Ec2 Forecast 2018

```
library(tidyverse)
library(devtools)
library(here)
library(stringr)
library(prophet)
library(dplyr)
library(anomalize)
library(lubridate)
```

```
#import flat file
file_raw = here::here('data/example-aws-cost-hr-data.csv')
data.raw = read_csv(file_raw)
#disable scientific notation
options(scipen=999)
```

```
#preprocess data
data.clean = dplyr::tbl_df(data.raw) %>%
  #change to numeric
  mutate_at(.,vars(-app),funs(as.numeric)) %>%
  #change na to 0 because timeseries
  replace(is.na(.), 0) %>%
  #don't need the total column %>%
  filter(.,app!='app Total') %>%
  #rename app to date
  rename(.,date=app) %>%
  #change date to date data type
  mutate(date = ymd(date))
```

```
#make usage data tidy (variables in columns)
data.usage = data.clean %>%
  select(.,-(`No Tagkey: app0($)`:`total cost`)) %>%
  #rename remove other columns
  rename_all(
    funs(
      #all lower
      stringr::str_to_lower(.) %>%
        #parse app name
        stringr::str_replace_all(., "\\(.*\\)", '')
    )
  ) %>%
  gather(.,instance,usage_hr,-date)
```

```
#make cost data tidy (variables in columns)
data.cost = data.clean %>%
  select(.,-(`No Tagkey: app0(hr)`:`total usage`)) %>%
  #rename remove other columns
  rename_all(
    funs(
      #all lower
      stringr::str_to_lower(.) %>%
        #parse app name
```

```
        stringr::str_replace_all(., "\\(.*\\)", '')
    )
  ) %>%
  gather(.,instance,cost_usd,-date)
```

```
#join usage and cost together
data.join = inner_join(data.cost,data.usage, by=c('date','instance')) %>%
  #calc min instance count assuming 24 hour operation
  mutate(min_instance_count= ceiling(usage_hr/24))

head(data.join)
```

```
## # A tibble: 6 x 5
##   date       instance         cost_usd usage_hr min_instance_count
##   <date>     <chr>               <dbl>    <dbl>              <dbl>
## 1 2017-08-01 no tagkey: app0     249.     1068.                 45
## 2 2017-08-02 no tagkey: app0     257.     1100.                 46
## 3 2017-08-03 no tagkey: app0     261.     1135.                 48
## 4 2017-08-04 no tagkey: app0     263.     1129.                 48
## 5 2017-08-05 no tagkey: app0     262.     1128.                 47
## 6 2017-08-06 no tagkey: app0     262.     1126.                 47
```

```
#tag first date cost is incurred for each instance
min.cost.date = data.join %>%
  #remove zero cost days
  filter(.,cost_usd > 0) %>%
  group_by(instance) %>%
  #get first cost date
  summarize(min_date = min(date))
```

```
#remove dates before first cost date
data.join.filter = left_join(data.join,min.cost.date, by = "instance") %>%
  filter(.,date>=min_date) %>%
  select(.,-min_date) %>%
  #add count of days history
  add_count(instance) %>%
  rename(.,days_history = n) %>%
  mutate(.,hist_group = ifelse(days_history<60,'short','long'))
```

```
#create groups to forecast in to minimize short history
forecast.group = data.join.filter %>%
  group_by(instance,hist_group) %>%
  #get cost information for that 30 days
  top_n(.,30,date) %>%
  #avg cost for last 30 days, total cost for last 30 days
  summarize(.,avg_cost_per_day = mean(cost_usd),cost = sum(cost_usd)) %>%
  mutate(.,cost_group = ifelse(avg_cost_per_day <= 1,'low','high')) %>%
  mutate(.,forecast_group = case_when(
    #these instances aren't generating costs
    cost==0 ~ 'inactive',
    #high cost and long history. Good candidate for individual forecast
    cost_group=='high' && hist_group=='long' ~ instance,
    #high cost and short history. Good candidate for grouping
    cost_group=='high' && hist_group=='short' ~ "high$_shortTS",
    #bucket everything else
```

```
    TRUE ~ "low$orshort"
  ))

head(forecast.group)
```

```
## # A tibble: 6 x 6
## # Groups:   instance [6]
##   instance hist_group avg_cost_per_day   cost cost_group forecast_group
##   <chr>    <chr>                 <dbl>  <dbl> <chr>      <chr>
## 1 app1     long                 620.   18612. high       app1
## 2 app10    long                   4.87   146. high       app10
## 3 app11    long                  31.6    949. high       app11
## 4 app12    long                  20.6    618. high       app12
## 5 app13    long                  36.5   1094. high       app13
## 6 app14    long                   0        0  low        inactive
```

```
#add forecasting groups to main data
data.join.filter = left_join(
                        data.join.filter
                       ,forecast.group %>% select(.,instance,forecast_group,cost_group)
                       ,by="instance"
                       ) %>%
  #remove deactived instances
  filter(.,forecast_group!='inactive')
```

```
#Create aggregate All bucket that captures all active instance activity
data.aggregate = data.join.filter %>%
  group_by(date) %>%
  summarise(
    instance = 'All',
    cost_usd = sum(cost_usd),
    usage_hr = sum(usage_hr),
    min_instance_count = sum(min_instance_count),
    days_history = max(days_history),
    hist_group = 'long',
    forecast_group = 'All',
    cost_group = 'high'
  )

head(data.aggregate)
```

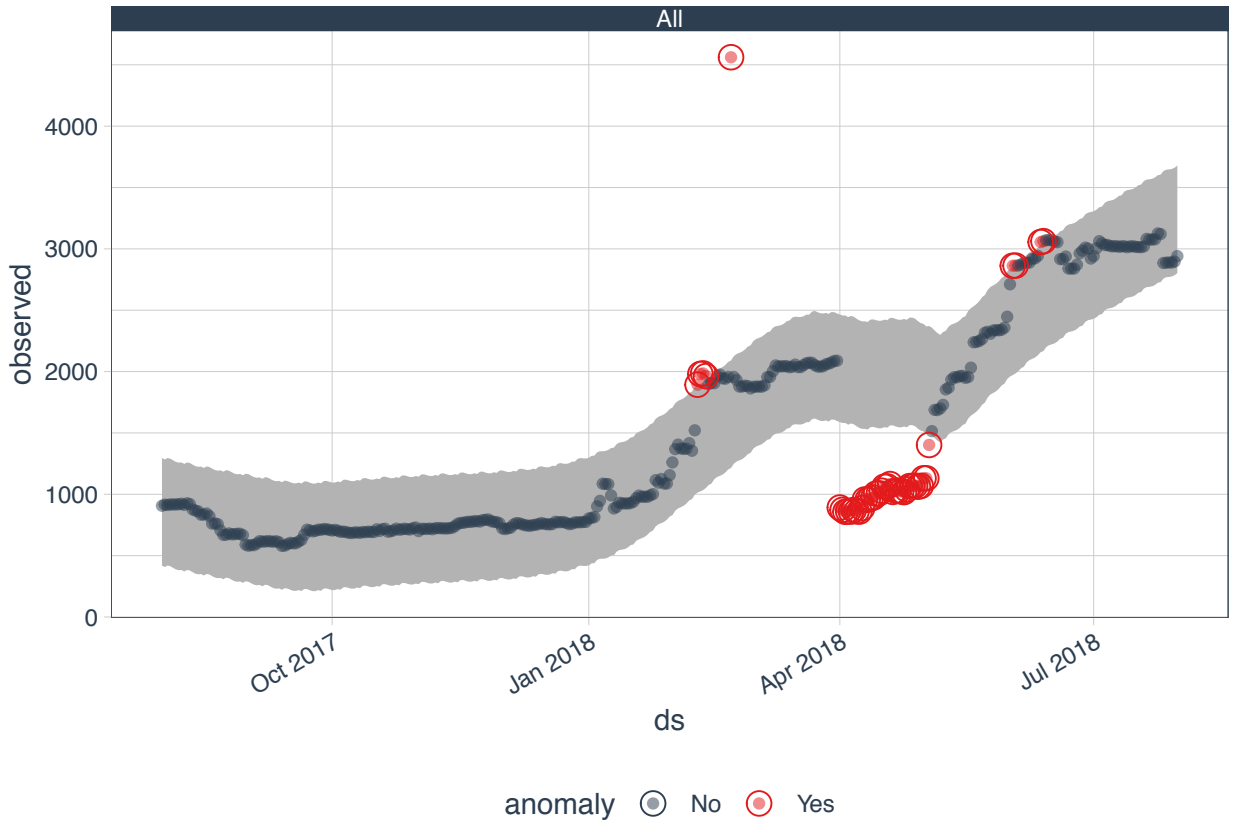
```
## # A tibble: 6 x 9
##   date       instance cost_usd usage_hr min_instance_count days_history
##   <date>     <chr>       <dbl>    <dbl>              <dbl>        <dbl>
## 1 2017-08-01 All           906.    2147.                105          365
## 2 2017-08-02 All           915.    2231.                108          365
## 3 2017-08-03 All           913.    2234.                109          365
## 4 2017-08-04 All           917.    2233.                109          365
## 5 2017-08-05 All           917.    2254.                110          365
## 6 2017-08-06 All           917.    2246.                109          365
## # ... with 3 more variables: hist_group <chr>, forecast_group <chr>,
## #   cost_group <chr>
```

```
#combine All and individual forecast groups
data.combine = dplyr::union(data.join.filter,data.aggregate)
```

```r
data.combine.group = data.combine %>%
  group_by(forecast_group,date) %>%
  summarise(
    cost_usd = sum(cost_usd)
  ) %>%
  #rename columns for prophet use
  rename(.,ds=date,y = cost_usd) %>%
  arrange(forecast_group,ds)
```

```r
#automatically remove outliers using anomalize
data.no.outlier = data.combine.group %>%
  filter(.,forecast_group!='inactive') %>%
  group_by(forecast_group) %>%
  #decompose time series
  time_decompose(y,merge = TRUE) %>%
  #tag anomalies/outliers
  anomalize(remainder, max_anoms = 0.20) %>%
  #set anomalies/outliers to NA
  mutate(.,y=ifelse(anomaly=='Yes',NA,y)) %>%
  select(.,ds,forecast_group,y)
```

```r
#Outlier Detection And Removal Example
data.combine.group %>%
    filter(.,forecast_group=="All") %>%
    group_by(forecast_group) %>%
    time_decompose(y) %>%
    anomalize(remainder) %>%
    time_recompose() %>%
    plot_anomalies(time_recomposed = TRUE, ncol = 3, alpha_dots = 0.5)
```




```
#create summary df to store forecast model objects
models.summary = data.no.outlier %>%
  group_by(forecast_group) %>%
  #nest timeseries into list data
  nest() %>%
  #create model m using linear growth
  mutate(m=map(data,~prophet(.x,growth="linear"))) %>%
  #create place holder for forecast predictions
  mutate(future=map(m,~make_future_dataframe(.x, periods = 153, include_history = TRUE))) %>%
  #generate forecasts
  mutate(forecast=map2(m,future,predict)) %>%
  #add graphs
  mutate(plot=pmap(list(m, forecast, forecast_group), ~plot(..1,..2,plot_cap=FALSE,ylabel='Cost_USD',xla
  #add forecast_method
  mutate(.,forecast_method = if_else(forecast_group=='All','Aggregate','Detailed'))
```

```
#Models View Sample
#show forecast groups
models.summary %>%
  select(.,forecast_method,forecast_group,everything()) %>%
  head()
```

```
## # A tibble: 6 x 7
##   forecast_method forecast_group data    m       future   forecast   plot
##   <chr>           <chr>          <list>  <list>  <list>   <list>     <lis>
## 1 Aggregate       All            <tibbl~ <S3: p~ <data.f~ <data.fra~ <S3:~
## 2 Detailed        app1           <tibbl~ <S3: p~ <data.f~ <data.fra~ <S3:~
## 3 Detailed        app10          <tibbl~ <S3: p~ <data.f~ <data.fra~ <S3:~
```

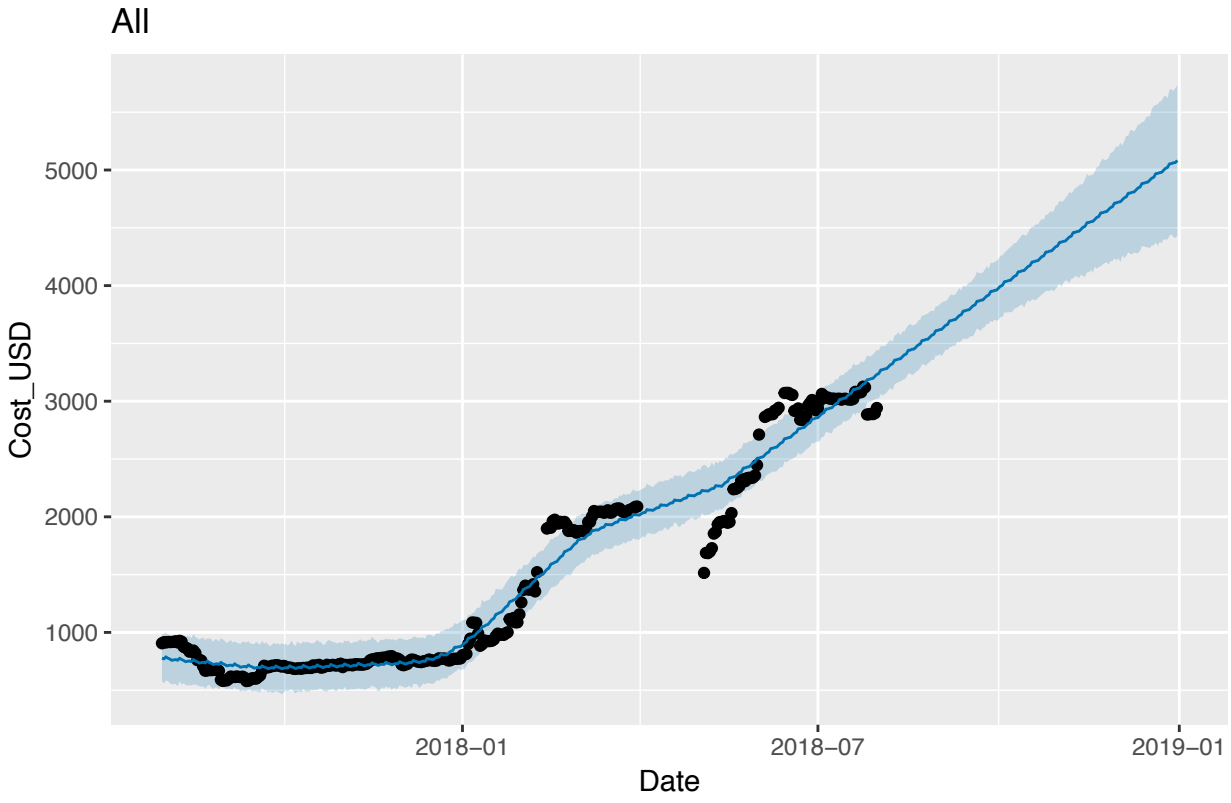
```
## 4 Detailed        app11          <tibbl~ <S3: p~ <data.f~ <data.fra~ <S3:~
## 5 Detailed        app12          <tibbl~ <S3: p~ <data.f~ <data.fra~ <S3:~
## 6 Detailed        app13          <tibbl~ <S3: p~ <data.f~ <data.fra~ <S3:~
```

Aggregate Level Forecast Cost_$

Activity for all items combined then forecasted together.

```
#Aggregate Forecast
aggregate.forecast = models.summary %>%
  filter(.,forecast_method == 'Aggregate')

#plot of aggregate level forecast
aggregate.forecast$plot
```

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




```
#monthly aggregate level forecast
aggregate.forecast %>%
  select(.,forecast) %>%
  unnest() %>%
  select(.,ds,yhat,yhat_lower,yhat_upper) %>%
  mutate(.,ds_month = floor_date(ds,"month")) %>%
  group_by(ds_month) %>%
  select(-ds) %>%
  summarize(
    yhat = sum(yhat),
    yhat_lower = sum(yhat_lower),
```

```
    yhat_upper = sum(yhat_upper)
  ) %>%
  filter(.,ds_month>=date("2018-08-01"))
```

```
## # A tibble: 5 x 4
##   ds_month               yhat yhat_lower yhat_upper
##   <dttm>                <dbl>      <dbl>      <dbl>
## 1 2018-08-01 00:00:00 106208.     99653.    112752.
## 2 2018-09-01 00:00:00 113854.    106695.    120783.
## 3 2018-10-01 00:00:00 129118.    119526.    138478.
## 4 2018-11-01 00:00:00 136079.    123487.    148403.
## 5 2018-12-01 00:00:00 152046.    134339.    169696.
```

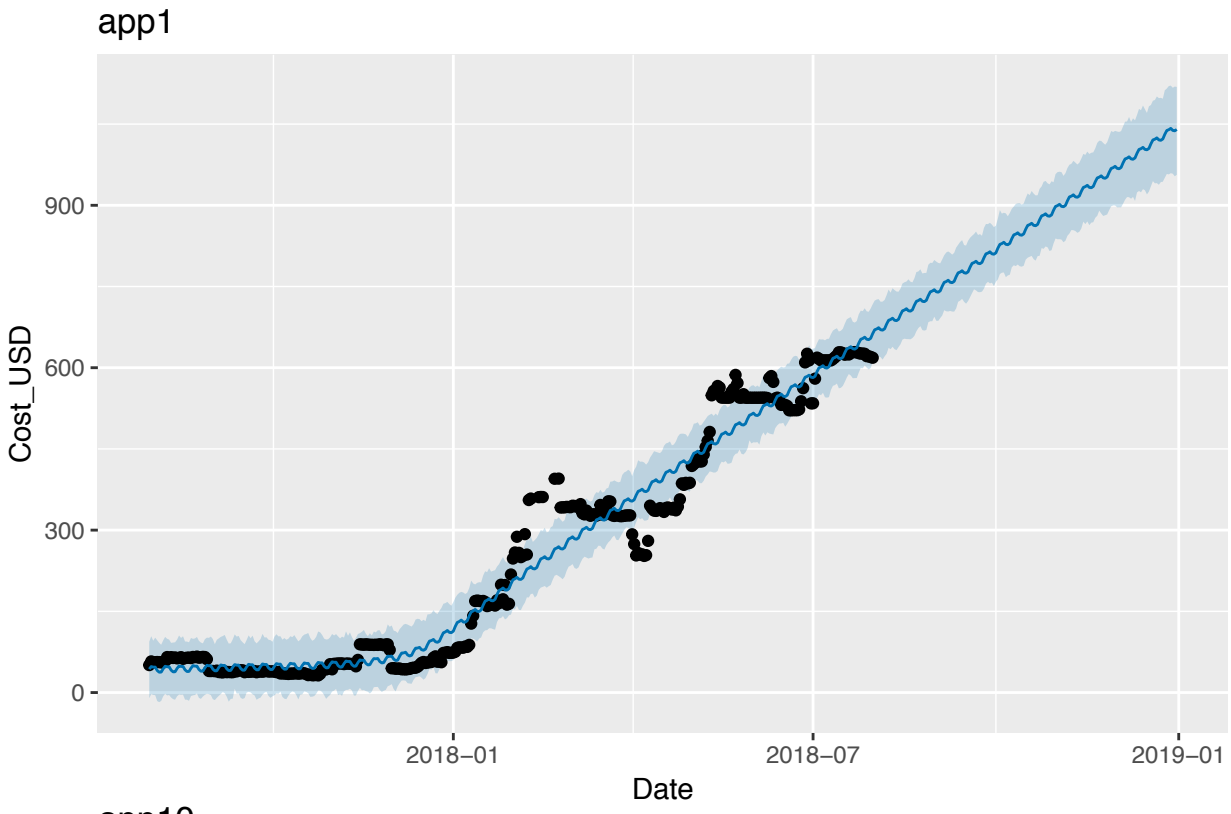
Detailed Level Forecast Cost_$

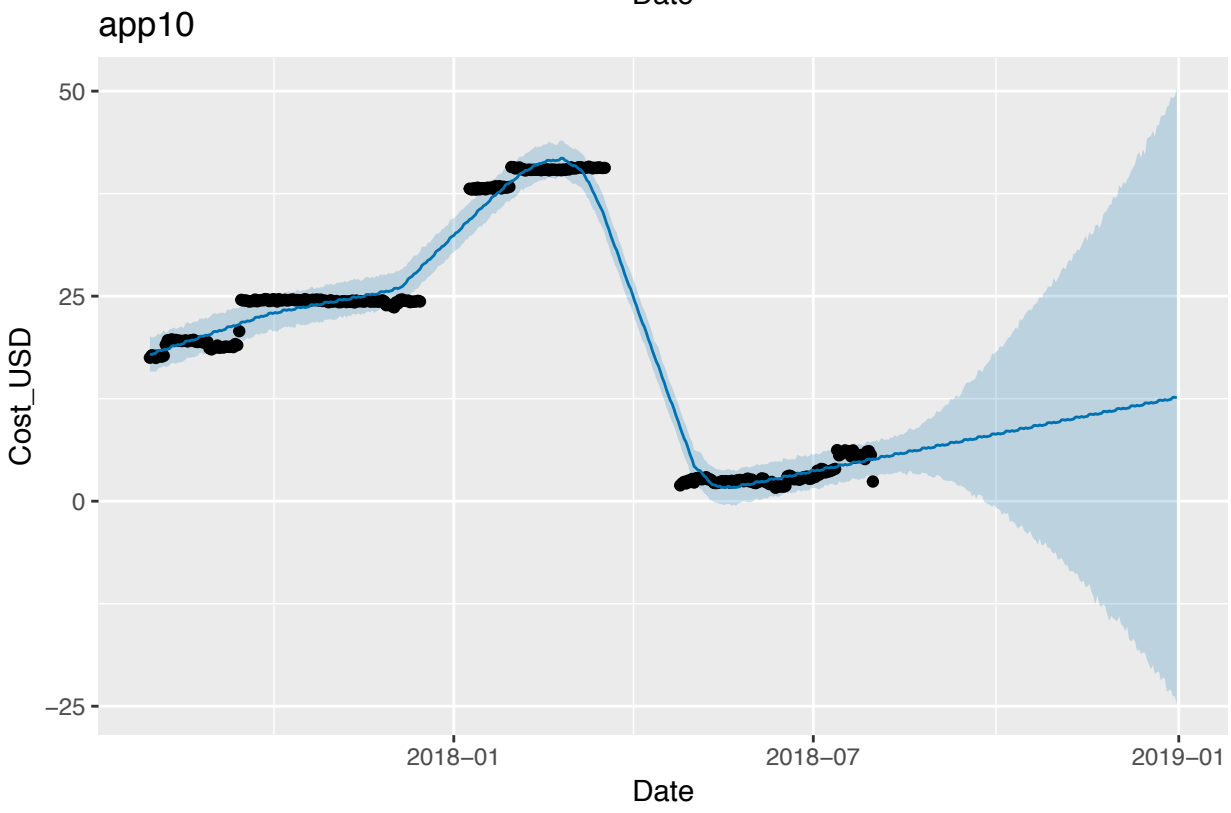
Groups forecasted individually then aggregated. Bottom Up Method.

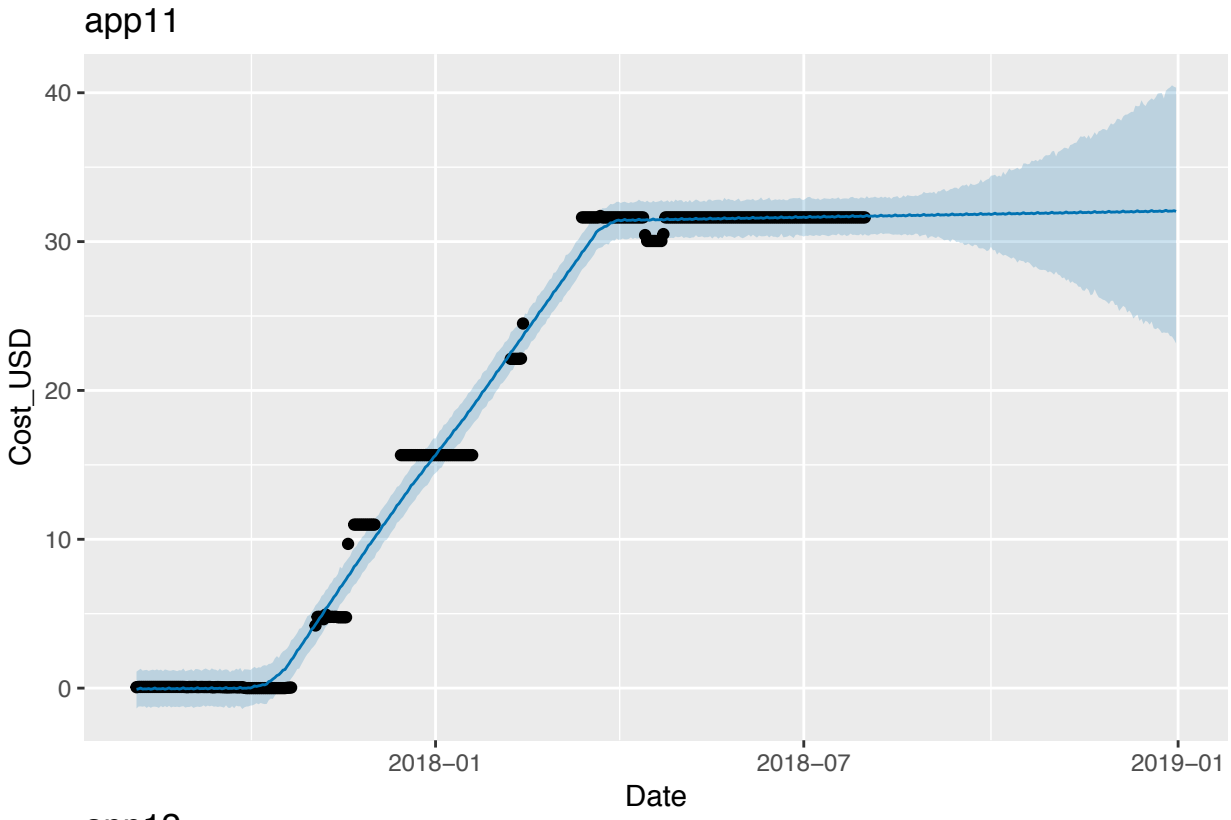
```
#Aggregate Forecast
detailed.forecast = models.summary %>%
  filter(.,forecast_method == 'Detailed')

#individual forecasts aggregated to to monthly level
detailed.forecast %>%
  select(.,forecast) %>%
  unnest() %>%
  select(.,ds,yhat,yhat_lower,yhat_upper) %>%
  mutate(.,ds_month = floor_date(ds,"month")) %>%
  group_by(ds_month) %>%
  select(-ds) %>%
  summarize(
    yhat = sum(yhat),
    yhat_lower = sum(yhat_lower),
    yhat_upper = sum(yhat_upper)
  ) %>%
  filter(.,ds_month>=date("2018-08-01"))
```

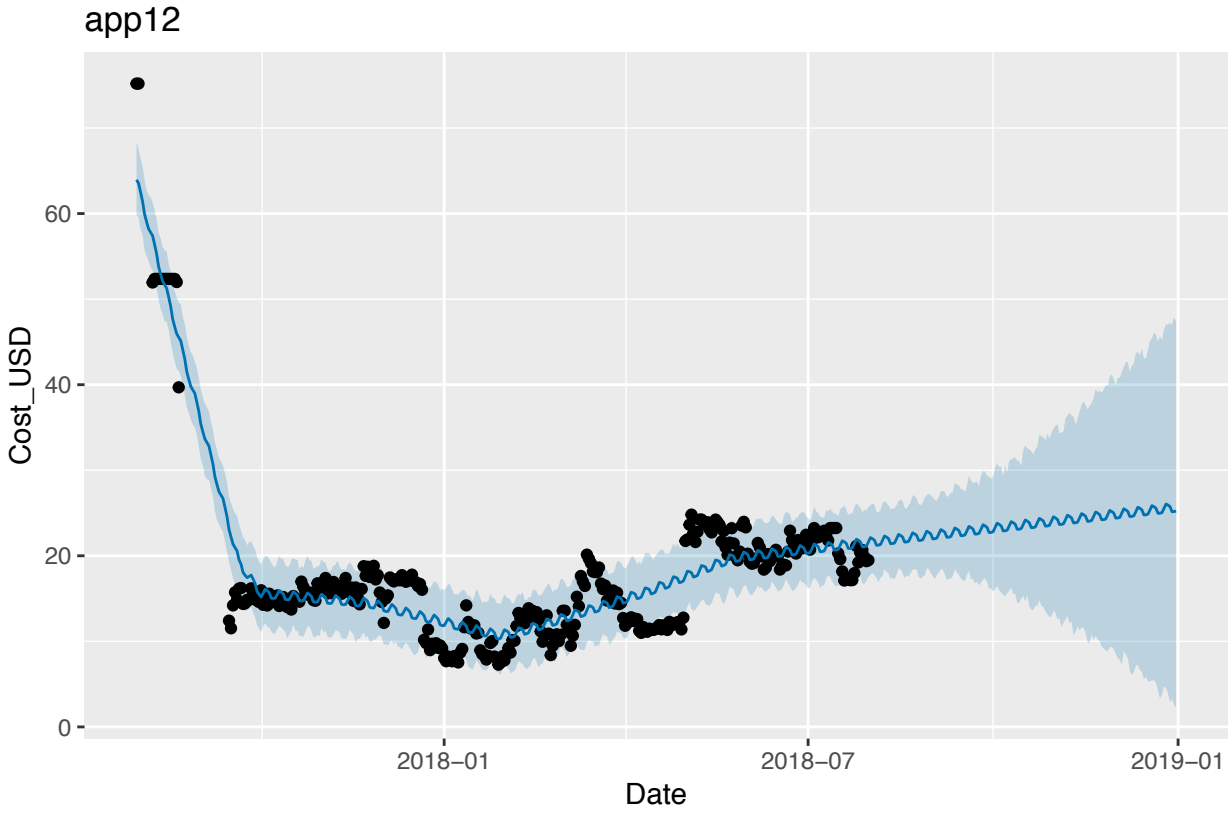
```
## # A tibble: 5 x 4
##   ds_month               yhat yhat_lower yhat_upper
##   <dttm>                <dbl>      <dbl>      <dbl>
## 1 2018-08-01 00:00:00 113607.    100050.    126951.
## 2 2018-09-01 00:00:00 125107.    110901.    139691.
## 3 2018-10-01 00:00:00 144916.    126881.    163135.
## 4 2018-11-01 00:00:00 155526.    133718.    177420.
## 5 2018-12-01 00:00:00 176344.    148225.    204573.
```

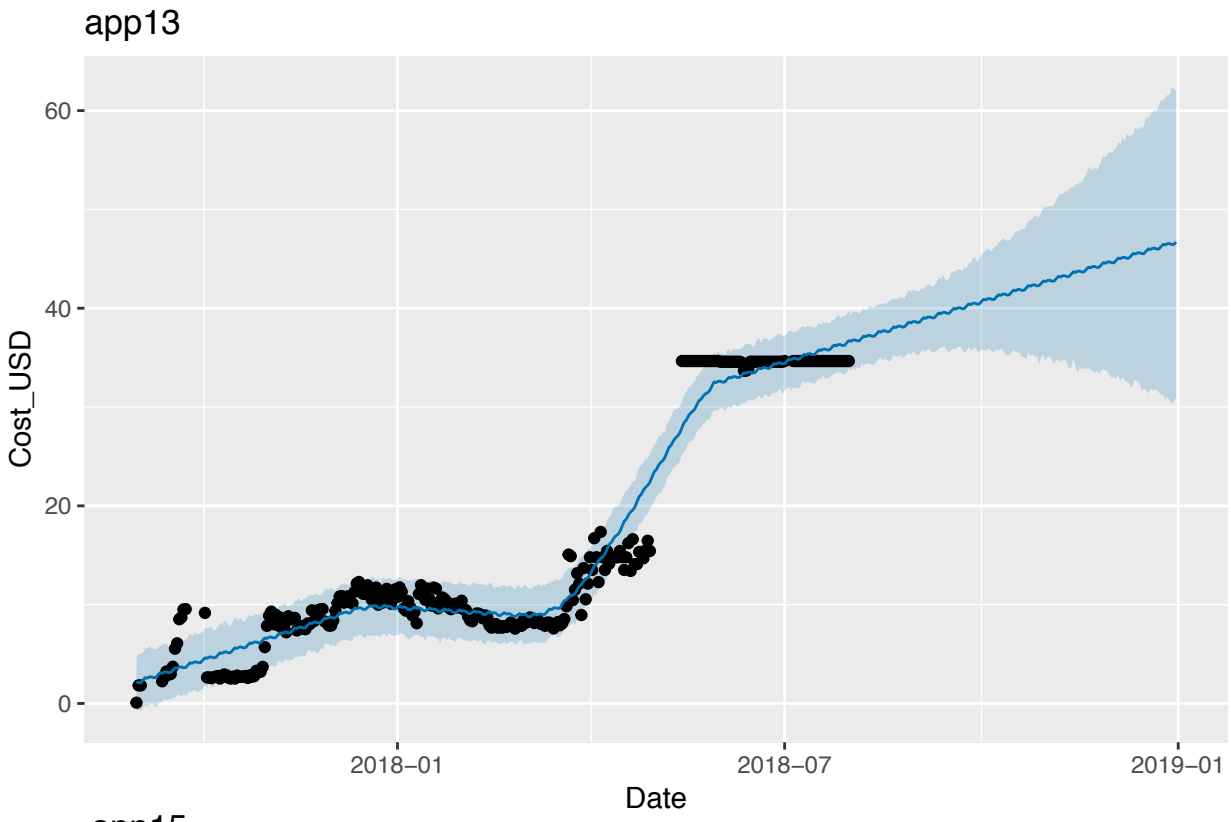
```
#plot of detailed level forecasts
detailed.forecast$plot
```

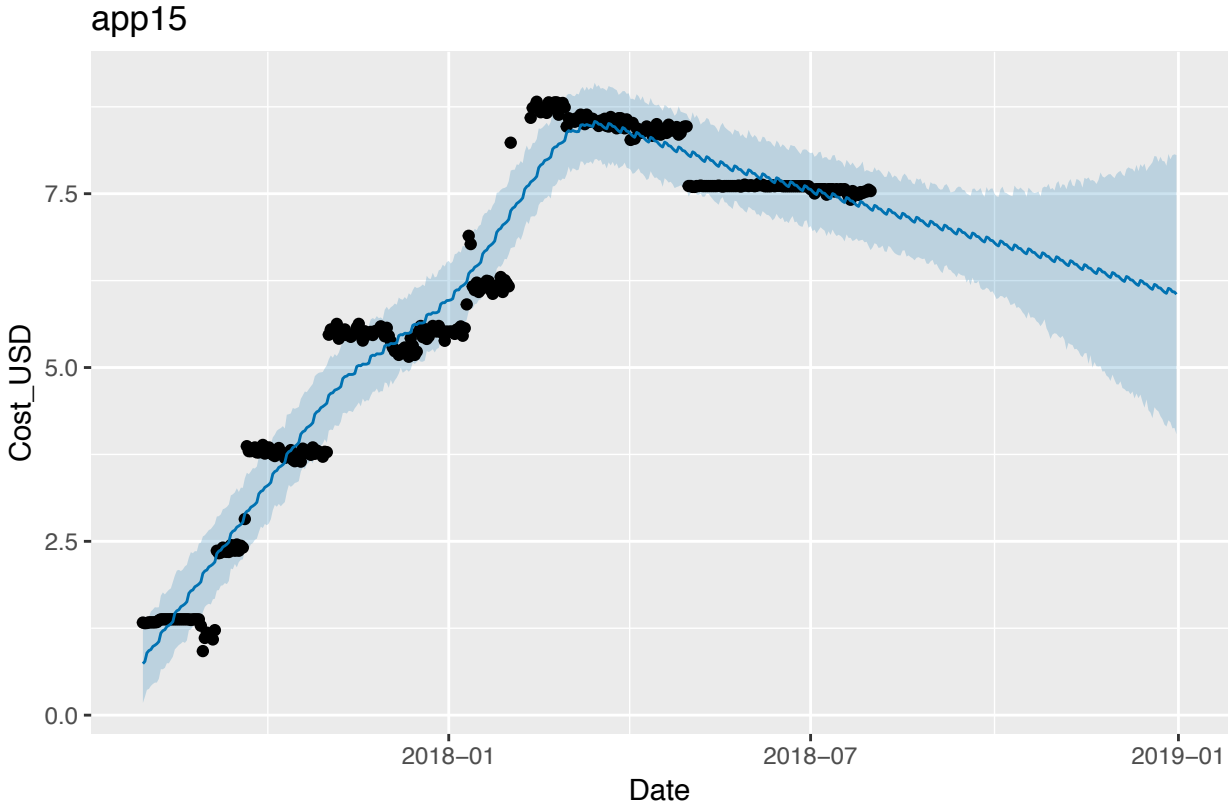

app1
900
600
300
0
Cost_USD
2018–01
2018–07
2019–01
Date

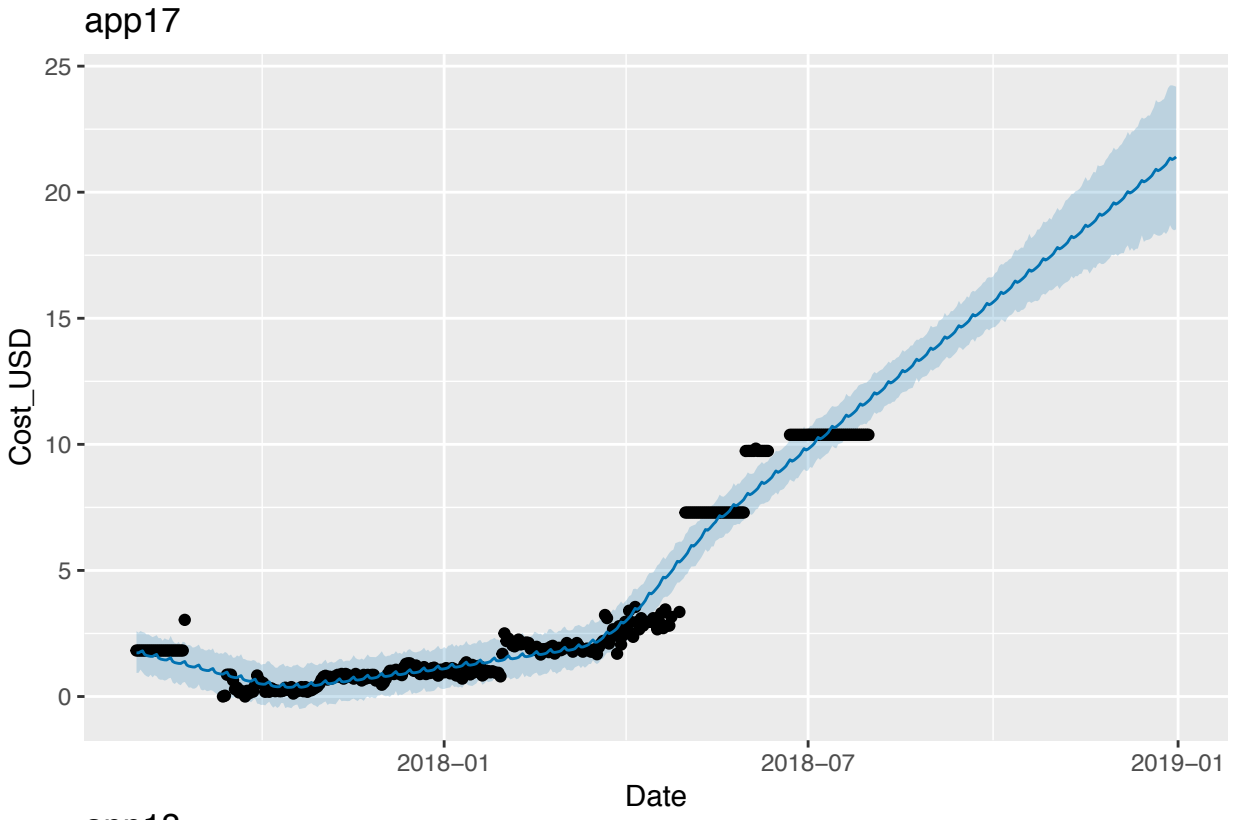

app10
50
25
0
−25
Cost_USD
2018–01
2018–07
2019–01
Date

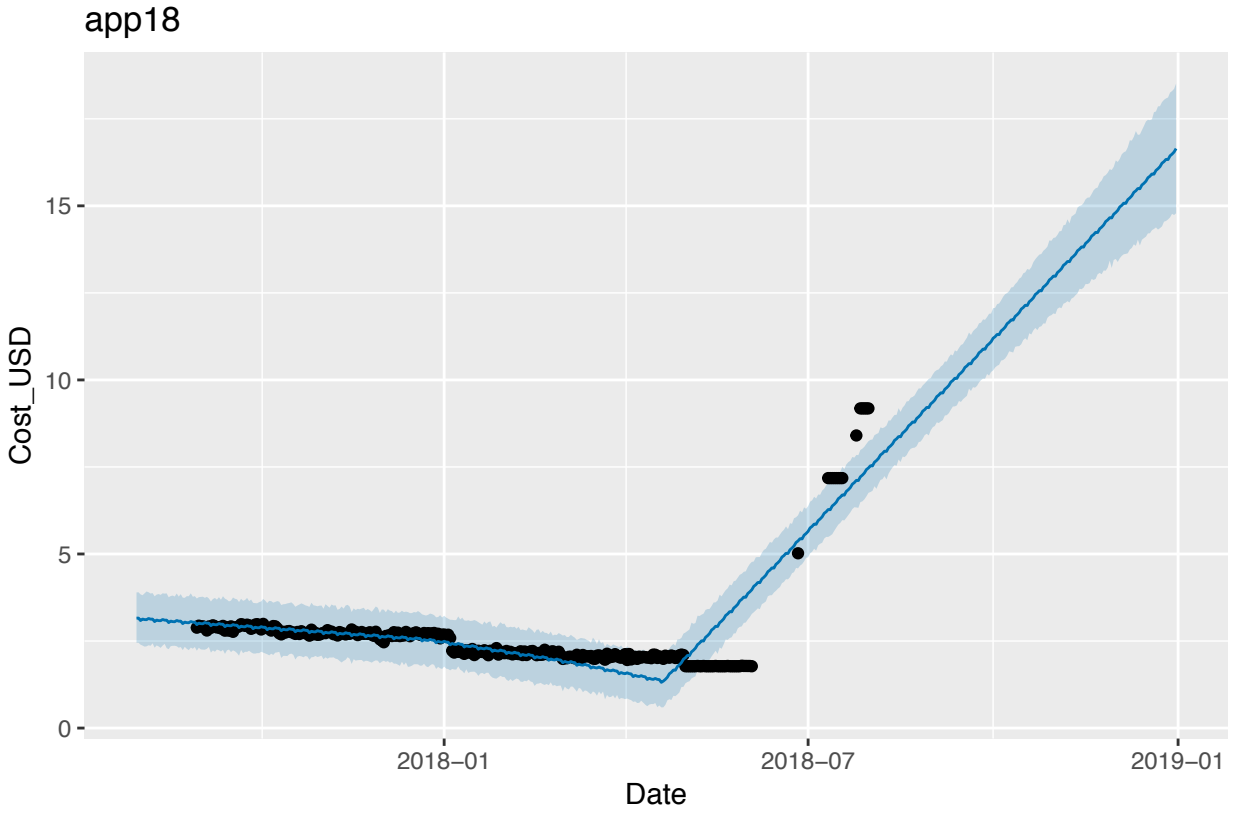

app11
40
30
20
10
0
Cost_USD
2018–01
2018–07
2019–01
Date

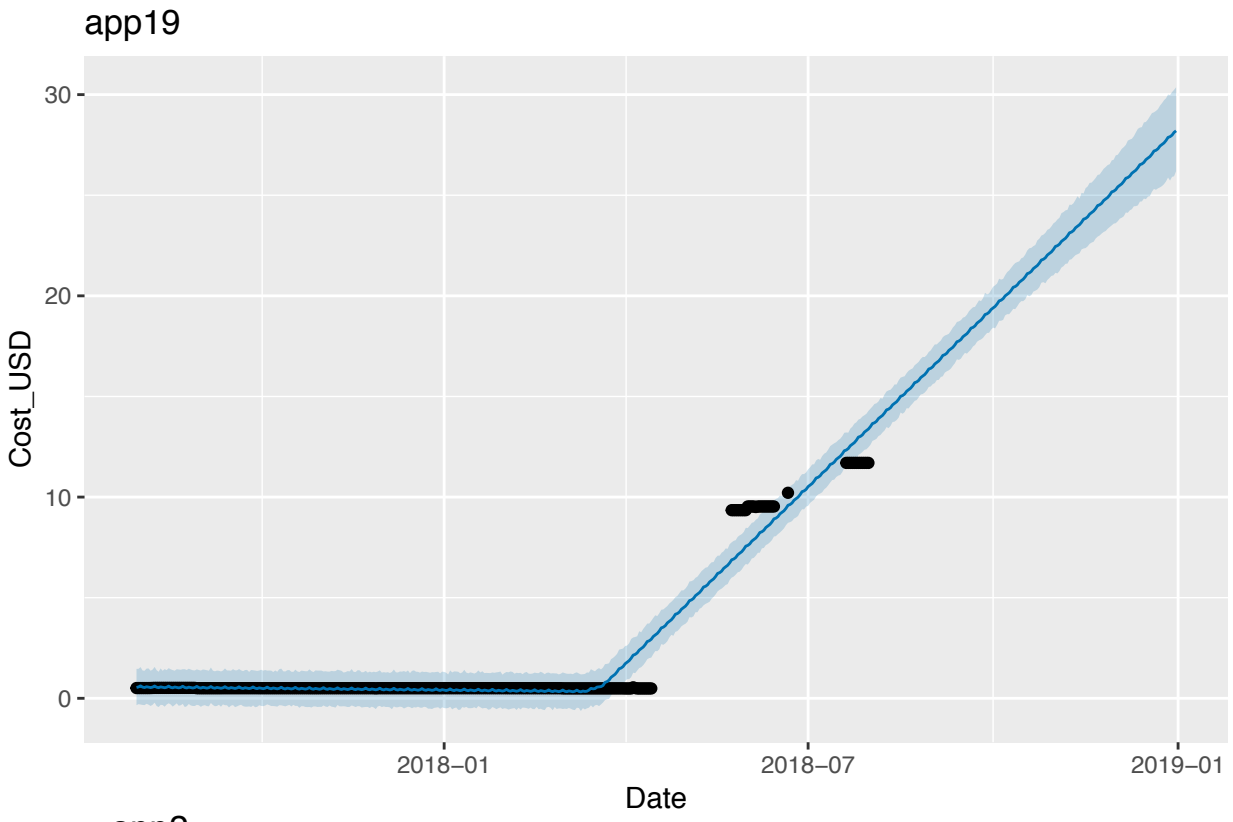

app12
60
40
20
0
Cost_USD
2018–01
2018–07
2019–01
Date

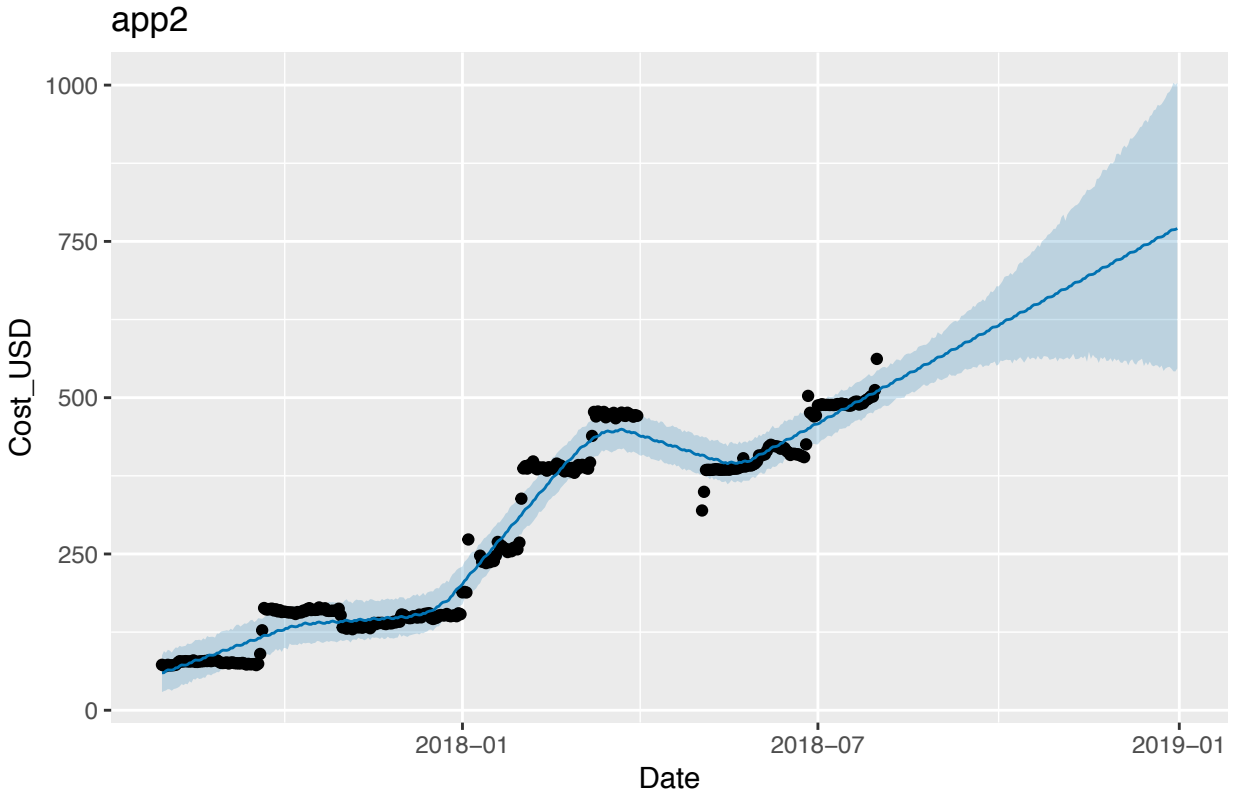

app13
60
40
20
0
Cost_USD
2018–01
2018–07
2019–01
Date

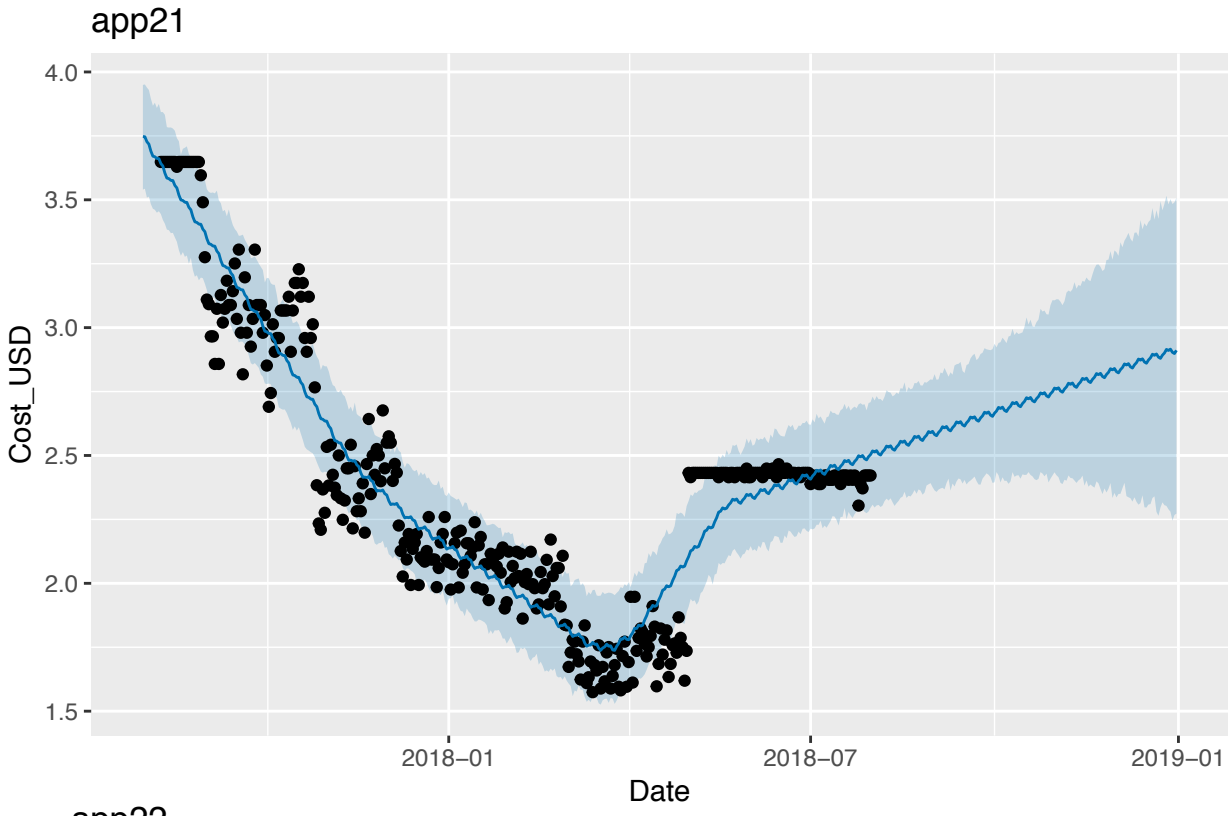

app15
7.5
5.0
2.5
0.0
Cost_USD
2018–01
2018–07
2019–01
Date

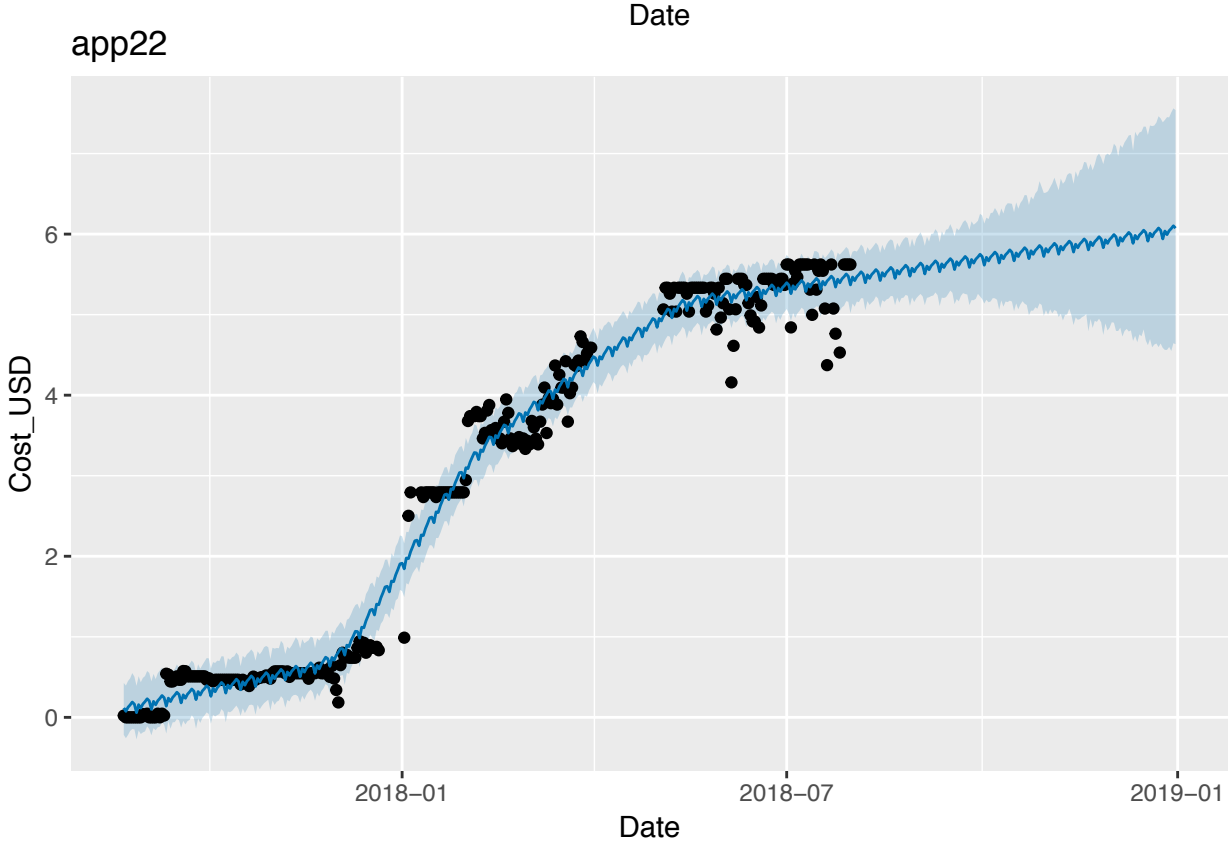

app17
25
20
15
10
5
0
Cost_USD
2018–01
2018–07
2019–01
Date

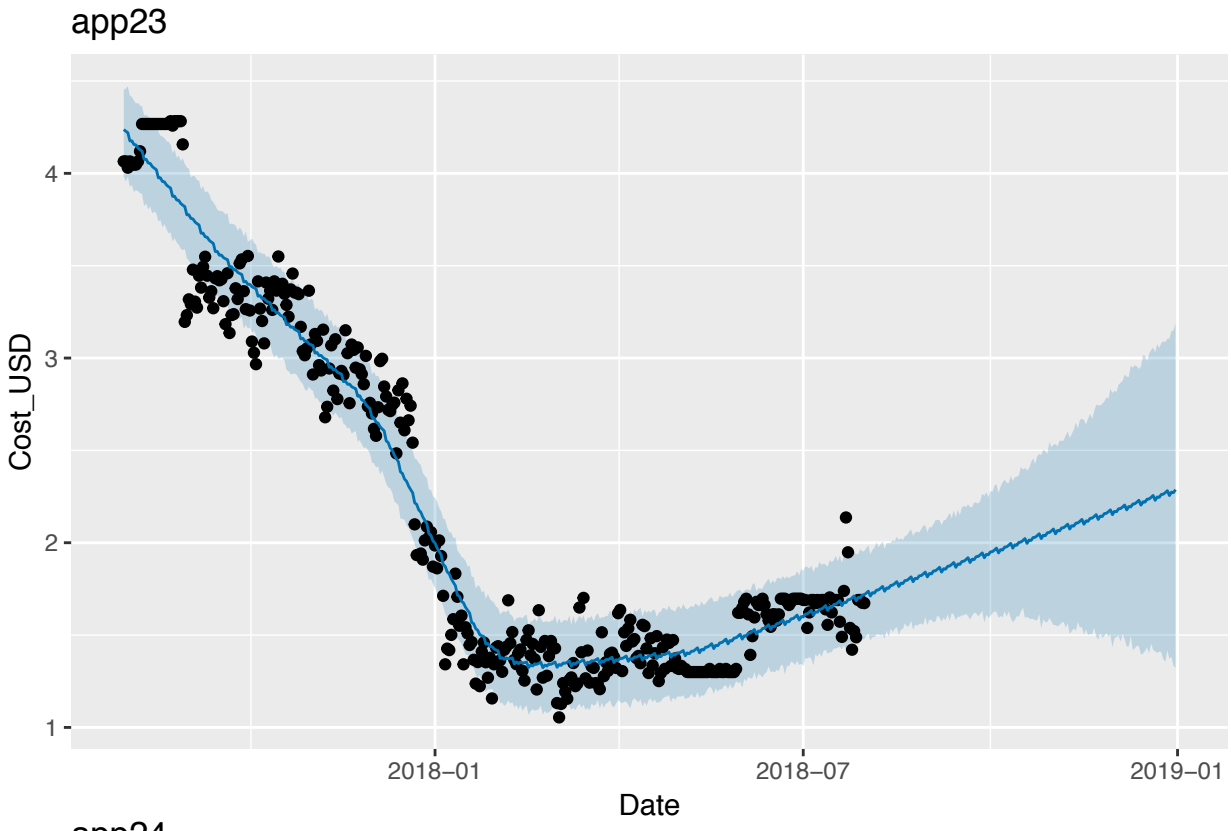

app18
15
10
5
0
Cost_USD
2018–01
2018–07
2019–01
Date

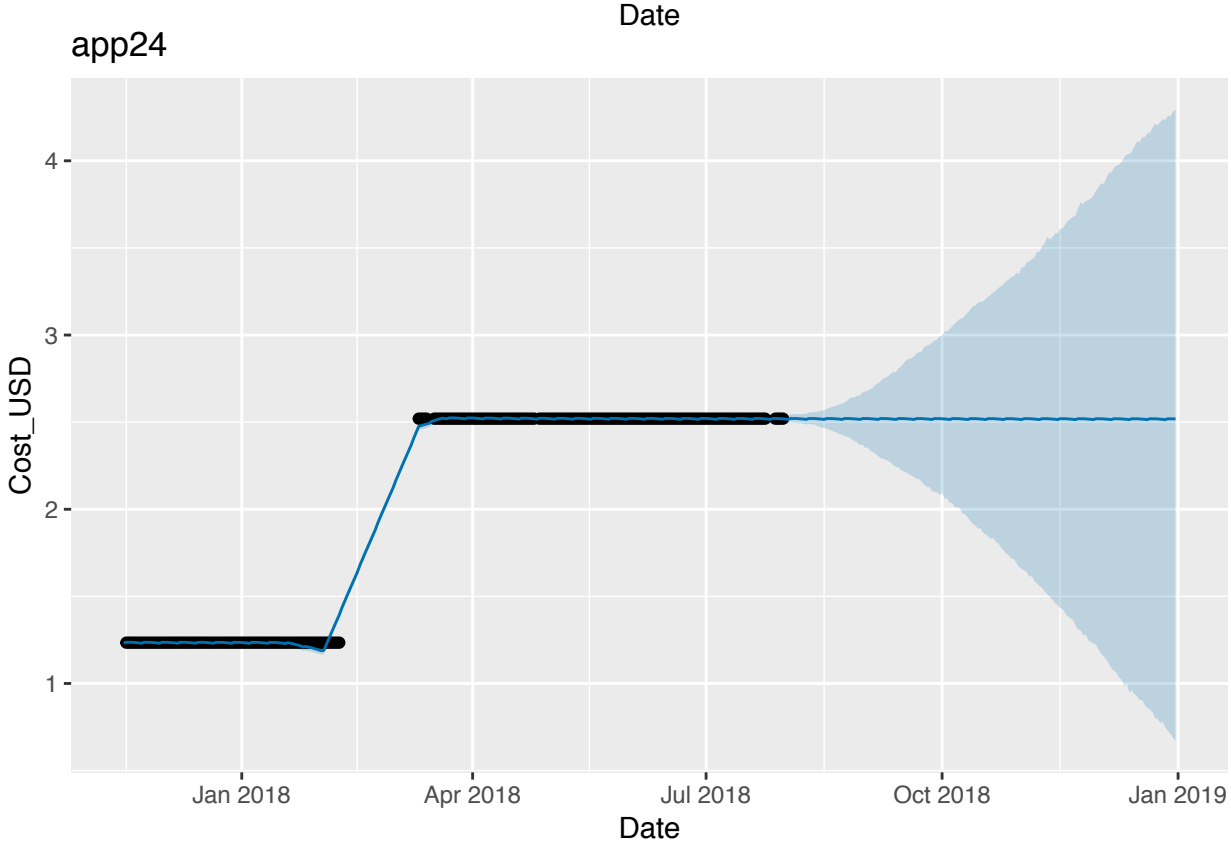

app19
30
20
10
0
Cost_USD
2018–01
2018–07
2019–01
Date

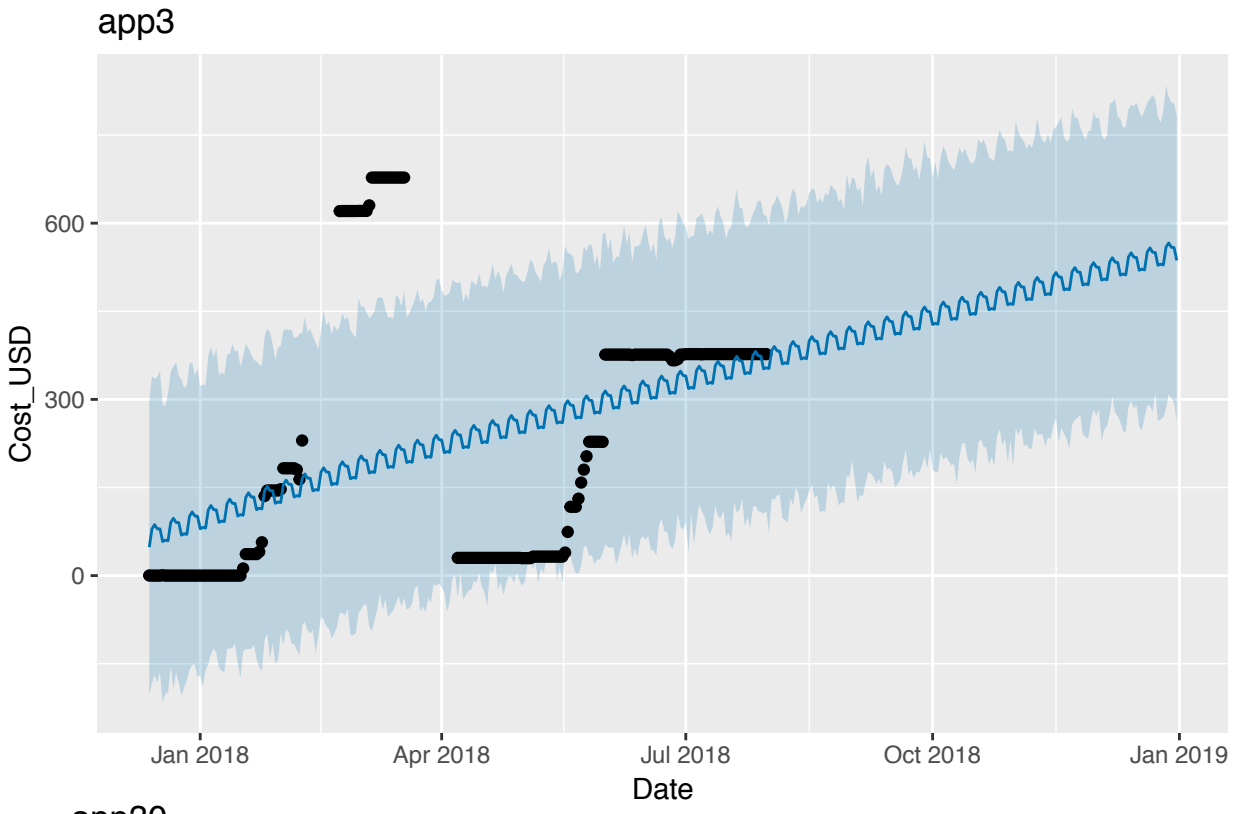

app2
1000
750
500
250
0
Cost_USD
2018–01
2018–07
2019–01
Date

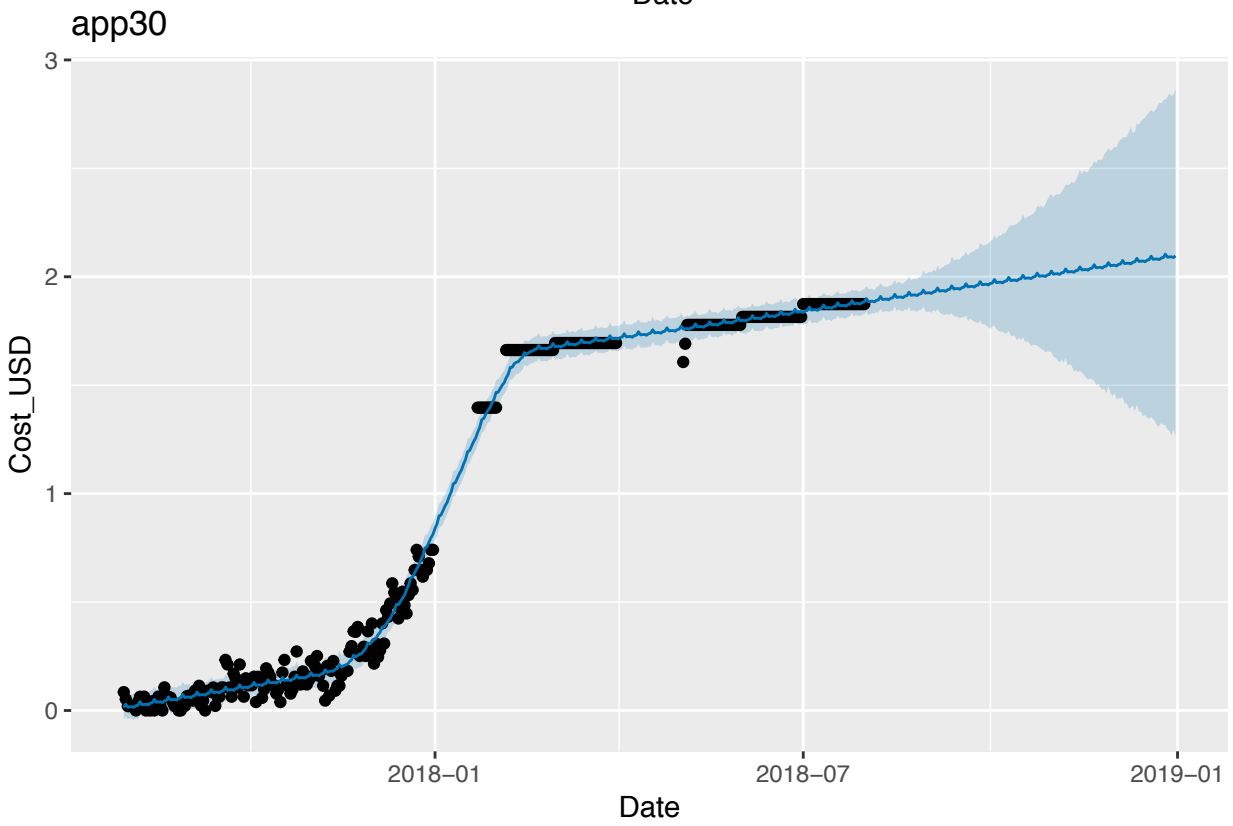

app21
4.0
3.5
3.0
2.5
2.0
1.5
Cost_USD
2018–01
2018–07
2019–01
Date

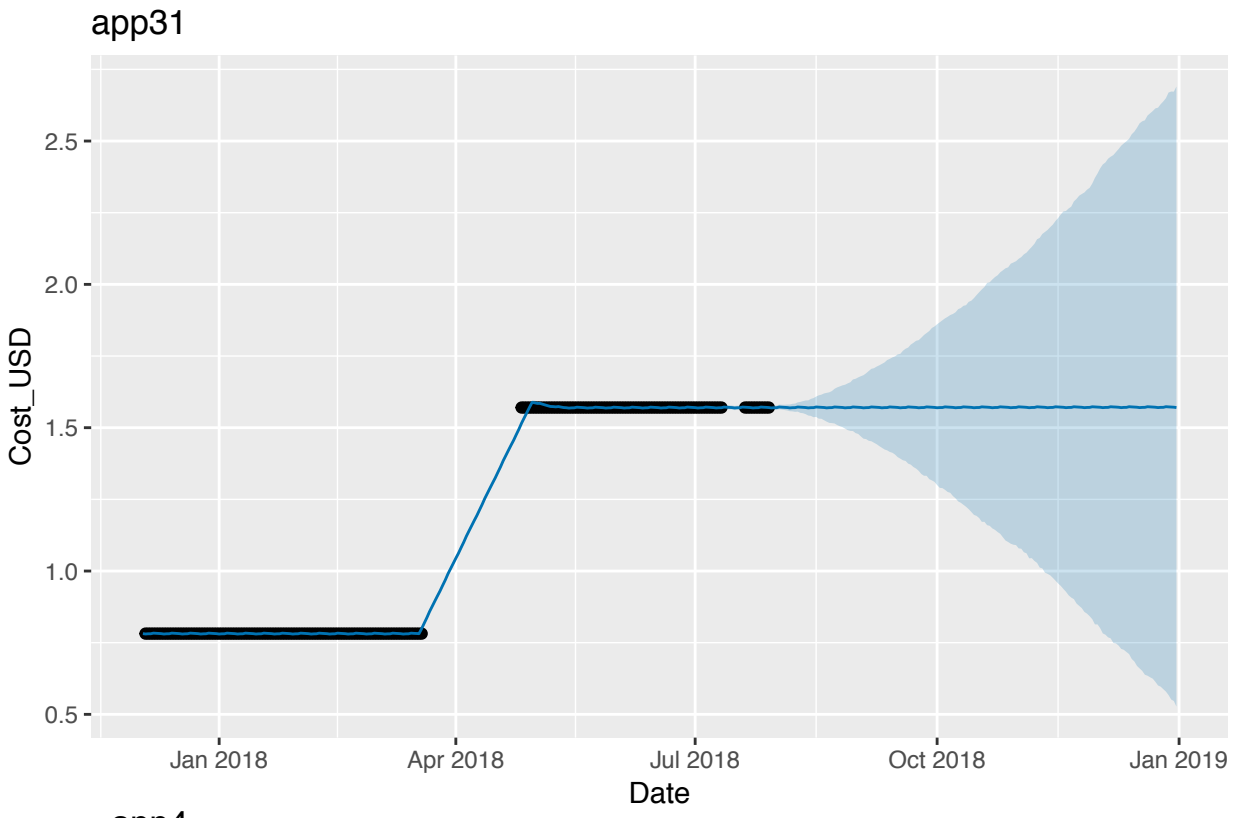

app22
6
4
2
0
Cost_USD
2018–01
2018–07
2019–01
Date

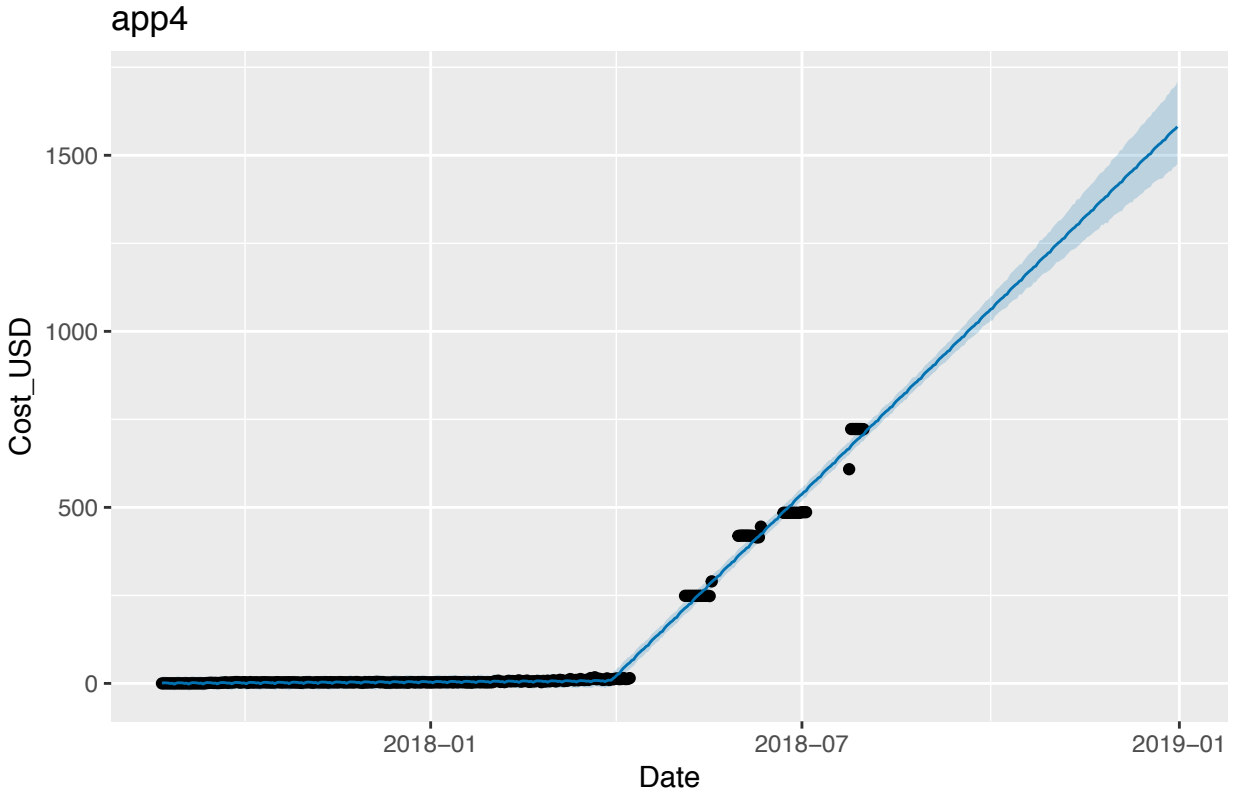

app23
Cost_USD
4
3
2
1
2018–01
2018–07
2019–01
Date

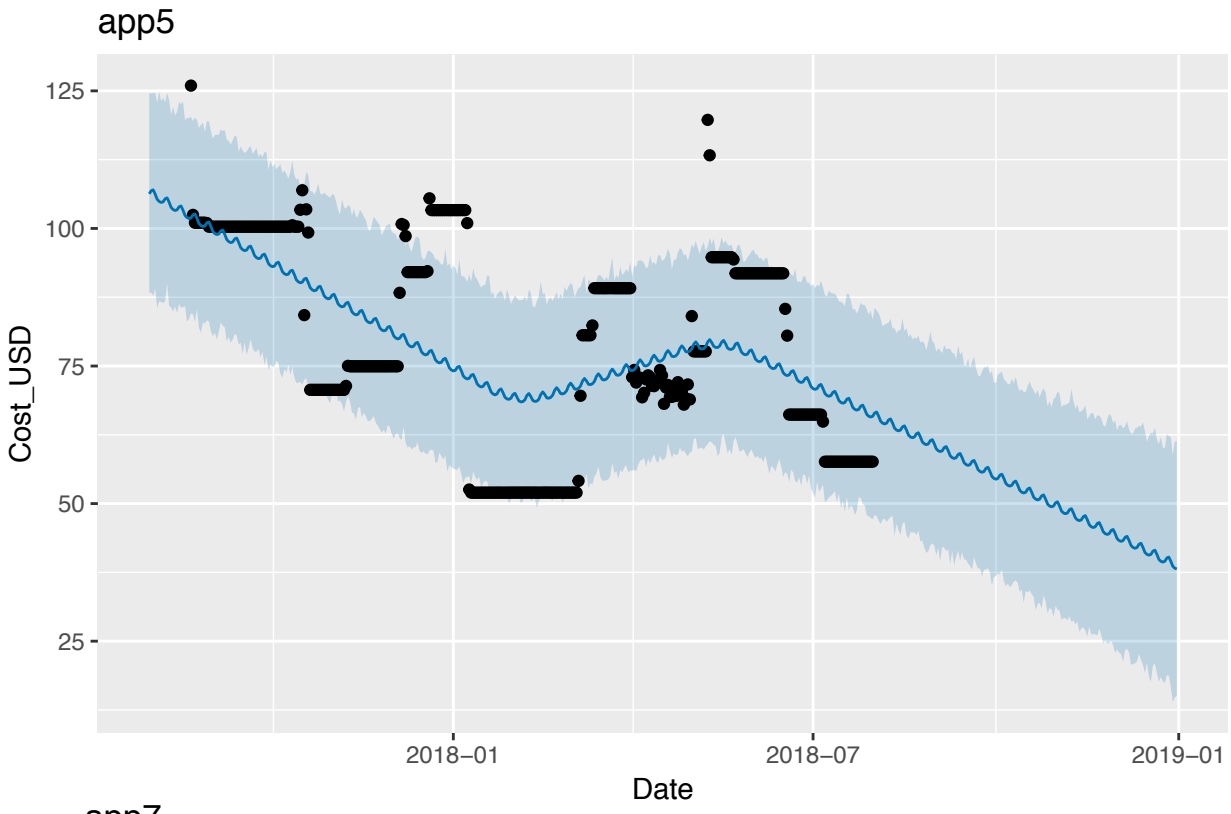

app24
Cost_USD
4
3
2
1
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
Date

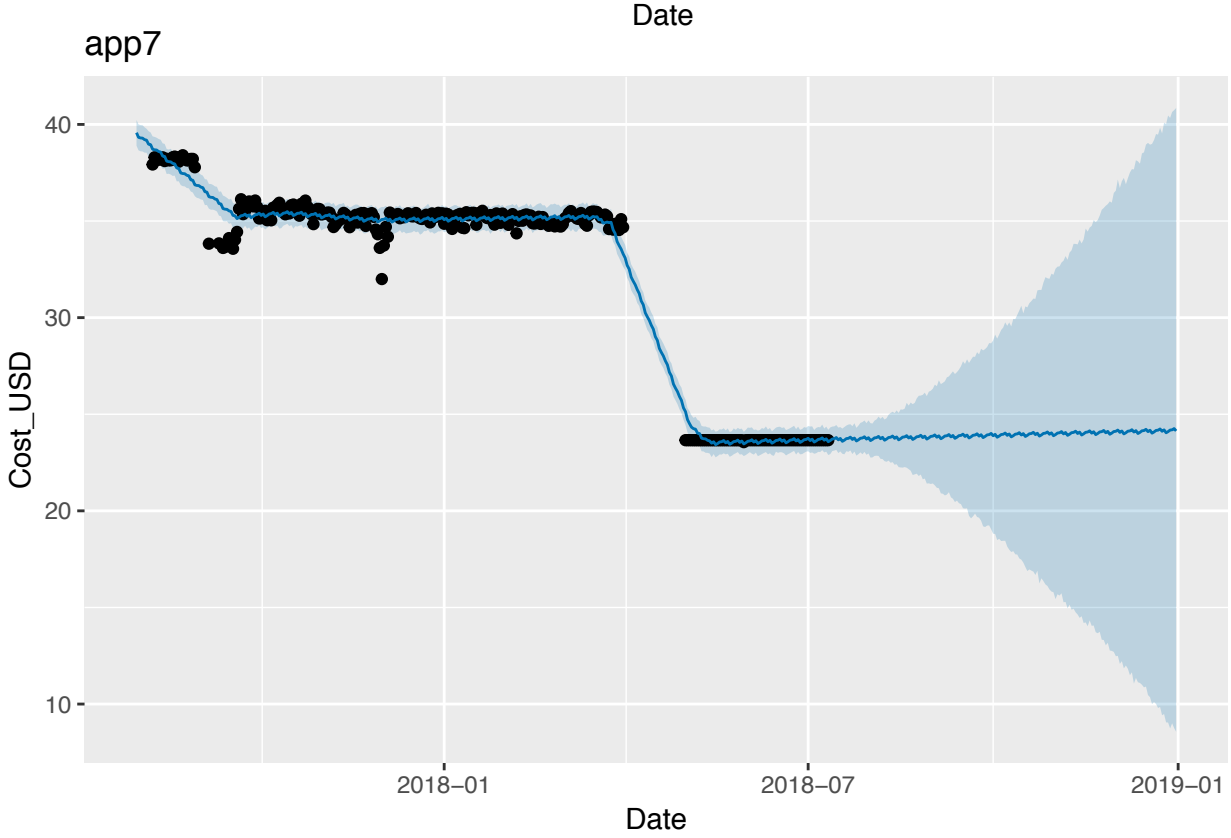

app3
600
300
0
Cost_USD
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
Date

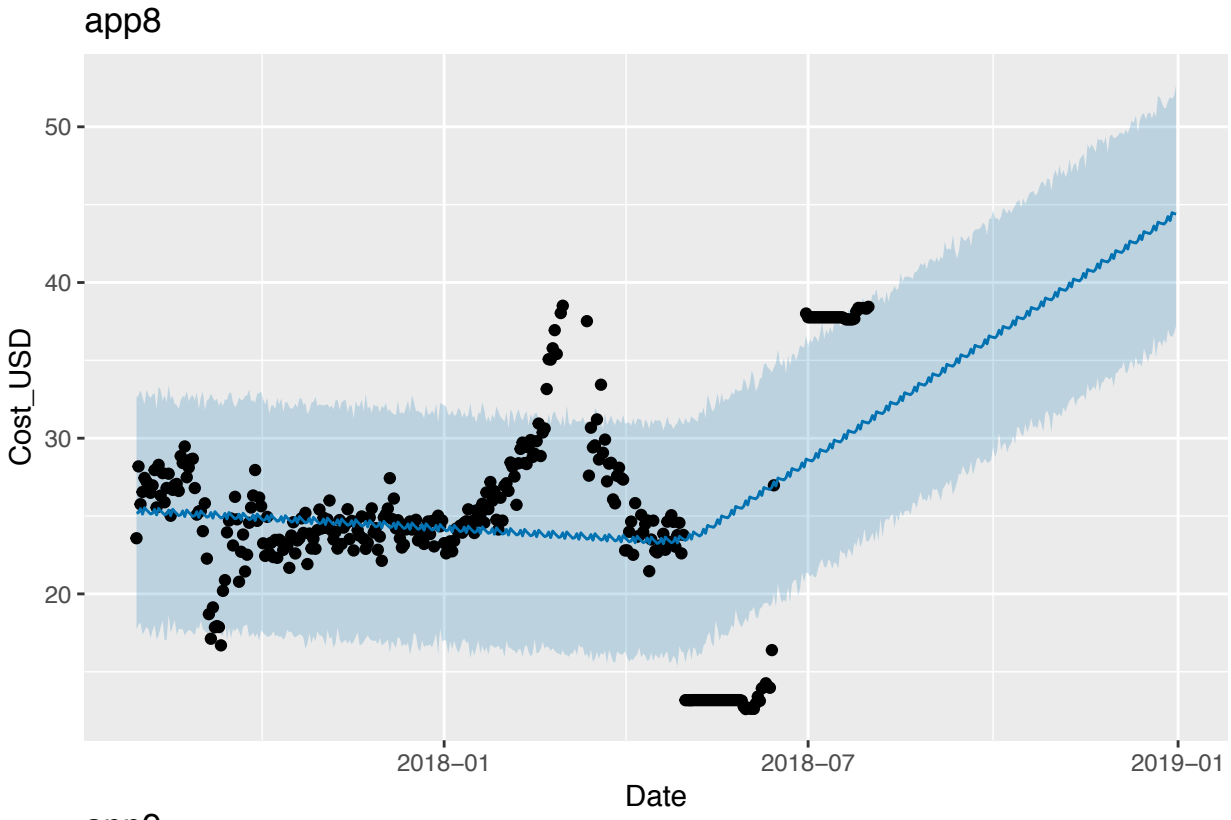

app30
3
2
1
0
Cost_USD
2018–01
2018–07
2019–01
Date

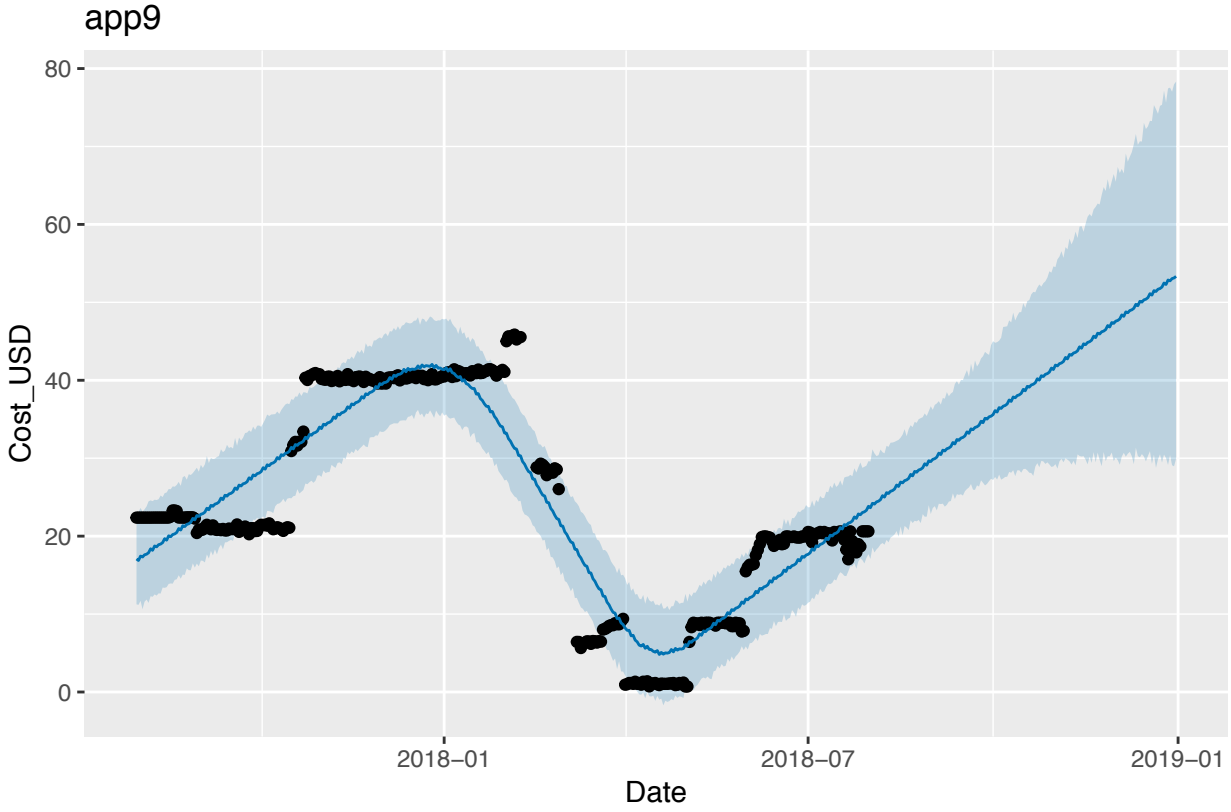

app31
2.5
2.0
1.5
1.0
0.5
Cost_USD
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
Date

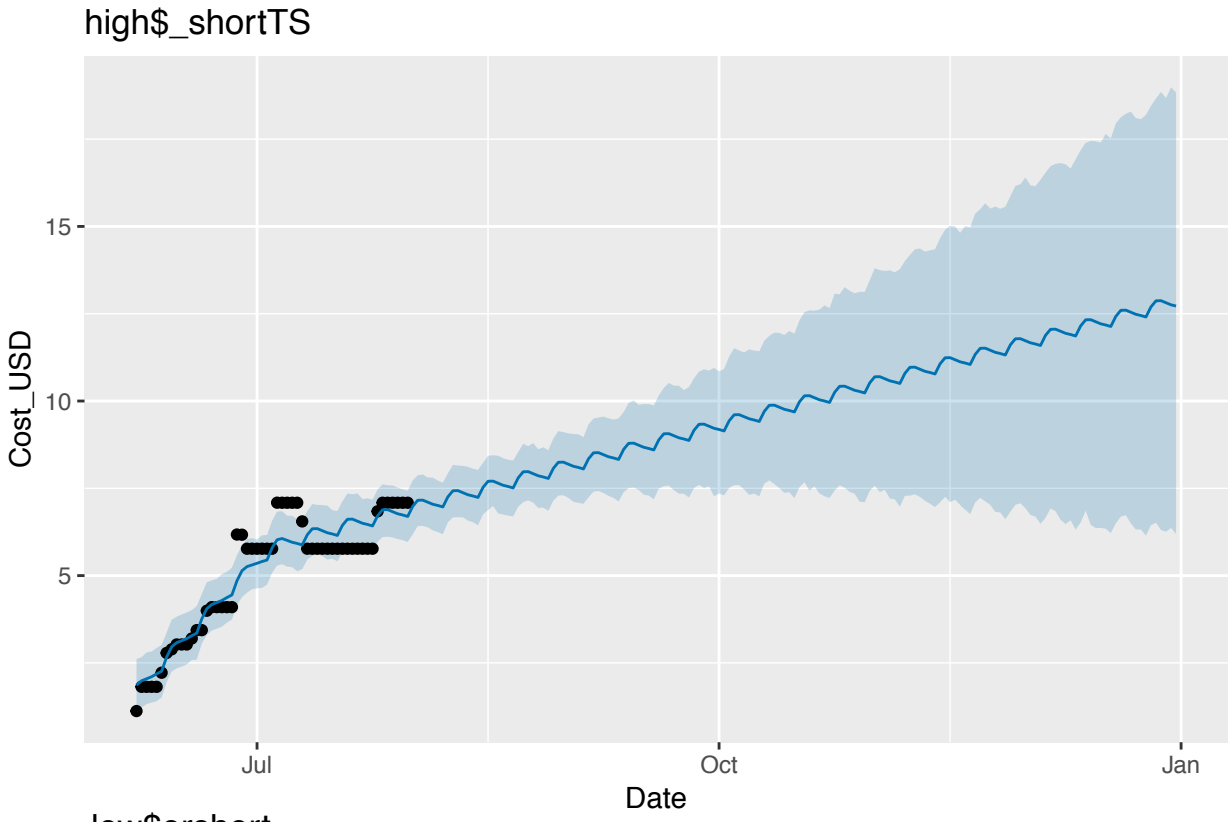

app4
1500
1000
500
0
Cost_USD
2018–01
2018–07
2019–01
Date

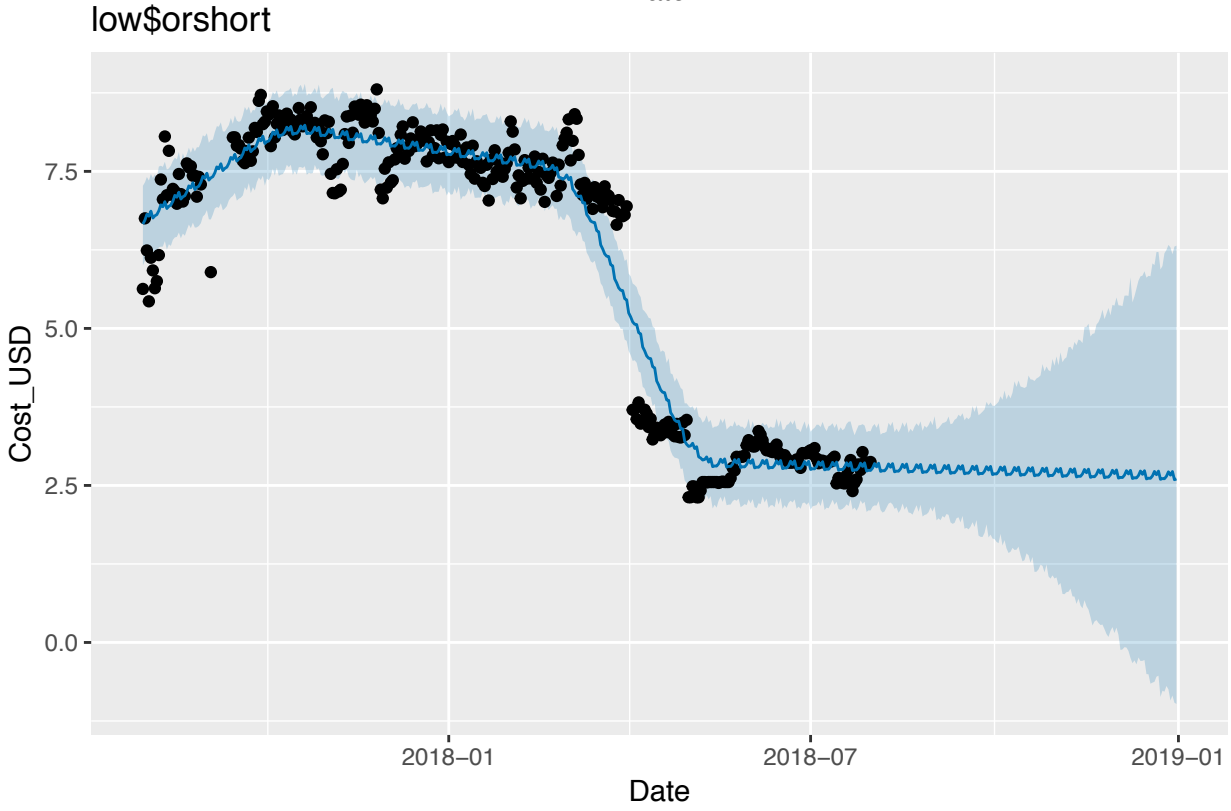

app5
125
100
75
50
25
Cost_USD
2018–01
2018–07
2019–01
Date

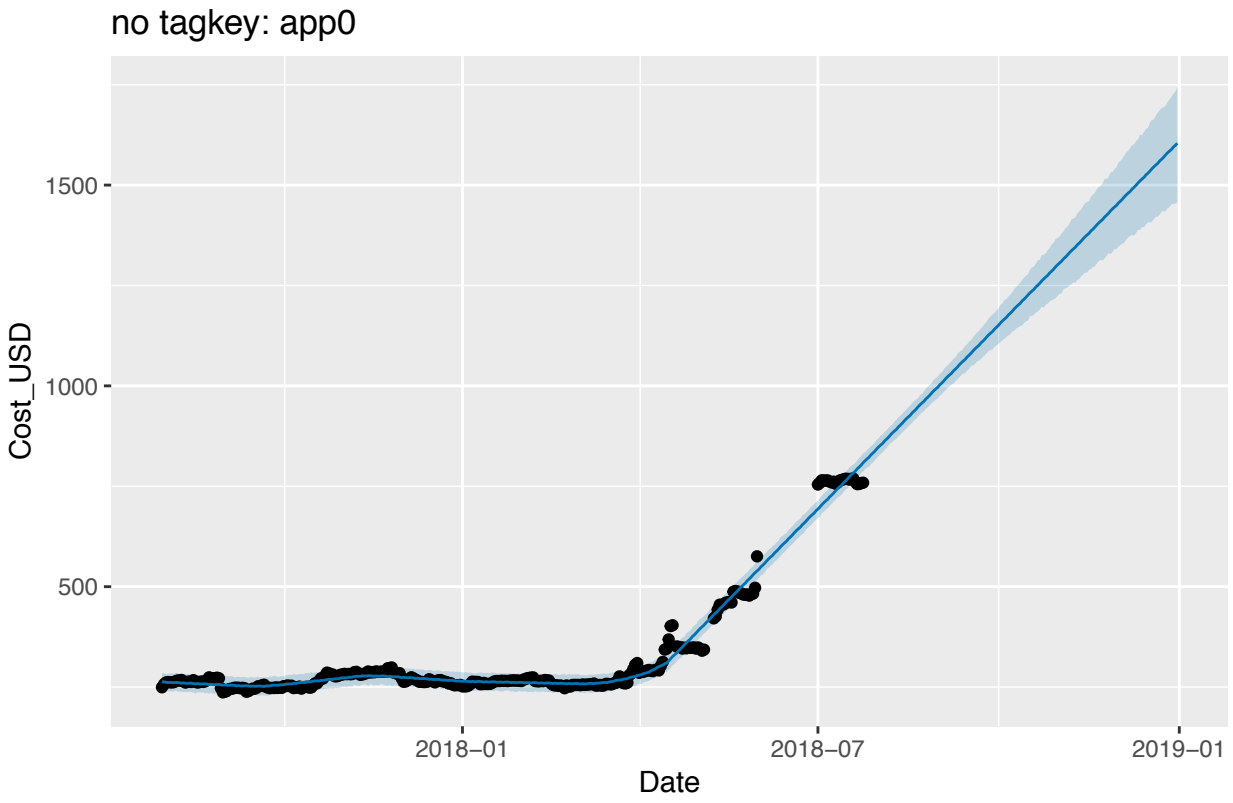

app7
40
30
20
10
Cost_USD
2018–01
2018–07
2019–01
Date


app8
50
40
30
20
Cost_USD
2018–01
2018–07
2019–01
Date


app9
80
60
40
20
0
Cost_USD
2018–01
2018–07
2019–01
Date


high$_shortTS
Cost_USD
15
10
5
Jul
Oct
Jan
Date


low$orshort
Cost_USD
7.5
5.0
2.5
0.0
2018–01
2018–07
2019–01
Date


no tagkey: app0
1500
1000
500
Cost_USD
2018–01
2018–07
2019–01
Date