Predicting Household Composition with TV Viewing Data

Generating Features of TV Viewing

Rafael Lüchinger

31 Januar 2019

Introduction

TV audiance in Switzerland is measured by Mediapulse AG. A representative panel of roughly 2000 households is constantly under measurement. These homes were carfully selected by a complex sampling design and all householdmembers have agreed to be part of the study. The TV viewing of each householdmember is individually recorded and detailed demografics are known for each person. This allows the market to target TV audiances by relevant characteristics like age gender and many more.

One issue with the panel approach is poor granularity. That means sometimes the system can not provide any audiance figures for a specific channel or airtime. It is likely that in the Swiss population of about 3.5 Mio. households at least a few people are watching even exotic programs at exotic times of the day. However, out of a panel of 2000 households chances are high that no one was watching that content. This is not a bias of the measurement but poor resolution.

A solution to this problem could be the inclusion of third party data. Set-Top-Boxes (STB) of TV-provider (Swisscom, UPC, etc.) are automatically recording the TV consumption in millions of Swiss homes and the data is returned to the providers servers (return path data, RPD). There are still many issues with these data that are currently adressed.

One major issue of RPD is that the viewing data is on household level, not on indvidual level. Household-level data is of little use to the market. Because it gives no insight in target groups based on age and gender and alike.

It is unlikely that RPD provider will ever measure the individual viewing or survey individual demografics within the subscribers homes. Apart from region code, the only information about the home is the viwing data itself. So the question arises if it is possible to predict the household composition based on viewing behavior.

The aim of this study is to explore the possiblity to predict the household composition within a household using TV viewing data. It seems to be a two-step-problem, first to find the number of householdmembers and then to assign age and gender to the individuals.

We will use the *Mediapulse TV-Panel* and its viewing data to study the subject. For all households in the panel its composition including household size and age and sex of each person is known. For each panel home the viewing data will be aggregated to household level. Different supervised machine learning algorithms will be fed with features extracted from that houesehold viewing data.

Target: Household Composition

Data Import

A R-Package `tv` is used to import the raw data of the *Mediapulse-TV-Panel*. The setup functions allows to specify the data to be read from disk into R. The import functions by default returns three data.tables:

1. `dem`: all individuals with their demografics
2. `view`: the TV viewing
3. `prog`: the program timetable with genre information

Date Range

TV viewing behavior is known to differ by season as well as by weekdays. During cold months TV viwing is more popular than during Sommer months. Similarly, on weekends people watch more TV than during the rest of the week. Not only differs the total viewing duration. Also the individual preferences for channels and programs might differ between weekend and workdays.

To extract freatures of TV viewing, we will consider a range of 8 weeks during 2017. This should be long enough to reflect the individual viewing patterns. We focus on cold months and a period free of holidays or special TV events (FIFA Wolrdcup, etc.). We make sure to get an equal number of each weekday.

```r
dayx <- as.Date('2017-11-12')
days <- seq(dayx - 28, dayx + 27, by = "day")
table(weekdays(days))
```

```
## 
##    Dienstag Donnerstag    Freitag   Mittwoch     Montag    Samstag 
##           8          8          8          8          8          8 
##     Sonntag 
##           8 
```

Household Composition

A Household is not necessarily under measurement on every day of our 8 weeks. Sometimes a household leaves the panel, then a new household will join the panel. Also for technical reasons a household may drop the panel just for a couple of days.

To create a dataset of households not all households within the 8 weeks were included. Rather, the sample of a single specific day is choosen. This day is a sunday and exactly in the middle of the 8 weeks. The difference is small anyway, e.g. routhly a 2000 versus 2100 households.

```r
library('tv')
id <- setup(days, obs = "ind", dem.var = c("sg","hhsize","age","sex"),
            dem.day = dayx, dem.uni = FALSE, view = FALSE, prg = FALSE)
import(id)
```

```r
(dem <- dem[(!guest)]) # excluding guests
```

```
##              day   hh ind    pin weight guest age sex hw sg hhsize
##    1: 2017-11-12    6   1    601 1.0831 FALSE  70   1  2  2      2
##    2: 2017-11-12    6   2    602 1.3365 FALSE  67   2  1  2      2
##    3: 2017-11-12    9   1    901 0.9552 FALSE  55   1  2  2      4
##    4: 2017-11-12    9   2    902 0.9935 FALSE  50   2  1  2      4
##    5: 2017-11-12    9   4    904 1.5404 FALSE  21   1  2  2      4
##   ---                                                             
## 4384: 2017-11-12 6201   2 620102 1.8489 FALSE  73   2  1  1      2
## 4385: 2017-11-12 6204   1 620401 0.7449 FALSE  52   1  2  2      4
## 4386: 2017-11-12 6204   2 620402 0.9444 FALSE  47   2  1  2      4
## 4387: 2017-11-12 6204   3 620403 1.2214 FALSE  17   1  2  2      4
## 4388: 2017-11-12 6204   4 620404 0.9262 FALSE  13   2  2  2      4
```

On our sample day 2017-11-12 the TV-Panel was formed by 2006 households and 4388 individuals living in these households. This gives a average householdsize of 2.19.

A note on the variale *hhsize*. Household size is not constant over time, the number of people living in a household can change by natural reasons like birth, death, moving in or out. Also the variable *hhsize* is not necessarily equal to the sum of individuals for the following reasons:

- babys 0-2 years old are not part of the panel
- guests are part of the data but not counted for household size
- houshold size is counted 1, 2, . . . , 5+, 5+ meaning housholds with 5 or more members

A simple transformation of the `dem` data.table presents for each household on a row its composition. There are 2006 households. We call the household ID `pin` and `sg` is the linguistic region (german, french, italian). Age and Sex of up to 8 householdmembers. There is no missing data.

```
hh <- dcast(dem, day + hh + sg + hhsize ~ ind, value.var = c("age","sex"), fill = 0L)
setnames(hh, 'hh', 'pin')
rm(id, dem)
hh
```

```
##              day  pin sg hhsize age_1 age_2 age_3 age_4 age_5 age_6 age_7
##    1: 2017-11-12    6  2      2    70    67     0     0     0     0     0
##    2: 2017-11-12    9  2      4    55    50     0    21    17     0     0
##    3: 2017-11-12   14  1      2    71    72     0     0     0     0     0
##    4: 2017-11-12   20  1      2    59    49     0     0     0     0     0
##    5: 2017-11-12   21  1      2    63    52     0     0     0     0     0
##   ---
## 2002: 2017-11-12 6196  1      2    74    76     0     0     0     0     0
## 2003: 2017-11-12 6197  1      2    40    47     0     0     0     0     0
## 2004: 2017-11-12 6200  1      2    63    72     0     0     0     0     0
## 2005: 2017-11-12 6201  1      2    71    73     0     0     0     0     0
## 2006: 2017-11-12 6204  2      4    52    47    17    13     0     0     0
##       age_8 sex_1 sex_2 sex_3 sex_4 sex_5 sex_6 sex_7 sex_8
##    1:     0     1     2     0     0     0     0     0     0
##    2:     0     1     2     0     1     2     0     0     0
##    3:     0     1     2     0     0     0     0     0     0
##    4:     0     1     2     0     0     0     0     0     0
##    5:     0     1     2     0     0     0     0     0     0
##   ---
## 2002:     0     2     1     0     0     0     0     0     0
## 2003:     0     2     2     0     0     0     0     0     0
## 2004:     0     2     1     0     0     0     0     0     0
## 2005:     0     1     2     0     0     0     0     0     0
## 2006:     0     1     2     1     2     0     0     0     0
```

Features: Viewing Behavior

Data Import

We use our knowledge and intuiton about TV viewing to generate features we believe would carry information about the household composition.

1. Dimension time
   - Weekend vs. Workingdays
   - daytime
2. Dimension content (genre)
   - type of channel
   - type of program

To split the data by weekend vs workingdays we do not specify anything but later simply use the date variable.

To split the data by daytime the `tv` package allows us to specify so called timebands.

We are interested in the viewing on household level. The `tv` package allows to specify this with the `setup(obs = "hh")`. Simply summing up all individuals viewing within a household doe not mean household level. If people watch together than this viewing counts only once.

```
id <- setup(
  day = days,
  guest = FALSE,
  obs = "hh",
  dem.var = "sg",
  tmb = list(
    '02to06' = c(start = '02:00:00', end = '05:59:59'),
    '06to08' = c(start = '06:00:00', end = '07:59:59'),
    '08to11' = c(start = '08:00:00', end = '10:59:59'),
    '11to13' = c(start = '11:00:00', end = '12:59:59'),
    '13to17' = c(start = '13:00:00', end = '16:59:59'),
    '17to20' = c(start = '17:00:00', end = '19:59:59'),
    '20to22' = c(start = '20:00:00', end = '21:59:59'),
    '22to24' = c(start = '22:00:00', end = '23:59:59'),
    '24to02' = c(start = '24:00:00', end = '25:59:59')
  )
  )
import(id)
```

sum viewing by weekpart and daytime

```
dem.add(dem, 'calendar')
nday <- dem[, .(N = uniqueN(day)), k=.(pin, wend)]
```

```
res <- calc(
  dt = view[dem, on=c('day',"pin")],
  by = c("day","tmb","pin"),
  period = "tmb.dur"
)

dem.add(res, 'calendar')

res <- res[, sum(dur), k=c("wend","tmb","pin")]
res[, nday := nday[res, on=c('wend','pin'), N]]
# na.omit(res, invert = TRUE)
```

```
res[, mean.dur := V1 / nday]
res[, wend := id$lab$wend[res, on='wend', label]]

X.tmb <- dcast(res, pin ~ wend + tmb, value.var = "mean.dur")
X.tmb <- X.tmb[hh[,.(pin)], on="pin"]
na.to.0(X.tmb) # na.omit(X.tmb, invert = TRUE)
```

```
##        pin Weekend_02to06 Weekend_06to08 Weekend_08to11 Weekend_11to13
##    1:    6        0.00000         0.0000     372.875000      285.25000
##    2:    9       88.31250        20.7500     621.500000     1053.93750
##    3:   14      328.12500        39.2500      12.000000       75.81250
##    4:   20     1019.66667       555.1333     824.466667     1216.93333
##    5:   21      607.37500       917.2500    3143.500000     2310.06250
##   ---
## 2002: 6196        0.00000         0.0000     907.125000     1435.25000
## 2003: 6197        0.00000         0.0000       8.222222        0.00000
## 2004: 6200        0.00000         0.0000      48.363636      311.09091
## 2005: 6201      102.33333         0.0000     492.111111       62.88889
## 2006: 6204       38.88889        50.0000    2802.888889     1403.55556
##       Weekend_13to17 Weekend_17to20 Weekend_20to22 Weekend_22to24
##    1:       766.3125       666.3750       2021.000      1782.2500
##    2:      1312.4375      1795.8125       2719.688      2357.1250
##    3:       378.0625      2086.2500       4219.188      1342.1875
##    4:      6214.6667      7163.6667       4844.333      3472.8667
##    5:      8009.3125      4244.0625       4007.312      6196.2500
##   ---
## 2002:     12953.7500     18287.6250      11247.875      2972.2500
## 2003:       118.8889       509.5556       1964.333       357.4444
## 2004:       877.3636      4763.4545       4599.909      5025.9091
## 2005:      3217.2222      4911.0000       3619.778       553.4444
## 2006:      1668.2222      5205.6667       5827.667      3320.1111
##       Weekend_24to02 Workday_02to06 Workday_06to08 Workday_08to11
##    1:      284.56250      5.8000000      0.0000000        0.00000
##    2:      493.68750     39.5500000     15.2000000        8.02500
##    3:        2.12500     15.8205128      0.5641026        0.00000
##    4:     1777.06667   1046.9750000    689.9500000      688.07500
##    5:     3049.75000     81.2250000   1711.8250000     2832.95000
##   ---
## 2002:        0.00000      0.6666667      0.0000000       47.94444
## 2003:       44.33333      0.0000000      0.0000000      435.71429
## 2004:     1509.54545      0.0000000      0.0000000        0.00000
## 2005:        0.00000      0.0000000      0.0000000      210.04000
## 2006:      333.44444      0.3809524   1942.9047619      224.04762
##       Workday_11to13 Workday_13to17 Workday_17to20 Workday_20to22
##    1:        18.6500       0.000000        150.125       1968.975
##    2:       451.6750    2048.475000       2485.550       2556.475
##    3:        37.5641      55.692308       2426.641       5507.026
##    4:       729.9750    2884.475000       5020.825       5629.200
##    5:      1160.0500    6320.900000       7995.600       5715.525
##   ---
## 2002:         3.0000  11054.611111      14862.833      11018.167
## 2003:       647.1429     923.761905       2295.571       3388.333
## 2004:         0.0000       1.576923       1771.885       5281.192
## 2005:        20.5200    1658.040000       3578.560       4574.600
## 2006:       532.0952     250.523810       3630.762       6373.238
##       Workday_22to24 Workday_24to02
##    1:      1585.4250       53.95000
##    2:      3339.2500      963.57500
##    3:      1410.5641        7.00000
##    4:      3629.1000     1159.57500
##    5:      5706.3750     1245.47500
##   ---
## 2002:      2541.1667        0.00000
## 2003:       840.4286      380.33333
## 2004:      5766.8846     1478.57692
## 2005:      1155.2800      109.00000
## 2006:      2336.5238       43.04762
```

```
setnames(X.tmb, -1, paste0("day_",tolower(names(X.tmb)[-1])))
```

sum viewing by channel groups

```
cols <- c("chn.type","chn.country","chn.lang")
view[, (cols) := id$lab$sta[view, on=c(id="base"), mget(cols)]]
ordercol(view, cols, "chn.name")
# view[, any(is.na(chn.type))]

res.type <- calc(
  dt = view[dem, on=c('day',"pin")],
  by = c("day","pin","chn.type"),
  period = "day"
)
res.type <- res.type[, sum(dur), k=c("chn.type","pin")]
res.type[, nday := nday[, .(N=sum(N)), k=pin][res.type, on='pin', N]]
# na.omit(res.type, invert = TRUE)
res.type[, mean.dur := V1 / nday]

X.chn.type <- dcast(res.type, pin ~ chn.type, value.var = "mean.dur")
X.chn.type <- X.chn.type[hh[,.(pin)], on="pin"]
na.to.0(X.chn.type) # na.omit(X.chn.type, invert = TRUE)
```

```
##        pin           0       Arts GeneralistPrivate GeneralistPublic
##    1:    6   29.053571   10.83929          696.0000         3160.232
##    2:    9   76.892857   61.44643         2666.9286         6800.214
##    3:   14    1.981818   49.20000          922.2364         7525.727
##    4:   20 6598.727273  270.72727         6132.2182         8481.509
##    5:   21 1321.732143  144.25000         8266.3214        20974.089
##   ---
## 2002: 6196  172.038462 1531.23077         3954.4615        18005.692
## 2003: 6197   11.200000    4.10000         4073.5333         2395.467
## 2004: 6200  211.540541  390.45946         1372.5405        11492.000
## 2005: 6201   46.264706  133.14706          244.7353        10056.912
## 2006: 6204   54.566667   10.50000         5775.6333         8309.333
##                 Kids LivestileIndoor LivestileOutdoor      Local MovieSeries
##    1:    0.0000000       11.053571         0.000000 275.946429  276.839286
##    2:    2.8035714      210.839286         0.000000  36.214286  227.607143
##    3:    0.8727273        1.181818         1.581818 562.927273    9.418182
##    4:  210.4363636      949.163636         6.709091  51.400000  235.127273
##    5:   96.7857143      422.821429        65.839286 770.017857  357.910714
##   ---
## 2002: 5935.7692308      138.461538      3209.769231 622.884615 3031.846154
## 2003:    0.0000000       48.333333         0.000000 598.033333    6.666667
## 2004:   14.1891892      167.675676        35.297297   0.000000  793.594595
## 2005:    0.0000000        0.000000         0.000000   1.794118   41.911765
## 2006:    0.0000000        6.733333         0.000000 418.366667    0.000000
##            Music   Nature        News  PayTV Religion     Sport
##    1:   0.000000   0.0000    7.446429    0.0 0.000000   0.00000
##    2: 556.750000 188.7679   25.964286    0.0 0.000000 640.62500
##    3:   1.836364   0.0000   98.636364    0.0 0.000000   0.80000
##    4:   0.000000   0.0000   63.218182    0.0 9.090909   0.00000
##    5:   0.000000   0.0000  229.142857    0.0 0.000000  39.57143
##   ---
## 2002: 169.076923   0.0000 5118.423077    0.0 0.000000 185.03846
## 2003:   0.000000   0.0000    1.400000    0.0 0.000000   0.00000
## 2004:   0.000000   0.0000  665.810811    0.0 0.000000   0.00000
## 2005:   0.000000   0.0000  263.264706    0.0 0.000000 955.50000
## 2006:   8.466667   0.0000    1.600000 2343.4 0.000000   0.00000
```

```
res.country <- calc(
  dt = view[dem, on=c("day","pin")],
  by = c("day","pin","chn.country"),
  period = "day"
)
res.country <- res.country[, sum(dur), k=c("chn.country","pin")]
res.country[, nday := nday[, .(N=sum(N)), k=pin][res.country, on='pin', N]]
# na.omit(res.country, invert = TRUE)
res.country[, mean.dur := V1 / nday]

X.chn.country <- dcast(res.country, pin ~ chn.country, value.var = "mean.dur")
X.chn.country <- X.chn.country[hh[,.(pin)], on="pin"]
na.to.0(X.chn.country) # na.omit(X.chn.country, invert = TRUE)
```

```
##        pin           0    foreign     swiss
##    1:    6   29.053571  2052.8036  2385.554
##    2:    9    5.107143  6521.5357  4968.411
##    3:   14    1.981818   354.5273  8819.891
##    4:   20 6598.727273  6758.6909  9650.909
##    5:   21 1321.732143 19363.1250 12003.625
##   ---
## 2002: 6196  172.038462 24724.4615 17178.192
## 2003: 6197   11.200000  4312.7333  2814.800
## 2004: 6200  211.540541  6699.0541  8232.514
## 2005: 6201   46.264706   468.1176 11229.147
## 2006: 6204   54.566667 10306.6333  6567.400
```

```
res.lang <- calc(
  dt = view[dem, on=c("day","pin")],
  by = c("day","pin","chn.lang"),
  period = "day"
)
res.lang <- res.lang[, sum(dur), k=c("chn.lang","pin")]
res.lang[, nday := nday[, .(N=sum(N)), k=pin][res.lang, on='pin', N]]
# na.omit(res.lang, invert = TRUE)
res.lang[, mean.dur := V1 / nday]

X.chn.lang <- dcast(res.lang, pin ~ chn.lang, value.var = "mean.dur")
X.chn.lang <- X.chn.lang[hh[,.(pin)], on="pin"]
na.to.0(X.chn.lang) # na.omit(X.chn.lang, invert = TRUE)
```

```
##        pin           0    english       french       german   italian
##    1:    6   29.053571 295.357143  4136.142857     5.821429  1.035714
##    2:    9    5.107143  56.303571 11382.392857    44.303571  5.803571
##    3:   14    1.981818   0.000000     1.309091  9173.109091  0.000000
##    4:   20 6598.727273  17.109091     0.000000 16392.490909  0.000000
##    5:   21 1321.732143   0.000000     0.000000 31366.750000  0.000000
##   ---
## 2002: 6196  172.038462  29.615385     4.730769 41865.000000  3.307692
## 2003: 6197   11.200000   0.000000     0.000000  7127.533333  0.000000
## 2004: 6200  211.540541   3.027027    28.540541 14854.810811 45.189189
## 2005: 6201   46.264706   0.000000     3.029412 11694.235294  0.000000
## 2006: 6204   54.566667   0.000000 16872.833333     1.200000  0.000000
##          other
##    1: 0.000000
##    2: 1.142857
##    3: 0.000000
##    4: 0.000000
##    5: 0.000000
##   ---
## 2002: 0.000000
## 2003: 0.000000
## 2004: 0.000000
## 2005: 0.000000
## 2006: 0.000000
```

sum viewing by program genre

```
view <- overlap.join(view, prog, type='prg')

res.genre <- calc(
  dt = view[dem, on=c("day","pin")],
  by = c("day","pin","genre"),
  period = "day"
)
res.genre <- res.genre[, sum(dur), k=c("genre","pin")]
res.genre[, nday := nday[, .(N=sum(N)), k=pin][res.genre, on='pin', N]]
# na.omit(res.genre, invert = TRUE)
res.genre[, mean.dur := V1 / nday]
```

```
X.genre <- dcast(res.genre, pin ~ genre, value.var = "mean.dur")
X.genre <- X.genre[hh[,.(pin)], on="pin"]
na.to.0(X.genre) # na.omit(X.genre, invert = TRUE)
```

```
##        pin commercial      info          kids   missing     movie     music
##    1:    6   192.9821  451.1607     0.0000000 1042.6964  714.3214  0.000000
##    2:    9   363.2143 1294.2500    10.5000000 1338.7679  908.1964  5.357143
##    3:   14   347.3636 2018.7273     0.5272727  176.0545    9.6000  5.672727
##    4:   20  1454.4909 2404.4364    25.0363636  835.6727  929.5091 16.800000
##    5:   21  2593.2321 3352.4464    63.4107143 6394.1964 2186.1607 78.910714
##   ---
## 2002: 6196  2977.7308 4364.5000 2319.4230769 3869.7308 1331.2692 78.538462
## 2003: 6197   405.9667 1234.3000     0.0000000  438.3333  624.5667  6.500000
## 2004: 6200   867.1892 1253.2432     0.7027027 2993.0811  755.6486  0.000000
## 2005: 6201   949.6176 1255.2353     1.3235294  276.7059  438.9412 30.470588
## 2006: 6204  1098.7667 1989.0333   970.7666667 1702.4000 1432.9000  2.133333
##            news       other    series   service        show       sport
##    1:  190.5179   0.0000000  500.7857  3.392857   17.30357  376.69643
##    2: 2008.1250   0.0000000  300.6786 27.357143  595.17857   38.75000
##    3: 2525.3273   0.0000000 1059.2000  6.000000 1624.89091  571.94545
##    4: 1329.4364   0.7454545 4194.3091  6.181818 1359.45455 3000.14545
##    5: 5712.0357  10.3571429  907.4107 31.607143 3598.89286  222.26786
##   ---
## 2002: 3276.1154 206.6153846 6412.7692 53.500000 1535.57692 2268.15385
## 2003:  376.3333   1.2333333 2654.5667  2.933333  500.56667   14.66667
## 2004: 1796.4865  17.3243243  870.1622 17.297297  522.45946 2181.72973
## 2005: 2107.1765  29.9411765 1855.2353 19.411765  384.91176 3897.00000
## 2006:  941.4000   2.3333333 2425.2667 10.266667  608.06667 1155.16667
##            talk   trailer
##    1:   0.00000  40.91071
##    2:   0.37500 134.80357
##    3: 239.61818 164.47273
##    4:  22.83636 443.45455
##    5: 228.41071 731.91071
##   ---
## 2002:  40.07692 910.46154
## 2003:  20.60000 171.83333
## 2004: 206.08108 279.59459
## 2005:   0.00000 360.82353
## 2006:   0.00000 236.26667
```

```
setnames(X.genre, -1, paste0("prg_",tolower(names(X.genre)[-1])))
```

Export Data

```
ordercol(hh, 'pin')
hh.composition <- hh[, day := NULL]
predictors <- X.tmb[X.chn[X.genre, on="pin"], on="pin"]
setnames(hh.composition, 'pin', 'hh')
setnames(predictors, 'pin', 'hh')
```

```r
setorder(hh.composition, 'hh')
setorder(predictors, 'hh')

# save(hh.composition, predictors, file = '~/diplom/data/data_predictors.RData')
```

```r
hh.composition
```

```
##         hh sg hhsize age_1 age_2 age_3 age_4 age_5 age_6 age_7 age_8 sex_1
##    1:    6  2      2    70    67     0     0     0     0     0     0     1
##    2:    9  2      4    55    50     0    21    17     0     0     0     1
##    3:   14  1      2    71    72     0     0     0     0     0     0     1
##    4:   20  1      2    59    49     0     0     0     0     0     0     1
##    5:   21  1      2    63    52     0     0     0     0     0     0     1
##   ---
## 2002: 6196  1      2    74    76     0     0     0     0     0     0     2
## 2003: 6197  1      2    40    47     0     0     0     0     0     0     2
## 2004: 6200  1      2    63    72     0     0     0     0     0     0     2
## 2005: 6201  1      2    71    73     0     0     0     0     0     0     1
## 2006: 6204  2      4    52    47    17    13     0     0     0     0     1
##       sex_2 sex_3 sex_4 sex_5 sex_6 sex_7 sex_8
##    1:     2     0     0     0     0     0     0
##    2:     2     0     1     2     0     0     0
##    3:     2     0     0     0     0     0     0
##    4:     2     0     0     0     0     0     0
##    5:     2     0     0     0     0     0     0
##   ---
## 2002:     1     0     0     0     0     0     0
## 2003:     2     0     0     0     0     0     0
## 2004:     1     0     0     0     0     0     0
## 2005:     2     0     0     0     0     0     0
## 2006:     2     1     2     0     0     0     0
```

```r
predictors[, 1:10]
```

```
##         hh day_weekend_02to06 day_weekend_06to08 day_weekend_08to11
##    1:    6            0.00000             0.0000         372.875000
##    2:    9           88.31250            20.7500         621.500000
##    3:   14          328.12500            39.2500          12.000000
##    4:   20         1019.66667           555.1333         824.466667
##    5:   21          607.37500           917.2500        3143.500000
##   ---
## 2002: 6196            0.00000             0.0000         907.125000
## 2003: 6197            0.00000             0.0000           8.222222
## 2004: 6200            0.00000             0.0000          48.363636
## 2005: 6201          102.33333             0.0000         492.111111
## 2006: 6204           38.88889            50.0000        2802.888889
##       day_weekend_11to13 day_weekend_13to17 day_weekend_17to20
##    1:          285.25000           766.3125           666.3750
##    2:         1053.93750          1312.4375          1795.8125
##    3:           75.81250           378.0625          2086.2500
##    4:         1216.93333          6214.6667          7163.6667
##    5:         2310.06250          8009.3125          4244.0625
##   ---
## 2002:         1435.25000         12953.7500         18287.6250
## 2003:            0.00000           118.8889           509.5556
## 2004:          311.09091           877.3636          4763.4545
## 2005:           62.88889          3217.2222          4911.0000
## 2006:         1403.55556          1668.2222          5205.6667
##       day_weekend_20to22 day_weekend_22to24 day_weekend_24to02
##    1:           2021.000          1782.2500          284.56250
##    2:           2719.688          2357.1250          493.68750
##    3:           4219.188          1342.1875            2.12500
##    4:           4844.333          3472.8667         1777.06667
##    5:           4007.312          6196.2500         3049.75000
##   ---
## 2002:          11247.875          2972.2500            0.00000
## 2003:           1964.333           357.4444           44.33333
## 2004:           4599.909          5025.9091         1509.54545
## 2005:           3619.778           553.4444            0.00000
## 2006:           5827.667          3320.1111          333.44444
```

```
cbind(names(predictors))
```

```
##       [,1]
##  [1,] "hh"
##  [2,] "day_weekend_02to06"
##  [3,] "day_weekend_06to08"
##  [4,] "day_weekend_08to11"
##  [5,] "day_weekend_11to13"
##  [6,] "day_weekend_13to17"
##  [7,] "day_weekend_17to20"
##  [8,] "day_weekend_20to22"
##  [9,] "day_weekend_22to24"
## [10,] "day_weekend_24to02"
## [11,] "day_workday_02to06"
## [12,] "day_workday_06to08"
## [13,] "day_workday_08to11"
## [14,] "day_workday_11to13"
## [15,] "day_workday_13to17"
## [16,] "day_workday_17to20"
## [17,] "day_workday_20to22"
## [18,] "day_workday_22to24"
## [19,] "day_workday_24to02"
## [20,] "chn_arts"
## [21,] "chn_generalistprivate"
## [22,] "chn_generalistpublic"
## [23,] "chn_kids"
## [24,] "chn_livestileindoor"
## [25,] "chn_livestileoutdoor"
## [26,] "chn_local"
## [27,] "chn_movieseries"
## [28,] "chn_music"
## [29,] "chn_nature"
## [30,] "chn_news"
## [31,] "chn_paytv"
## [32,] "chn_religion"
```

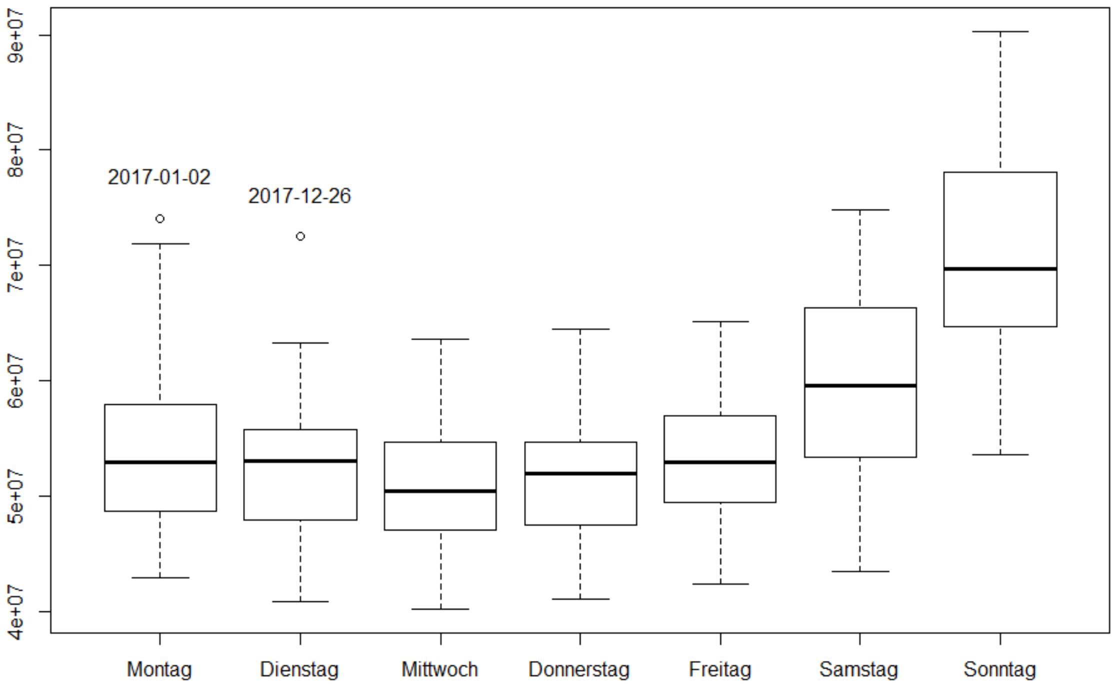

Figure 1: Amount of TV viewing by weekdays during 2017. More viewing on weekends, festival days behave like sundays

```
## [33,] "chn_sport"
## [34,] "chn_foreign"
## [35,] "chn_swiss"
## [36,] "chn_english"
## [37,] "chn_french"
## [38,] "chn_german"
## [39,] "chn_italian"
## [40,] "chn_other"
## [41,] "prg_commercial"
## [42,] "prg_info"
## [43,] "prg_kids"
## [44,] "prg_missing"
## [45,] "prg_movie"
## [46,] "prg_music"
## [47,] "prg_news"
## [48,] "prg_other"
## [49,] "prg_series"
## [50,] "prg_service"
## [51,] "prg_show"
## [52,] "prg_sport"
## [53,] "prg_talk"
## [54,] "prg_trailer"
```

Appendix

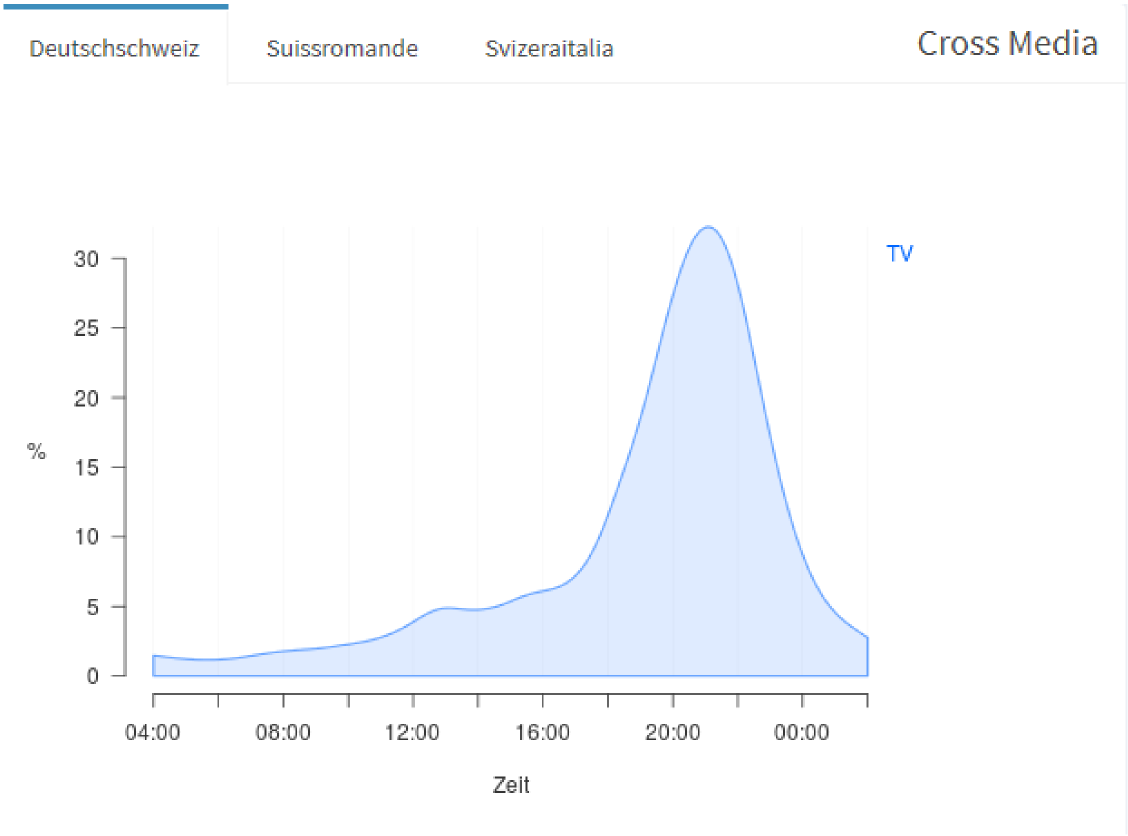

Figure 2: Relative amount of TV viewing during a day. Averaged across one Year.