Analysing Anomaly Detection Methods For Time Series Using R

Structure

1. Context and Motivation
2. Anomalies and Outcomes
3. Detection Methods and Analysing them using R

Context and Motivation

Welcome | Sign in or Register

Gas & electricity | Broadband, TV & home phone | Mobiles | Banking | Insurance

Switch energy & save

It's quick and easy to save up to £679/yr*

Compare your gas & electricity

Fixed energy plan ending? ▸

Gas and electricity ▸

Electricity prices ▸

Broadband, TV, landline

Fast, reliable broadband from £2.50

Compare broadband deals

Broadband deals ▸

Broadband & home phone ▸

Broadband & TV packages ▸

Mobiles

Latest deals from only £10

Credit cards

Compare credit cards

Car insurance

Quick quote and save today!

Mortgages

Compare mortgage rates

Situations

1. A tv show talks about energy savings and causes a big spike in online traffic
2. A bug is released onto an important page in our funnel, causing a big drop in users completing our energy process
3. A product is taken off our energy tariff price comparison page, causing a drop in users proceeding to the next step.
4. News stories over several days causes an increase in Google traffic to one of our guides pages.

How these changes are often noticed

1. ‘Oh, traffic has gone up on the site!’
2. ‘Why has this KPI dropped in the last two hours?’
3. ‘Hmm, this landing page started getting loads of traffic 3 days ago’

Challenges

- Real time data streams can't be checked constantly in a reliable way by people with many other things to do.
- It is time consuming to check thousands of time series regularly for changes

Anomalies and Outcomes

Anomaly: What is it?

- Anomaly: something that is different from what is expected, or not in agreement with something else.
- Expected ranges and values are defined differently by different methods.
- Our definition of anomaly will change with the methods.

Outcomes

1. Quickly and reliably detect large and important anomalies
2. Eventually and reliably detect less important anomalies
3. To indicate where to look to find out why the anomaly occurred
4. Do the above without many false positives

Detection Methods and Analysing them in R

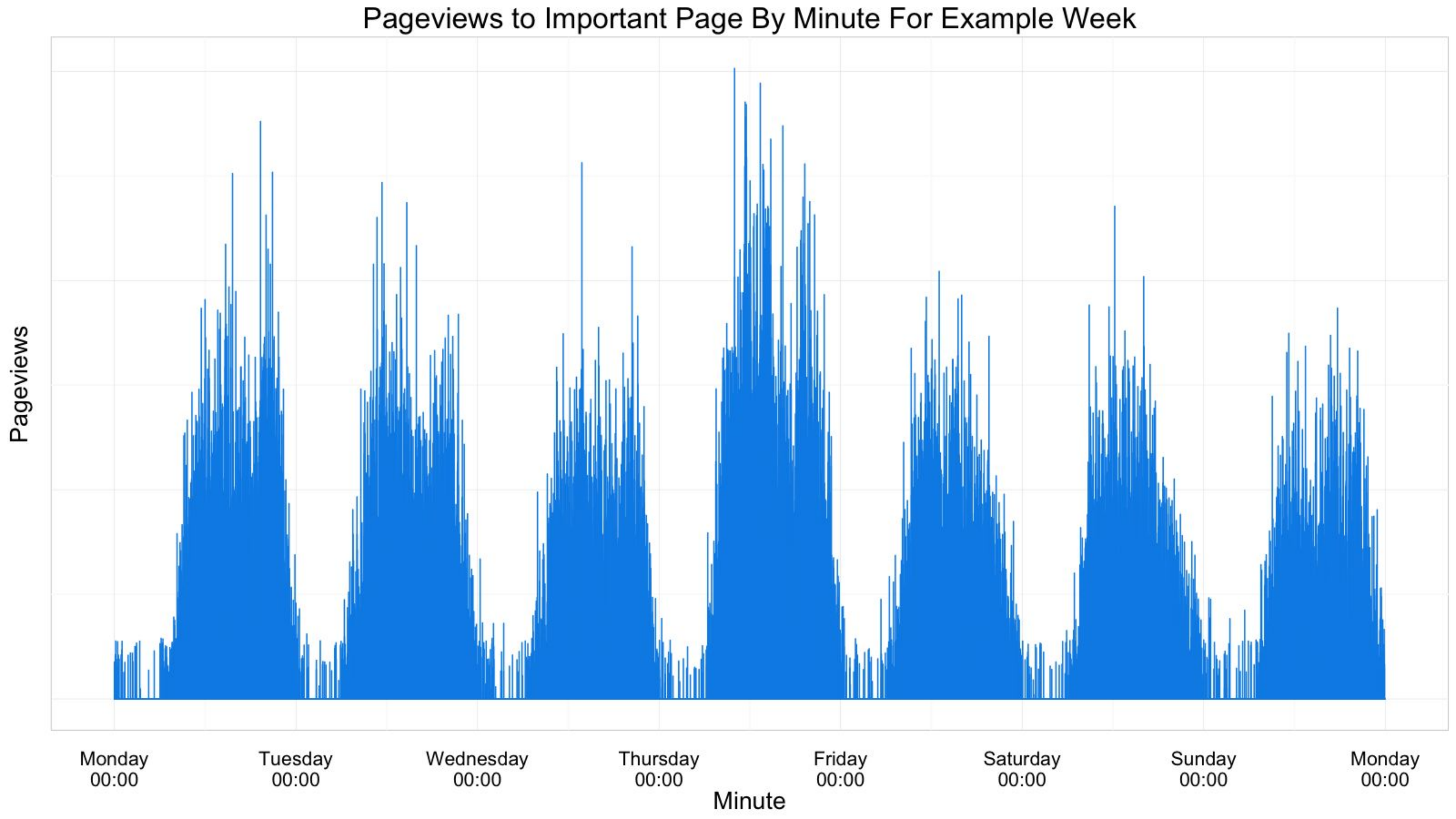
A Few Approaches

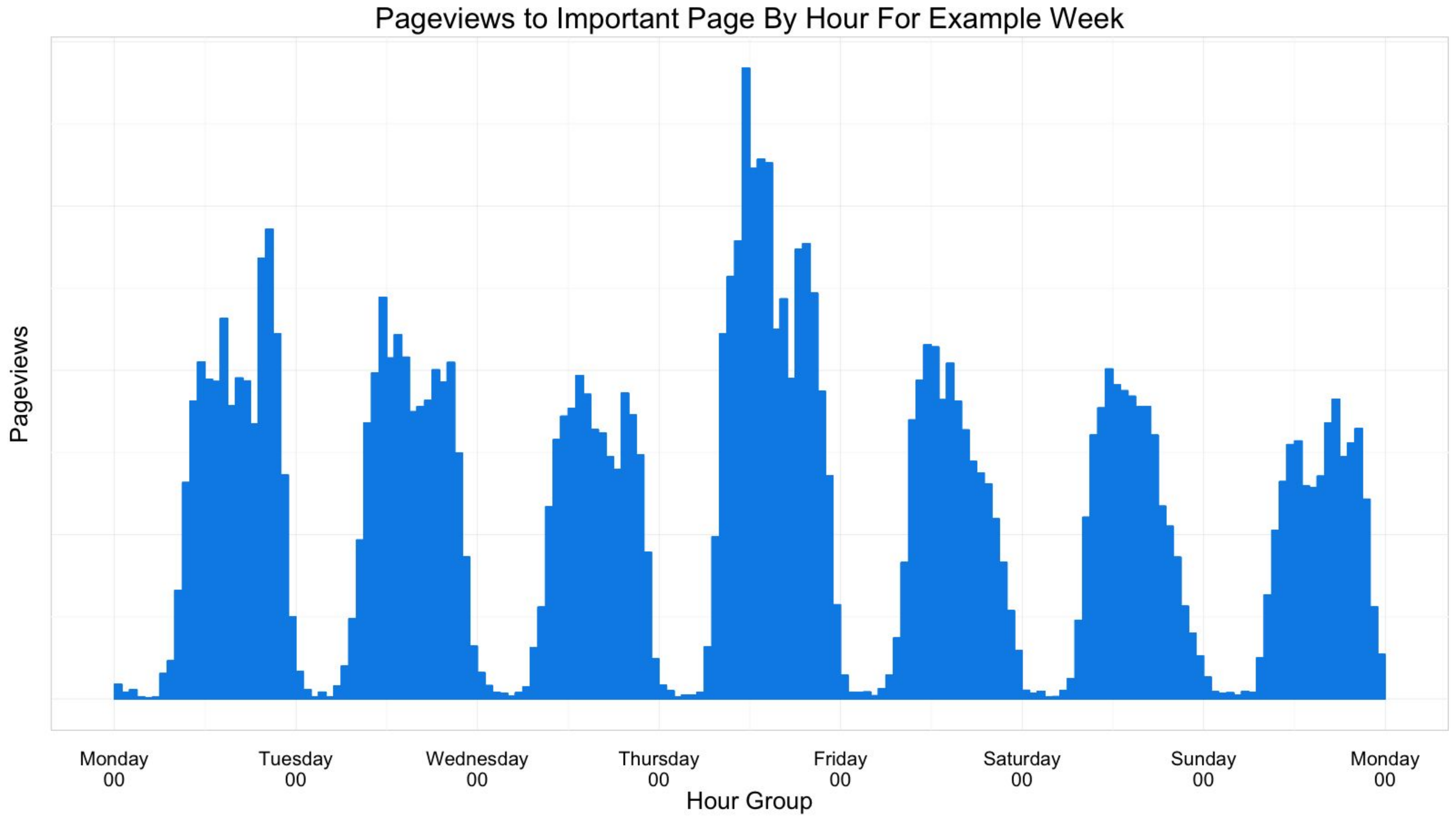
1. Manually setting max/min thresholds based on time
2. Robust Principal Component Analysis
3. Changepoint Detection

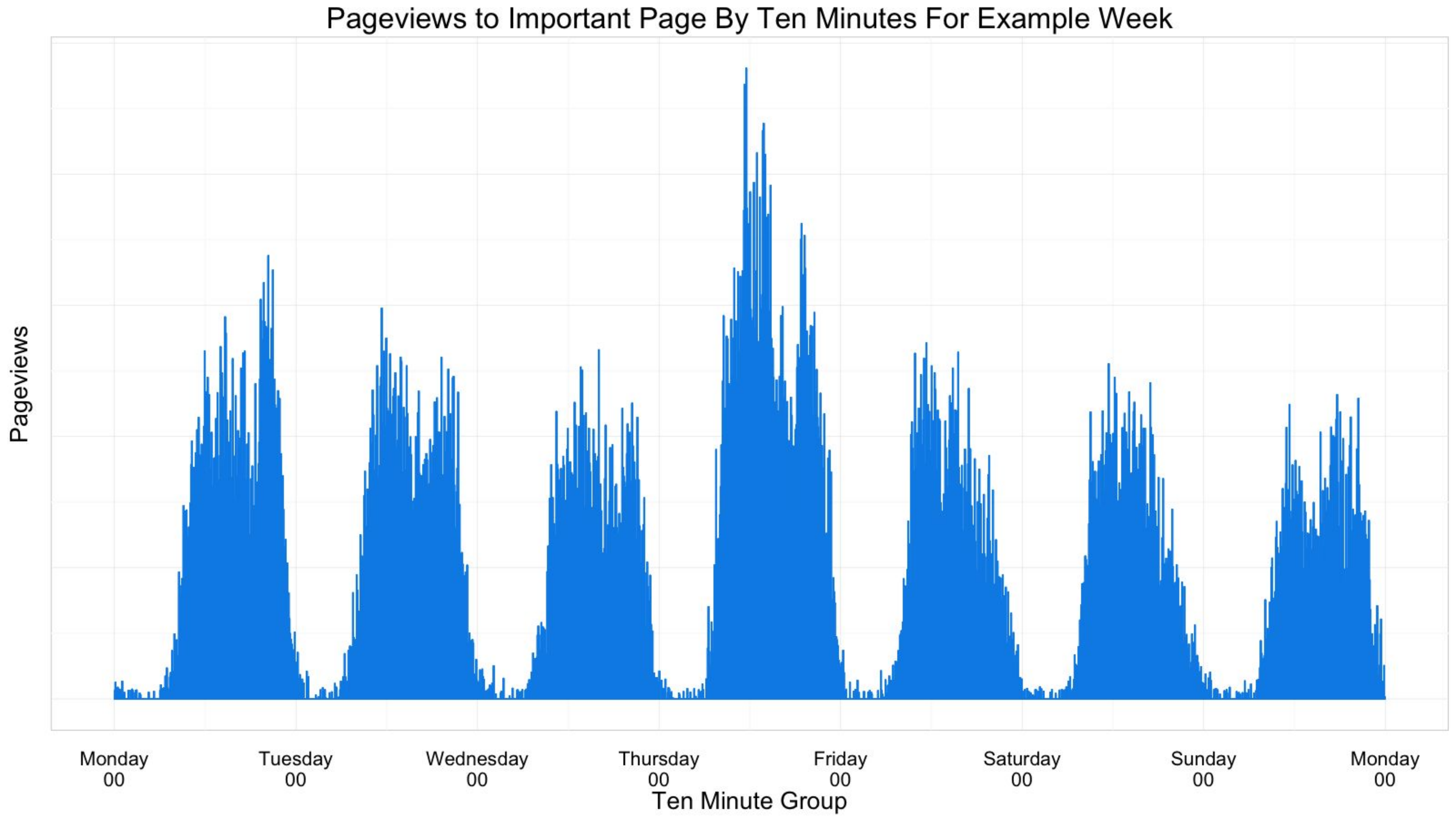
Setting Maximum and Minimum Thresholds Based on Time

Setting Max and Min Thresholds

1. Choose a metric you record in real time
2. (Optional) transform into a new metric over a period of recent time.
3. Set upper and lower limits for this metric - if the metric goes outside these bounds, detect an anomaly.
    - Limits can depend on time. If you have seasonal data, then the limits can depend on the time in each season.
    - Send an alert to interrupt key people


Pageviews to Important Page By Minute For Example Week
Pageviews
Monday
00:00
Tuesday
00:00
Wednesday
00:00
Thursday
00:00
Friday
00:00
Saturday
00:00
Sunday
00:00
Monday
00:00
Minute

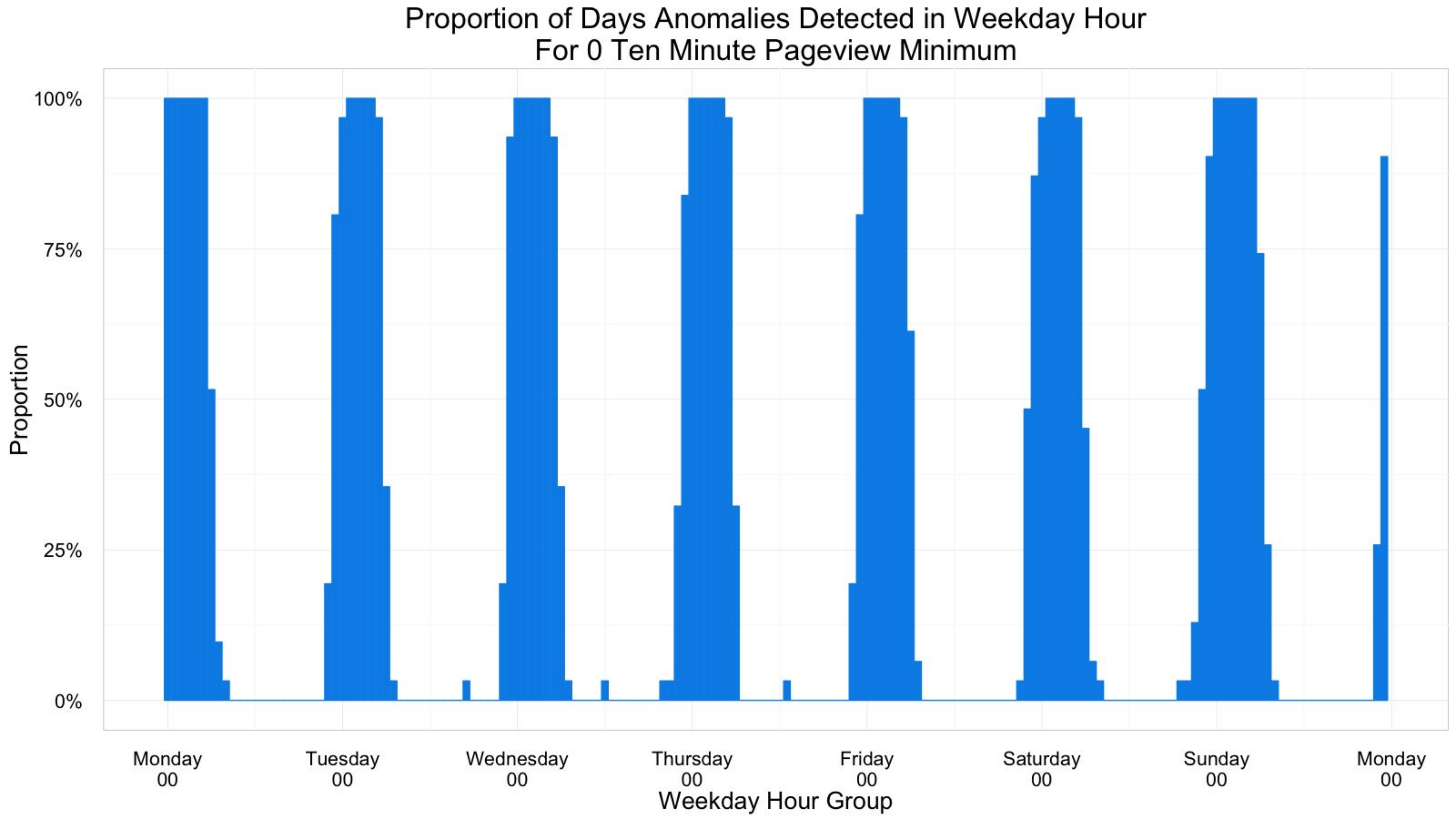

Pageviews to Important Page By Hour For Example Week
Pageviews
Monday
00
Tuesday
00
Wednesday
00
Thursday
00
Friday
00
Saturday
00
Sunday
00
Monday
00
Hour Group



Pageviews to Important Page By Ten Minutes For Example Week
Pageviews
Monday 00
Tuesday 00
Wednesday 00
Thursday 00
Friday 00
Saturday 00
Sunday 00
Monday 00
Ten Minute Group

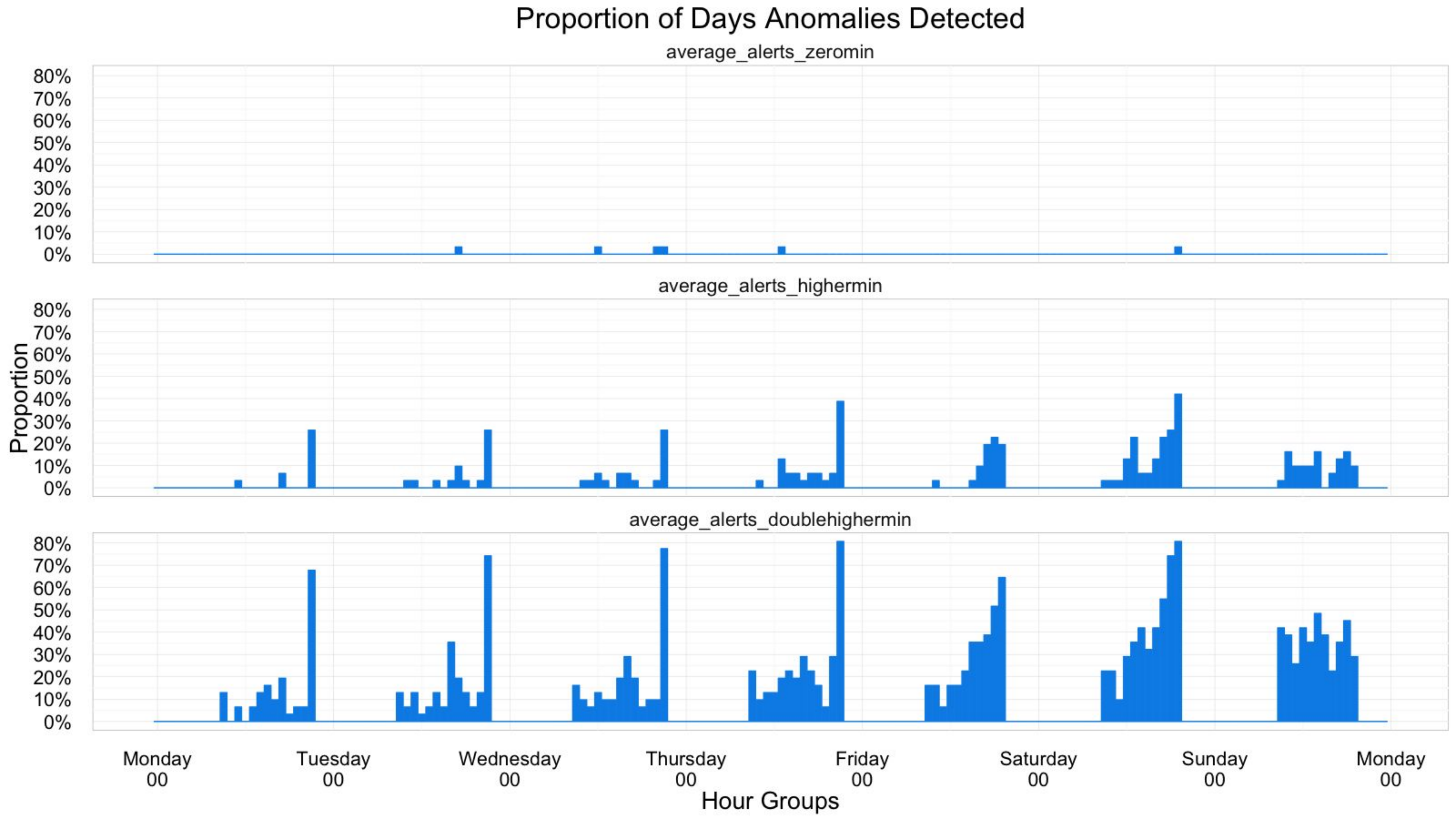
Start Simple

- Extracted 31 weeks of data
- Have ten minute rolling sum buckets
- Create a detector function which classifies a point as anomalous if it is equal to 0
- Group by weekday - hour combination
- For each of these hours, find out what proportion of days in our 31 week dataset would have an alert fired off.

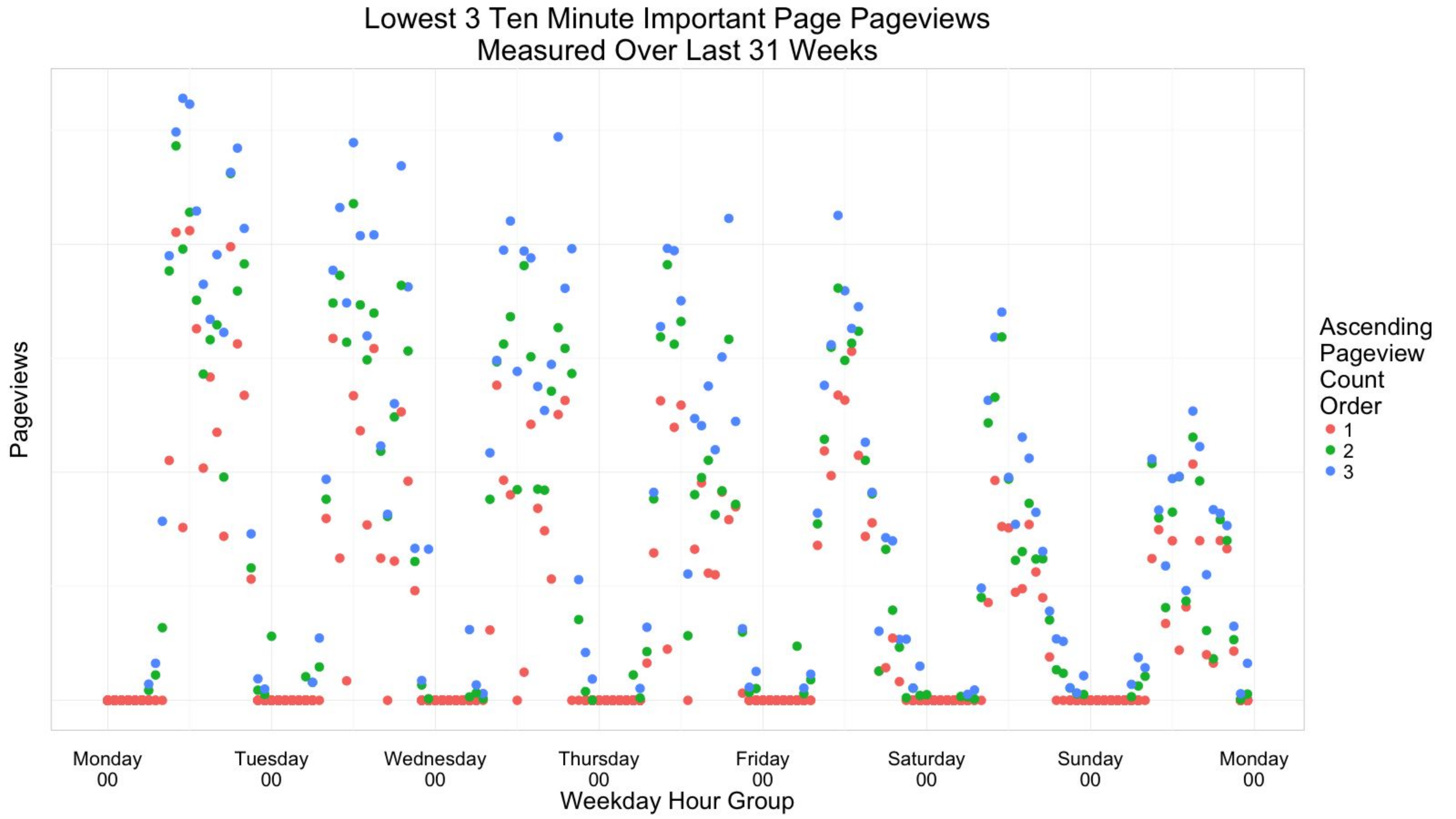

Proportion of Days Anomalies Detected in Weekday Hour
For 0 Ten Minute Pageview Minimum
Proportion
100%
75%
50%
25%
0%
Monday 00
Tuesday 00
Wednesday 00
Thursday 00
Friday 00
Saturday 00
Sunday 00
Monday 00
Weekday Hour Group

Next...

- Have a program check the datastream every few minutes outside of those nighttime hours, send an alert if zero pageviews detected.
- We can do better.
- Ignore nighttime hours, try a couple of arbitrary thresholds

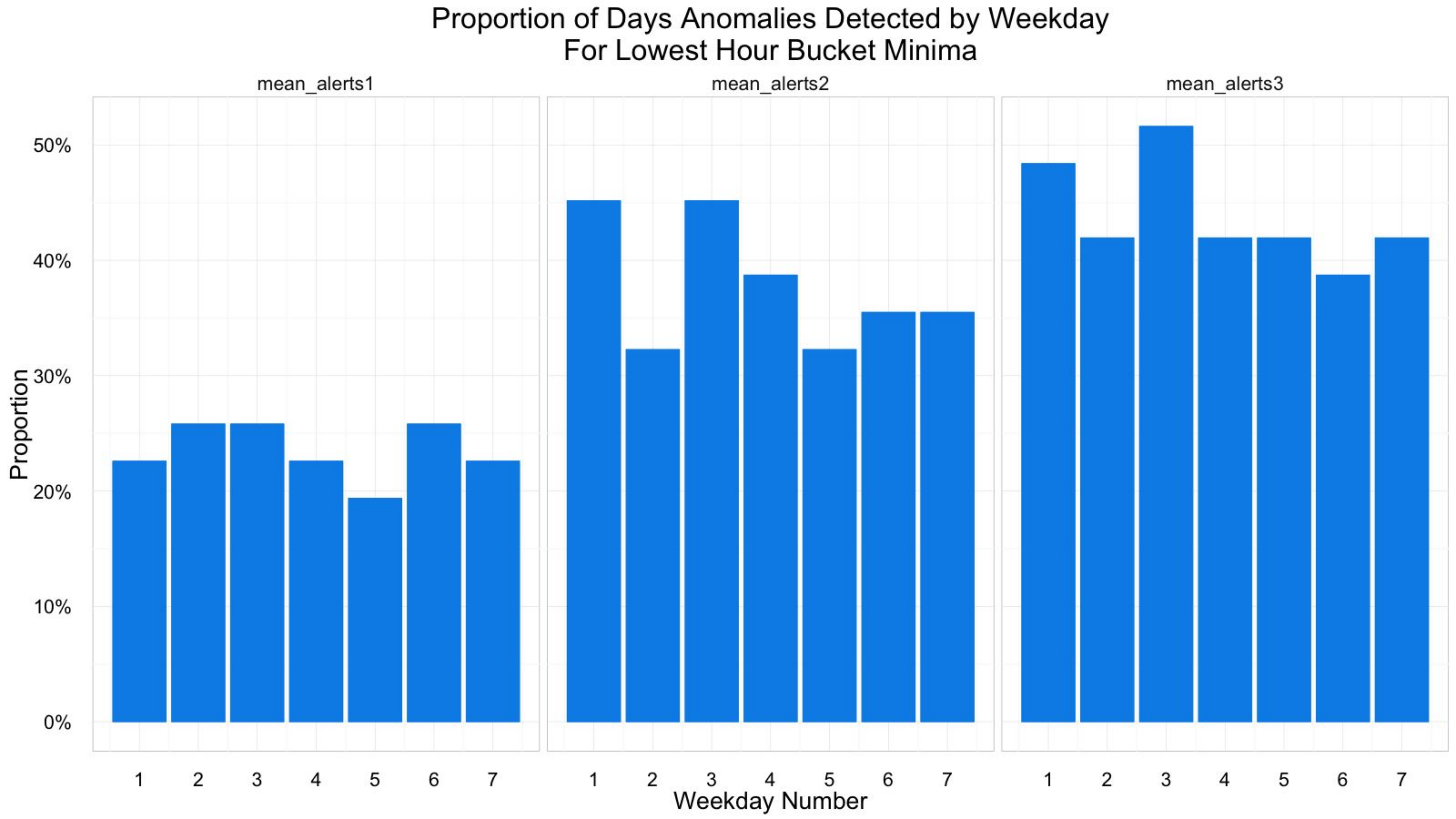


Proportion of Days Anomalies Detected
average_alerts_zeromin
average_alerts_highermin
average_alerts_doublehighermin
Proportion
0%
10%
20%
30%
40%
50%
60%
70%
80%
Monday 00
Tuesday 00
Wednesday 00
Thursday 00
Friday 00
Saturday 00
Sunday 00
Monday 00
Hour Groups


Lowest 3 Ten Minute Important Page Pageviews
Measured Over Last 31 Weeks
Pageviews
Monday 00
Tuesday 00
Wednesday 00
Thursday 00
Friday 00
Saturday 00
Sunday 00
Monday 00
Weekday Hour Group
Ascending Pageview Count Order
1
2
3

Next Step

- Find the minimum pageviews for each weekday hour bucket for each week
- Create 3 detectors:
    - First: detect anomaly if pageviews is less than or equal to the **lowest recorded** value for that weekday hour bucket in the historical dataset
    - Second: detect anomaly if pageviews is less than or equal to the **second lowest** recorded value for that weekday hour bucket in the historical dataset
    - Third: detect anomaly if pageviews is less than or equal to the **third lowest** recorded value for that weekday hour bucket in the historical dataset


Proportion of Days Anomalies Detected by Weekday
For Lowest Hour Bucket Minima
mean_alerts1
mean_alerts2
mean_alerts3
Proportion
50%
40%
30%
20%
10%
0%
1
2
3
4
5
6
7
Weekday Number

Useful Packages

- Dplyr or Data Table
- Lubridate and Zoo
- RcppRoll and Zoo
- Reshape2
- Ggplot2

Robust Principal Component Analysis

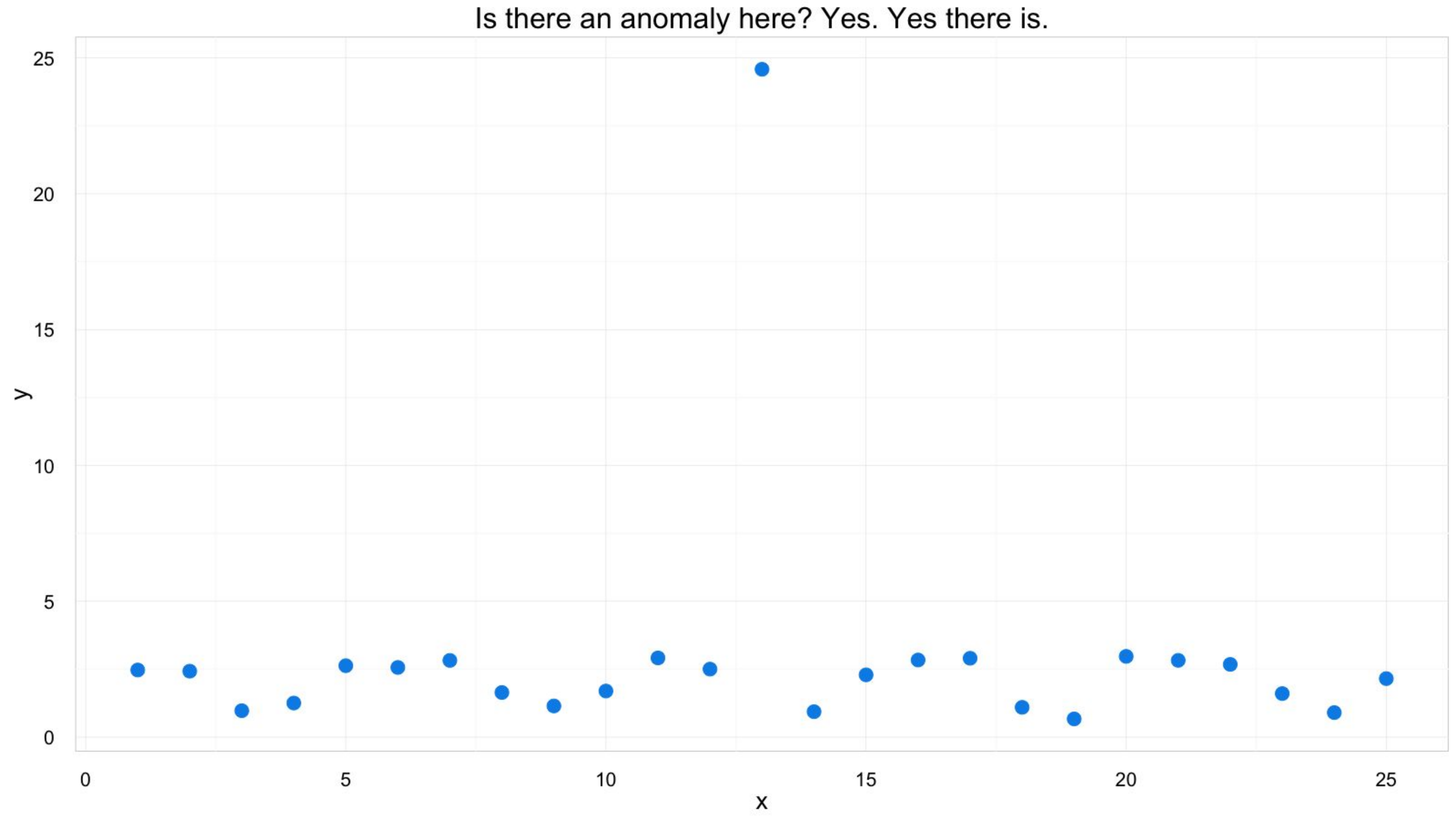
Robust Principal Component Analysis

Data Cleaning:

1. Start with a timeseries
2. Choose an interval of time that corresponds to a season in the series
3. Create a matrix D with each row as one of these intervals, in order
4. Whiten each column - mean to zero and variance to 1

Actual RPCA step:

5. Find a way of representing this matrix as a low rank matrix L added to a sparse matrix S added to an error matrix E. Transform into original space.


Is there an anomaly here? Yes. Yes there is.
y
25
20
15
10
5
0
0
5
10
15
20
25
x

	t1	t2	t3	t4	t5
s1	2.47	2.43	0.97	1.26	2.63
s2	2.56	2.82	1.64	1.15	1.70
s3	2.92	2.50	24.58	0.94	2.29
s4	2.84	2.90	1.10	0.67	2.97
s5	2.83	2.68	1.60	0.90	2.16

=

	t1	t2	t3	t4	t5
s1	2.47	2.43	0.97	1.26	2.63
s2	2.56	2.82	1.64	1.15	1.70
s3	2.92	2.50	1.58	0.94	2.29
s4	2.84	2.90	1.10	0.67	2.97
s5	2.83	2.68	1.60	0.90	2.16

+

	t1	t2	t3	t4	t5
s1	0	0	0	0	0
s2	0	0	0	0	0
s3	0	0	23	0	0
s4	0	0	0	0	0
s5	0	0	0	0	0

Use Netflix Surus Package

```
> library(devtools)
> install_github(repo = "Surus", username = "Netflix", subdir = "resources/R/RAD")
```

```
> ?AnomalyDetection.rpca
```

```
AnomalyDetection.rpca(X, frequency = 7, dates = NULL, autodiff = T,
  forcediff = F, scale = T, L.penalty = 1,
  s.penalty = 1.4/sqrt(max(frequency, ifelse(is.data.frame(X), nrow(X),
  length(X))/frequency)), verbose = F)
```

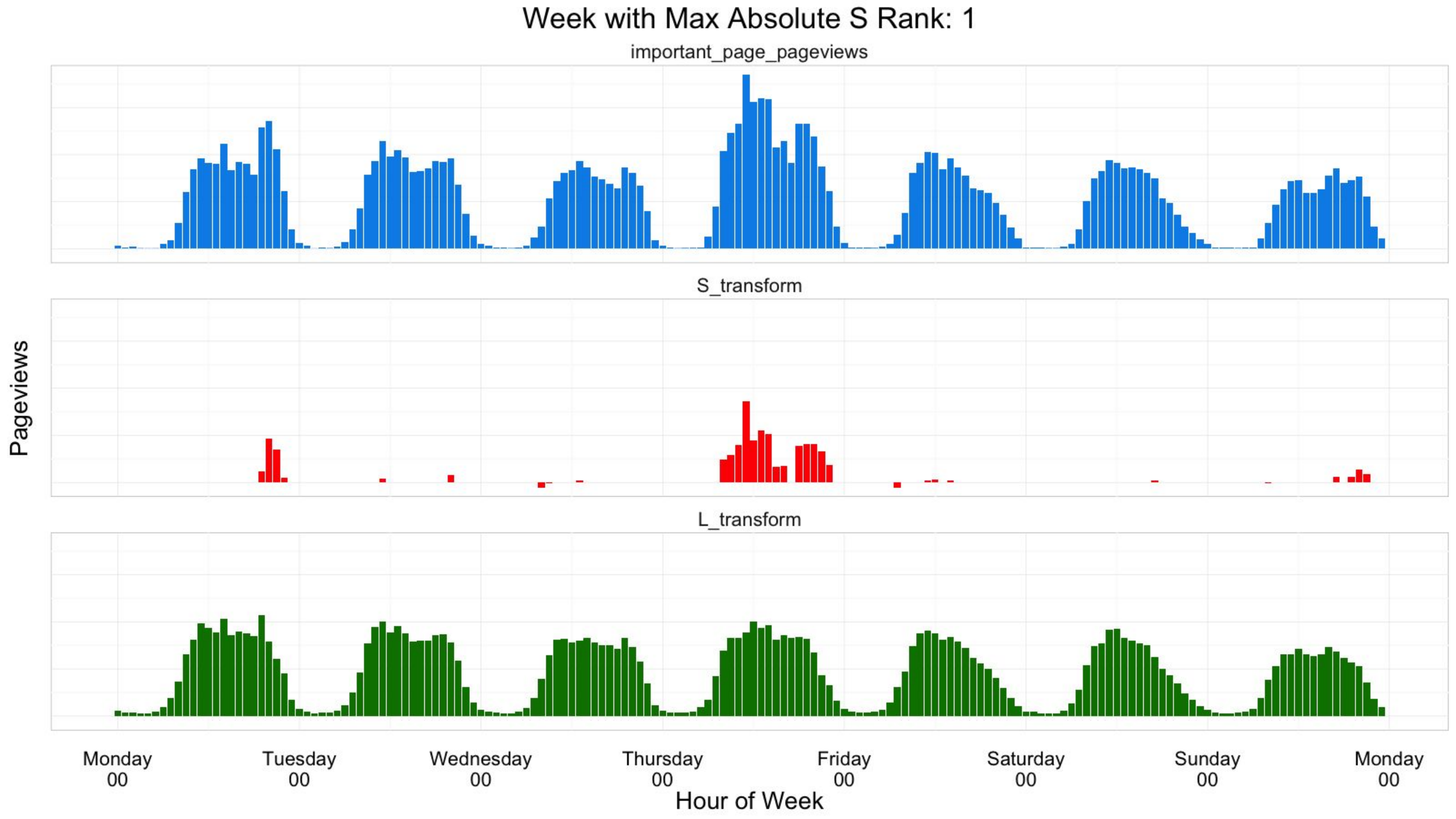
Use Netflix Surus Package

Value

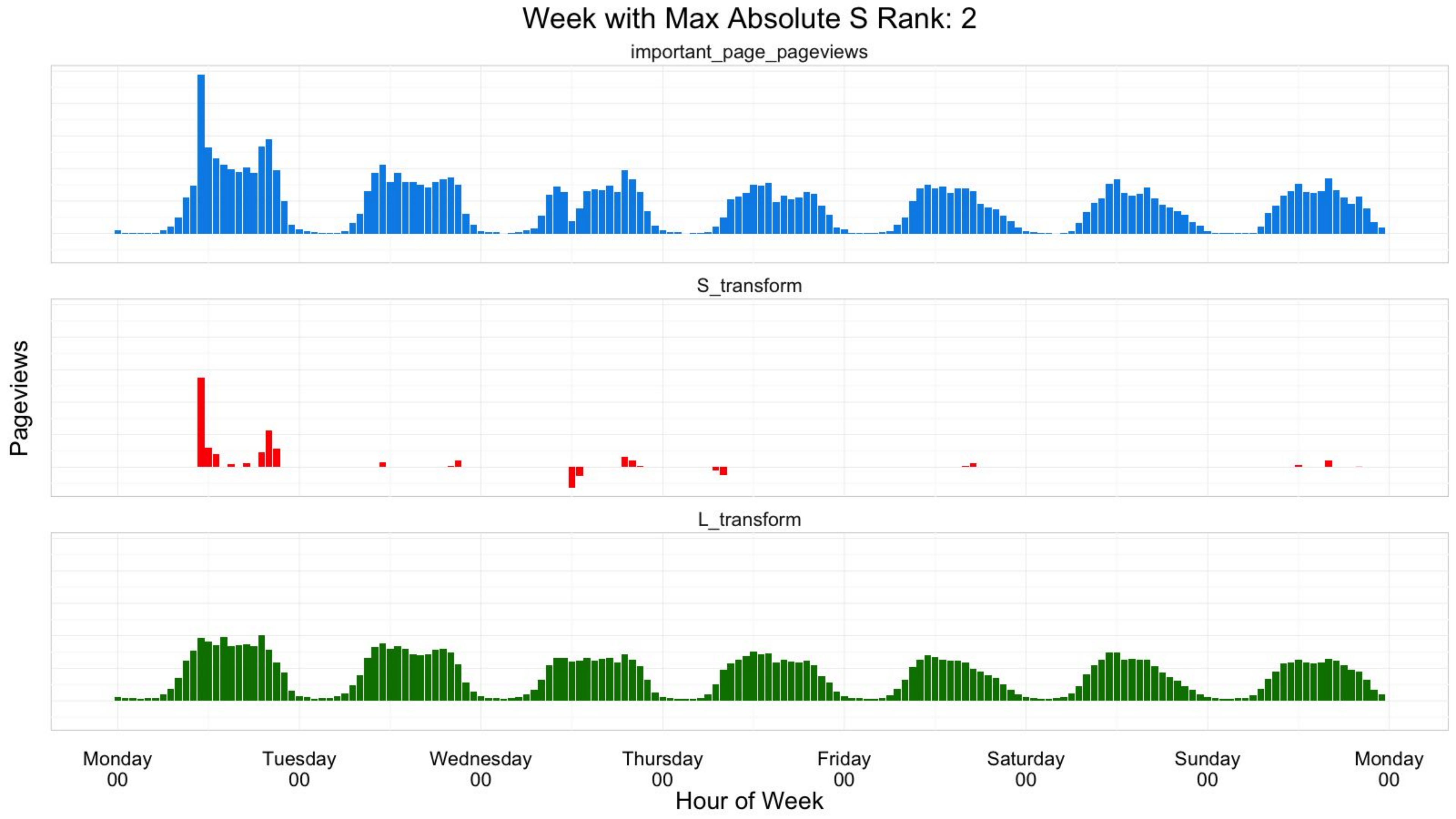
- X_transform. The transformation applied to the time series, can be the identity or could be differencing
- L_transform. The low rank component in the transformed space
- S_transform. The sparse outliers in the transformed space
- E_transform. The noise in the transformed space

Apply to Data

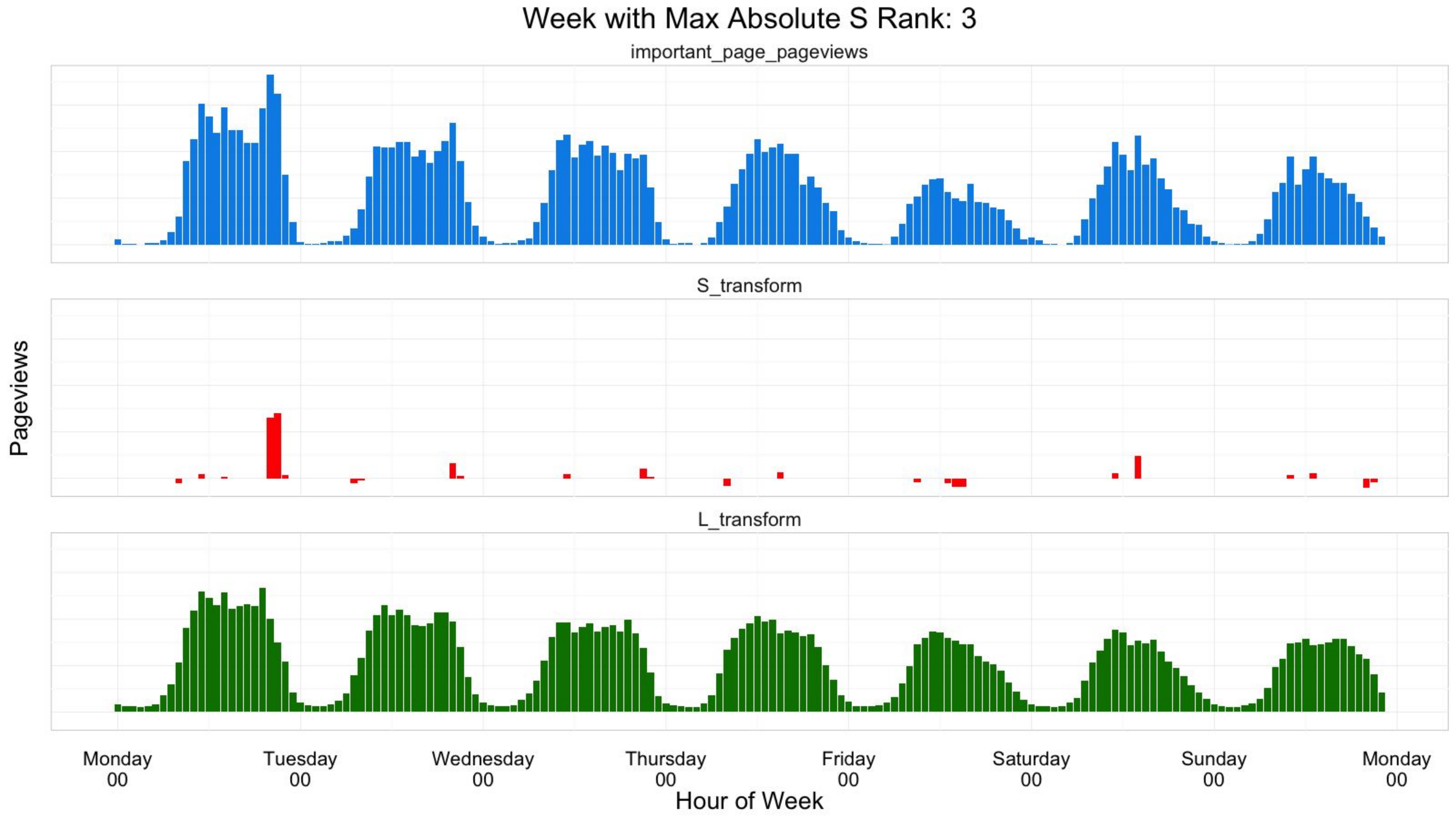
- 31 weeks, 168 hours each week of pageviews
- Use default S penalty
- Prioritise our weeks.
- Rank each week with the maximum absolute S value recorded



Week with Max Absolute S Rank: 1
important_page_pageviews
S_transform
L_transform
Pageviews
Monday 00
Tuesday 00
Wednesday 00
Thursday 00
Friday 00
Saturday 00
Sunday 00
Monday 00
Hour of Week



Week with Max Absolute S Rank: 2
important_page_pageviews
S_transform
L_transform
Pageviews
Monday 00
Tuesday 00
Wednesday 00
Thursday 00
Friday 00
Saturday 00
Sunday 00
Monday 00
Hour of Week



Week with Max Absolute S Rank: 3
important_page_pageviews
S_transform
L_transform
Pageviews
Monday 00
Tuesday 00
Wednesday 00
Thursday 00
Friday 00
Saturday 00
Sunday 00
Monday 00
Hour of Week

```
AnomalyDetection.rpca(X, frequency = 7, dates = NULL, autodiff = T,
  forcediff = F, scale = T, L.penalty = 1,
  s.penalty = 1.4/sqrt(max(frequency, ifelse(is.data.frame(X), nrow(X),
  length(X))/frequency)), verbose = F)
```

Function Being Minimized

0.5*EuclideanNorm(X-L-S)^2

+ LPenalty*NuclearNorm(L)

+ SPenalty*L1Norm(S)

Changepoints

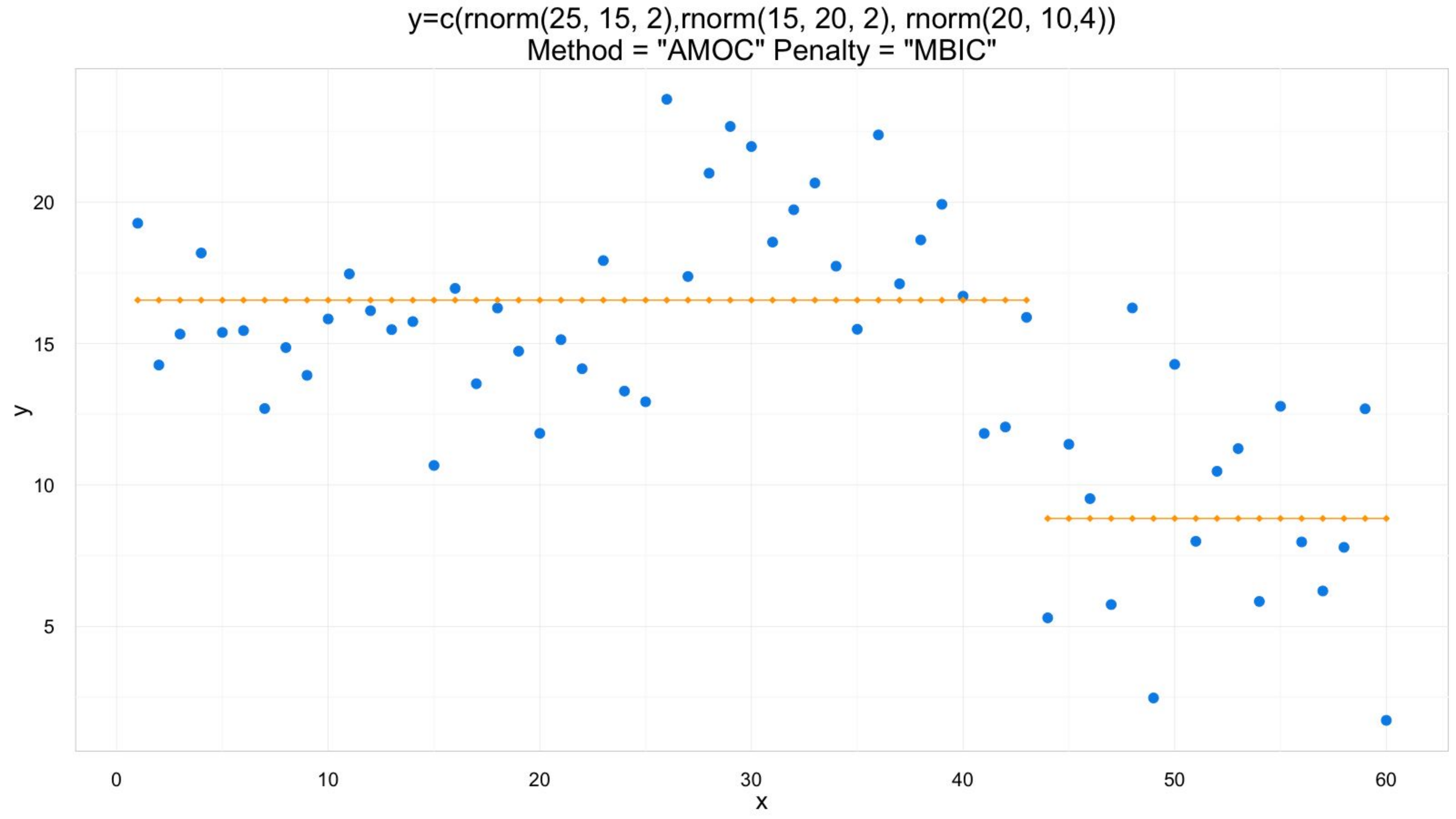
Changepoints for Means

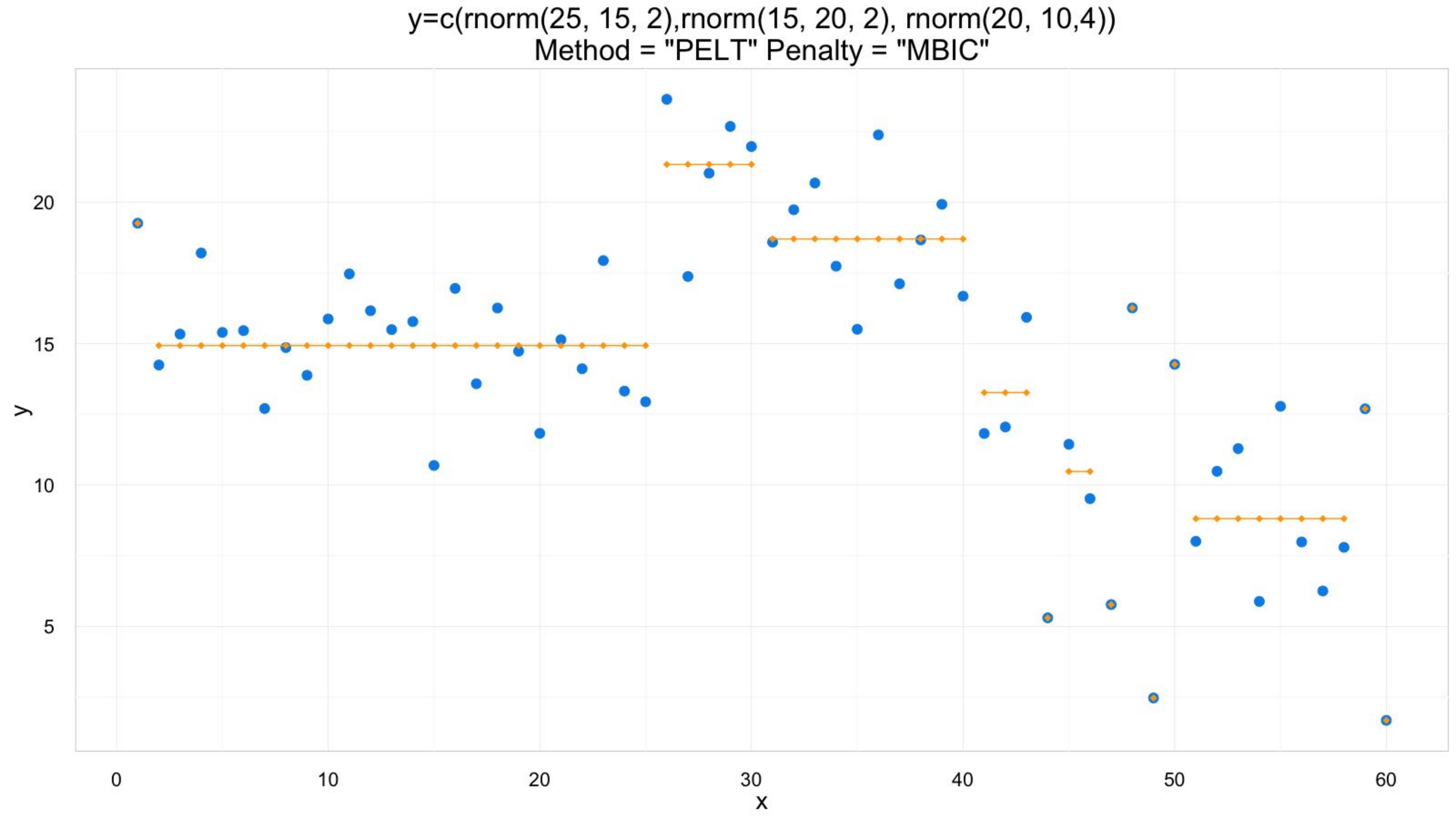
1. Choose a timeseries
2. Assume that a known probability distribution with unknown mean generates the data at each point.
3. Find a likely partition on which the mean is constant on each segment.

Changepoint Package

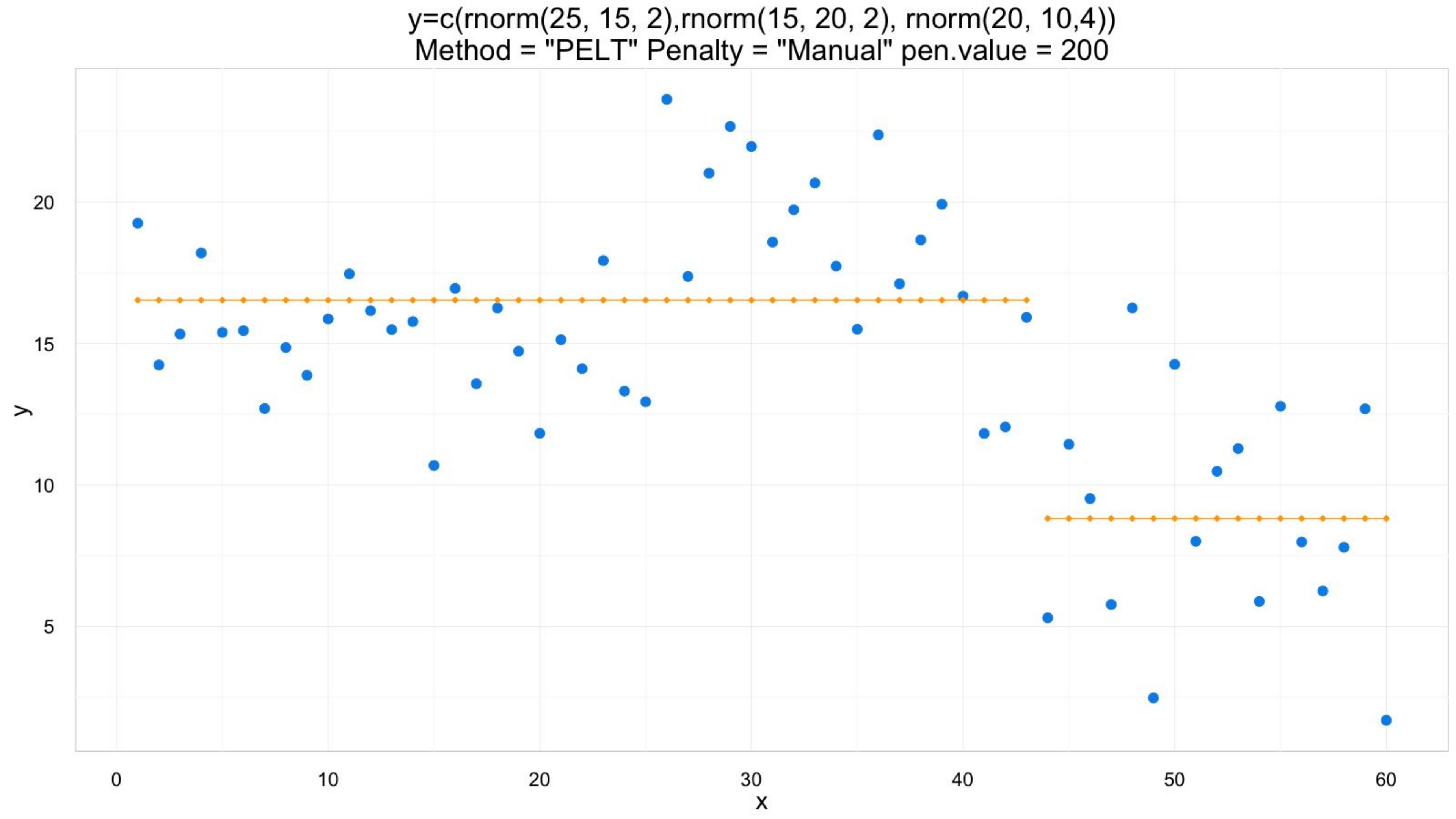
- Available on CRAN
- Can find changepoints for both variances and means
- Use cpt.mean for detecting changes in mean.

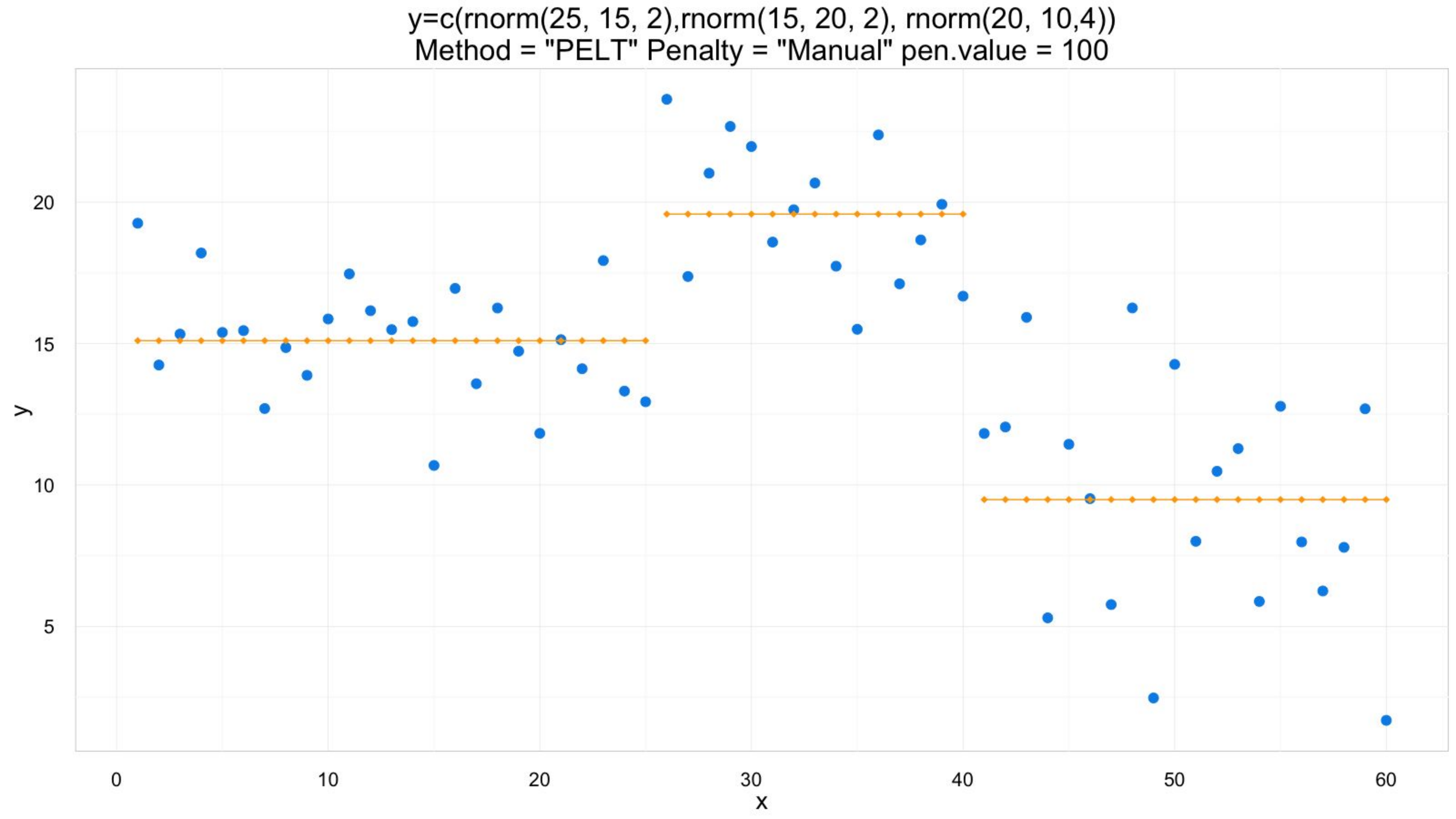
```
cpt.mean(data,penalty="MBIC",pen.value=0,method="AMOC",Q=5,test.stat="Normal",class=TRUE,
param.estimates=TRUE,minseglen=1)
```


y=c(rnorm(25, 15, 2),rnorm(15, 20, 2), rnorm(20, 10,4))
Method = "AMOC" Penalty = "MBIC"
y
x
20
15
10
5
0
10
20
30
40
50
60



y=c(rnorm(25, 15, 2),rnorm(15, 20, 2), rnorm(20, 10,4))
Method = "PELT" Penalty = "MBIC"
y
x
0
10
20
30
40
50
60
5
10
15
20


y=c(rnorm(25, 15, 2),rnorm(15, 20, 2), rnorm(20, 10,4))
Method = "PELT" Penalty = "Manual" pen.value = 200
y
20
15
10
5
0
10
20
30
40
50
60
x


y=c(rnorm(25, 15, 2),rnorm(15, 20, 2), rnorm(20, 10,4))
Method = "PELT" Penalty = "Manual" pen.value = 100
y
x
0
10
20
30
40
50
60
5
10
15
20

Cpt.mean outputs vectors with length = no. of changepoints.

```
cpt.mean.vec <- function(x, method = 'PELT', penalty = 'MBIC', v) {
  m <- cpt.mean(x, method = method, penalty = penalty, pen.value = v)
  m_cpts <- m@cpts
  reps <- m@cpts - lag(m@cpts, default = 0)
  m_next_cpts <- rep(m_cpts, reps)
  m_means <- m@param.est$mean
  m_next_step_size <- rep(c(m_means[2:length(m_means)]- m_means[1:(length(m_means)-1)], 0),reps)
  m_current_means <- rep(m_means, reps)
  output <- data.frame(current_means = m_current_means, next_cpt = m_next_cpts, next_step_size = m_next_step_size)
  return(output)
}
```

Useful Working Pattern

```
> head(df, 15)
   segment timestamp group metric
1        1         1     A   23.7
2        1         2     A   22.0
3        1         3     A   18.2
4        1         4     A   19.3
5        1         5     A   17.8
6        2         6     A   14.9
7        2         7     A   21.9
8        2         8     A   26.3
9        2         9     A   26.1
10       2        10     A   27.8
11       1         1     B   21.3
12       1         2     B   28.6
13       1         3     B   37.0
14       1         4     B   28.0
15       1         5     B   26.3
```

Useful Working Pattern

```
> dfdm <- cbind(df, detection_metrics); head(dfdm, 15);
   segment timestamp group metric detection_metric1 detection_metric2
1        1         1     A   23.7              21.3             0.073
2        1         2     A   22.0              21.3             0.069
3        1         3     A   18.2              21.3             0.106
4        1         4     A   19.3              21.3             0.002
5        1         5     A   17.8              21.3             0.191
6        2         6     A   14.9              21.3             0.106
7        2         7     A   21.9              21.3             0.165
8        2         8     A   26.3              21.3             0.004
9        2         9     A   26.1              21.3             0.135
10       2        10     A   27.8              21.3             0.105
11       1         1     B   21.3              24.7             0.236
12       1         2     B   28.6              24.7             0.339
13       1         3     B   37.0              24.7             0.213
14       1         4     B   28.0              24.7             0.357
15       1         5     B   26.3              24.7             0.321
```

Useful Working Pattern

```
> dfdm <- group_by(dfdm, segment, group) %>%
+   mutate(max_seggroup_detect2_value = max(detection_metric2)) %>%
+   ungroup %>%
+   mutate(group_segment_ranking = dense_rank(-max_seggroup_detect2_value))
```

Useful Working Pattern

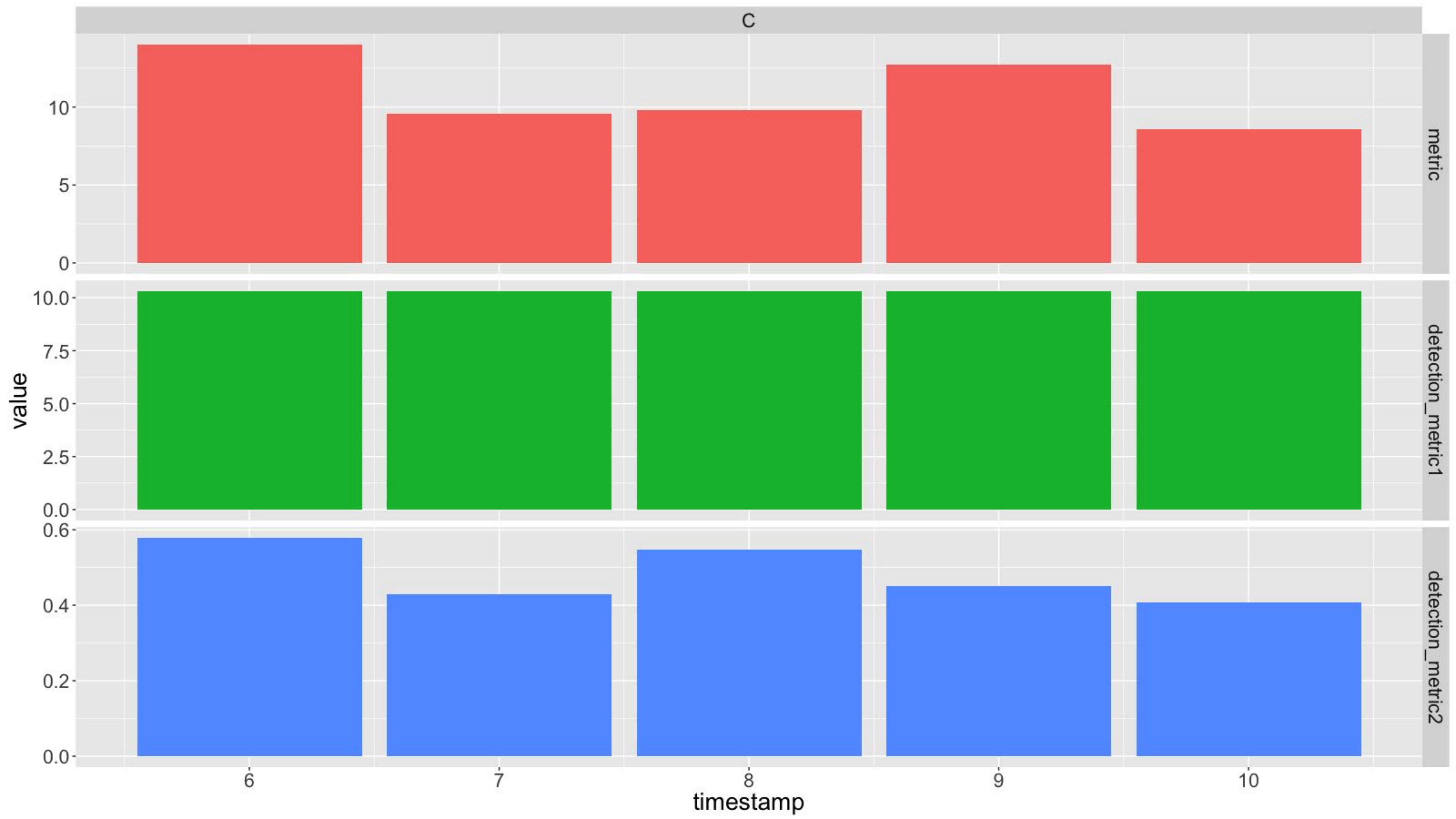
```
> head(dfdm, 15)
# A tibble: 15 x 9
   segment timestamp  group metric detection_metric1 detection_metric2 max_seggroup_detect2_value group_segment_ranking max_detect2_value
     <int>     <int> <fctr>  <dbl>             <dbl>             <dbl>                      <dbl>                 <int>             <dbl>
1        1         1      A   23.7              21.3             0.073                      0.191                     5             0.191
2        1         2      A   22.0              21.3             0.069                      0.191                     5             0.191
3        1         3      A   18.2              21.3             0.106                      0.191                     5             0.191
4        1         4      A   19.3              21.3             0.002                      0.191                     5             0.191
5        1         5      A   17.8              21.3             0.191                      0.191                     5             0.191
6        2         6      A   14.9              21.3             0.106                      0.165                     6             0.165
7        2         7      A   21.9              21.3             0.165                      0.165                     6             0.165
8        2         8      A   26.3              21.3             0.004                      0.165                     6             0.165
9        2         9      A   26.1              21.3             0.135                      0.165                     6             0.165
10       2        10      A   27.8              21.3             0.105                      0.165                     6             0.165
11       1         1      B   21.3              24.7             0.236                      0.357                     4             0.357
12       1         2      B   28.6              24.7             0.339                      0.357                     4             0.357
13       1         3      B   37.0              24.7             0.213                      0.357                     4             0.357
14       1         4      B   28.0              24.7             0.357                      0.357                     4             0.357
15       1         5      B   26.3              24.7             0.321                      0.357                     4             0.357
```

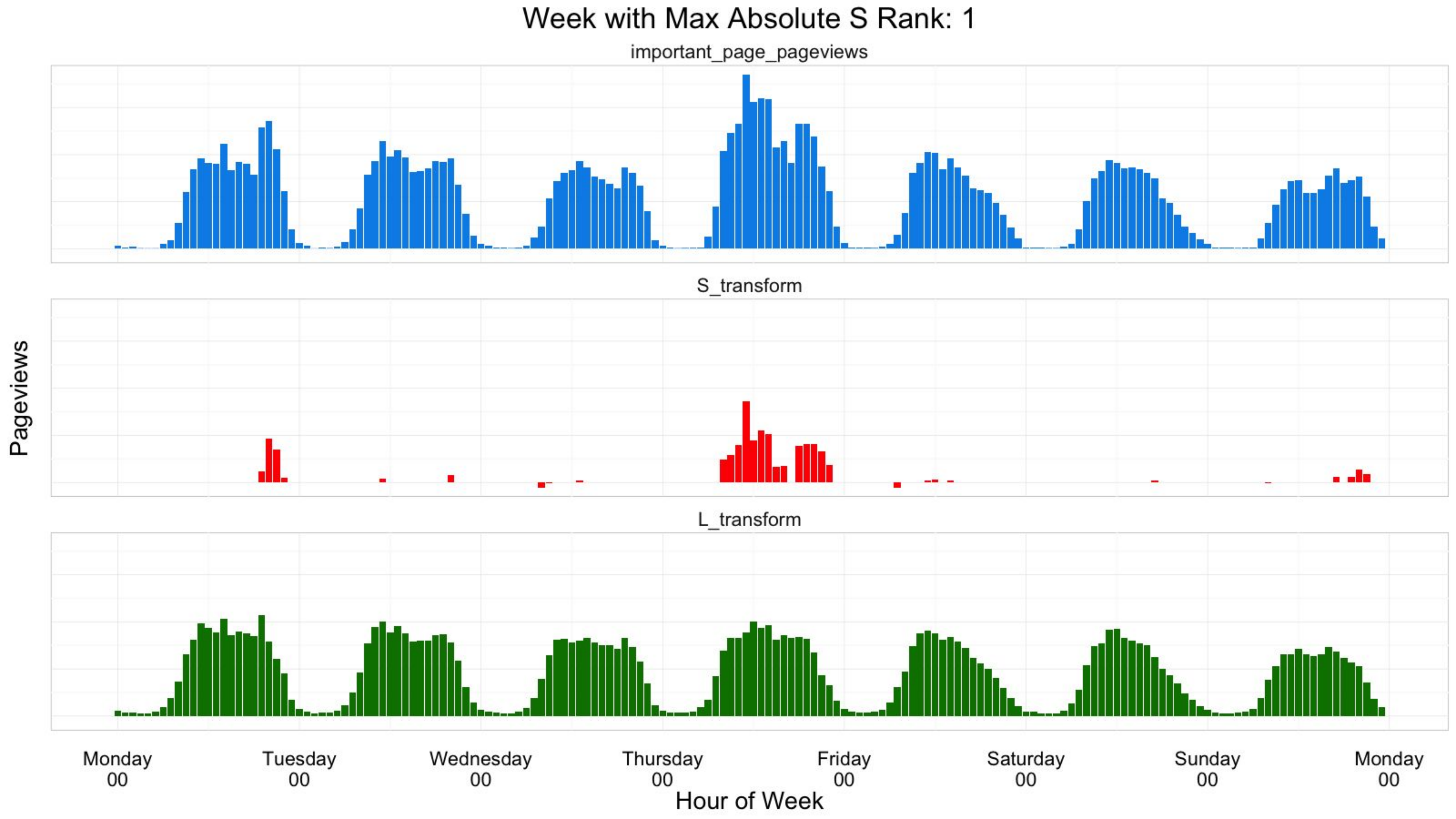
Useful Working Pattern

```
> dfdm.melt <- melt(dfdm, c('segment', 'timestamp', 'group', 'group_segment_ranking'),
+                   measure.vars = c('metric', 'detection_metric1', 'detection_metric2'))
> head(dfdm.melt, 15)
   segment timestamp group group_segment_ranking variable value
1        1         1     A                     5   metric  23.7
2        1         2     A                     5   metric  22.0
3        1         3     A                     5   metric  18.2
4        1         4     A                     5   metric  19.3
5        1         5     A                     5   metric  17.8
6        2         6     A                     6   metric  14.9
7        2         7     A                     6   metric  21.9
8        2         8     A                     6   metric  26.3
9        2         9     A                     6   metric  26.1
10       2        10     A                     6   metric  27.8
11       1         1     B                     4   metric  21.3
12       1         2     B                     4   metric  28.6
13       1         3     B                     4   metric  37.0
14       1         4     B                     4   metric  28.0
15       1         5     B                     4   metric  26.3
```

Useful Working Pattern

```
> filter(dfdm.melt, group_segment_ranking == 1) %>%
+   ggplot(aes(timestamp, value, fill = variable, group = variable)) +
+   geom_bar(stat = 'identity') +
+   facet_grid(variable~group, scale = 'free') +
+   guides(fill = FALSE) +
+   theme(text = element_text(size = 20))
```


C
metric
detection_metric1
detection_metric2
value
timestamp
0
5
10
0.0
2.5
5.0
7.5
10.0
0.0
0.2
0.4
0.6
6
7
8
9
10


Week with Max Absolute S Rank: 1
important_page_pageviews
S_transform
L_transform
Pageviews
Monday
00
Tuesday
00
Wednesday
00
Thursday
00
Friday
00
Saturday
00
Sunday
00
Monday
00
Hour of Week

Summary - Real Time

- Do it. Start simple and iterate.
- Investigate boring but reliable methods first.
- Use historical data to figure out how often alerts would be fired off

Summary - Outside of Real-Time

- Prioritise, don’t classify
- RPCA (RAD package) for finding spikes and dips
- Changepoints (changepoint package) for finding longer-term changes
- Adjust penalty values to control sensitivity.
- Rank time series with understandable metrics first based on:
    - Changepoint mean differences
    - S values
    - Different penalty values

Thank You!

shannon.wirtz@uswitch.com

We’re recruiting: https://www.uswitch.com/vacancies/