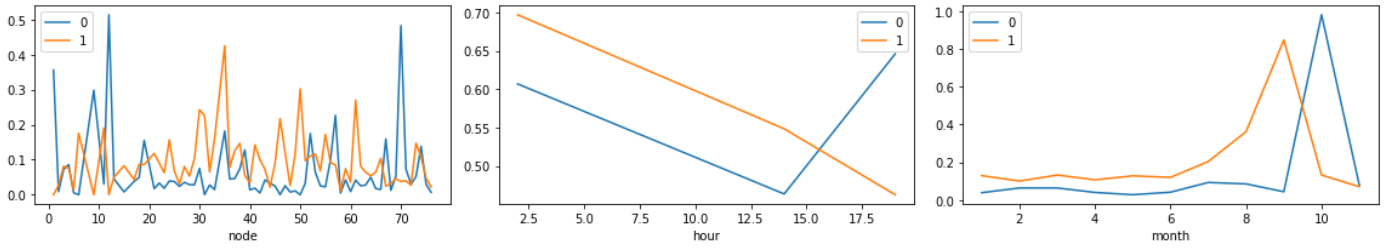




In [25]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(datafile, outlier_nodes=[34, 8, 1
```

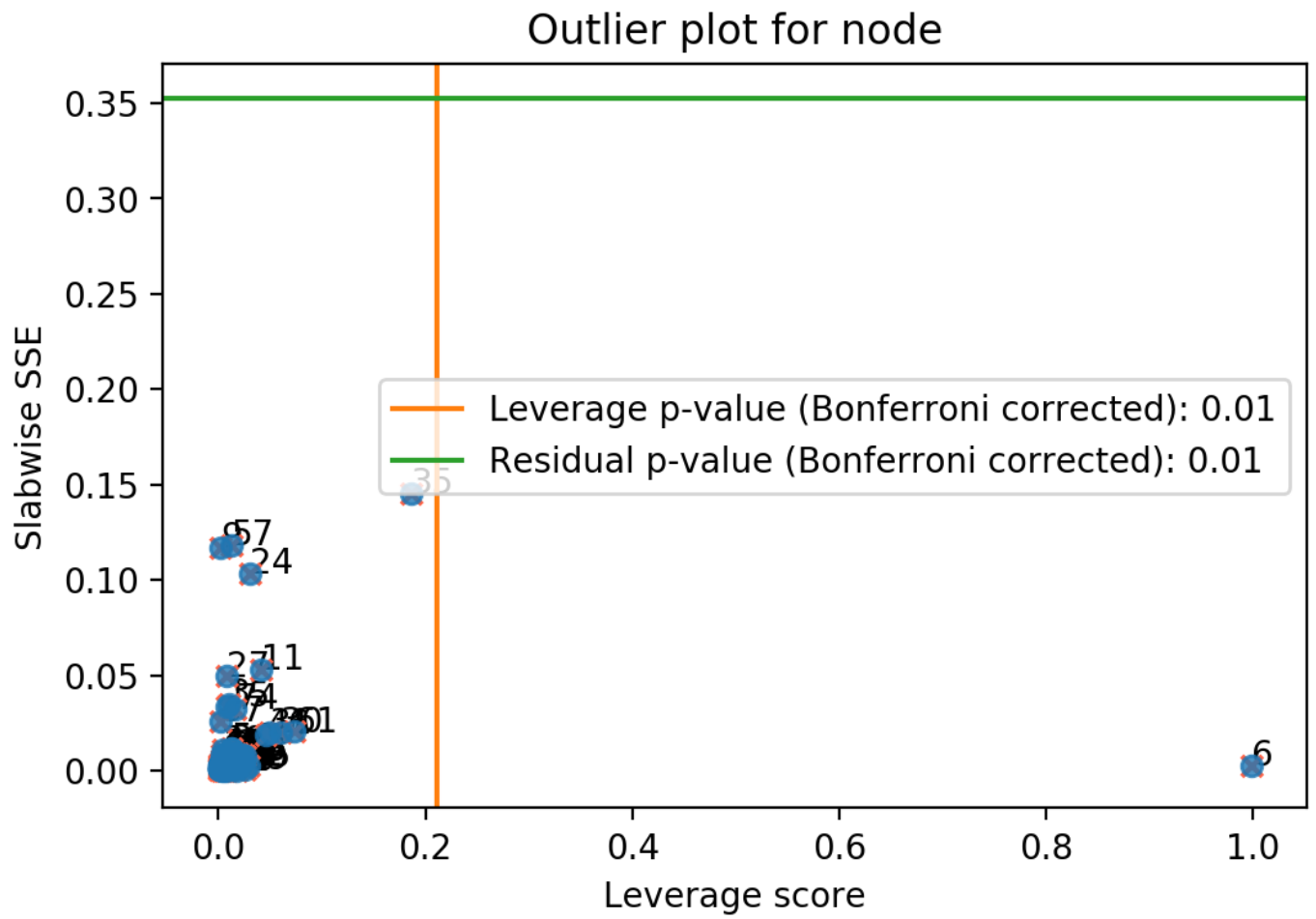
```
Decomposed dataset in 53s.
```

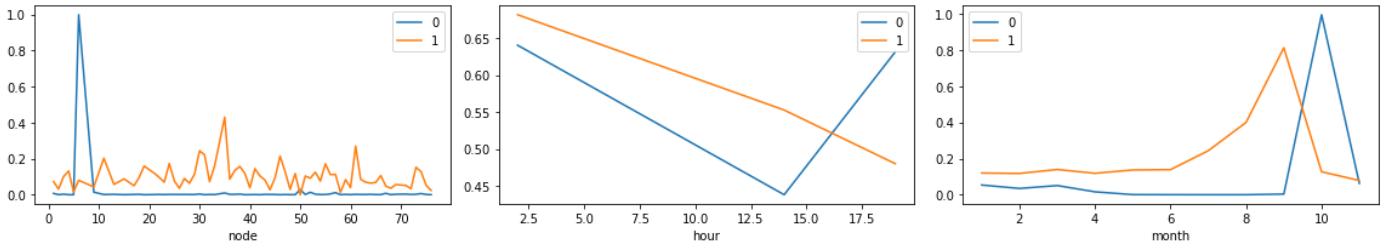




In [26]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(
    datafile, outlier_nodes=[34, 8, 16, 20, 7, 14, 10, 12, 70], plot=True
)
```

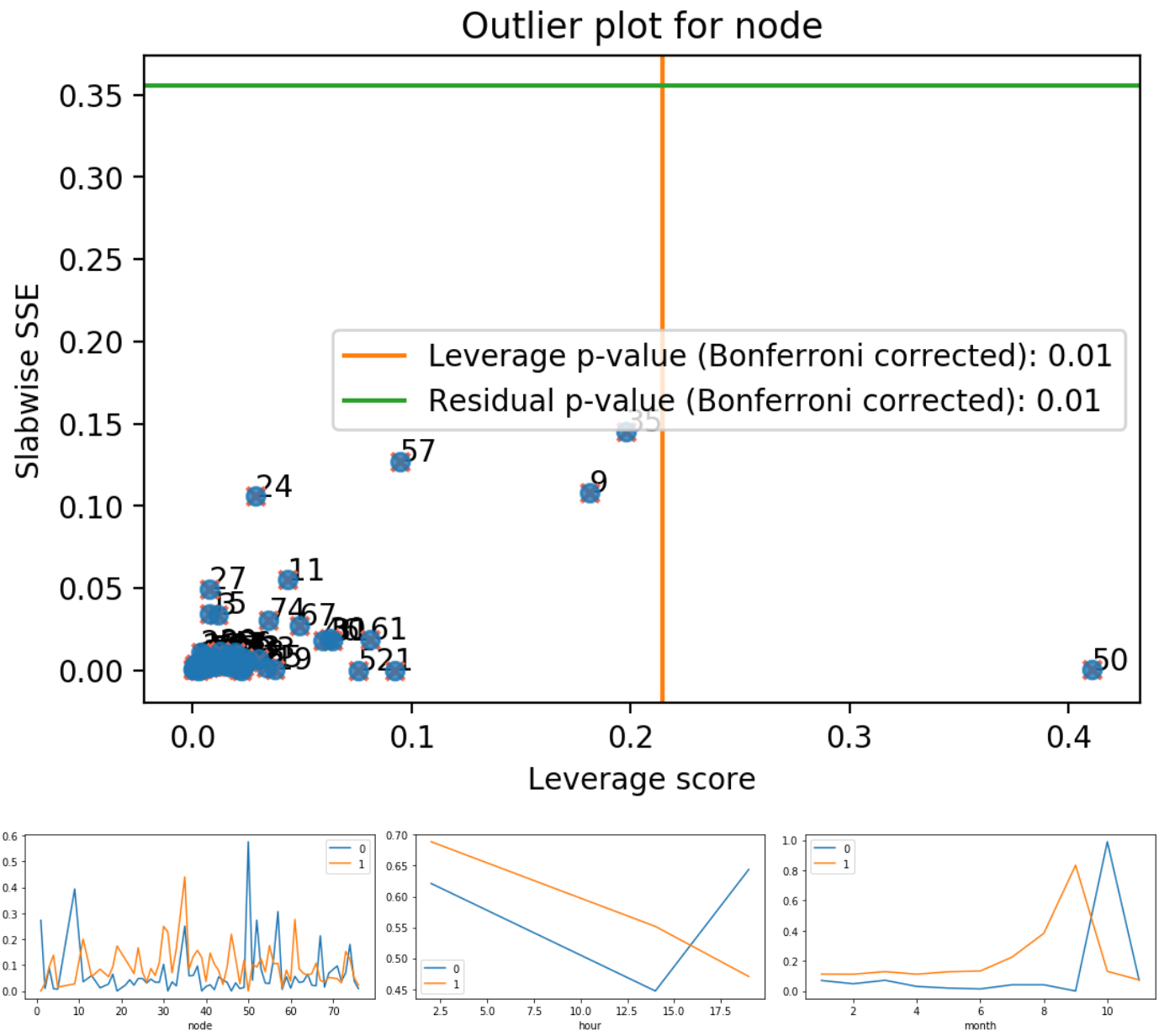
```
Decomposed dataset in 50s.
```





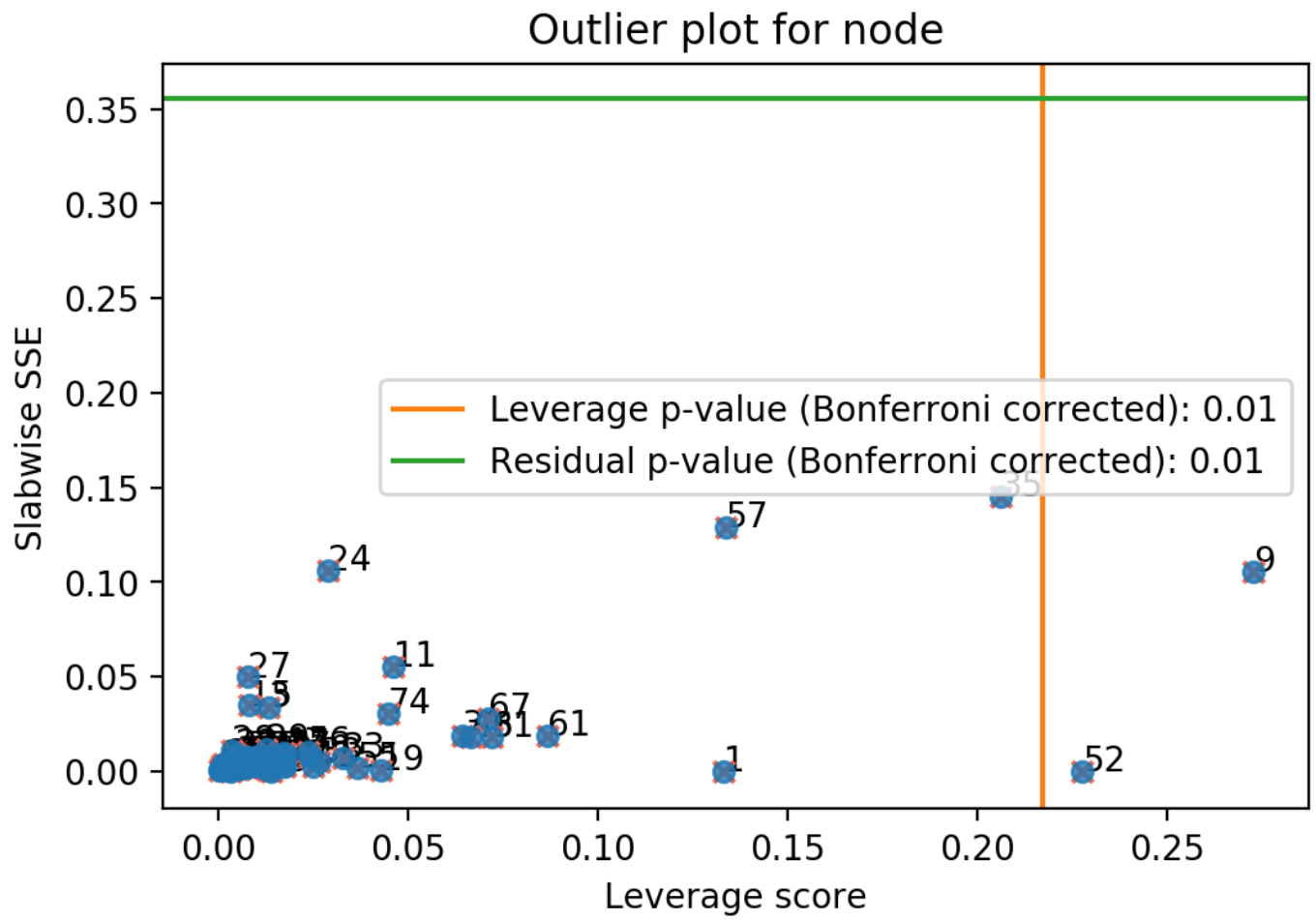
In [27]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(
    datafile, outlier_nodes=[34, 8, 16, 20, 7, 14, 10, 12, 70, 6], plot=True
)
```

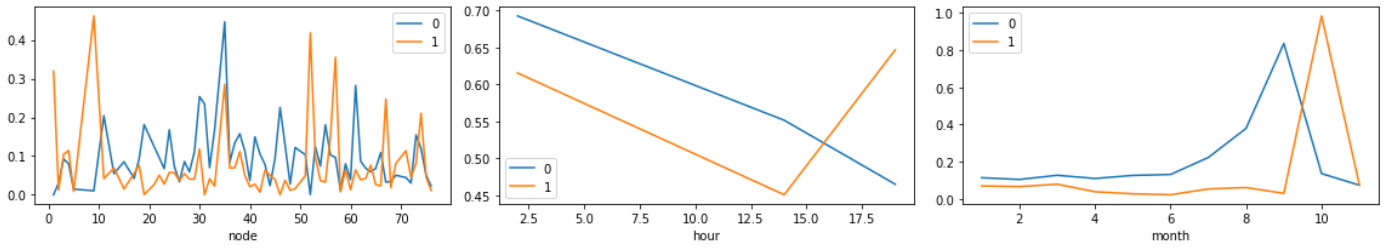
```
Decomposed dataset in 46s.
```



In [28]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(
    datafile, outlier_nodes=[34, 8, 16, 20, 7, 14, 10, 12, 70, 6, 50], plot=True
)
```

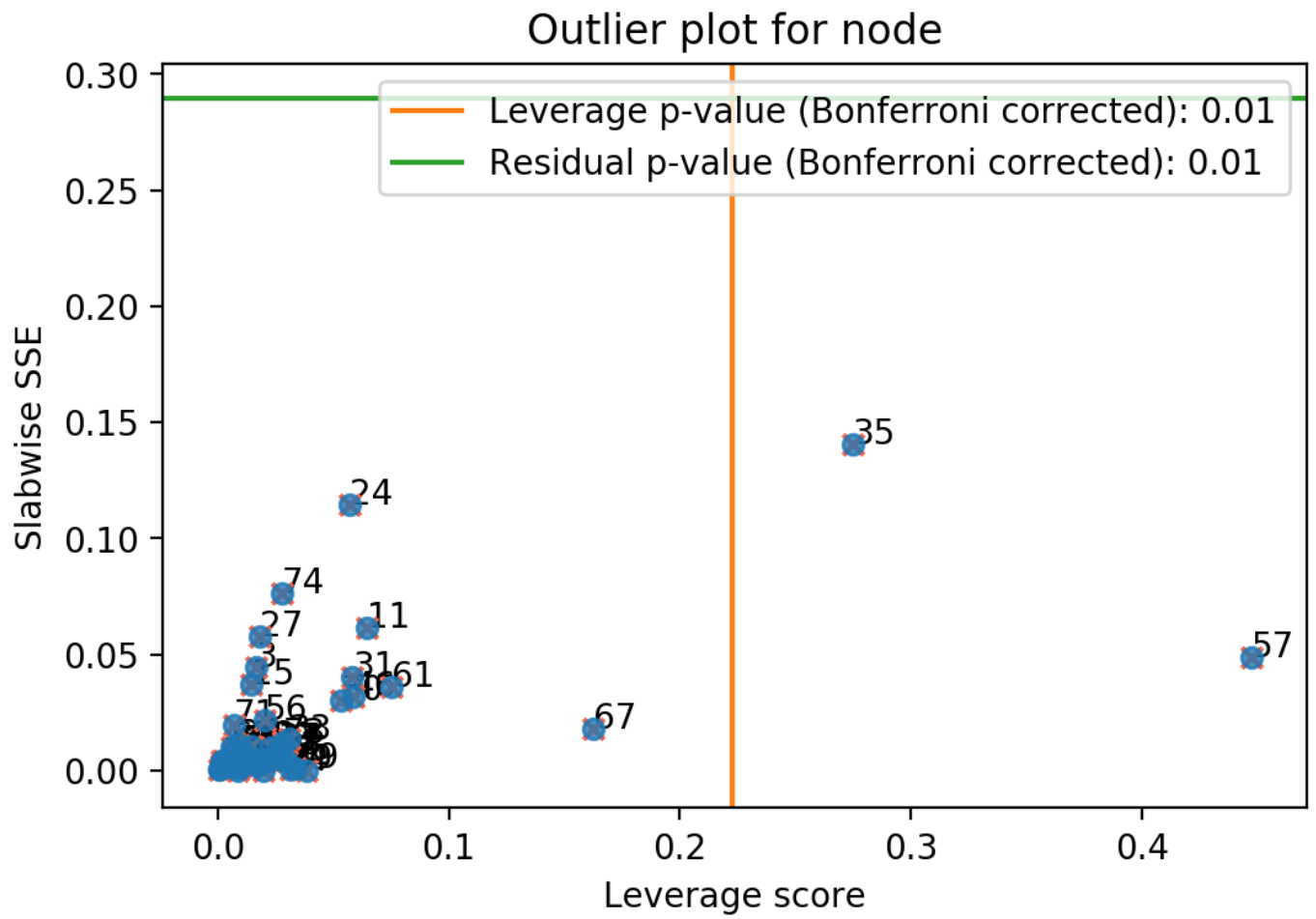
```
Decomposed dataset in 49s.
```





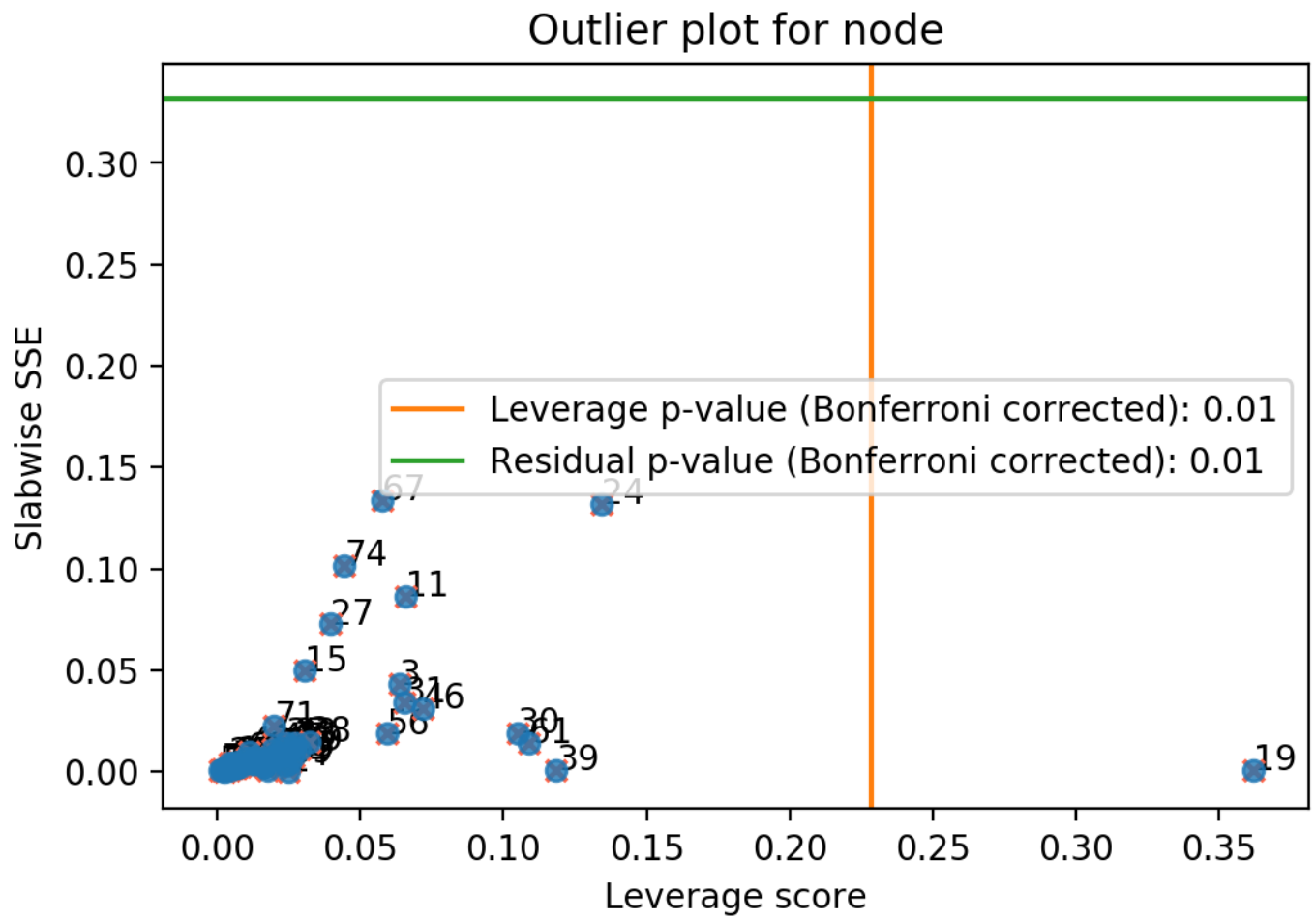
In [29]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(
    datafile, outlier_nodes=[34, 8, 16, 20, 7, 14, 10, 12, 70, 6, 50, 9, 52], plot=True
)
```

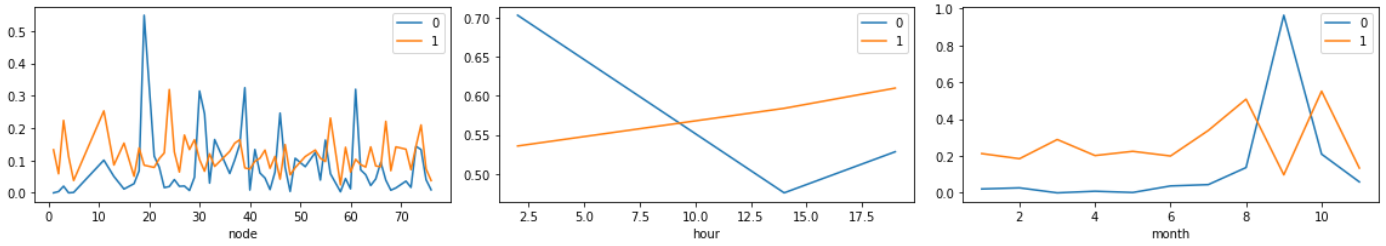
```
Decomposed dataset in 48s.
```



In [30]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(
    datafile, outlier_nodes=[34, 8, 16, 20, 7, 14, 10, 12, 70, 6, 50, 9, 52, 57, 35], pl
)
```

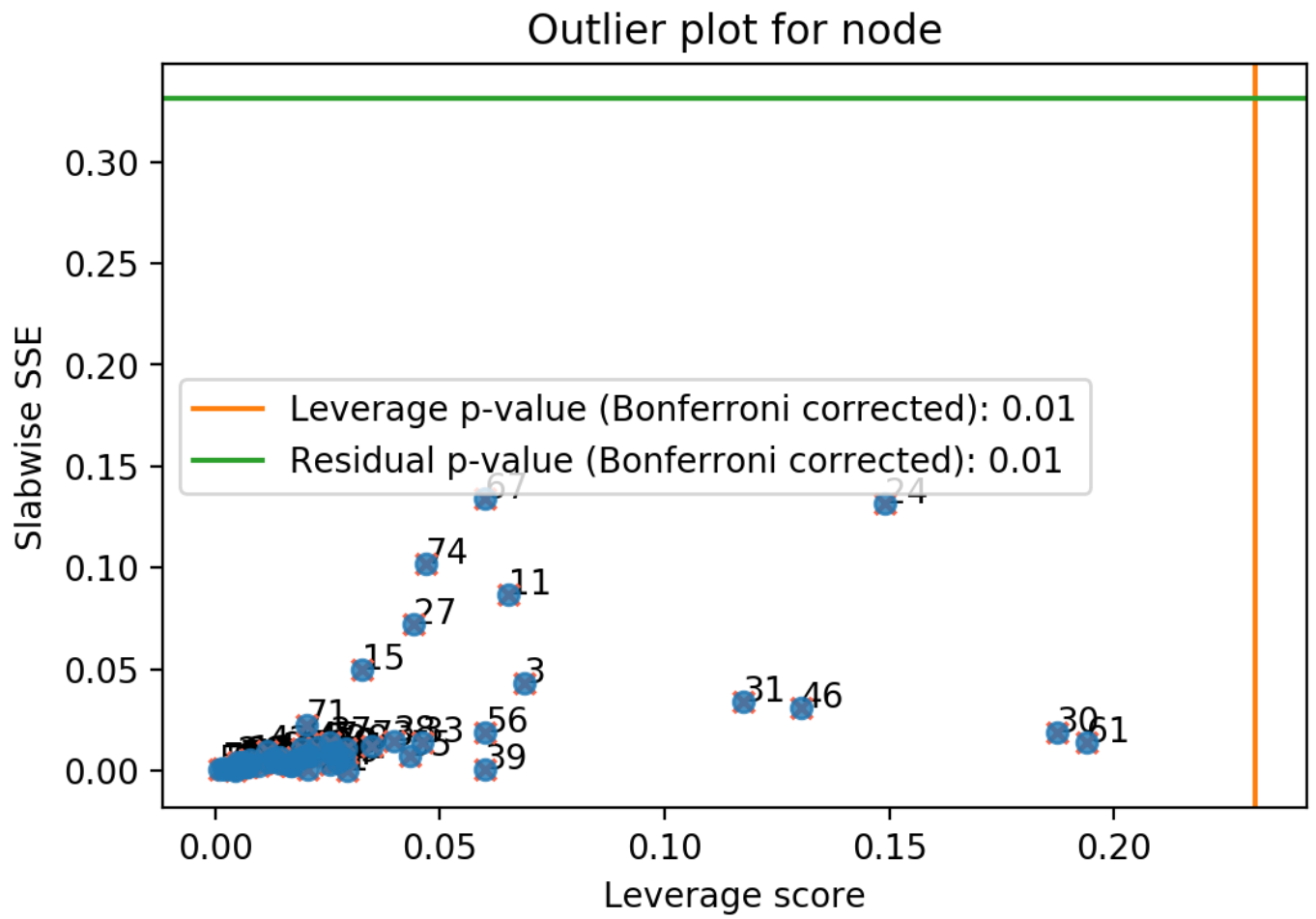
```
Decomposed dataset in 48s.
```

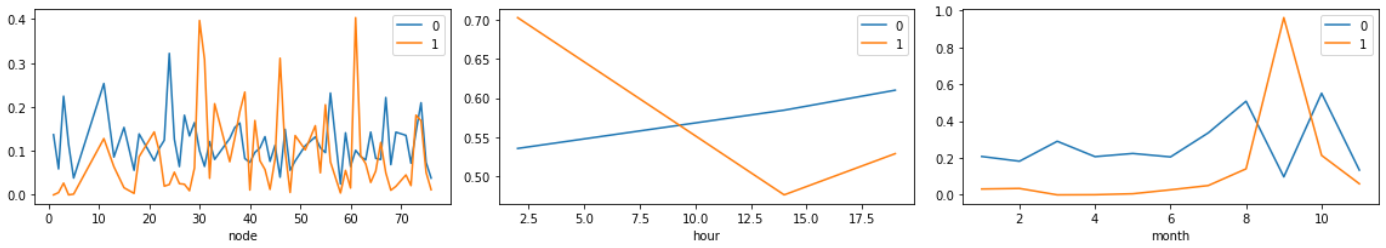




In [31]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(
    datafile, outlier_nodes=[34, 8, 16, 20, 7, 14, 10, 12, 70, 6, 50, 9, 52, 57, 35, 19]
)
```

```
Decomposed dataset in 46s.
```





In [32]:
```
print("Dataset shape:", dataset.shape)
```

```
Dataset shape: (60, 3, 11)
```

Paper plot

Below, we create the plot used in the paper datafile = "op2_dl_76_mode3_NHM.mat"

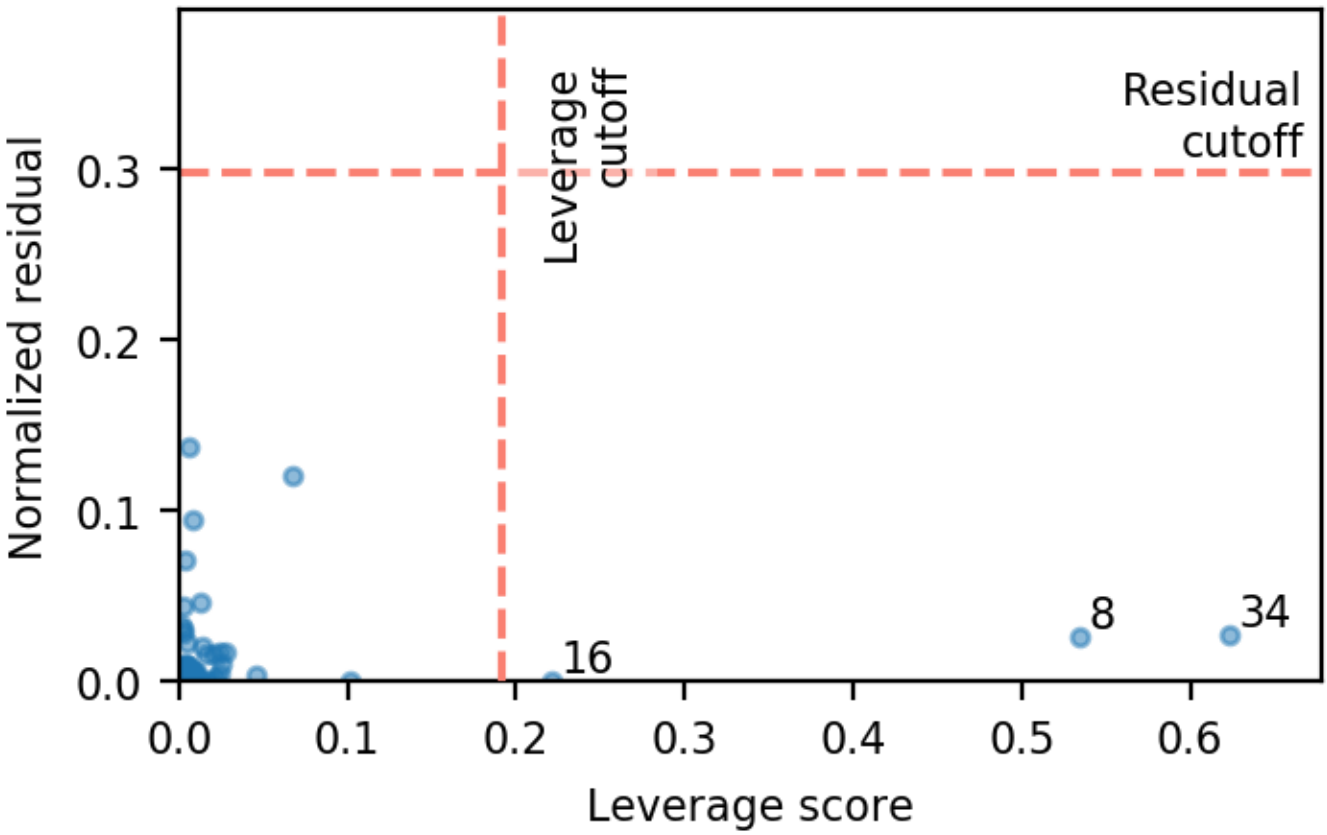
In [33]:
```
cp_tensor, dataset, leverage, residuals = run_analysis(
    datafile, outlier_nodes=[], plot=False
)
```

```
Decomposed dataset in 62s.
```

In [34]:
```
leverage_threshold = tlviz.outliers.get_leverage_outlier_threshold(leverage, 'bonferroni
residual_threshold = tlviz.outliers.get_slabwise_sse_outlier_threshold(residuals, 'bonfe
```

In [35]:
```
cutoff_color = "salmon"

fig, ax = plt.subplots(figsize=(3.5, 3.5/1.6), dpi=200)
ax.scatter(leverage, residuals, s=10, alpha=0.5)
ax.axvline(leverage_threshold, linestyle='--', zorder=3, color=cutoff_color)
ax.axhline(residual_threshold, linestyle='--', color=cutoff_color)
```

```
for node, l, r in zip(cp_tensor[1][0].index, leverage, residuals):
    if l > leverage_threshold or r > residual_threshold:
        ax.annotate(node, (l + 0.005, r + 0.005), fontsize=8)



ax.set_ylim(0, ax.get_ylim()[1] + 0.08)
ax.set_xlim(left=0, right=ax.get_xlim()[1] + 0.02)

xlim = ax.get_xlim()
ax.text(
    xlim[1] - 0.01,
    residual_threshold + 0.01,
    "Residual\ncutoff",
    horizontalalignment="right",
    fontsize=8
)

ylim = ax.get_ylim()
ax.text(
    leverage_threshold + 0.08,
    ylim[1] - 0.03,
    "Leverage\ncutoff",
    verticalalignment="top",
    horizontalalignment="right",
    rotation=90,
    bbox=dict(facecolor=(1., 1., 1., 0.4), edgecolor=(1., 1., 1., 0.)),
    zorder=2,
    fontsize=8
)
ax.set_xlabel("Leverage score", size=8)
ax.set_ylabel("Normalized residual", size=8)
fig.subplots_adjust(left=.13, bottom=.18, right=0.98, top=0.98, wspace=None, hspace=None

for tick in ax.xaxis.get_majorticklabels():
    tick.set_fontsize(8)
for tick in ax.yaxis.get_majorticklabels():
    tick.set_fontsize(8)


fig.savefig("outlier_detection.png")
fig.savefig("outlier_detection.pdf")
```


Normalized residual
0.0
0.1
0.2
0.3
Leverage
cutoff
Residual
cutoff
16
8
34
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Leverage score